MAML is a Noisy Contrastive Learner

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Model-agnostic meta-learning (MAML) is one of the most popular and widely adopted meta-learning algorithms nowadays, which achieves remarkable success in various learning problems. Yet, with the unique design of nested inner-loop and outer-loop updates which govern the task-specific and meta-model-centric learning respectively, the underlying learning objective of MAML still remains implicit and thus impedes a more straightforward understanding of it. In this paper, we provide a new perspective of the working mechanism of MAML. We discover that MAML is analogous to a meta-learner using a supervised contrastive objective function, where the query features are pulled towards the support features of the same class and against those of different classes, in which such contrastiveness is experimentally verified via an analysis based on the cosine similarity. Moreover, we reveal that the vanilla MAML algorithm has an undesirable interference term originating from the random initialization and the cross-task interaction. We therefore propose a simple but effective technique, zeroing trick, to alleviate such interference, where extensive experiments are then conducted on both miniImagenet and Omniglot datasets to demonstrate the consistent improvement brought by our proposed technique thus validating its effectiveness.

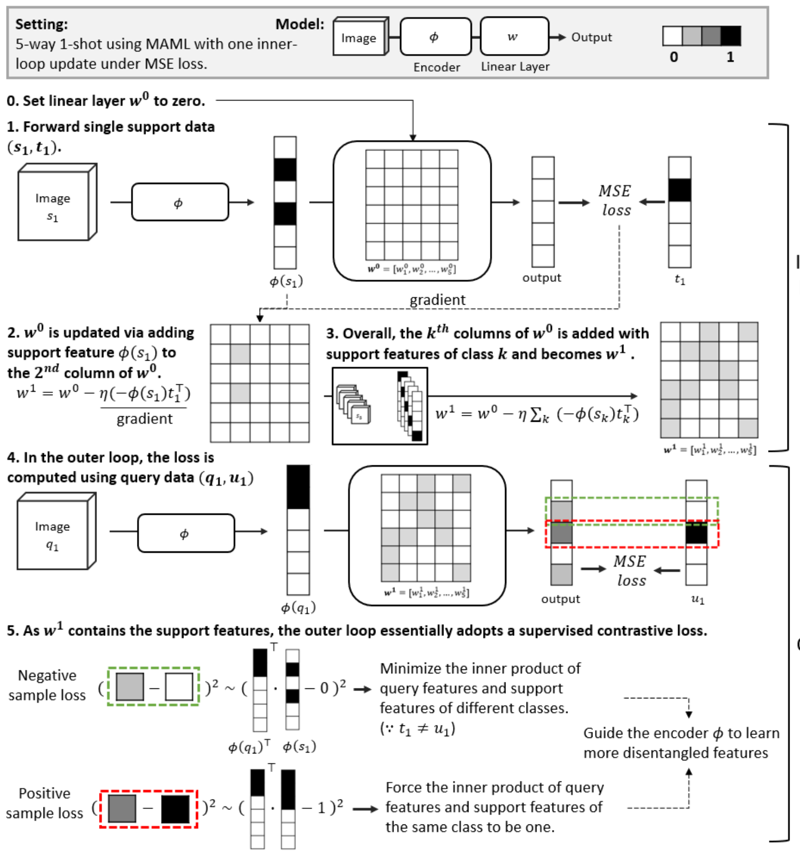
1 Introduction

Humans can learn from very few samples. They can readily establish their cognition and understanding to novel tasks, environments, or domains even with very limited experience in the corresponding circumstances. *Meta-learning*, a subfield of machine learning aims at equipping machines with such capacity to effectively accommodate new scenarios [1, 2]. Machines learn to extract task-agnostic information so that their performance on unseen tasks can be improved [3].

One highly influential meta-learning algorithm is Model Agnostic Meta-Learning (MAML) [4], which has inspired numerous follow-up extensions [5, 6, 7, 8, 9, 10]. MAML estimates a set of model parameters such that an adaptation of the model to a new task only requires some updates to those parameters. We take the few-shot classification task as an example to review the algorithmic procedure of MAML. A few-shot classification problem refers to classifying samples from some classes (i.e. query data) after seeing a few examples per class (i.e. support data). In a meta-learning scenario, we consider a distribution of tasks, where each task is a few-shot classification problem and different tasks have different target classes. MAML aims to meta-train the base-model based on training tasks (i.e., the meta-training dataset) and evaluate the performance of the base-model on the testing tasks sampled from a held-out unseen dataset (i.e. the meta-testing dataset). In meta-training, MAML follows a bi-level optimization scheme composed of the inner loop and the outer loop, as shown in Algorithm A (please refer to Section 2 for detailed definition). In the inner loop (also known as *fast adaptation*), the base-model θ is updated to θ' using the support set. In the outer loop, a loss is evaluated on θ' using the query set, and its gradient is computed with respect to θ to update the base-model. Since the outer loop requires computing the gradient of gradient , it is called second-order MAML (SOMAML). To prevent computing the Hessian matrix, [4] proposes first-order MAML (FOMAML) that uses the gradient computed with respect to θ' to update the base-model.

The widely accepted intuition behind MAML is that the models are *encouraged* to learn a general-purpose representations which are broadly applicable not only to the seen tasks but also to novel tasks [4, 11, 12]. [11] confirms this perspective by showing that during fast adaptation, the majority of changes made are in the final linear layers, while the convolution layers (as the feature encoder) remain almost static. This implies the models trained with MAML learn a good feature representation and that they only have to change the linear mapping from features to outputs during the fast adaptation . Similar ideas of freezing feature extractor during the inner loop have also been explored in [13, 14, 15], and has been held as assumption in theoretical works such as [16, 17, 18].

While this intuition sounds satisfactory, we further ask the most fundamental questions: (1) In what sense does MAML *guide* any model to learn general-purpose representations? (2) And how do the inner loop and outer loop in the training mechanism of MAML collaboratively prompt to achieve so? (3) What is the role of the support and query data and how do they, if any, interact with each other? In this paper, we answer these questions and give new insights on the working mechanism of MAML, which turns out to be closely connected to supervised contrastive learning (SCL)[1].



Setting:
5-way 1-shot using MAML with one inner-loop update under MSE loss.
Model:
Image
ϕ
Encoder
w
Linear Layer
Output
0
1
0. Set linear layer w^0 to zero.
1. Forward single support data (s_1, t_1).
Image s_1
$\phi(s_1)$
$w^0 = [w_1^0, w_2^0, \dots, w_5^0]$
output
MSE loss
t_1
gradient
Inner loop
2. w^0 is updated via adding support feature $\phi(s_1)$ to the 2nd column of w^0.
$w^1 = w^0 - \eta(-\phi(s_1)t_1^\top)$
gradient
3. Overall, the k^{th} columns of w^0 is added with support features of class k and becomes w^1.
$w^1 = w^0 - \eta \sum_k (-\phi(s_k)t_k^\top)$
$w^1 = [w_1^1, w_2^1, \dots, w_5^1]$
4. In the outer loop, the loss is computed using query data (q_1, u_1)
Image q_1
$\phi(q_1)$
$w^1 = [w_1^1, w_2^1, \dots, w_5^1]$
output
MSE loss
u_1
Outer loop
5. As w^1 contains the support features, the outer loop essentially adopts a supervised contrastive loss.
Negative sample loss
$(\cdot - \cdot)^2 \sim (\cdot - 0)^2$
$\phi(q_1)^\top$ $\phi(s_1)$
Minimize the inner product of query features and support features of different classes. $(\because t_1 \neq u_1)$
Guide the encoder ϕ to learn more disentangled features
Positive sample loss
$(\cdot - \cdot)^2 \sim (\cdot - 1)^2$
Force the inner product of query features and support features of the same class to be one.

Figure 1: A step-by-step illustration showing the SCL objective underlying MAML. Assume the linear layer $\mathbf{w}^0$ to be zero, we find that, during the inner loop, the i^{th} column of $\mathbf{w}^0$ is added with the support features of class i. In the outer loop, the output of a query sample is the inner product of $\phi(q_1)$ and $\mathbf{w}^1$, which is essentially the inner product of the query features and all the support features. Thus, MAML displays the characteristic of supervised contrastiveness, where the support set acts as the positive and negative samples.

Here, we provide a sketch of our analysis in Figure 1. We consider a setting of (a) a 5-way 1-shot paradigm of few-shot learning, (b) the mean square error (MSE) between the one-hot encoding of

[1]We use the term *supervised contrastiveness* to refer to the setting of using groundtruth label information to differentiate positive samples and negative samples [19]. This setting is different from (unsupervised/self-supervised) *contrastive learning*.

groundtruth label and the outputs as the objective function, and (c) MAML with a single inner-loop update. At the beginning of the inner loop, we set the linear layer $\mathbf{w}^0$ to zero. Then, the inner loop update of $\mathbf{w}^0$ is equivalent to adding the support features to $\mathbf{w}^0$. In the outer loop, the output of a query sample q_1 is actually the inner product between the query feature $\phi(q_1)$ and all support features (the learning rate is omitted for now). As the groundtruth is an one-hot vector, the encoder is trained to either minimize the inner product between the query features and the support features (when they are from different classes, as shown in the green box), or to pull the inner product between the query features and the support features to 1 (when they have the same label, as shown in the red box). Under this derivation, the inner loop and the outer loop together manifest a SCL objective. Particularly, as the vanilla implementation of MAML uses non-zero (random) initialization for the linear layer, we will show such initialization leads to a noisy SCL objective which would impede the training.

