# MAML IS A NOISY CONTRASTIVE LEARNER

**Anonymous authors**
Paper under double-blind review

## 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 respectively govern the task-specific and meta-model-centric learning, 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 to the working mechanism of MAML and 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, our analysis reveals 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 the extensive experiments are then conducted on both miniImagenet and Omniglot datasets to demonstrate the consistent improvement brought by our proposed technique thus well 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 (Vilalta & Drissi, 2002; Grant et al., 2018). Machines learn to extract task-agnostic information so that their performance on unseen tasks can be improved (Hospedales et al., 2020).

One highly influential meta-learning algorithm is Model Agnostic Meta-Learning (MAML) (Finn et al., 2017), which has inspired numerous follow-up extensions (Nichol et al., 2018; Rajeswaran et al., 2019; Liu et al., 2019; Finn et al., 2019; Jamal & Qi, 2019; Javed & White, 2019). 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-learn the base-model based on training tasks (i.e., the meta-training dataset) and evaluate the performance of the base-model on the 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 $\theta$ is updated to $\theta'$ using the support set. In the outer loop, a loss is evaluated on $\theta'$ using the query set, and its gradient is computed with respect to $\theta$ to update the base-model. Since the outer loop requires computing the gradient of gradient (as the update in the inner loop is included in the entire computation graph), it is called second-order MAML (SOMAML). To prevent computing the Hessian matrix, Finn et al. (2017) proposes first-order MAML (FOMAML) that uses the gradient computed with respect to the inner-loop-updated parameters $\theta'$ 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 (Finn et al., 2017; Raghu et al., 2020; Goldblum et al., 2020b). Raghu et al. (2020) 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 (Lee et al., 2019; Bertinetto et al., 2019; Liu et al., 2020), and has been held as assumption in theoretical works (Du et al., 2021; Tripuraneni et al., 2020; Chua et al., 2021).

While this intuition sounds satisfactory, we step further and ask the following fundamental questions: (1) In what sense does MAML *guide* any model to learn general-purpose representations? And how do the inner loop and outer loop in the training mechanism of MAML collaboratively prompt to achieve so? (2) 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].

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 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 the 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 give a formal definition of SCL, present a more general case of MAML with cross entropy loss, and show the underlying learning protocol of vanilla MAML as an interfered SCL in Section 2. We then experimentally verify the supervised contrastiveness of MAML, and propose 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 implicitly an 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 Khosla et al. (2020), 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

[1]We use the term *supervised contrastiveness* to refer to the setting of using groundtruth label information to differentiate positive samples and negative samples (Khosla et al., 2020). This setting is different from (unsupervised/self-supervised) *contrastive learning*.

