# Meta-Learning with Task-Adaptive Loss Function for Few-Shot Learning
## – *Supplementary Document* –

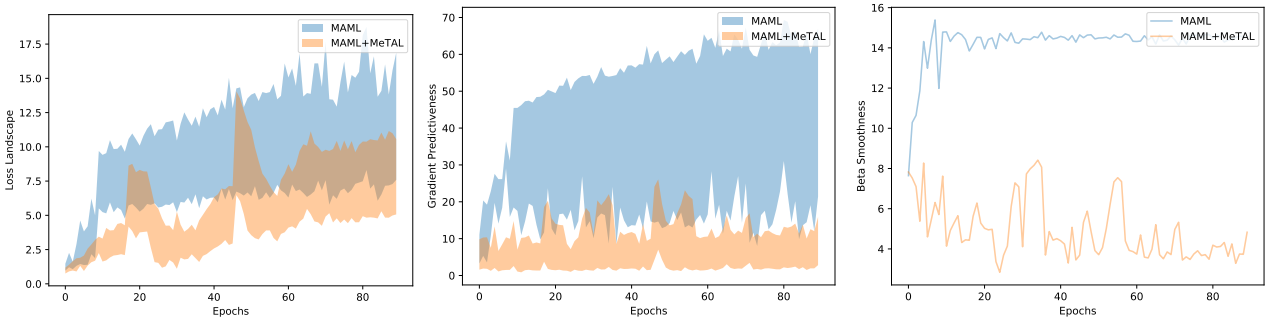
Sungyong Baik[1] Janghoon Choi[1] Heewon Kim[1] Dohee Cho[1] Jaesik Min[2] Kyoung Mu Lee[1]

[1]ASRI, Department of ECE, Seoul National University [2]Hyundai Motor Group

[1]{dsybaik, ultio791, ghimhw, jdh12245, kyoungmu}@snu.ac.kr [2]jaesik.min@hyundai.com

## A. Loss Landscape Visualization

To further stress the effectiveness of MeTAL, we analyze and compare the loss landscapes from MAML [6] and MeTAL. To this end, we employ a loss landscape visualization scheme [24] used for MAML analysis in [4]. Santurkar*et al.* [24] analyze the stability and smoothness of optimization landscape by measuring loss variations (*i.e*. loss landscape), changes in gradients (*i.e*. gradient predictiveness), and the maximum difference in gradients (*i.e*. "effective" $\beta$-smoothness). Figure A demonstrates loss variations (a), changes in gradients (b), and the maximum difference in gradients (c) are measured for meta-validation set at the first inner-loop step and averaged for each meta-training epoch on 5-way 5-shot miniImageNet classification. The thinner shades in (a), (b), and the lower value in (c) indicate the smoother loss landscape. MeTAL (orange) shows a relatively smoother loss landscape (*i.e*. the learned loss function is relatively well-behaved), compared to MAML (blue).




(a) loss landscape (b) gradient predictiveness (c) "effective" $\beta$ smoothness

Figure A: Inner-loop loss landscape visualization

## B. Few-Shot Classification

In this section, we provide more few-shot classification results on CIFAR100-derived [11] benchmarks and more thorough comparisons with the recent optimization-based meta-learning algorithms.

### B.1. CIFAR100-based Datasets

In addition to Table 1 in the main text, we further validate the effectiveness of MeTAL on other few-shot classification benchmarks: namely, CIFAR-FS [5] and FC100 [18]. Both CIFAR-FS and FC100 are derived from CIFAR100 [11] and composed of images with low-resolution of $32 \times 32$. Bertinetto *et al.* [5] uses a procedure similar to miniImageNet [21] and randomly samples and splits the original CIFAR100 dataset to obtain CIFAR-FS. On the other hand, FC100 [18] is obtained by using a dataset construction process similar to tieredImageNet [22], in which class hierarchies are used to split