In this paper, we present a more general case of MAML with cross entropy loss and show the underlying learning protocol of the vanilla MAML algorithm as an interfered SCL in Section 2. We then experimentally verify the supervised contrastiveness of MAML, and proposed to mitigate the interference with our simple but effective technique of the zero-initialization and zeroing trick (cf. Section 3). In summary, our main contributions are three-fold:

- We show that MAML is an implicitly SCL algorithm and that the noise comes from the randomly initialized linear layer and the cross-task interaction, compromising the capability of the encoder.
- We verify the inherent contrastiveness of MAML based on the cosine similarity analysis.
- We experimentally validate our analysis and show that applying the zeroing trick induces a notable improvement in testing accuracy during training. We also show that during meta-testing, a pronounced increase in accuracy occurs when the zeroing trick is applied.

2 Why MAML is Implicitly a Noisy Supervised Contrastive Algorithm?

2.1 Preliminary: Supervised Contrastive Learning

In [19], supervised contrastive learning (SCL) is described as "contrasts the set of all samples from the same class as positives against the negatives from the remainder of the batch" and "embeddings from the same class are pulled closer together than embeddings from different classes." For a sample s, label information is leveraged to indicate positive samples (i.e., samples having the same label as sample s) and negative samples (i.e., samples having different labels to sample s). The loss of SCL is designed to increase the similarity (or decrease metric distance) of embeddings of positive samples and to decrease the similarity (or increase the metric distance) of embeddings of negative samples [20, 21, 19]. In essence, SCL combines supervised learning and contrastive learning, and it differs from supervised learning in that the loss contains a measurement of the similarity or metric distance between the embedding of a sample and embedding of its positive/negative sample pairs.

Now we give a formal definition of SCL. For a set of N samples drawn from a n-class dataset. Let $i \in I = \{1, ..., N\}$ be the index of an arbitrary sample. Let $A(i) = I \setminus \{i\}$, $P(i)$ be the set of indices of all positive samples of sample i, and $N(i) = A(i) \setminus P(i)$ be the set of indices of all negative samples of sample i. Let z_i indicates the embedding of sample i.

Definition 1 *Let M_{sim} be a measurement of similarity (e.g., inner product, cosine similarity). Training algorithms that adopt loss of the following form belong to SCL:*

$$L_{SCL} = \sum_{p \in P(i)} c_p^- M_{sim}(z_i, z_p) + \sum_{n \in N(i)} c_n^+ M_{sim}(z_i, z_n) + c \tag{1}$$

where $c_p^- < 0$ and $c_n^+ > 0$ for all n and p, and c is a constant independent of samples. A training algorithm that follows Eq.(1)*, but with either (a) $c_n^+ < 0$ for some n or (b) c is a constant dependent of samples, belongs to noisy SCL.*

2.2 Problem Setup

We provide the detailed derivation to show that MAML is implicitly a noisy SCL, where we adopt the few-shot classification as the example application. Consider drawing a batch of tasks

$\{T_1, \dots, T_{N_{batch}}\}$ from a meta-training task distribution D. Each task T_n contains a support set S_n and a query set Q_n, where $S_n = \{(s_m, t_m)\}_{m=1}^{N_{way} \times N_{shot}}$, $Q_n = \{(q_m, u_m)\}_{m=1}^{N_{way} \times N_{query}}$, $s_m, q_m \in \mathbf{R}^{N_{in}}$ are data samples, and $t_m, u_m \in \{1, ..., N_{way}\}$ are labels. We denote N_{way} the number of classes in each task, and $\{N_{shot}, N_{query}\}$ respectively the number of support and query samples per class. The architecture of our base-model comprises of a convolutional encoder $\phi : \mathbf{R}^{N_{in}} \to \mathbf{R}^{N_f}$ (parameterized by φ), a fully connected linear head $\mathbf{w} \in \mathbf{R}^{N_f \times N_{way}}$, and a Softmax output layer, where N_f is the dimension of the feature space. We denote the k^{th} column of $\mathbf{w}$ as $\mathbf{w_k}$. Note that the base-model parameters θ consist of φ and $\mathbf{w}$.

As shown in Algorithm A, both SOMAML and FOMAML adopt a training strategy comprising the inner loop and the outer loop. At the beginning of a meta-training iteration, we sample N_{batch} tasks. For each task T_n, we perform inner loop update using the inner loop loss (c.f. Eq. (2)) evaluated on the support data, and then evaluate the outer loop loss (c.f. Eq. (3)) on the updated base-model using the query data. In the i^{th} step of the inner loop, the parameters $\{\phi^{i-1}, \mathbf{w}^{i-1}\}$ are updated to $\{\phi^i, \mathbf{w}^i\}$ using the multi-class cross entropy loss evaluated on the support dataset S_n as

$$L_{\{\phi^i, \mathbf{w}^i\}, S_n} = \mathop{\mathbf{E}}_{(s,t) \sim S_n} \sum_{j=1}^{N_{way}} 1_{j=t} [-\log \frac{\exp(\phi^i(s)^\top \mathbf{w_j}^i)}{\sum_{k=1}^{N_{way}} \exp(\phi^i(s)^\top \mathbf{w_k}^i)}] \tag{2}$$

After N_{step} inner loop updates, we compute the outer loop loss using the query data Q_n:

$$L_{\{\phi^{N_{step}}, \mathbf{w}^{N_{step}}\}, Q_n} = \mathop{\mathbf{E}}_{(q,u) \sim Q_n} [-\log \frac{\exp(\phi^{N_{step}}(q)^\top \mathbf{w_u}^{N_{step}})}{\sum_{k=1}^{N_{way}} \exp(\phi^{N_{step}}(q)^\top \mathbf{w_k}^{N_{step}})}] \tag{3}$$

Then, we sum up the outer loop losses of all tasks, and perform gradient descent to update the base-model's initial parameters $\{\phi^0, \mathbf{w}^0\}$.

In the following, we aim to show the noisy supervised contrastiveness entailed in MAM L's objective function. To achieve this goal, we assume that *the encoder ϕ is frozen during the inner loop*. Without loss of generality, we consider a base-model trained with MAML with $N_{batch} = 1$ and $N_{step} = 1$, and we discuss the generalized version in Section 2.5. For simplicity, the k^{th} channel softmax predictive output $\frac{\exp(\phi(s)^\top \mathbf{w_k}^0)}{\sum_{j=1}^{N_{way}} \exp(\phi(s)^\top \mathbf{w_j}^0)}$ ($\frac{\exp(\phi(q)^\top \mathbf{w_k}^1)}{\sum_{j=1}^{N_{way}} \exp(\phi(q)^\top \mathbf{w_j}^1)}$) of sample s (q) is denoted as s_k (q_k).

2.3 Inner Loop and Outer Loop Update of Linear Layer and Encoder

In this section, we primarily focus on the update of parameters in the case of FOMAML. The full derivation and discussion of SOMAML are provided in Section B.

Inner loop update of the linear layer. In the inner loop, the linear layer $\mathbf{w}^0$ is updated to $\mathbf{w}^1$ with a learning rate η as shown in Eq. (4) in both FOMAML and SOMAML. In contrast to the example in Figure 1, the columns of the linear layer are added with the weighted sum of the features extracted from support samples (i.e., support features). Compared to $\mathbf{w_k}^0$, $\mathbf{w_k}^1$ is pushed towards the support features of the same class (i.e., class k) with strength of $1 - \mathrm{s}_k$, while being pulled away from the support features of different classes with strength of s_k.

$$\mathbf{w_k}^1 = \mathbf{w_k}^0 - \eta \frac{\partial L_{\{\phi, \mathbf{w}^0\}, S}}{\partial \mathbf{w_k}^0} = \mathbf{w_k}^0 + \eta \mathop{\mathbf{E}}_{(s,t) \sim S} (1_{k=t} - \mathrm{s}_k) \phi(s) \tag{4}$$

Outer loop update of the linear layer. In the outer loop, $\mathbf{w}^0$ is updated using the query data with a learning rate ρ. For FOMAML, the final linear layer is updated as follows.

$$\mathbf{w'_k}^0 = \mathbf{w_k}^0 - \rho \frac{\partial L_{\{\phi, \mathbf{w}^1\}, Q}}{\partial \mathbf{w_k}^1} = \mathbf{w_k}^0 + \rho \mathop{\mathbf{E}}_{(q,u) \sim Q} (1_{k=u} - \mathrm{q}_k) \phi(q) \tag{5}$$

Note that the computation of q_k requires the inner-loop updated $\mathbf{w}^1$. Generally speaking, Eq. (5) resembles Eq. (4). It is obvious that, in the outer loop, the query features are added weightedly to the linear layer, and the strength of change relates to the output value.

Outer loop update of the encoder. Using the chain rule, the gradient of the outer loop loss with respect to φ is given by $\frac{\partial L_{\{\phi, \mathbf{w}^1\}, Q}}{\partial \varphi} = \mathbf{E}_{(q,u) \sim Q} \frac{\partial L_{\{\phi, \mathbf{w}^1\}, Q}}{\partial \phi(q)} \frac{\partial \phi(q)}{\partial \varphi} + \mathbf{E}_{(s,t) \sim S} \frac{\partial L_{\{\phi, \mathbf{w}^1\}, Q}}{\partial \phi(s)} \frac{\partial \phi(s)}{\partial \varphi}$, where the second term can be neglected when FOMAML is considered. Below, we take a deeper look at the backpropagated error of one query data $(q, u) \sim Q$. The full derivation is provided in Appendix B.3.

$$\frac{\partial L_{\{\phi,\mathbf{w}^1\},q}}{\partial \phi(q)} = \sum_{j=1}^{N_{way}} (\mathrm{q}_j - 1_{j=u}) \mathbf{w_j}^0 + \eta \underset{(s,t)\sim S}{\mathbf{E}} [-(\sum_{j=1}^{N_{way}} \mathrm{q}_j \mathrm{s}_j) + \mathrm{s}_u + \mathrm{q}_t - 1_{t=u}] \phi(s) \tag{6}$$

2.4 MAML is a Noisy Contrastive Learner

Reformulating the outer loop loss for the encoder as a noisy SCL loss. We observe from Eq. (6) that the actual loss for the encoder (evaluated on a single query data $(q, u) \sim Q$) is as follows:

$$L_{\{\phi,\mathbf{w}^1\},q} = \sum_{j=1}^{N_{way}} \underbrace{(\mathrm{q}_j - 1_{j=u}) \mathbf{w_j}^{0\top}}_{\text{stop gradient}} \phi(q) + \eta \underset{(s,t)\sim S}{\mathbf{E}} \underbrace{[-(\sum_{j=1}^{N_{way}} \mathrm{q}_j \mathrm{s}_j) + \mathrm{s}_u + \mathrm{q}_t - 1_{t=u}] \phi(s)^\top}_{\text{stop gradient}} \phi(q) \tag{7}$$

To better deliberate the effect of each term in the reformulated outer loop loss, we define the first term in Eq. (7) as *interference term*, the second term as *noisy contrastive term*, and the coefficients $-(\sum_{j=1}^{N_{way}} \mathrm{q}_j \mathrm{s}_j) + \mathrm{s}_u + \mathrm{q}_t - 1_{t=u}$ as *contrastive coefficients*.