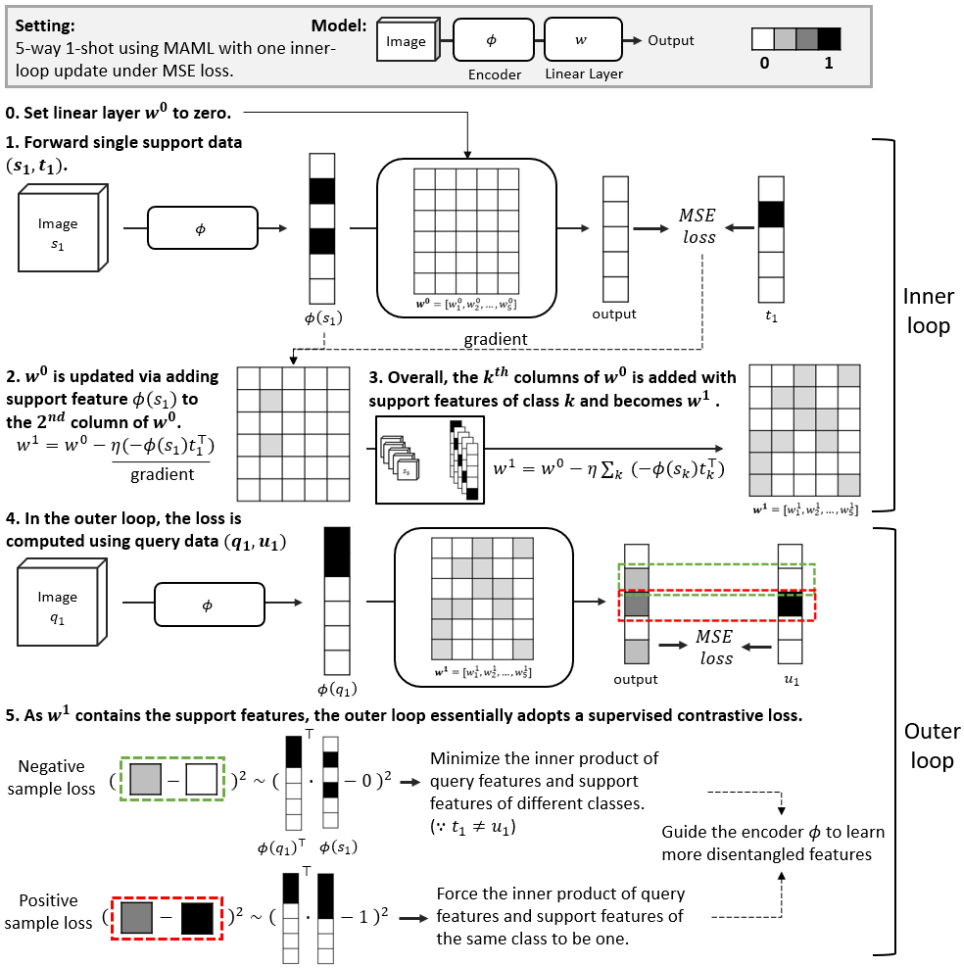


Figure 1: A step-by-step illustration showing the SCL objective underlying MAML. Assume the linear layer $\mathbf{w}^0$ to be zero, we find that, during the inner loop, the $i^{th}$ column of $\mathbf{w}^0$ is added with the support features of class $i$. In the outer loop, the output of a query sample 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.

classes." For a sample $s$, label information is leveraged to indicate positive samples (samples having the same label as sample $s$) and negative samples (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 (Schroff et al., 2015; Sohn, 2016; Khosla et al., 2020). 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 $\{T_1, \ldots, T_{N_{batch}}\}$ from a 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 $\varphi$), 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 $\theta$ consist of $\varphi$ and $\mathbf{w}$.

As shown in Algorithm A, MAML adopts a training strategy comprising the inner loop and the outer loop. At the beginning of a 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

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

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 $\phi$ 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 $\mathrm{s}_k$ ($\mathrm{q}_k$).

### 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 $\eta$ 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 $\mathrm{s}_k$.

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

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

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

Note that the computation of $\mathrm{q}_k$ 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 $\varphi$ (i.e., the parameters of the encoder) 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.

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

### 2.4 MAML IS A NOISY CONTRASTIVE LEARNER

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

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

For SOMAML, we have

$$L_{\{\phi,\mathbf{w}^1\},q} = \sum_{j=1}^{N_{way}} \underbrace{(\mathrm{q}_j - \mathbf{1}_{\mathrm{j=u}})\mathbf{w_j}^{0\top}}_{\text{stop gradient}} \phi(q) + \eta \mathop{\mathbf{E}}_{(s,t)\sim S} \underbrace{[-(\sum_{j=1}^{N_{way}} \mathrm{q}_j \mathrm{s}_j) + \mathrm{s}_u + \mathrm{q}_t - \mathbf{1}_{\mathrm{t=u}}]}_{\text{stop gradient}} \phi(s)^\top \phi(q) \tag{8}$$

With these two reformulations, we can observe the essential difference in FOMAML and SOMAML: how the support features are engaged in the computation graph. This observation can help explain why models trained with SOMAML generally converge faster. In Figure 8, we show empirically that the reformulated losses yield the same results as using the original loss in MAML. To better deliberate the effect of each term in the reformulated outer loop loss, we define the first term in Eq. (7) or Eq. (8) as *interference term*, the second term as *noisy contrastive term*, and the coefficients $-(\sum_{j=1}^{N_{way}} \mathrm{q}_j \mathrm{s}_j) + \mathrm{s}_u + \mathrm{q}_t - \mathbf{1}_{\mathrm{t=u}}$ as *contrastive coefficients*.

**Understanding the interference term.** In the case of $j = u$, the outer loop loss forces the model to minimize $(\mathrm{q}_j - \mathbf{1})\mathbf{w_j}^{0\top}\phi(q)$. 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., 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 that the encoder would be updated to maximize the inner product between $\phi(q)$ and the support features of the same class. 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.

**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. (24) (multiple inner loop update steps) follows Definition 1. For SOMAML, Eq. (8) follows Definition 1.

**Introduction of the zeroing trick makes Eq.** (7) **and Eq.** (8) **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 2. With the zeroing trick, the original outer loop loss (of FOMAML) becomes

$$L_{\{\phi,\mathbf{w}^1\},q} = \eta \mathop{\mathbf{E}}_{(s,t)\sim S} \underbrace{(\mathrm{q}_t - \mathbf{1}_{\mathrm{t=u}})\phi(s)^\top}_{\text{stop gradient}} \phi(q) \tag{9}$$

For SOMAML, the original outer loop loss becomes

$$L_{\{\phi,\mathbf{w}^1\},q} = \eta \mathop{\mathbf{E}}_{(s,t)\sim S} \underbrace{(\mathrm{q}_t - \mathbf{1}_{\mathrm{t=u}})}_{\text{stop gradient}} \phi(s)^\top \phi(q) \tag{10}$$

The zeroing trick brings two nontrivial effects: (a) eliminating the interference term in both Eq. (7) and Eq. (8); (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 $\mathrm{q}_t - \mathbf{1}_{\mathrm{t=u}}$, 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. (9) and Eq. (10) 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, Goldblum et al. (2020a) and Wang et al. (2021) 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.