| # query | # non-query | # distractor | MeTAL | ALFA+MeTAL |
|---|---|---|---|---|
| 15 | 0 | 0 | $70.52 \pm 0.29\%$ | $74.10 \pm 0.43\%$ |
| 0 | 5 | 0 | $68.90 \pm 0.39\%$ | $71.62 \pm 0.34\%$ |
| 0 | 10 | 0 | $69.76 \pm 0.48\%$ | $72.33 \pm 0.27\%$ |
| 0 | 15 | 0 | $70.40 \pm 0.34\%$ | $73.48 \pm 0.20\%$ |
| 5 | 10 | 0 | $70.02 \pm 0.45\%$ | $72.98 \pm 0.32\%$ |
| 10 | 5 | 0 | $70.06 \pm 0.41\%$ | $73.21 \pm 0.36\%$ |
| 0 | 0 | 5 | $67.69 \pm 0.39\%$ | $69.72 \pm 0.45\%$ |
| 0 | 0 | 10 | $67.02 \pm 0.48\%$ | $70.02 \pm 0.49\%$ |
| 0 | 0 | 15 | $66.97 \pm 0.34\%$ | $70.63 \pm 0.46\%$ |
| 5 | 5 | 5 | $67.58 \pm 0.47\%$ | $71.50 \pm 0.45\%$ |

Table A: Investigation on the effectiveness of MeTAL under various semi-supervised few-shot classification scenarios. In addition to support examples, different combination of unlabeled examples are used in inner-loop optimization. miniImageNet 5-*way* 5-*shot* classification accuracy is reported with a 4-CONV backbone. # implies the number of unlabeled examples per each *way*. Each distractor image is sampled from a different class.

the original dataset to simulate more challenging few-shot learning scenarios. The overall results and comparisons with recent optimization-based meta-learning algorithms on the two datasets are presented in Table B.

The table illustrates that, for the same base learner backbone (4-CONV or ResNet12), MeTAL proves a state-of-the-art performance when used together with ALFA [3], which is a recently introduced inner-loop optimizer for optimization-based meta-learning algorithms. Specifically, MeTAL is shown to outperform even recent methods that use pretrained networks or larger networks (WRN-28-10), especially on 5-shot classification. Note that SIB [9] and SIB + E3BM [16] also attempt to explicitly utilize transductive setting, similar to MeTAL. MeTAL (with the same backbone or sometimes even smaller backbone) outperforms SIB and its variants, which use pretrained networks, suggesting that the task-adaptive loss function by MeTAL effectively extracts useful information from the unlabeled examples. Furthermore, MeTAL exhibits similar tendency that can be observed in Table1 in the main text: MeTAL provides consistent performance improvement across different baselines, base learner backbone, and datasets. These experimental results reinforce our claim that learning a good loss function, which has been significantly less explored, is just as beneficial for generalization as learning a good initialization or a good optimizer.

### B.2. Detailed comparisons on ImageNet-based Datasets

We augment Table 1 in the main text with more comparisons with other recent optimization-based meta-learners, as presented in Table C. Similar to results on CIFAR-based benchmarks, MeTAL is shown to outperform most recent methods with the similar base learner backbone (4-CONV or ResNet12), including methods that utilize pretrained feature extractor, which may limit their application or effectiveness to classification problems only. Providing the competitive performance without relying on pretraining or data augmentation, our proposed method MeTAL demonstrates its effectiveness in achieving generalization.

## C. Semi-Supervised Inner-Loop Optimization

Recently, metric-based meta-learning algorithms, such as the method from [15], have attempted to make full use of the *unlabeled query* set by exploiting its feature similarity with the labeled support set. Under such scenario (*a.k.a. transductive* setting), many recent metric-based meta-learning algorithms have achieved outstanding performance. On the other hand, the transductive setting or transductive inference is rarely explored among optimization-based learners. Recent few works [2, 9] have applied transductive inference to optimization-based methods to utilize the information available from the *unlabeled query* set. However, these works only explore finetuning to the given query set, without considering a scenario, where they may exist a batch of unlabeled data prior to the inference or meta-test time [22]. We perform a small ablation study that shows MeTAL does not learn to finetune to the given query set but rather learns to extract information from the unlabeled images.

To this end, we introduce a semi-supervised few-shot classification setting, similar to [22]. Similar to how Ren *et al.* [22] has set up semi-supervised few-shot classification, we divide unlabeled data into query set, non-query set, and distractor set.