Understanding the interference term. In the case of $j = u$, the outer loop loss forces the model to minimize $(\mathrm{q}_j - 1)\mathbf{w_j}^{0\top} \phi(q)$. This can be problematic because (a) at the beginning of training, $\mathbf{w}^0$ is assigned with random values and (b) $\mathbf{w}^0$ is added with query features of previous tasks as shown in Eq. (5). Consequently, $\phi(q)$ is pushed to a direction composed of previous query features or to a random direction, introducing an unnecessary *cross-task interference* or an *initialization interference* that slows down the training of the encoder. Noting that the cross-task interference also occurs at the testing period, since, at testing stage, $\mathbf{w}^0$ is already added with query features of training tasks, which can be an interference to testing tasks.

Understanding the noisy contrastive term. When the query and support data have the same label, e.g., class 1, then the contrastive coefficient becomes $-\sum_{j=2}^{N_{way}} \mathrm{q}_j \mathrm{s}_j - (1-\mathrm{q}_1)(1-\mathrm{s}_1) < 0$, indicating the encoder is updated to maximize the inner product between $\phi(q)$ and the support features of class 1. However, when the query and support data are in different classes, the sign of the contrastive coefficient can sometimes be negative. The outer loop loss thus cannot well contrast the query features against the support features of different classes, making this loss term not an ordinary SCL loss.

To better illustrate the influence of the interference term and the noisy contrastive term, we provide an ablation experiment in Section B.7.

Theorem 1 *With the assumption of (a) no inner loop update of the encoder, FOMAML is a noisy SCL algorithm. With assumptions of (a) no inner loop update of the encoder and (b) a single inner-loop update, SOMAML is a noisy SCL algorithm.*

Proof: For FOMAML, both Eq. (7) (one inner loop update step) and Eq. (22) (multiple inner loop update steps) follows Definition 1. For SOMAML, Eq. (14) follows Definition 1.

Introduction of the zeroing trick makes Eq. (7) **SCL losses.** To tackle the interference term and make the contrastive coefficients more accurate, we introduce the zeroing trick: setting the $\mathbf{w}^0$ to be zero after each outer loop update, as shown in Algorithm A. With the zeroing trick, the original outer loop loss (of FOMAML) becomes

$$L_{\{\phi,\mathbf{w}^1\},q} = \eta \underset{(s,t)\sim S}{\mathbf{E}} \underbrace{(\mathrm{q}_t - 1_{t=u}) \phi(s)^\top}_{\text{stop gradient}} \phi(q) \tag{8}$$

The zeroing trick brings two nontrivial effects: (a) eliminating the interference term in Eq. (7); (b) making the contrastive coefficients follow SCL. For (b), since all the predictive values of support data become the same, i.e., $s_k = \frac{1}{N_{way}}$, the contrastive coefficient becomes $\mathrm{q}_t - 1_{t=u}$, which is negative when the support and query data have the same label, and positive otherwise. With the zeroing trick, the contrastive coefficient follows the SCL loss.

Corollary 1 *With mild assumptions of (a) no inner loop update of the encoder, (b) a single inner-loop update and (c) training with the zeroing trick (i.e., the linear layer is zeroed at the end of each outer loop), both FOMAML and SOMAML are SCL algorithms.*

Proof: Both Eq. (8) and Eq. (14) follow Definition 1.

The introduction of the zeroing trick makes the relationship between MAML and SCL more straightforward. Generally speaking, by connecting MAML and SCL, we can better understand other MAML-based meta-learning studies. For example, [22] and [23] combine MAML and adversarial training by using adversarially perturbed query data in the outer loop to compute the outer loop loss, which, from our perspective, can be directly connected to using adversarially perturbed data for SCL.

In Section 3, we experimentally validate this intuition and show that zero-initialization of $\mathbf{w}^0$, reduction in the initial norm of $\mathbf{w}^0$, or the application of zeroing trick can speed up the learning profile. This is applicable to both SOMAML and FOMAML.

2.5 Generalization of our Analysis

The analysis of the outer loop update of the encoder is derived under the condition of $N_{batch} = 1$ and $N_{step} = 1$ in the inner loop. We include the full analysis in Appendix C while keeping the assumption of freezing encoder during the inner loop. This assumption can hardly be dropped because the behavior of the updated encoder is intractable. Moreover, the assumption of the frozen encoder is empirically reasonable since previous works of [11, 13, 14, 15] yield comparable performance with keeping the encoder fixed during the inner loop.

3 Experimental Results

3.1 Setup

We conduct our experiments on the miniImagenet dataset [24, 25] and the Omniglot dataset [26]. For the results on the Omniglot dataset, please refer to Appendix E. For the miniImagenet, it contains 84×84 RGB images of 100 classes from the ImageNet dataset with 600 samples per class. We split the dataset into 64, 16 and 20 classes for training, validation, and testing as proposed in [25]. We do not perform hyperparameter search and thus are not using the validation data. For all our experiments of applying MAML into few-shot classification problem, where we adopt two experimental settings: 5-way 1-shot and 5-way 5-shot, with the batch size N_{batch} being 4 and 2, respectively[4]. The few-shot classification accuracy is calculated by averaging the results over 400 tasks in the test phase. For model architecture, optimizer and other experimental details, please refer to Appendix D.1.

3.2 Cosine Similarity Analysis Verifies the Implicit Contrastiveness in MAML

In Section 2, we show that the encoder is updated so that the query features are pushed towards the support features of the same class and pulled away from those of different classes. Here we verify this supervised contrastiveness experimentally. Consider a relatively overfitting scenario where there are five classes of images and for each class there are 20 support images and 20 query images. We fix the support and query set (i.e. the data is not resampled every iteration) to verify the concept that the support features work as positive and negative samples. Channel shuffling is used to avoid the undesirable channel memorization effect [9, 27]. We train the model using FOMAML and examine how well the encoder can separate the data of different classes in the feature space by measuring the averaged cosine similarities between the features of each class. The results are averaged over 10 random seeds.

As shown in the top row of Figure 2, the model trained with MAML0 learns to separate the features of different classes. Moreover, the contrast between the diagonal and the off-diagonal entries of the heatmap increases as we remove the initialization interference (by zero-initializing $\mathbf{w}^0$, shown in the middle row) and remove the cross-task interference (by applying the zeroing trick, shown in the bottom row). The result agrees with our analysis that MAML implicitly contains the interference term which can impede the encoder from learning a good feature representation.

3.3 Zeroing Linear Layer at Testing Time Increases Testing Accuracy

Before starting our analysis on benchmark datasets, we note that the cross-task interference can also occur during meta-testing. In the meta-testing stage, the base-model is updated in the inner loop using S and then the performance is evaluated using Q, where S and Q are drawn from a held-out unseen dataset. Recall that at the end of the outer loop (in meta-training stage), the query features are added

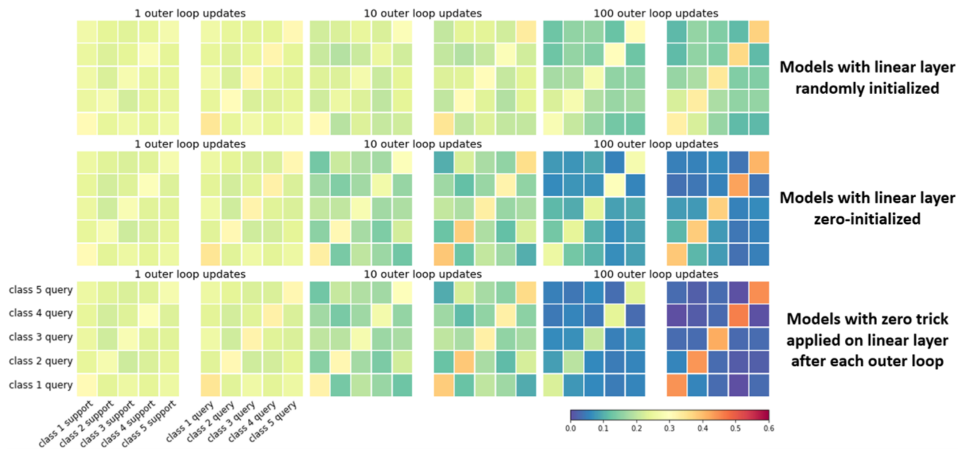


Figure 2: **The supervised contrastiveness entailed in MAML manifests when zero initialization or the zeroing trick is applied.** The value in the heatmap is calculated by averaging the pairwise cosine similarities between query features or between query features and support features. We consider the setting of having randomly initialized $\mathbf{w}^0$ (top), zero-initialized $\mathbf{w}^0$ (middle), and the zeroing trick (bottom), and experiment with various numbers of outer loop updates.

weightedly to the linear layer. In other words, at the beginning of meta-testing, the linear layer of the model is already added with the features of previous training tasks, which can drastically influence the performance on the unseen tasks. To validate this idea, we show that zeroing the linear layer at the beginning of the testing time increases the testing accuracy of the model trained with FOMAML.

