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 $\{T_1, \ldots T_{N_{batch}}\}$ from $D$
 3:     for $n = 1, 2, \ldots, N_{batch}$ do
 4:         $\{S_n, Q_n\} \leftarrow$ sample from $T_n$
 5:         $\theta_n = \theta$
 6:         for $i = 1, 2, \ldots, N_{step}$ do
 7:             $\theta_n \leftarrow \theta_n - \eta \nabla_{\theta_n} L_{\theta_n, S_n}$
 8:         end for
 9:     end for
10:     Update $\theta \leftarrow \theta - \rho \sum_{n=1}^{N_{batch}} \nabla_{\theta} L_{\theta_n, Q_n}$
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 $\{T_1, \ldots T_{N_{batch}}\}$ from $D$
 3:     for $n = 1, 2, \ldots, N_{batch}$ do
 4:         $\{S_n, Q_n\} \leftarrow$ sample from $T_n$
 5:         $\theta_n = \theta$
 6:         for $i = 1, 2, \ldots, N_{step}$ do
 7:             $\theta_n \leftarrow \theta_n - \eta \nabla_{\theta_n} L_{\theta_n, S_n}$
 8:         end for
 9:     end for
10:     Update $\theta \leftarrow \theta - \rho \sum_{n=1}^{N_{batch}} \nabla_{\theta_n} L_{\theta_n, Q_n}$
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 w $\leftarrow$ 0 (the zeroing trick)
 2: while not done do
 3:     Sample tasks $\{T_1, \ldots T_{N_{batch}}\}$ from $D$
 4:     for $n = 1, 2, \ldots, N_{batch}$ do
 5:         $\{S_n, Q_n\} \leftarrow$ sample from $T_n$
 6:         $\theta_n = \theta$
 7:         for $i = 1, 2, \ldots, N_{step}$ do
 8:             $\theta_n \leftarrow \theta_n - \eta \nabla_{\theta_n} L_{\theta_n, S_n}$
 9:         end for
10:     end for
11:     Update $\theta \leftarrow \theta - \rho \sum_{n=1}^{N_{batch}} \nabla_{\theta} L_{\theta_n, Q_n}$
12:     Set 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

$$\begin{aligned}
\mathbf{w'^0_k} &= \mathbf{w_k}^0 - \rho\frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial \mathbf{w_k}^0} = \mathbf{w_k}^0 - \rho\sum_{m=1}^{N_{way}}\frac{\partial \mathbf{w_m}^1}{\partial \mathbf{w_k}^0}\cdot\frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial \mathbf{w_m}^1} \\
&= \mathbf{w_k}^0 - \rho\frac{\partial \mathbf{w_k}^1}{\partial \mathbf{w_k}^0}\cdot\frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial \mathbf{w_k}^1} - \rho\sum_{m\neq k}^{N_{way}}\frac{\partial \mathbf{w_m}^1}{\partial \mathbf{w_k}^0}\cdot\frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial \mathbf{w_m}^1} \\
&= \mathbf{w_k}^0 + \rho[I - \eta \underset{(s,t)\sim S}{\mathbf{E}}(\mathrm{s}_k - \mathrm{s}_k^2)\phi(s)\phi(s)^T] \underset{(q,u)\sim Q}{\mathbf{E}}(\mathbf{1}_{\mathrm{k=u}} - \mathrm{q}_k)\phi(q) \\
&\qquad + \rho\eta\sum_{m\neq k}[\underset{(s,t)\sim S}{\mathbf{E}}(\mathrm{s}_m\mathrm{s}_k)\phi(s)\phi(s)^T][\underset{(q,u)\sim Q}{\mathbf{E}}(\mathbf{1}_{\mathrm{m=u}} - \mathrm{q}_m)\phi(q)] \\
&= \mathbf{w_k}^0 + \rho[I - \eta \underset{(s,t)\sim S}{\mathbf{E}}\mathrm{s}_k\phi(s)\phi(s)^T] \underset{(q,u)\sim Q}{\mathbf{E}}(\mathbf{1}_{\mathrm{k=u}} - \mathrm{q}_k)\phi(q) \\
&\qquad + \rho\eta\sum_{m=1}^{N_{way}}[\underset{(s,t)\sim S}{\mathbf{E}}(\mathrm{s}_m\mathrm{s}_k)\phi(s)\phi(s)^T][\underset{(q,u)\sim Q}{\mathbf{E}}(\mathbf{1}_{\mathrm{m=u}} - \mathrm{q}_m)\phi(q)]
\end{aligned} \tag{11}$$