| Model | Base learner | CIFAR-FS | | FC100 | |
|---|---|---|---|---|---|
| | Backbone | 1-shot | 5-shot | 1-shot | 5-shot |
| BOIL [17] | 4-CONV | $58.03 \pm 0.43\%$ | $73.61 \pm 0.32\%$ | $38.93 \pm 0.45\%$ | $51.66 \pm 0.32\%$ |
| MAML + gcp-sampling [14] | 4-CONV | $57.62 \pm 0.97\%$ | $72.51 \pm 0.72\%$ | - | - |
| MAML++ + gcp-sampling [14] | 4-CONV | $60.14 \pm 0.97\%$ | $73.98 \pm 0.74\%$ | - | - |
| SIB * [9] | 4-CONV | $68.7 \pm 0.6\%$ | $77.1 \pm 0.4\%$ | - | - |
| META-RHKS-I [28] | 4-CONV | - | - | $38.90 \pm 1.90\%$ | $51.47 \pm 0.86\%$ |
| META-RHKS-II [28] | 4-CONV | - | - | $\mathbf{41.20 \pm 2.17}\%$ | $51.36 \pm 0.96\%$ |
| MAML + $\mathbf{E}^3$BM [16] | 4-CONV | - | - | $39.9 \pm 1.8\%$ | $52.6 \pm 0.9\%$ |
| MAML‡ | 4-CONV | $57.63 \pm 0.73\%$ | $73.95 \pm 0.84\%$ | $35.89 \pm 0.72\%$ | $49.31 \pm 0.47\%$ |
| **MeTAL (Ours)** | 4-CONV | $59.16 \pm 0.56\%$ | $74.62 \pm 0.42\%$ | $37.46 \pm 0.39\%$ | $51.34 \pm 0.25\%$ |
| ALFA + MAML [3] | 4-CONV | $59.96 \pm 0.49\%$ | $76.79 \pm 0.42\%$ | $37.99 \pm 0.48\%$ | $53.01 \pm 0.49\%$ |
| ALFA + **MeTAL (Ours)** | 4-CONV | $\mathbf{69.19 \pm 0.27}\%$ | $\mathbf{79.33 \pm 0.28}\%$ | $\mathbf{42.24 \pm 0.47}\%$ | $\mathbf{55.36 \pm 0.16}\%$ |
| MetaOpt † [13] | ResNet12 | $72.0 \pm 0.7\%$ | $84.2 \pm 0.5\%$ | $41.1 \pm 0.6\%$ | $55.5 \pm 0.6\%$ |
| MAML‡ | ResNet12 | $63.81 \pm 0.54\%$ | $77.07 \pm 0.42\%$ | $37.29 \pm 0.40\%$ | $50.70 \pm 0.35\%$ |
| **MeTAL (Ours)** | ResNet12 | $67.97 \pm 0.47\%$ | $82.17 \pm 0.38\%$ | $39.98 \pm 0.39\%$ | $53.85 \pm 0.36\%$ |
| ALFA + MAML [3] | ResNet12 | $66.79 \pm 0.47\%$ | $83.62 \pm 0.37\%$ | $41.46 \pm 0.49\%$ | $55.82 \pm 0.50\%$ |
| ALFA + **MeTAL (Ours)** | ResNet12 | $\mathbf{76.32 \pm 0.43}\%$ | $\mathbf{86.73 \pm 0.31}\%$ | $\mathbf{44.54 \pm 0.50}\%$ | $\mathbf{58.44 \pm 0.42}\%$ |
| SIB * [9] | WRN-28-10 | $80.0 \pm 0.6\%$ | $85.3 \pm 0.4\%$ | - | - |
| SIB + $\mathbf{E}^3$BM* [16] | WRN-28-10 | - | - | $46.0 \pm 0.6\%$ | $57.1 \pm 0.4\%$ |

* Pretrained
† Trained with data augmentation.
‡ Reproduced.

Table B: 5-way 1-shot and 5-way 5-shot classification test accuracy on CIFAR-based datasets: CIFAR-FS and FC100.

Query set is a set of examples whose classes are to be estimated. Non-query set is a set of examples that belong to the same task (same set of classes) as the query set. The difference from the query set is that the non-query set is available before the inference time or query set is given Distractor set is a set of examples that belong to different tasks (different classes).

Table A reports the 5-way 5-shot classification test accuracy with a 4-CONV base learner backbone on miniImageNet when various combinations of three types of unlabeled examples are used, instead of original 15 query examples per class (first row in the table), during the inner-loop optimization. The table shows that the classification accuracy increases with the number of non-query unlabeled examples, implying that MeTAL learns to generalize better by extracting relevant information from the unlabeled examples, instead of finetuning/overfitting to the given set of unlabeled examples. Surprisingly, MeTAL manages, to some extent, the performance under the presence of irrelevant or maybe even destructive distractor sets. In particular, the accuracy of MeTAL does not drop significantly with increasing number of distractor sets. This further corroborates that MeTAL, unlike other methods, does not finetune or overfit to the given unlabeled examples but rather attempts to obtain better generalization.