As illustrated in Figure 3, compared to directly entering meta-testing (i.e. the subplot at the left), additionally zeroing the linear layer at the beginning of each meta-testing time (i.e. the subplot at the right) increases the testing accuracy of the model whose linear layer is randomly initialized or zero-initialized (denoted by the red and orange curves, respectively). And the difference in testing performance sustains across the whole training session. In the following experiments, we evaluate the testing performance only with zeroing the linear layer at the beginning of the meta-testing stage. By zeroing the last linear layer, the potential interference brought by the prior (of the linear layer) is ignored. Then, we can focus on the capacity of the encoder in learning a good feature representation.

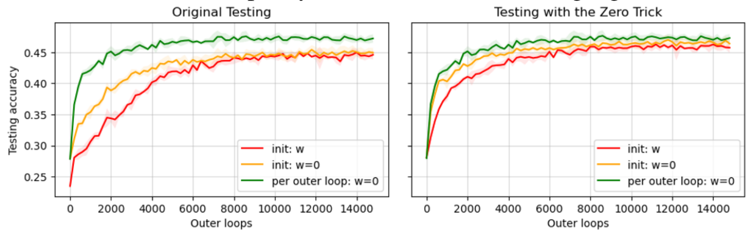


Figure 3: **Using the zeroing trick at testing stage improves the testing accuracy.** The left/right subplot shows the performance of models without/with their $\mathbf{w}^0$ zeroed at the beginning of testing time. The curves in red: $\mathbf{w}^0$ is randomly initialized. The curves in yellow: $\mathbf{w}^0$ is zeroed at initialization. The curves in green: the models trained with the zeroing trick in the training stage.

3.4 Single Inner Loop Update Suffices when Models are Trained with the Zeroing Trick

In Eq. (4) and Eq. (17), we show that the features of the support data are added to the linear layer in the inner loop. Larger number of inner loop update steps can better offset the effect of interference brought by a non-zeroed linear layer. In other words, when the models are trained with the zeroing trick, a larger number of inner loop updates can not bring any benefit. We validate this intuition in Figure 4 under a 5-way 1-shot setting. In the original FOMAML, the models trained with a single inner loop update step (denoted as red curve) converge slower than those trained with update step of 7 (denoted as purple curve). On the contrary, when the models are trained with the zeroing trick, models with various inner loop update steps converge at the same speed.

3.5 Effect of Initialization and the Zeroing Trick

In Eq. (7), we observe an interference derived from the historical task features or random initialization. We validate our formula by examining the effects of (1) reducing the norm of $\mathbf{w}^0$ at initialization;

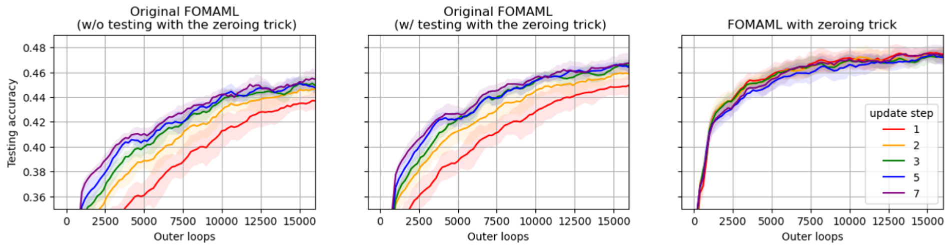


Figure 4: **With the zeroing trick, a larger number of inner loop update steps is not necessary.** In MAML, a larger number of inner loop update steps yield better testing accuracy (refer to figure at the leftmost), even with zeroing trick applied in the testing stage (refer to figure at the middle). However, models trained using the zeroing trick do not show this trend (refer to the rightmost figure).

(2) applying the zeroing trick (i.e. zeroing $\mathbf{w}^0$ after each outer loop update) which forces the model to forget the information of previous query features. From Figure 5, the testing accuracy is higher when the norm of $\mathbf{w}^0$ at initialization is lower. Compared to random initialization, reducing the norm via down-scaling $\mathbf{w}^0$ by 0.7 yields visible differences in testing performance. As for the case of zeroing$\mathbf{w}^0$ per outer loop (the purple curve), the resultant testing accuracy outperforms that of the models with a randomly initialized $\mathbf{w}^0$. The results with SOMAML are shown in Figure 6.

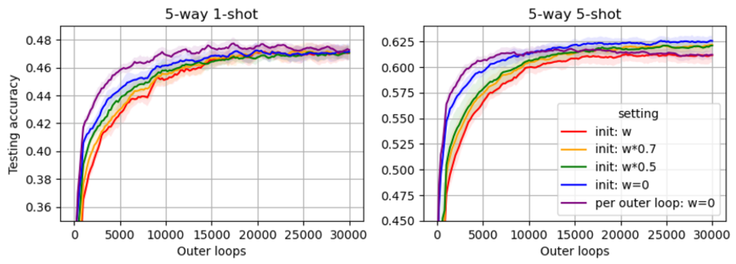


Figure 5: **Effect of initialization and the zeroing trick.** Both reducing the norm of $\mathbf{w}^0$ and zeroing $\mathbf{w}^0$ each outer loop increases the accuracy. The curves in red: models with $\mathbf{w}^0$ randomly initialized. The curves in orange/green: reducing the norm of $\mathbf{w}^0$ at initialization by a factor of 0.7/ 0.5. The curve in blue: $\mathbf{w}^0$ is zero-initialized. The curve in blue: models trained with the zeroing trick.

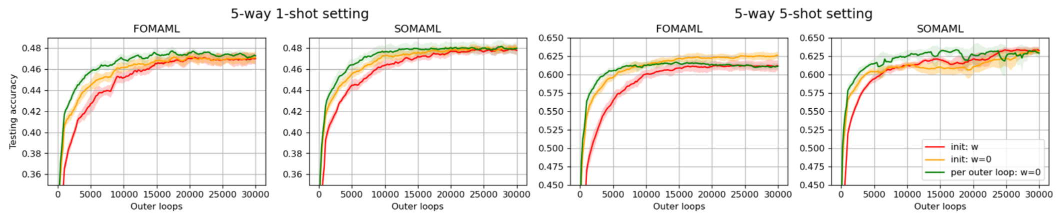


Figure 6: **Both FOMAML and SOMAML benefit from the zeroing trick**. We examine if reducing or removing the interference can increase the testing performance in models trained with FOMAML and SOMAML. The results suggest that SOMAML suffers from the interference term. Note that the second subplots from the right shows lower testing performance of models trained with the zeroing trick as compared to the zero-initialized model. This may result from the overfitting problem.

4 Conclusion

In this paper, we perform an extensive study to demystify how the seminal MAML algorithm guides the encoder to learn a general-purpose feature representation and what is the interaction between the support and query set. Our analysis shows that MAML is implicitly a supervised contrastive learner using the support features as positive and negative samples to direct the update of the encoder. Moreover, we unveil an interference term hidden in the MAML algorithm originated from the random initialization and the cross-task interaction, which can impede the representation learning. Driven by our analysis, removing the interference term by a simple zeroing trick renders the model unbiased to seen or unseen tasks. With this zeroing trick, we show constant improvements in both the training and testing profiles, with experiments conducted on the miniImagenet and Omniglot datasets.

References

[1] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial Intelligence Review*, 2002.

[2] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths, "Recasting gradient-based meta-learning as hierarchical bayes," in *International Conference on Learning Representations (ICLR)*, 2018.

[3] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *arXiv preprint arXiv:2004.05439*, 2020.

[4] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[5] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[6] A. Rajeswaran, C. Finn, S. Kakade, and S. Levine, "Meta-learning with implicit gradients," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[7] H. Liu, R. Socher, and C. Xiong, "Taming maml: Efficient unbiased meta-reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2019.

[8] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, "Online meta-learning," in *International Conference on Machine Learning (ICML)*, 2019.

[9] M. A. Jamal and G.-J. Qi, "Task agnostic meta-learning for few-shot learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[10] K. Javed and M. White, "Meta-learning representations for continual learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[11] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid learning or feature reuse? towards understanding the effectiveness of maml," in *International Conference on Learning Representations (ICLR)*, 2020.

[12] M. Goldblum, S. Reich, L. Fowl, R. Ni, V. Cherepanova, and T. Goldstein, "Unraveling meta-learning: Understanding feature representations for few-shot tasks," in *International Conference on Machine Learning (ICML)*, 2020.

[13] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[14] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," in *International Conference on Learning Representations (ICLR)*, 2019.

[15] C. Liu, C. Xu, Y. Wang, L. Zhang, and Y. Fu, "An embarrassingly simple baseline to one-shot learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[16] S. S. Du, W. Hu, S. M. Kakade, J. D. Lee, and Q. Lei, "Few-shot learning via learning the representation, provably," in *International Conference on Learning Representations (ICLR)*, 2021.

[17] N. Tripuraneni, C. Jin, and M. I. Jordan, "Provable meta-learning of linear representations," *arXiv preprint arXiv:2002.11684*, 2020.

[18] K. Chua, Q. Lei, and J. D. Lee, "How fine-tuning allows for effective meta-learning," *arXiv preprint arXiv:2105.02221*, 2021.

[19] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[20] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[21] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[22] M. Goldblum, L. Fowl, and T. Goldstein, "Adversarially robust few-shot learning: A meta-learning approach," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[23] R. Wang, K. Xu, S. Liu, P.-Y. Chen, T.-W. Weng, C. Gan, and M. Wang, "On fast adversarial robustness adaptation in model-agnostic meta-learning," in *International Conference on Learning Representations (ICLR)*, 2021.