We can further simplify Eq. (11) to Eq. (12) with the help of the zeroing trick.

$$\mathbf{w'^0_k} = \rho[I - \eta \underset{(s,t)\sim S}{\mathbf{E}}\mathrm{s}_k\phi(s)\phi(s)^T] \underset{(q,u)\sim Q}{\mathbf{E}}(\mathbf{1}_{\mathrm{k=u}} - \mathrm{q}_k)\phi(q) \tag{12}$$

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. (11) turns out to be zero (c.f. Eq. (13)). The equality of Eq. (13) holds since the summation of the (softmax) outputs is one.

$$\begin{aligned}
&\frac{\rho\eta}{N_{way}^2}\sum_{m=1}^{N_{way}}[\underset{(s,t)\sim S}{\mathbf{E}}\phi(s)\phi(s)^T][\underset{(q,u)\sim Q}{\mathbf{E}}(\mathbf{1}_{\mathrm{m=u}} - \mathrm{q}_m)\phi(q)] \\
&= \frac{\rho\eta}{N_{way}^2}[\underset{(s,t)\sim S}{\mathbf{E}}\phi(s)\phi(s)^T] \underset{(q,u)\sim Q}{\mathbf{E}}\phi(q)\sum_{m=1}^{N_{way}}(\mathbf{1}_{\mathrm{m=u}} - \mathrm{q}_m) = 0
\end{aligned} \tag{13}$$

B.2 Discussion of the difference between FOMAML and SOMAML

With the derivation of Eq. (12), 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 Song et al. (2020); Rothfuss et al. (2019); Liu et al. (2019). 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.

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

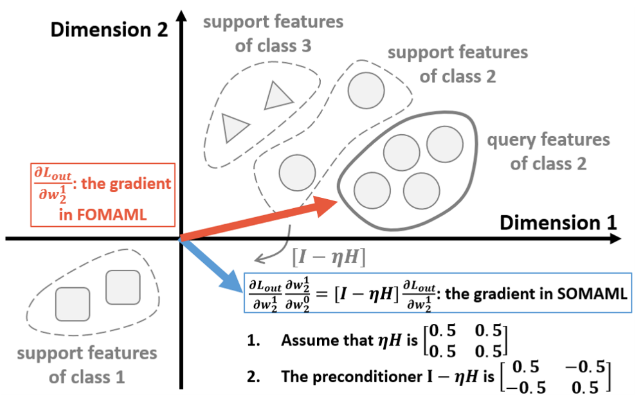
Dimension 2
support features of class 3
support features of class 2
query features of class 2
$\frac{\partial L_{out}}{\partial w_2^1}$: the gradient in FOMAML
Dimension 1
$[I - \eta H]$
$\frac{\partial L_{out}}{\partial w_2^1}\frac{\partial w_2^1}{\partial w_2^0} = [I - \eta H]\frac{\partial L_{out}}{\partial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

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).

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. $\mathbf{E}_{(q,u)\sim Q}(\mathbf{1}_{\mathrm{k=u}} - \mathrm{q}_k)\phi(q)$) 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:

$$\begin{aligned}
-\frac{\partial L_{\{\phi,\mathbf{w}^1\},Q}}{\partial \phi(q)} &= \mathbf{w_u}^1 - \sum_{j=1}^{N_{way}} (\mathrm{q}_j \mathbf{w_j}^1) = \sum_{j=1}^{N_{way}} (\mathbf{1}_{\mathrm{j=u}} - \mathrm{q}_j)\mathbf{w_j}^1 \\
&= \sum_{j=1}^{N_{way}} (\mathbf{1}_{\mathrm{j=u}} - \mathrm{q}_j)\mathbf{w_j}^0 + \eta \sum_{j=1}^{N_{way}} [\mathbf{1}_{\mathrm{j=u}} - \mathrm{q}_j][\underset{(s,t)\sim S}{\mathbf{E}} (\mathbf{1}_{\mathrm{j=t}} - \mathrm{s}_j)\phi(s)] \qquad (14) \\
&= \sum_{j=1}^{N_{way}} (\mathbf{1}_{\mathrm{j=u}} - \mathrm{q}_j)\mathbf{w_j}^0 + \eta \underset{(s,t)\sim S}{\mathbf{E}} [(\sum_{j=1}^{N_{way}} \mathrm{q}_j \mathrm{s}_j) - \mathrm{s}_u - \mathrm{q}_t + \mathbf{1}_{\mathrm{t=u}}]\phi(s)
\end{aligned}$$