Now we give responses to the questions asked in Section 1:

**In what sense does MAML guide any model to learn general-purpose representations?** Under the assumption of no inner loop update for the encoder, MAML is a noisy SCL algorithm. The effectiveness of MAML in enabling models to learn general-purpose representations can be attributed to the SCL characteristics of MAML.

**How do the inner loop and outer loop in the training mechanism of MAML collaboratively prompt to achieve so?** MAML adopts the inner loop and the outer loop to perform noisy SCL in a sequential manner. In the inner loop, the features of support data are preserved in the linear layer. In the outer loop, the predictive output of the query data thus contains the inner product between the support features and the query features.

**What is the role of the support and query data and how do they, if any, interact with each other?** We show that the original loss in MAML can be reformulated as a loss term containing the inner products of the embedding of the support and query data. In FOMAML, the support features act as the reference, while the query features are updated to move towards the support features of the same class and against those of the different classes.

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 (Raghu et al., 2020; Lee et al., 2019; Bertinetto et al., 2019; Liu et al., 2020) 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 (Vinyals et al., 2016; Ravi & Larochelle, 2017) and the Omniglot dataset (Lake et al., 2015). For the results on the Omniglot dataset, please refer to Appendix E. For the miniImagenet, it contains $84 \times 84$ 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 (Ravi & Larochelle, 2017). 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(Finn et al., 2017). 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. 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. 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.

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 linear layer (top), zero-initialized linear layer (middle), and the zeroing trick (bottom), and experiment with various numbers of outer loop updates.

### 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 support data $S$ and then the performance is evaluated using query data $Q$, 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 weightedly to the linear layer $\mathbf{w}^0$. In other words, at the beginning of meta-testing, $\mathbf{w}^0$ is already added with the query features of previous training tasks, which can drastically influence the performance on the unseen tasks. To validate this idea, we apply the zeroing trick at testing time (which we refer to zeroing $\mathbf{w}^0$ at the beginning of the testing time) and show such trick 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 linear layer, the potential interference brought by the prior (of the linear layer) is ignored. Then, we can fully focus on the capacity of the encoder in learning a good feature representation.

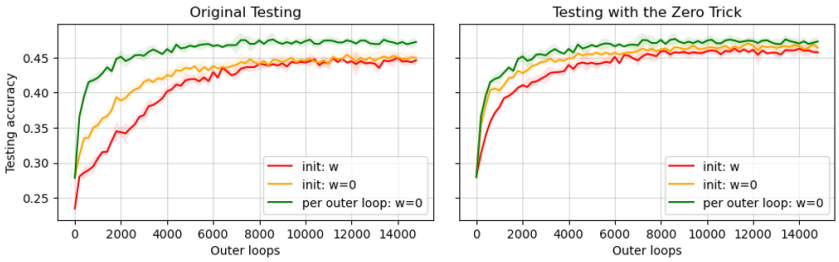


Figure 3: **Using the zeroing trick at testing stage improves the testing accuracy.** The left/right subplot shows the performance of models without/with $\mathbf{w}^0$ zeroed at the beginning of testing time. The curves in red: $\mathbf{w}^0$ is randomly initialized. The curves in yellow: $\mathbf{w}^0$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

### 3.4 SINGLE INNER LOOP UPDATE SUFFICES WHEN MODELS ARE TRAINED WITH THE ZEROING TRICK

In Eq. (4) and Eq. (19), 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; (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 linear layer. The results with SOMAML are shown in Figure 6.

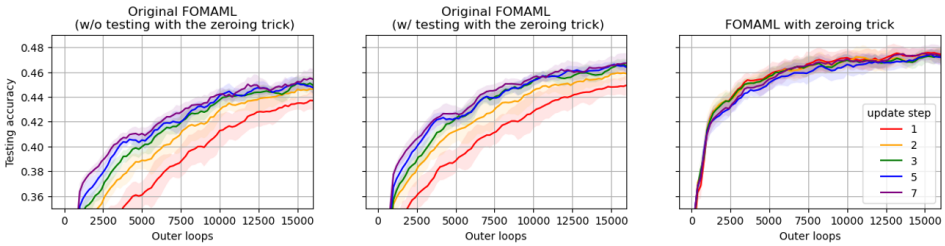


Figure 4: **With the zeroing trick, a larger number of inner loop update steps is not necessary.** In original MAML, a larger number of inner loop update steps is preferred as it generally yields better testing accuracy (refer to figure at the leftmost), even with zeroing trick applied in the testing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