[24] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[25] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *International Conference on Learning Representations (ICLR)*, 2017.

[26] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, 2015.

[27] J. Rajendran, A. Irpan, and E. Jang, "Meta-learning requires meta-augmentation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[28] X. Song, W. Gao, Y. Yang, K. Choromanski, A. Pacchiano, and Y. Tang, "Es-maml: Simple hessian-free meta learning," in *International Conference on Learning Representations (ICLR)*, 2020.

[29] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel, "Promp: Proximal meta-policy search," in *International Conference on Learning Representations (ICLR)*, 2019.

[30] A. Antoniou, H. Edwards, and A. Storkey, "How to train your maml," in *International Conference on Learning Representations (ICLR)*, 2019.

[31] L. Long, "Maml-pytorch implementation," https://github.com/dragen1860/MAML-Pytorch, 2018.

[32] M. Yin, G. Tucker, M. Zhou, S. Levine, and C. Finn, "Meta-learning without memorization," in *International Conference on Learning Representations (ICLR)*, 2020.

[33] T. Deleu, "Model-agnostic meta-learning," https://github.com/tristandeleu/pytorch-maml, 2020.

Appendix

A Original MAML and MAML with the Zeroing Trick

Algorithm 1 Second-order MAML

```
Require: Task distribution $D$
Require: $\eta, \rho$: inner loop and outer loop learning rates
Require: Randomly initialized base-model parameters $\theta$
1: while not done do
2:     Sample tasks $\{T_1, \dots T_{N_{batch}}\}$ from $D$
3:     for $n = 1, 2, \dots, N_{batch}$ do
4:         $\{S_n, Q_n\} \leftarrow$ sample from $T_n$
5:         $\theta'_n = \theta$
6:         for $i = 1, 2, \dots, N_{step}$ do
7:             $\theta'_n = \theta'_n - \eta \nabla_{\theta'_n} L_{\theta'_n, Sn}$
8:         end for
9:     end for
10:    Update $\theta \leftarrow \theta - \rho \sum_{n=1}^{N_{batch}} \nabla_{\theta} L_{\theta'_n, Qn}$
11: end while
```

Algorithm 2 First-order MAML

```
Require: Task distribution $D$
Require: $\eta, \rho$: inner loop and outer loop learning rates
Require: Randomly initialized base-model parameters $\theta$
1: while not done do
2:     Sample tasks $\{T_1, \dots T_{N_{batch}}\}$ from $D$
3:     for $n = 1, 2, \dots, N_{batch}$ do
4:         $\{S_n, Q_n\} \leftarrow$ sample from $T_n$
5:         $\theta'_n = \theta$
6:         for $i = 1, 2, \dots, N_{step}$ do
7:             $\theta'_n = \theta'_n - \eta \nabla_{\theta'_n} L_{\theta'_n, Sn}$
8:         end for
9:     end for
10:    Update $\theta \leftarrow \theta - \rho \sum_{n=1}^{N_{batch}} \nabla_{\theta'_n} L_{\theta'_n, Qn}$
11: end while
```

Algorithm 3 Second-order MAML with the zeroing trick

```
    Require: Task distribution $D$
    Require: $\eta, \rho$: inner loop and outer loop learning rates
    Require: Randomly initialized base-model parameters $\theta$
 1: Set $\mathbf{w} \leftarrow 0$ (the zeroing trick)
 2: while not done do
 3:     Sample tasks $\{T_1, \dots T_{N_{batch}}\}$ from $D$
 4:     for $n = 1, 2, \dots, N_{batch}$ do
 5:         $\{S_n, Q_n\} \leftarrow$ sample from $T_n$
 6:         $\theta'_n = \theta$
 7:         for $i = 1, 2, \dots, N_{step}$ do
 8:             $\theta'_n = \theta'_n - \eta \nabla_{\theta'_n} L_{\theta'_n, Sn}$
 9:         end for
10:     end for
11:     Update $\theta \leftarrow \theta - \rho \sum_{n=1}^{N_{batch}} \nabla_\theta L_{\theta'_n, Qn}$
12:     Set $\mathbf{w} \leftarrow 0$ (the zeroing trick)
13: end while
```

B Supplementary Derivation

In this section, we provide the full derivations that supplement the main paper. We consider the case of $N_{batch} = 1$, $N_{step} = 1$ and the assumption of frozen encoder (no inner loop update for the encoder). We provide the outer loop update of the linear layer under SOMAML in Section B.1 and discuss the main difference in FOMAML and SOMAML in detail in Section B.2. Next, we offer the full derivation of the outer loop update of the encoder in Section B.3. Then, we reformulate the outer loop loss for the encoder in both FOMAML and SOMAML in Section B.4 and Section B.5. Finally, we show the performance of the models trained using the reformulated loss in Section B.6.

B.1 The Derivation of Outer Loop Update for the Linear Layer Using SOMAML

Here, we provide the complete derivation of the outer loop update for the linear layer. Using SOMAML with support set S and query set Q, the update of the linear layer follows

$$\begin{aligned}
\mathbf{w_k'^0} &= \mathbf{w_k}^0 - \rho \frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial \mathbf{w_k}^0} = \mathbf{w_k}^0 - \rho \sum_{m=1}^{N_{way}} \frac{\partial \mathbf{w_m}^1}{\partial \mathbf{w_k}^0} \cdot \frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial \mathbf{w_m}^1} \\
&= \mathbf{w_k}^0 - \rho \frac{\partial \mathbf{w_k}^1}{\partial \mathbf{w_k}^0} \cdot \frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial \mathbf{w_k}^1} - \rho \sum_{m \neq k}^{N_{way}} \frac{\partial \mathbf{w_m}^1}{\partial \mathbf{w_k}^0} \cdot \frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial \mathbf{w_m}^1} \\
&= \mathbf{w_k}^0 + \rho[I - \eta \mathop{\mathbf{E}}_{(s,t)\sim S} (\mathrm{s}_k - \mathrm{s}_k^2)\phi(s)\phi(s)^T] \mathop{\mathbf{E}}_{(q,u)\sim Q} (1_{k=u} - \mathrm{q}_k)\phi(q) \\
&\quad + \rho\eta \sum_{m \neq k} [\mathop{\mathbf{E}}_{(s,t)\sim S} (\mathrm{s}_m \mathrm{s}_k)\phi(s)\phi(s)^T][\mathop{\mathbf{E}}_{(q,u)\sim Q} (1_{m=u} - \mathrm{q}_m)\phi(q)] \\
&= \mathbf{w_k}^0 + \rho[I - \eta \mathop{\mathbf{E}}_{(s,t)\sim S} \mathrm{s}_k \phi(s)\phi(s)^T] \mathop{\mathbf{E}}_{(q,u)\sim Q} (1_{k=u} - \mathrm{q}_k)\phi(q) \\
&\quad + \rho\eta \sum_{m=1}^{N_{way}} [\mathop{\mathbf{E}}_{(s,t)\sim S} (\mathrm{s}_m \mathrm{s}_k)\phi(s)\phi(s)^T][\mathop{\mathbf{E}}_{(q,u)\sim Q} (1_{m=u} - \mathrm{q}_m)\phi(q)]
\end{aligned} \tag{9}$$

We can further simplify Eq. (9) to Eq. (10) with the help of the zeroing trick.

$$\mathbf{w_k'^0} = \rho[I - \eta \mathop{\mathbf{E}}_{(s,t)\sim S} \mathrm{s}_k \phi(s)\phi(s)^T] \mathop{\mathbf{E}}_{(q,u)\sim Q} (1_{k=u} - \mathrm{q}_k)\phi(q) \tag{10}$$

This is because the zeroing trick essentially turns the logits of all support samples to zero, and consequently the predicted probability (softmax) output s_m becomes $\frac{1}{N_{way}}$ for all channel m. Therefore, the third term in Eq. (9) turns out to be zero (c.f. Eq. (11)). The equality of Eq. (11) holds since the summation of the (softmax) outputs is one.

$$\begin{aligned}&\frac{\rho\eta}{N_{way}^2}\sum_{m=1}^{N_{way}}[\underset{(s,t)\sim S}{\mathbf{E}}\phi(s)\phi(s)^T][\underset{(q,u)\sim Q}{\mathbf{E}}(1_{m=u}-\mathrm{q}_m)\phi(q)] \\ &=\frac{\rho\eta}{N_{way}^2}[\underset{(s,t)\sim S}{\mathbf{E}}\phi(s)\phi(s)^T]\underset{(q,u)\sim Q}{\mathbf{E}}\phi(q)\sum_{m=1}^{N_{way}}(1_{m=u}-\mathrm{q}_m)=0\end{aligned} \tag{11}$$

B.2 Discussion of the difference between FOMAML and SOMAML

With the derivation of Eq. (10), we can turn to the difference in using SOMAML and FOMAML to update the linear layer. There are plenty of works dedicated to approximating or estimating the second-order derivatives in the MAML algorithm in a more computational-efficient or accurate manner [28, 29, 7]. However, most of them do not interpret the effect of the Hessian matrix. Here, we provide a new perspective that, for the final linear layer, the Hessian matrix is to inform the crowdedness of the feature space. For simplicity, we denote the matrix $\mathbf{E}_{(s,t)\sim S}\,\mathrm{s}_k\phi(s)\phi(s)^\top$ as H.