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:

$$L_{\{\phi,\mathbf{w}^1\},q} = \sum_{j=1}^{N_{way}} \underbrace{(q_j - \mathbf{1}_{\mathrm{j=u}})\mathbf{w_j}^{0\top}}_{\text{stop gradient}} \phi(q) - \eta \underset{(s,t)\sim S}{\mathbf{E}} \underbrace{[(\sum_{j=1}^{N_{way}} \mathrm{q}_j \mathrm{s}_j) - \mathrm{s}_u - \mathrm{q}_t + \mathbf{1}_{\mathrm{t=u}}]\phi(s)^\top}_{\text{stop gradient}} \phi(q) \qquad (15)$$

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

$$L_{\{\phi,\mathbf{w}^1\},q} = \sum_{j=1}^{N_{way}} \underbrace{(q_j - \mathbf{1}_{\mathrm{j=u}})\mathbf{w_j}^{0\top}}_{\text{stop gradient}} \phi(q) - \eta \underset{(s,t)\sim S}{\mathbf{E}} \underbrace{[(\sum_{j=1}^{N_{way}} \mathrm{q}_j \mathrm{s}_j) - \mathrm{s}_u - \mathrm{q}_t + \mathbf{1}_{\mathrm{t=u}}]}_{\text{stop gradient}} \phi(s)^\top \phi(q) \tag{16}$$

B.5 Introduction of the zeroing trick makes Eq. (7) and Eq. (8) SCL losses.

Apply the zeroing trick to Eq. (7) and Eq. (8), we can derive the actual loss Eq. (17) and Eq. (18) that the encoder follows.

$$L_{\{\phi,\mathbf{w}^1\},q} = \eta \underset{(s,t)\sim S}{\mathbf{E}} \underbrace{(\mathrm{q}_t - \mathbf{1}_{\mathrm{t=u}})\phi(s)^\top}_{\text{stop gradient}} \phi(q) \tag{17}$$

$$L_{\{\phi,\mathbf{w}^1\},q} = \eta \underset{(s,t)\sim S}{\mathbf{E}} \underbrace{(\mathrm{q}_t - \mathbf{1}_{\mathrm{t=u}})}_{\text{stop gradient}} \phi(s)^\top \phi(q) \tag{18}$$

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. (8)

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. (17) and Eq. (18) 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.

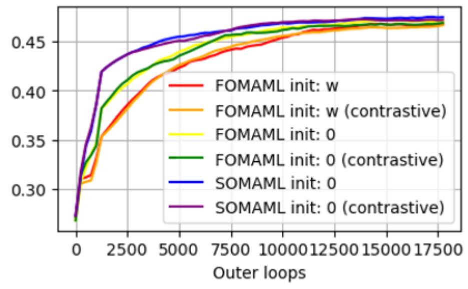

Figure 8: **Updating the encoder using the reformulated outer loop loss**. We experimentally validate that the testing accuracy of models trained using MAML (with no inner loop update of encoder) consists with that of models using their corresponding supervised contrastive losses, i.e., Eq. (7), Eq. (17) and Eq. (18).

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

Reformulating the loss of MAML into a noisy SCL form enables us to further investigate the net effect brought by the interference term and the noisy contrastive term, which we presume both effects to be negative. To investigate the effect of the interference term, 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", as is discussed in Section 2.4. To mitigate this noise, we consider the loss in Eq. (7) with the term $-(\sum_{j=1}^{N_{way}} \mathrm{q}_j \mathrm{s}_j) + \mathrm{s}_u$ dropped from the contrastive coefficient, and denote it as "nCTT $\times$". On the other hand, we also implement a loss with "ITF $\times$, nCTT $\times$", which is actually Eq. (9). We adopt the same experimental setting as Section B.6.