## D. Visual Tracking

To further demonstrate the applicability and flexibility of MeTAL, we apply our proposed method in visual tracking. Visual tracking is a challenging problem, in which the goal is to track the target whose bounding box is given only in the first frame of the video. As such problem setting is inherently a few-shot learning problem, one of the most flexible few-shot learning methodologies MAML [6] has gained attention from visual tracking community. In particular, Park *et al.* [19] has employed MAML to one of the existing tracking algorithms, such as CREST [25] to better adapt to object appearance changes throughout video frames, naming the newly obtained tracker MetaCREST. We use their publicly released code and apply MeTAL to MetaCREST to evaluate the capability and flexibility of the proposed task-adaptive loss function under more realistic and challenging scenarios, as shown in Table D and Figure B. Both quantitatively and qualitatively, MeTAL demonstrates performance improvement over MetaCREST, validating the effectiveness and flexibility of MeTAL in learning

| Model | Base learner / Backbone | miniImageNet 1-shot | miniImageNet 5-shot | tiredImageNet 1-shot | tiredImageNet 5-shot |
|---|---|---|---|---|---|
| MAML + $E^3BM$ [16] | 4-CONV | $53.2 \pm 1.8\%$ | $65.1 \pm 0.9\%$ | $52.1 \pm 1.8\%$ | $70.2 \pm 0.9\%$ |
| Meta-RHKS-I [28] | 4-CONV | $51.10 \pm 1.82\%$ | $66.19 \pm 0.80\%$ | - | - |
| Meta-RHKS-II [28] | 4-CONV | $50.03 \pm 2.09\%$ | $65.40 \pm 0.91\%$ | - | - |
| BOIL [17] | 4-CONV | $49.61 \pm 0.16\%$ | $66.45 \pm 0.37\%$ | $48.58 \pm 0.27\%$ | $69.37 \pm 0.12\%$ |
| MAML + Meta-Dropout [12] | 4-CONV | $51.93 \pm 0.67\%$ | $67.42 \pm 0.52\%$ | - | - |
| Meta-SGD + Meta-Dropout [12] | 4-CONV | $50.87 \pm 0.63\%$ | $65.55 \pm 0.57\%$ | - | - |
| ModGrad [8] | 4-CONV | $53.20 \pm 0.86\%$ | $69.17 \pm 0.69\%$ | | |
| MAML + gcp-sampling [14] | 4-CONV | $49.65 \pm 0.85\%$ | $65.37 \pm 0.70\%$ | - | - |
| MAML++ + gcp-sampling [14] | 4-CONV | $52.34 \pm 0.81\%$ | $69.21 \pm 0.68\%$ | - | - |
| MAML + L2F [4] | 4-CONV | $52.10 \pm 0.50\%$ | $69.38 \pm 0.46\%$ | $54.40 \pm 0.50\%$ | $73.34 \pm 0.44\%$ |
| SIB* [9] | 4-CONV | $\mathbf{58.0} \pm \mathbf{0.6}\%$ | $70.7 \pm 0.4\%$ | - | - |
| MAML‡ | 4-CONV | $49.64 \pm 0.31\%$ | $64.99 \pm 0.27\%$ | $50.98 \pm 0.26\%$ | $66.25 \pm 0.19\%$ |
| **MeTAL (Ours)** | 4-CONV | $52.63 \pm 0.37\%$ | $70.52 \pm 0.29\%$ | $54.34 \pm 0.31\%$ | $70.40 \pm 0.21\%$ |
| ALFA + MAML [3] | 4-CONV | $50.58 \pm 0.51\%$ | $69.12 \pm 0.47\%$ | $53.16 \pm 0.49\%$ | $70.54 \pm 0.46\%$ |
| ALFA + **MeTAL (Ours)** | 4-CONV | $\mathbf{57.75} \pm \mathbf{0.38}\%$ | $\mathbf{74.10} \pm \mathbf{0.43}\%$ | $\mathbf{60.29} \pm \mathbf{0.37}\%$ | $\mathbf{75.88} \pm \mathbf{0.29}\%$ |
| Warp-MAML [7] | 4-CONV(128)§ | $52.3 \pm 0.8\%$ | $68.4 \pm 0.6\%$ | $57.2 \pm 0.9\%$ | $74.1 \pm 0.7\%$ |
| SIB* [9] | 4-CONV(128)§ | $63.26 \pm 1.07\%$ | $75.73 \pm 0.71\%$ | - | - |
| MAML + L2F | ResNet12 | $57.48 \pm 0.49\%$ | $74.68 \pm 0.43\%$ | $63.94 \pm 0.48\%$ | $77.61 \pm 0.41\%$ |
| MetaOpt † [13] | ResNet12 | $62.64 \pm 0.61\%$ | $78.63 \pm 0.46\%$ | $65.99 \pm 0.72\%$ | $81.56 \pm 0.53\%$ |
| SIB + IFSL* † [27] | ResNet10 | $\mathbf{67.10} \pm \mathbf{0.56}\%$ | $78.88 \pm 0.35\%$ | $\mathbf{77.64} \pm \mathbf{0.58}\%$ | $85.09 \pm 0.35\%$ |
| MAML‡ | ResNet12 | $58.60 \pm 0.42\%$ | $69.54 \pm 0.38\%$ | $59.82 \pm 0.41\%$ | $73.17 \pm 0.32\%$ |
| **MeTAL (Ours)** | ResNet12 | $59.64 \pm 0.38\%$ | $76.20 \pm 0.19\%$ | $63.89 \pm 0.43\%$ | $80.14 \pm 0.40\%$ |
| ALFA + MAML [3] | ResNet12 | $59.74 \pm 0.49\%$ | $77.96 \pm 0.41\%$ | $64.62 \pm 0.49\%$ | $82.48 \pm 0.38\%$ |
| ALFA + **MeTAL (Ours)** | ResNet12 | $66.61 \pm 0.28\%$ | $\mathbf{81.43} \pm \mathbf{0.25}\%$ | $70.29 \pm 0.40\%$ | $\mathbf{86.17} \pm \mathbf{0.35}\%$ |
| LEO-trainval* [23] | WRN-28-10 | $61.76 \pm 0.08\%$ | $77.59 \pm 0.12\%$ | $66.33 \pm 0.05\%$ | $81.44 \pm 0.09\%$ |
| LEO + L2F* [4] | WRN-28-10 | $62.12 \pm 0.13\%$ | $78.13 \pm 0.15\%$ | $68.00 \pm 0.11\%$ | $83.02 \pm 0.08\%$ |
| SIB* [9] | WRN-28-10 | $70.0 \pm 0.6\%$ | $79.2 \pm 0.4\%$ | - | - |
| ModGrad [8] | WRN-28-10 | $65.72 \pm 0.21\%$ | $81.17 \pm 0.20\%$ | - | - |
| SIB + $E^3BM$* [16] | WRN-28-10 | $71.4 \pm 0.5\%$ | $81.2 \pm 0.4\%$ | $75.6 \pm 0.6\%$ | $84.3 \pm 0.4\%$ |
| SIB + IFSL* † [27] | WRN-28-10 | $\mathbf{71.31} \pm \mathbf{0.56}\%$ | $\mathbf{81.73} \pm \mathbf{0.34}\%$ | $\mathbf{81.97} \pm \mathbf{0.56}\%$ | $\mathbf{88.19} \pm \mathbf{0.34}\%$ |