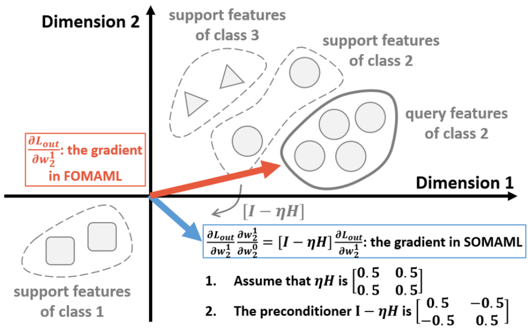

Dimension 2
support features of class 3
support features of class 2
query features of class 2
$\frac{\partial L_{out}}{\partial w_2^1}$: the gradient in FOMAML
Dimension 1
$[I-\eta H]$
$\frac{\partial L_{out}}{\partial w_2^1}\frac{\partial w_2^1}{\partial w_2^0}=[I-\eta H]\frac{\partial L_{out}}{\partial w_2^1}$: the gradient in SOMAML
support features of class 1
1. Assume that ηH is $\begin{bmatrix}0.5 & 0.5\\0.5 & 0.5\end{bmatrix}$
2. The preconditioner $\mathrm{I}-\eta H$ is $\begin{bmatrix}0.5 & -0.5\\-0.5 & 0.5\end{bmatrix}$

Figure 7: Illustration of the difference between FOMAML and SOMAML in updating the linear layer. In Eq. 5, the gradient of $\mathbf{w_k}^0$ is a weighted sum of the support features (denoted as the red arrow). Regarding SOMAML, if $\mathbf{w}^0$ is zeroed, then the gradient is preconditioned by $I-\eta H$ matrix. In this figure, the support features predominantly occupy the first and the third quadrant, thus the eigenvector with the largest eigenvalue of the Hessian matrix points at the $[1,1]$ direction. Then, $I-\eta H$ matrix biases the gradient towards regions of lower variance (denoted as the blue arrow).

To this end, we conduct an analysis here. Without loss of generality, we introduce a zero mean assumption: $\mathbf{E}_{(s,t)\sim S}\,\phi(s)$ is a zero vector. Then, H is essentially a covariance matrix weighted by the predicted probability (softmax) output. Since a covariance matrix is positive semi-definite, we can decompose H into $R\Lambda R^\top$ where columns of R are the eigenvectors of H and the diagonal entries of Λ are the eigenvalues of H. Recall that in the Pricipal Component Analysis (PCA) algorithm, larger eigenvalues of the covariance matrix indicates larger variance of the features projected onto the corresponding eigenvector. Consequently, the term $\rho R(I-\eta\Lambda)R^\top$ generally preconditions the gradient (i.e. $\mathbf{E}_{(q,u)\sim Q}(1_{k=u}-\mathrm{q}_k)\phi(q)$) to the eigen-directions whose variances are relatively small. This preconditioner helps the features to avoid the crowdedness and to explore the subspace of smaller variance. We illustrate the concept of such preconditioning in Figure 7.

B.3 The Full Derivation of the Outer Loop Update of the Encoder.

As the encoder ϕ is parameterized by φ, the outer loop gradient with respect to φ is given by $\frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial\varphi}=\mathbf{E}_{(q,u)\sim Q}\frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial\phi(q)}\frac{\partial\phi(q)}{\partial\varphi}+\mathbf{E}_{(s,t)\sim S}\frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial\phi(s)}\frac{\partial\phi(s)}{\partial\varphi}$. We take a deeper look at the backpropagated error $\frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial\phi(q)}$ of the feature of one query data $(q,u)\sim Q$, based on the following form:

$$\begin{aligned}-\frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial\phi(q)}&=\mathbf{w_u}^1-\sum_{j=1}^{N_{way}}(\mathrm{q}_j\mathbf{w_j}^1)=\sum_{j=1}^{N_{way}}(1_{j=u}-\mathrm{q}_j)\mathbf{w_j}^1\\ &=\sum_{j=1}^{N_{way}}(1_{j=u}-\mathrm{q}_j)\mathbf{w_j}^0+\eta\sum_{j=1}^{N_{way}}[1_{j=u}-\mathrm{q}_j][\underset{(s,t)\sim S}{\mathbf{E}}(1_{j=t}-\mathrm{s}_j)\phi(s)]\\ &=\sum_{j=1}^{N_{way}}(1_{j=u}-\mathrm{q}_j)\mathbf{w_j}^0+\eta\underset{(s,t)\sim S}{\mathbf{E}}[(\sum_{j=1}^{N_{way}}\mathrm{q}_j\mathrm{s}_j)-\mathrm{s}_u-\mathrm{q}_t+1_{t=u}]\phi(s)\end{aligned} \tag{12}$$

B.4 Reformulation of the Outer Loop Loss for the Encoder as Noisy SCL Loss.

We can derive the actual loss (evaluated on a single query data $(q, u) \sim Q$) that the encoder uses under FOMAML scheme as follows:

$$L_{\{\phi,\mathbf{w}^1\},q} = \sum_{j=1}^{N_{way}} \underbrace{(q_j - 1_{j=u})\mathbf{w_j}^{0\top}}_{\text{stop gradient}} \phi(q) - \eta \mathop{\mathbf{E}}_{(s,t)\sim S} \underbrace{[(\sum_{j=1}^{N_{way}} \mathrm{q}_j \mathrm{s}_j) - \mathrm{s}_u - \mathrm{q}_t + 1_{t=u}]\phi(s)^\top}_{\text{stop gradient}} \phi(q) \tag{13}$$

For SOMAML, we need to additionally plug Eq. (4) into Eq. (3).

$$L_{\{\phi,\mathbf{w}^1\},q} = \sum_{j=1}^{N_{way}} \underbrace{(q_j - 1_{j=u})\mathbf{w_j}^{0\top}}_{\text{stop gradient}} \phi(q) - \eta \mathop{\mathbf{E}}_{(s,t)\sim S} \underbrace{[(\sum_{j=1}^{N_{way}} \mathrm{q}_j \mathrm{s}_j) - \mathrm{s}_u - \mathrm{q}_t + 1_{t=u}]}_{\text{stop gradient}} \phi(s)^\top \phi(q) \tag{14}$$

B.5 Introduction of the zeroing trick makes Eq. (7) and Eq. (14) SCL losses.

Apply the zeroing trick to Eq. (7) and Eq. (14), we can derive the actual loss Eq. (15) and Eq. (16) that the encoder follows.

$$L_{\{\phi,\mathbf{w}^1\},q} = \eta \mathop{\mathbf{E}}_{(s,t)\sim S} \underbrace{(\mathrm{q}_t - 1_{t=u})\phi(s)^\top}_{\text{stop gradient}} \phi(q) \tag{15}$$

$$L_{\{\phi,\mathbf{w}^1\},q} = \eta \mathop{\mathbf{E}}_{(s,t)\sim S} \underbrace{(\mathrm{q}_t - 1_{t=u})}_{\text{stop gradient}} \phi(s)^\top \phi(q) \tag{16}$$

With these two equations, we can observe the essential difference in FOMAML and SOMAML: in FOMAML, the loss forces the query features to move closer to the support features of the same class; in SOMAML, the loss forces both the query and the support features of the same class to move closer to each other. This observation can explain why models trained with SOMAML generally converge faster. To simply wrap up, using the zeroing trick and considering a single update step in the inner loop, we show in Section B.2 and Section B.5, the explicit difference of FOMAML and SOMAML from the perspective of features.

B.6 Explicitly Computing the Reformulating Loss Using Eq. (7) and Eq. (14)

Under the assumption of no inner loop update of the encoder, we show that MAML can be reformulated as a loss taking noisy SCL form. Below, we consider a setting of 5-way 1-shot miniImagenet few-shot classification task under the condition of no inner loop update of the encoder. We empirically show that explicitly computing the reformulated losses of Eq. (7), Eq. (15) and Eq. (16) yield almost the same curves as MAML (with the assumption of no inner loop update of the encoder). Please note that the reformulated losses are used to update the encoders, for the linear layer $\mathbf{w}^0$, we explicitly update it using Eq. (5). Note that although the performance models training using FOMAML, FOMAML with the zeroing trick, and SOMAML converge to similar testing accuracy, the overall testing performance during the training process is distinct. The results are averaged over three random seeds.

B.7 The Effect of Interference Term and Noisy Contrastive Term

Reformulating the loss of MAML into a noisy SCL form also enables us to further investigate the effects brought by the interference term and the noisy contrastive term. To investigate the effect of the interference term in Figure 9, we simply consider the loss in Eq. (7) with the interference term dropped (denoted as "ITF $\times$"). As for the noisy contrastive term, the noise comes from the fact that "when the query and support data are in different classes, the sign of the contrastive coefficient can sometimes be negative", discussed in Section 2.4. To mitigate this noise, we consider the loss in Eq. (7) with dropping $-(\sum_{j=1}^{N_{way}} \mathrm{q}_j \mathrm{s}_j) + \mathrm{s}_u$ from the contrastive coefficient, and denote it as

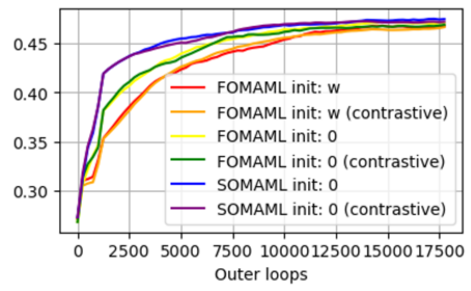


Figure 8: **Updating the encoder using the reformulated outer loop loss**. We experimentally validate that the testing accuracy of models trained using MAML (with no inner loop update of encoder) consists with that using their supervised contrastive counterpart, i.e., Eq. (7), Eq. (15) and Eq. (16).

"nCTT $\times$". On the other hand, we also implement a loss with "ITF $\times$, nCTT $\times$", which is actually Eq. (8). We consider a setting of FOMAML with 5-way 1-shot few-shot learning problem on the miniImagenet. The results are averaged over three random seeds.