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.

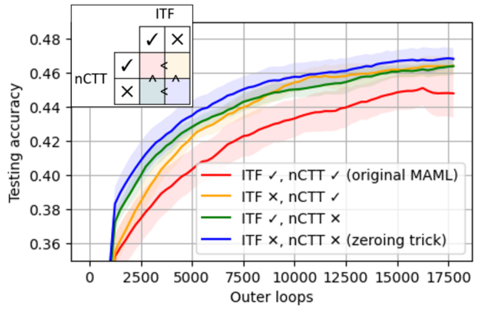

Figure 9: **The effect of the interference term and the noisy contrastive term.** We perform an ablation study of the reformulated loss in Eq. (7) by dropping the interference term (denoted as "ITF") or dropping the noisy part in the noisy contrastive term (denoted as "nCTT"). To better visualize the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

To better understand how noisy is the noisy contrastive term, i.e., how many times the the sign of the contrastive coefficient is negative when the query and support data are in different classes, we explicitly record the ratio of the contrastive term being positive or negative. We adopt the same experimental setting as Section B.6.

The result is shown in Figure 10. When the zeroing trick is applied, the ratio of contrastive term being negative (shown as the red curve on the right subplot) is 0.2, which is $\frac{1}{N_{way}}$ where $N_{way} = 5$ in our setting. On the other hand, when the zeroing trick is not applied, the ratio of contrastive term being negative (shown as the orange color on the right subplot) is larger than 0.2. This additional experiment necessitates the application of the zeroing trick.

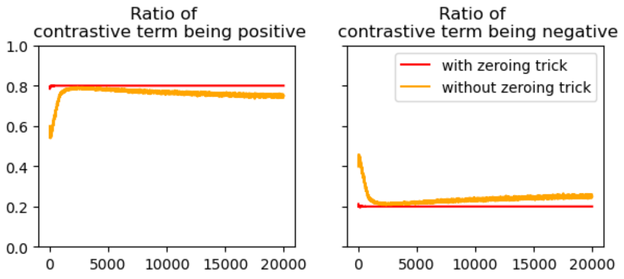

Figure 10: **The ratio of the contrastive term being positive or negative during training.** With the zeroing trick, the MAML becomes a SCL algorithm, so its ratio of negative contrastive term is 0.2. The original MAML, however, is a noisy SCL algorithm, and thus its ratio of negative contrastive term is larger than 0.2

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. (19) and Eq. (20).

$$\mathbf{w_k}^{i,n} = \mathbf{w_k}^{i-1,n} - \eta \frac{\partial L_{\{\phi,\mathbf{w}^{i-1,n}\},S_n}}{\partial \mathbf{w_k}^{i-1,n}} = \mathbf{w_k}^{i-1,n} + \eta \mathop{\mathbf{E}}_{(s,t)\sim S_n} (1_{k=t} - \mathrm{s}_k^{i-1,n})\phi(s) \tag{19}$$

$$\mathbf{w_k}^{N_{step},n} = \mathbf{w_k}^0 - \eta \sum_{i=1}^{N_{step}} \mathop{\mathbf{E}}_{(s,t)\sim S_n} (1_{k=t} - \mathrm{s}_k^{i-1,n})\phi(s) \tag{20}$$

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

$$\begin{aligned} \mathbf{w'_k}^0 &= \mathbf{w_k}^0 - \rho \sum_{n=1}^{N_{batch}} \frac{\partial L_{\{\phi,\mathbf{w_k}^{N_{step},n}\},Q_n}}{\partial \mathbf{w_k}^0} \\ &= \mathbf{w_k}^0 - \rho \sum_{n=1}^{N_{batch}} \sum_{p_0=k, p_1\in I,\dots,p_{N_{way}}\in I} [(\prod_{i=0}^{N_{step}-1} \frac{\partial \mathbf{w_{p_{i+1}}}^{i+1,n}}{\partial \mathbf{w_{p_i}}^{i,n}}) \frac{\partial L_{\{\phi,\mathbf{w}^{N_{step},n}\},Q_n}}{\partial \mathbf{w_{p_{N_{step}}}}^{N_{step},n}}] \end{aligned} \tag{21}$$