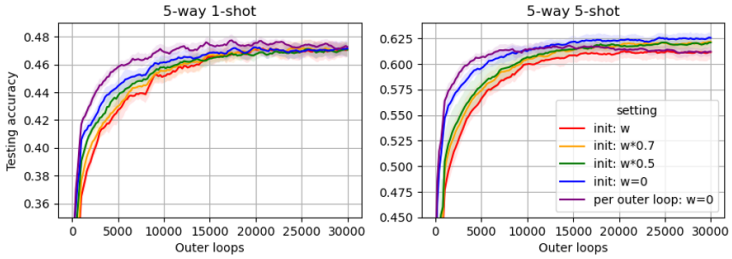


Figure 5: **Effect of initialization and the zeroing trick on testing performance.** Both reducing the norm of $\mathbf{w}^0$ and zeroing $\mathbf{w}^0$ each outer loop (i.e., the zeroing trick) increase the testing accuracy. The curves in red: models with $\mathbf{w}^0$ randomly initialized. The curves 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.

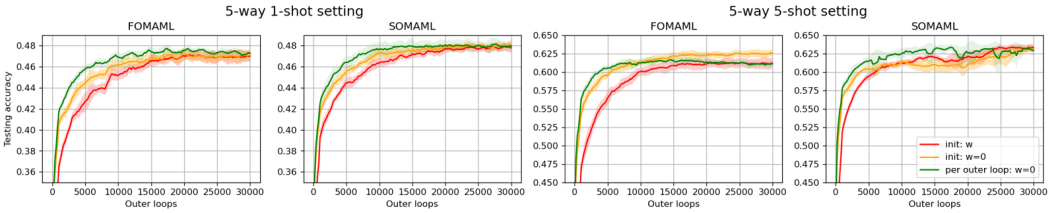


Figure 6: **Both FOMAML and SOMAML benefit from the zeroing trick**. We examine if reducing or removing the interference can increase the testing performance in models trained with FOMAML and SOMAML. The results suggest that SOMAML suffers from the interference term. Note that the second subplots from 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. The curves in red: models trained with original MAML. The curve in orange: $\mathbf{w}^0$ is zero-initialized. The curve in green: models trained with the zeroing trick.

## 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 MAML originated from the random initialization or from 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


Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. In *International Conference on Learning Representations (ICLR)*, 2019.

Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations (ICLR)*, 2019.

Kurtland Chua, Qi Lei, and Jason D Lee. How fine-tuning allows for effective meta-learning. *arXiv preprint arXiv:2105.02221*, 2021.

Tristan Deleu. Model-agnostic meta-learning. `https://github.com/tristandeleu/pytorch-maml`, 2020.

Simon S Du, Wei Hu, Sham M Kakade, Jason D Lee, and Qi Lei. Few-shot learning via learning the representation, provably. In *International Conference on Learning Representations (ICLR)*, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning (ICML)*, 2019.

Micah Goldblum, Liam Fowl, and Tom Goldstein. Adversarially robust few-shot learning: A meta-learning approach. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020a.

Micah Goldblum, Steven Reich, Liam Fowl, Renkun Ni, Valeriia Cherepanova, and Tom Goldstein. Unraveling meta-learning: Understanding feature representations for few-shot tasks. In *International Conference on Machine Learning (ICML)*, 2020b.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations (ICLR)*, 2018.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

Muhammad Abdullah Jamal and Guo-Jun Qi. Task agnostic meta-learning for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Khurram Javed and Martha White. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Chen Liu, Chengming Xu, Yikai Wang, Li Zhang, and Yanwei Fu. An embarrassingly simple baseline to one-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Hao Liu, Richard Socher, and Caiming Xiong. Taming maml: Efficient unbiased meta-reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2019.

Liangqu Long. Maml-pytorch implementation. `https://github.com/dragen1860/MAML-Pytorch`, 2018.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 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 (ICLR)*, 2020.

Janarthanan Rajendran, Alex Irpan, and Eric Jang. Meta-learning requires meta-augmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. In *International Conference on Learning Representations (ICLR)*, 2019.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. Es-maml: Simple hessian-free meta learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Nilesh Tripuraneni, Chi Jin, and Michael I Jordan. Provable meta-learning of linear representations. *arXiv preprint arXiv:2002.11684*, 2020.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 2002.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Ren Wang, Kaidi Xu, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Chuang Gan, and Meng Wang. On fast adversarial robustness adaptation in model-agnostic meta-learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. In *International Conference on Learning Representations (ICLR)*, 2020.