In Figure 9, we show the testing profiles of the original reformulated loss (i.e., the curve in red, labeled as "ITF ✓, nCTT ✓"), dropping the interference term (i.e., the curve in orange, labeled as "ITF $\times$, nCTT ✓"), dropping the noisy part of the contrastive term (i.e., the curve in green, labeled as "ITF ✓, nCTT $\times$") or dropping both (i.e., the curve in blue, labeled as "ITF $\times$, nCTT $\times$"). We can see that either dropping the interference term or dropping dropping the noisy part of contrastive coefficients yield profound benefit.

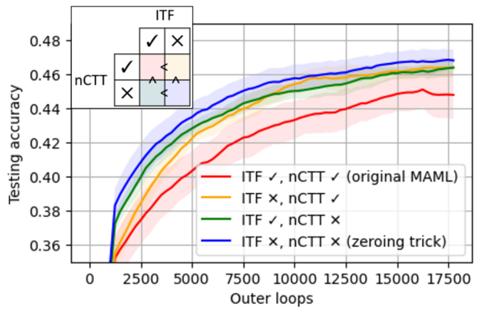


Figure 9: **The effect of the interference term and the noisy contrastive term.** We perform an ablation study of the reformulated loss in Eq. (7) by dropping the interference term (denoted as "ITF") or dropping the noisy part in the noisy contrastive term (denoted as "nCTT"). To better visualize the benefit, we compare the testing accuracy in the inlet.

C A Generalization of our Analysis

In this section, we derive a more general case of the encoder update in the outer loop. We consider drawing N_{batch} tasks from the task distribution D and having N_{step} update steps in the inner loop while keeping the assumption of frozen encoder in the inner loop.

To derive a more general case, we use $\mathbf{w_k}^{i,n}$ to denote the k^{th} column of $\mathbf{w}^{i,n}$, where $\mathbf{w}^{i,n}$ is updated from $\mathbf{w^0}$ using support data S_n for i inner-loop steps. For simplicity, the k^{th} channel softmax predictive output $\frac{\exp(\phi(s)^\top \mathbf{w_k}^{i,n})}{\sum_{j=1}^{N_{way}} \exp(\phi(s)^\top \mathbf{w_j}^{i,n})}$ of sample s (using $\mathbf{w}^{i-1,n}$) is denoted as $\mathrm{s}_k^{i,n}$.

Inner Loop Update for the Linear Layer We yield the inner loop update for the final linear layer in Eq. (17) and Eq. (18).

$$\mathbf{w_k}^{i,n} = \mathbf{w_k^{i-1,n}} - \eta \frac{\partial L_{\{\phi, \mathbf{w^{i-1,n}}\}, S_n}}{\partial \mathbf{w_k}^{i-1,n}} = \mathbf{w_k}^{i-1,n} + \eta \mathop{\mathbf{E}}_{(s,t)\sim S_n} (1_{k=t} - \mathrm{s}_k^{i-1,n})\phi(s) \tag{17}$$

$$\mathbf{w_k}^{N_{step},n} = \mathbf{w_k}^0 - \eta \sum_{i=1}^{N_{step}} \mathop{\mathbf{E}}_{(s,t)\sim S_n} (1_{k=t} - \mathrm{s}_k^{i-1,n})\phi(s) \tag{18}$$

Outer Loop Update for the Linear Layer We derive the outer loop update for the linear layer in SOMAML, with denoting $I = \{1, 2, ..., N_{way}\}$:

$$\begin{aligned} \mathbf{w'_k}^0 &= \mathbf{w_k}^0 - \rho \sum_{n=1}^{N_{batch}} \frac{\partial L_{\{\phi, \mathbf{w_k}^{N_{step},n}\}, Q_n}}{\partial \mathbf{w_k}^0} \\ &= \mathbf{w_k}^0 - \rho \sum_{n=1}^{N_{batch}} \sum_{p_0=k, p_1\in I, ..., p_{N_{way}}\in I} [(\prod_{i=0}^{N_{step}-1} \frac{\partial \mathbf{w_{p_{i+1}}}^{i+1,n}}{\partial \mathbf{w_{p_i}}^{i,n}}) \frac{\partial L_{\{\phi, \mathbf{w}^{N_{step},n}\}, Q_n}}{\partial \mathbf{w_{p_{N_{step}}}}^{N_{step},n}}] \end{aligned} \tag{19}$$

When it comes to FOMAML, we have

$$\mathbf{w'^0_k} = \mathbf{w_k}^0 - \rho \sum_{n=1}^{N_{batch}} \frac{\partial L_{\{\phi, \mathbf{w_k}^{N_{step},n}\}, Q_n}}{\partial \mathbf{w_k}^{N_{step},n}} = \mathbf{w^0_k} + \rho \sum_{n=1}^{N_{batch}} \mathop{\mathbf{E}}_{(q,u)\sim Q_n} (1_{k=u} - \mathrm{q}_k^{N_{step},n})\phi(q) \tag{20}$$

Outer Loop Update for the Encoder We derive the outer loop update of the encoder under FOMAML as below. We consider the back-propagated error of the feature of one query data $(q, u) \sim Q_n$.

Note that the third equality below holds by leveraging Eq. (17).

$$
\begin{aligned}
-\frac{\partial L_{\{\phi,\mathbf{w}^{N_{step},n}\},Q_n}}{\partial \phi(q)} &= \mathbf{w_u}^{N_{step},n} - \sum_{i=1}^{N_{way}} (\mathrm{q}_i^{N_{step},n} \mathbf{w_i}^{N_{step},n}) \\
&= \sum_{i=1}^{N_{way}} (1_{i=u} - \mathrm{q}_i^{N_{step},n}) \mathbf{w_i}^{N_{step},n} \\
&= \sum_{i=1}^{N_{way}} (1_{i=u} - \mathrm{q}_i^{N_{step},n}) [\mathbf{w_i^0} + \eta \sum_{p=1}^{N_{step}} \underset{(s,t)\sim S_n}{\mathbf{E}} (1_{i=t} - \mathrm{s}_i^{p-1,n}) \phi(s)] \\
&= \sum_{i=1}^{N_{way}} (1_{i=u} - \mathrm{q}_i^{N_{step},n}) \mathbf{w_i^0} \\
&\quad + \eta \sum_{i=1}^{N_{way}} (1_{i=u} - \mathrm{q}_i^{N_{step},n}) \sum_{p=1}^{N_{step}} \underset{(s,t)\sim S_n}{\mathbf{E}} (1_{i=u} - \mathrm{s}_i^{p-1,n}) \phi(s) \\
&= \sum_{i=1}^{N_{way}} (1_{i=u} - \mathrm{q}_i^{N_{step},n}) \mathbf{w_i^0} \\
&\quad + \eta \underset{(s,t)\sim S_n}{\mathbf{E}} \sum_{p=1}^{N_{step}} [(\sum_{j=1}^{N_{way}} \mathrm{q}_j^{N_{step},n} \mathrm{s}_j^{p-1,n}) - \mathrm{s}_u^{p-1,n} - \mathrm{q}_t^{N_{step},n} + 1_{t=u}] \phi(s)
\end{aligned}
\tag{21}
$$

Reformulating the Outer Loop Loss for the Encoder as Noisy SCL Loss. From Eq. (21), we can derive the generalized loss (of one query sample $(q, u) \sim Q_n$) that the encoder uses under FOMAML scheme.

$$
\begin{aligned}
L_{\{\phi,\mathbf{w}^{N_{step},n}\},q} &= \sum_{i=1}^{N_{way}} \underbrace{(1_{i=u} - \mathrm{q}_i^{N_{step},n}) \mathbf{w_i^0}^\top}_{\text{stop gradient}} \phi(q) \\
&+ \eta \underset{(s,t)\sim S_n}{\mathbf{E}} \sum_{p=1}^{N_{step}} \underbrace{[(\sum_{j=1}^{N_{way}} \mathrm{q}_j^{N_{step},n} \mathrm{s}_j^{p-1,n}) - \mathrm{s}_u^{p-1,n} - \mathrm{q}_t^{N_{step},n} + 1_{t=u}] \phi(s)^\top}_{\text{stop gradient}} \phi(q)
\end{aligned}
\tag{22}
$$

D Experiments on MiniImagenet Dataset

D.1 Experimental Details in MiniImagenet Dataset

The model architecture contains four basic blocks and one fully connected linear layer, where each block comprises a convolution layer with a kernel size of 3×3 and filter size of 64, batch normalization, ReLU nonlineartity and 2×2 max-poling. The models are trained with the softmax cross entropy loss function using the Adam optimizer with an outer loop learning rate of 0.001 [30]. The inner loop step size η is set to 0.01. The models are trained for 30000 iterations [11]. The results are averaged over four random seeds, and we use the shaded region to indicate the standard deviation. Each experiment is run on either a single NVIDIA 1080-Ti or V100 GPU. The detailed implementation is based on [31] (MIT License).

D.2 The Zeroing Trick Mitigates the Channel Memorization Problem

The channel memorization problem [9, 27] is a known issue occurring in a non-mutually-exclusive task setting, e.g., the task-specific class-to-label is not randomly assigned, and thus the label can be inferred from the query data alone [32]. Consider a 5-way K-shot experiment where the number of training classes is $5 \times$ L. Now we construct tasks by assigning the label t to a class sampled from class tL to $(t+1)$L. It is conceivable that the model will learn to directly map the query data to the label without using the information of the support data and thus fails to generalize to unseen tasks. This phenomenon can be explained from the perspective that the t^{th} column of the final linear layer already accumulates the query features from tLth to $(t+1)$Lth classes. Zeroing the final linear layer implicitly forces the model to use the imprinted information from the support features for inferring the label and thus mitigates this problem. We use the miniImagenet dataset and consider the case of L $= 12$. As shown in Figure 10, the zeroing trick prevents the model from the channel memorization problem whereas zero-initialization of the linear layer only works out at the beginning. Besides, the performance of models trained with the zeroing trick under this non-mutually-exclusive task setting equals the ones under the conventional few-shot setting as shown in Figure 5. As the zeroing trick clears out the final linear layer and equalizes the value of logits, our result essentially accords with [9] that proposes a regularizer to maximize the entropy of prediction of the meta-initialized model.