When it comes to FOMAML, we have

$$\mathbf{w'_k}^0 = \mathbf{w_k}^0 - \rho \sum_{n=1}^{N_{batch}} \frac{\partial L_{\{\phi,\mathbf{w_k}^{N_{step},n}\},Q_n}}{\partial \mathbf{w_k}^{N_{step},n}} = \mathbf{w_k^0} + \rho \sum_{n=1}^{N_{batch}} \mathop{\mathbf{E}}_{(q,u)\sim Q_n} (1_{k=u} - \mathrm{q}_k^{N_{step},n})\phi(q) \tag{22}$$

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. (19).

$$\begin{aligned} -\frac{\partial L_{\{\phi,\mathbf{w}^{N_{step},n}\},Q_n}}{\partial \phi(q)} &= \mathbf{w_u}^{N_{step},n} - \sum_{i=1}^{N_{way}} (\mathrm{q}_i^{N_{step},n} \mathbf{w_i}^{N_{step},n}) \\ &= \sum_{i=1}^{N_{way}} (1_{i=u} - \mathrm{q}_i^{N_{step},n}) \mathbf{w_i}^{N_{step},n} \\ &= \sum_{i=1}^{N_{way}} (1_{i=u} - \mathrm{q}_i^{N_{step},n})[\mathbf{w_i^0} + \eta \sum_{p=1}^{N_{step}} \mathop{\mathbf{E}}_{(s,t)\sim S_n} (1_{i=t} - \mathrm{s}_i^{p-1,n})\phi(s)] \\ &= \sum_{i=1}^{N_{way}} (1_{i=u} - \mathrm{q}_i^{N_{step},n}) \mathbf{w_i^0} \\ &\quad + \eta \sum_{i=1}^{N_{way}} (1_{i=u} - \mathrm{q}_i^{N_{step},n}) \sum_{p=1}^{N_{step}} \mathop{\mathbf{E}}_{(s,t)\sim S_n} (1_{i=u} - \mathrm{s}_i^{p-1,n})\phi(s) \\ &= \sum_{i=1}^{N_{way}} (1_{i=u} - \mathrm{q}_i^{N_{step},n}) \mathbf{w_i^0} \\ &\quad + \eta \mathop{\mathbf{E}}_{(s,t)\sim S_n} \sum_{p=1}^{N_{step}} [(\sum_{j=1}^{N_{way}} \mathrm{q}_j^{N_{step},n} \mathrm{s}_j^{p-1,n}) - \mathrm{s}_u^{p-1,n} - \mathrm{q}_t^{N_{step},n} + 1_{t=u}]\phi(s) \end{aligned} \tag{23}$$

Reformulating the Outer Loop Loss for the Encoder as Noisy SCL Loss. From Eq. (23), we can derive the generalized loss (of one query sample $(q, u) \sim Q_n$) that the encoder uses under FOMAML scheme.

$$
\begin{aligned}
L_{\{\phi,\mathbf{w}^{N_{step},n}\},q} = & \sum_{i=1}^{N_{way}} \underbrace{(1_{i=u} - \mathrm{q}_i^{N_{step},n})\mathbf{w_i^0}^\top}_{\text{stop gradient}} \phi(q) \\
& + \eta \mathop{\mathbf{E}}_{(s,t)\sim S_n} \sum_{p=1}^{N_{step}} \underbrace{[(\sum_{j=1}^{N_{way}} \mathrm{q}_j^{N_{step},n}\mathrm{s}_j^{p-1,n}) - \mathrm{s}_u^{p-1,n} - \mathrm{q}_t^{N_{step},n} + 1_{t=u}]\phi(s)^\top}_{\text{stop gradient}} \phi(q)
\end{aligned} \tag{24}
$$

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 (Antoniou et al., 2019). The inner loop step size η is set to 0.01. The models are trained for 30000 iterations (Raghu et al., 2020). 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 Long (2018) (MIT License).