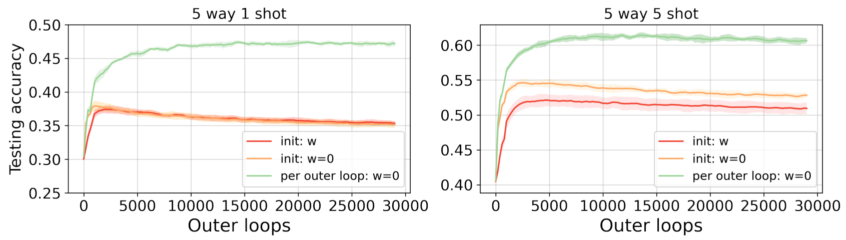


Figure 10: **The performance of the models trained on non-mutually exclusive tasks.** The models are trained under a non-mutually exclusive tasks setting where there is a one-to-one assignment between class and channel. Under this circumstance, the zeroing trick tackles the channel memorization problem and yields a performance similar to conventional few-shot settings.

E Experiments on Omniglot Dataset

Omniglot is a hand-written character dataset containing 1623 character classes, each with 20 drawn samples from different people [26]. The dataset set is splitted into training (1028 classes), validation (172 classes) and testing (423 classes) sets [24]. Since we follow [4] for setting hyperparamters, we do not use the the validation data. The character images are resized to 28 × 28. For all our experiments, we adopt two experimental settings: 5-way 1-shot and 5-way 5-shot where the batch size N_{batch} is 32 and N_{query} is 15 for both cases [4]. The inner loop learning rate η is 0.4. The models are trained for 30000 or 20000 iterations using FOMAML or SOMAML, respectively. For channel memorization problem, the models are trained for 10000 iterations using FOMAML. The few-shot classification accuracy is calculated by averaging the results over 500 tasks in the test stage. The model architecture follows the architecture used to train on miniImagenet, but we substitute the convolution with max-pooling with strided convolution operation as in [4]. The loss function, optimizer, and outer loop learning rate are the same as the ones used in the experiments on miniImagenet. Each experiment is run on either a single NVIDIA 1080-Ti or V100 GPU. The results are averaged over four random seeds, and the standard deviation is illustrated with the shaded region. The models are trained using FOMAML unless stated otherwise. The detailed implementation is based on [33] (MIT License).

We revisit the application of the zeroing trick at the testing stage on Omniglot in Figure 11 and observe the increasing testing accuracy, in which such results are compatible with the ones on miniImagenet (cf. Figure 3 in the main manuscript). In the following experiments, we evaluate the testing performance only after applying the zeroing trick.

In Figure 12, the distinction between the performance of models trained with the zeroing trick and zero-initialized models is less prominent as compared to miniImagenet (cf. Figure 5 in the main manuscript) in 5-way 1-shot setting. This result can be explained by the larger inner loop learning rate $\eta = 0.4$ used in Omniglot. We also show the testing performance of models trained using SOMAML in Figure 13, where there is little distinction in performance (in comparison to the results on miniImagenet, cf. Figure 6 in the main manuscript) between the models trained with the zeroing trick and the ones trained with random initialization.

For channel memorization task, we construct non-mutually-exclusive training tasks by assigning the label t (where $1 \leq t \leq 5$ in 5-way setting) to a class sampled from class tL to $(t+1)$L where L is 205 on Omniglot. The class-to-channel assignment is not applied to the testing tasks. The result is shown in Figure 14. For a detailed discussion, please refer to Section D.2 in the main manuscript.

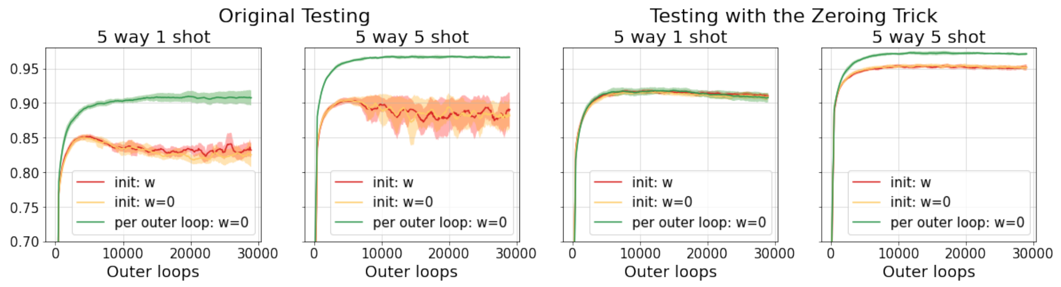


Figure 11: **Zeroing the final linear layer before testing time improves the testing accuracy on Omniglot.** The two subplots on the left: original testing setting. The two subplots at the right: the final linear layer is zeroed before testing time. The curves in red: the models whose linear layer is randomly initialized. The curves in yellow: the models whose linear layer is zeroed at initialization. The curves in green: the models whose linear layer is zeroed after each outer loop at training stage.

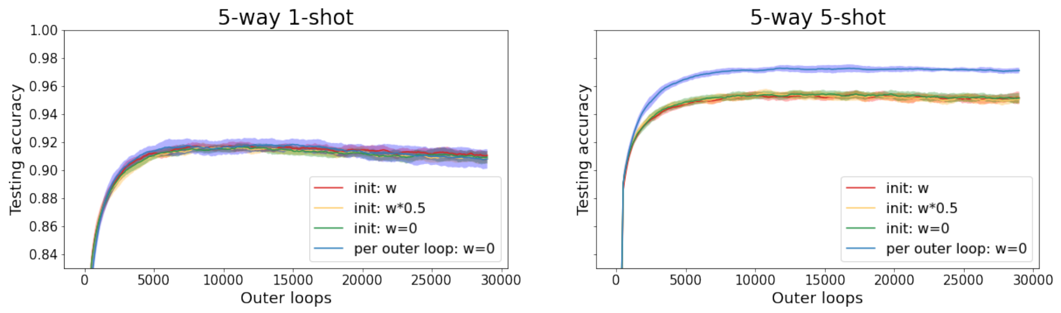


Figure 12: **Effect of initialization and the zeroing trick in testing accuracy on Omniglot.** The test performance of the models with reducing the initial norm of the weights of final linear layer is similar to that with the final linear layer being zero-initialized. The distinction in performance between models trained using the zeroing trick and zero-initialized model is more prominent in 5-way 5-shot setting.

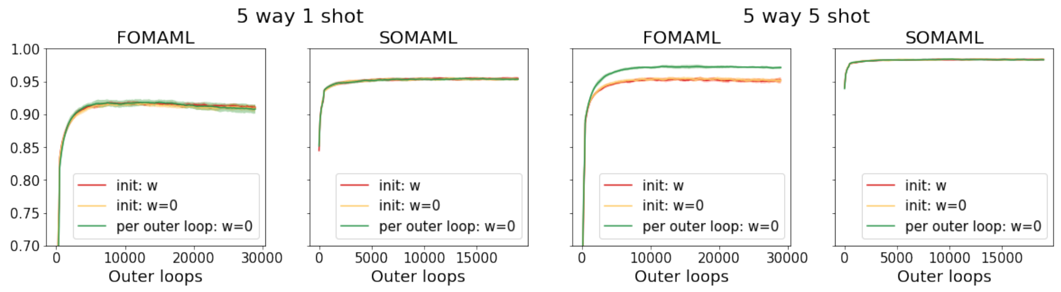


Figure 13: **The effect of the zeroing trick on models trained using FOMAML and SOMAML on Omniglot**. The results suggest that the effect of the zeroing trick is more prominent in models trained with FOMAML and less visible when models are trained with SOMAML on Omniglot.

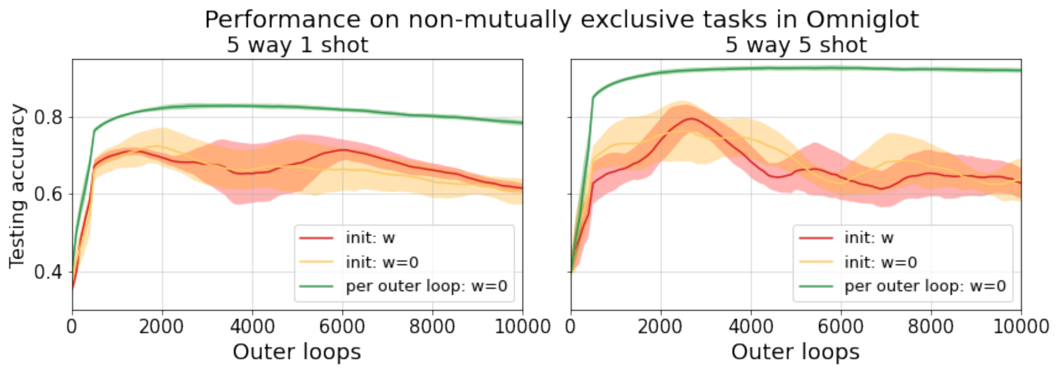


Figure 14: **The performance of the models trained on non-mutually exclusive tasks on Omniglot.** The results are compatible to those on miniImagenet (cf. Figure 14 in the main manuscript), suggesting that the zeroing trick alleviates the channel memorization problem. Besides, the testing accuracy of the models trained on non-mutually exclusive task setting with the zeroing trick (i.e. the green curves in Figure 14) is lower than that of the models trained on conventional few-shot setting with the zeroing trick (i.e. the green curves in Figure 10), whereas in miniImagenet the distinction in performance between the models trained on these two settings is relatively small.