D.2 The Zeroing Trick Mitigates the Channel Memorization Problem

The channel memorization problem (Jamal & Qi, 2019; Rajendran et al., 2020) 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 (Yin et al., 2020). Consider a 5-way K-shot experiment where the number of training classes is $5 \times \mathrm{L}$. Now we construct tasks by assigning the label t to a class sampled from class $t\mathrm{L}$ to $(t+1)\mathrm{L}$. 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 $t\mathrm{L}^{th}$ to $(t+1)\mathrm{L}^{th}$ 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 $\mathrm{L} = 12$. As shown in Figure 11, 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 Jamal & Qi (2019) that proposes a regularizer to maximize the entropy of prediction of the meta-initialized model.

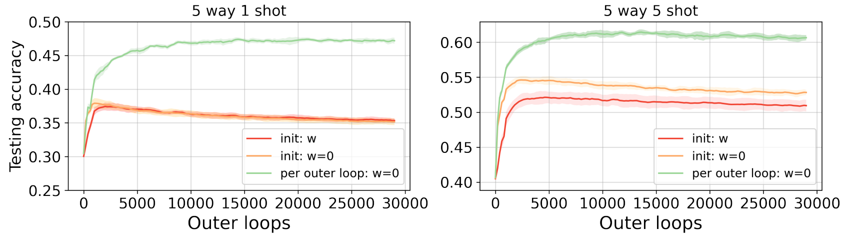

Figure 11: **The performance of the models trained on non-mutually exclusive tasks.** The models are trained under a non-mutually exclusive tasks setting where there is a one-to-one assignment between class and channel. Under this circumstance, the zeroing trick tackles the channel memorization problem 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 (Lake et al., 2015). The dataset set is splitted into training (1028 classes), validation (172 classes) and testing (423 classes) sets (Vinyals et al., 2016). Since we follow Finn et al. (2017) 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 (Finn et al., 2017). 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 Finn et al. (2017). 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 Deleu (2020) (MIT License).

We revisit the application of the zeroing trick at the testing stage on Omniglot in Figure 12 and observe the increasing testing accuracy, in which such results are compatible with the ones on mini-Imagenet (cf. Figure 3 in the main manuscript). In the following experiments, we evaluate the testing performance only after applying the zeroing trick.

In Figure 13, 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 14, 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 15. For a detailed discussion, please refer to Section D.2 in the main manuscript.

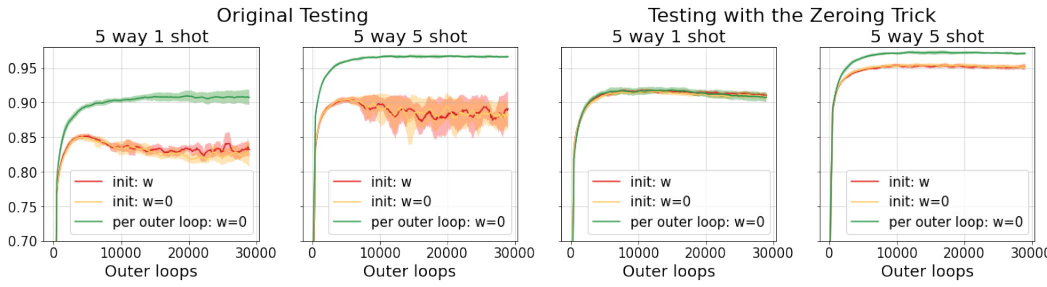

Figure 12: **Zeroing the final linear layer before testing time improves the testing accuracy on Omniglot.** The two subplots on the left: original testing setting. The two subplots at the right: the final linear layer is zeroed before testing time. The curves in red: the models whose linear layer is 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.

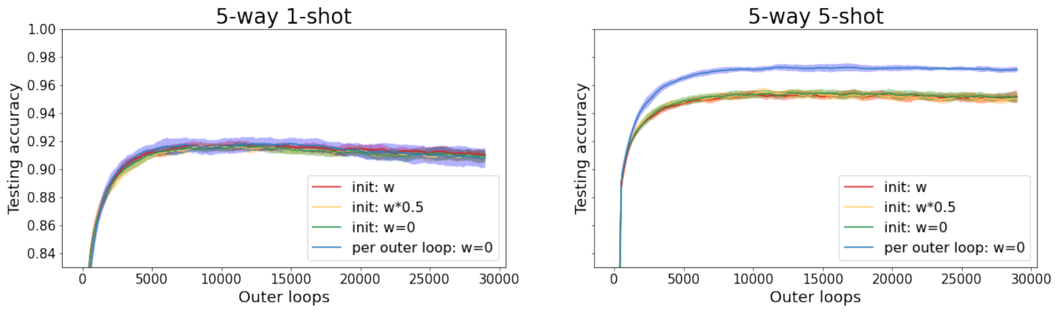

Figure 13: **Effect of initialization and the zeroing trick in testing accuracy on Omniglot.** The test performance of the models with reducing the initial norm of the weights of final linear layer is similar to that with the final linear layer being zero-initialized. The distinction in performance between models trained using the zeroing trick and zero-initialized model is more prominent in 5-way 5-shot setting.

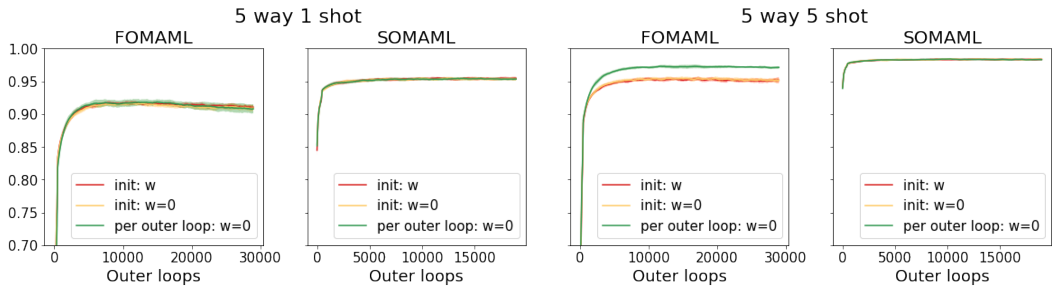

Figure 14: **The effect of the zeroing trick on models trained using FOMAML and SOMAML on Omniglot**. The results suggest that the effect of the zeroing trick is more prominent in models trained with FOMAML and less visible when models are trained with SOMAML on Omniglot.

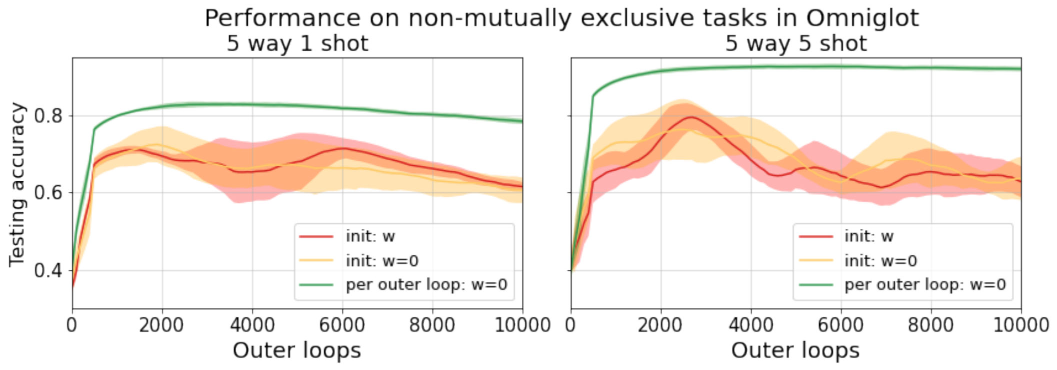

Figure 15: **The performance of the models trained on non-mutually exclusive tasks on Omniglot.** The results are compatible to those on miniImagenet (cf. Figure 15 in the main manuscript), suggesting that the zeroing trick alleviates the channel memorization problem. Besides, the testing accuracy of the models trained on non-mutually exclusive task setting with the zeroing trick (i.e. the green curves in Figure 15) is lower than that of the models trained on conventional few-shot setting with the zeroing trick (i.e. the green curves in Figure 11), whereas in miniImagenet the distinction in performance between the models trained on these two settings is relatively small.