* Pretrained
† Trained with data augmentation.
‡ Reproduced.
§ Larger 4-CONV architecture with 128 filters.

Table C: 5-way 1-shot and 5-way 5-shot classification test accuracy on ImageNet-based datasets: miniImageNet and tiered-ImageNet.

a loss function that provides better generalization for each task. Note that in visual tracking, it is difficult to handle query examples (a new frame) as we could in few-shot classification or simple few-shot regression. As such, we do not employ semi-supervised inner-loop optimization and thus only evaluate the effectiveness of a task-adaptive loss function.

## E. Visualization of Affine Transformation Parameters

In addition to Figure 2 in the main text, we illustrate the affine transformation parameters generated by our proposed adapter meta-network $g_\psi$ for other loss learner network parameters in Figure C. Exhibit consistent tendency with Figure 2, MeTAL demonstrates dynamic behaviour across inner-loop steps and tasks. Interestingly, MeTAL learns to dynamically change the offset ($\beta$) of the second layer bias while minimizing the scaling.

| Model | Precision | Success rate |
|---|---|---|
| MetaCREST [19] | 0.7994 | 0.6029 |
| MetaCREST + MeTAL | **0.8253** | **0.6143** |

Table D: Precision and success rate measured over 100 sequences in the OTB2015 dataset [26] by using one-pass evaluation (OPE) protocol.

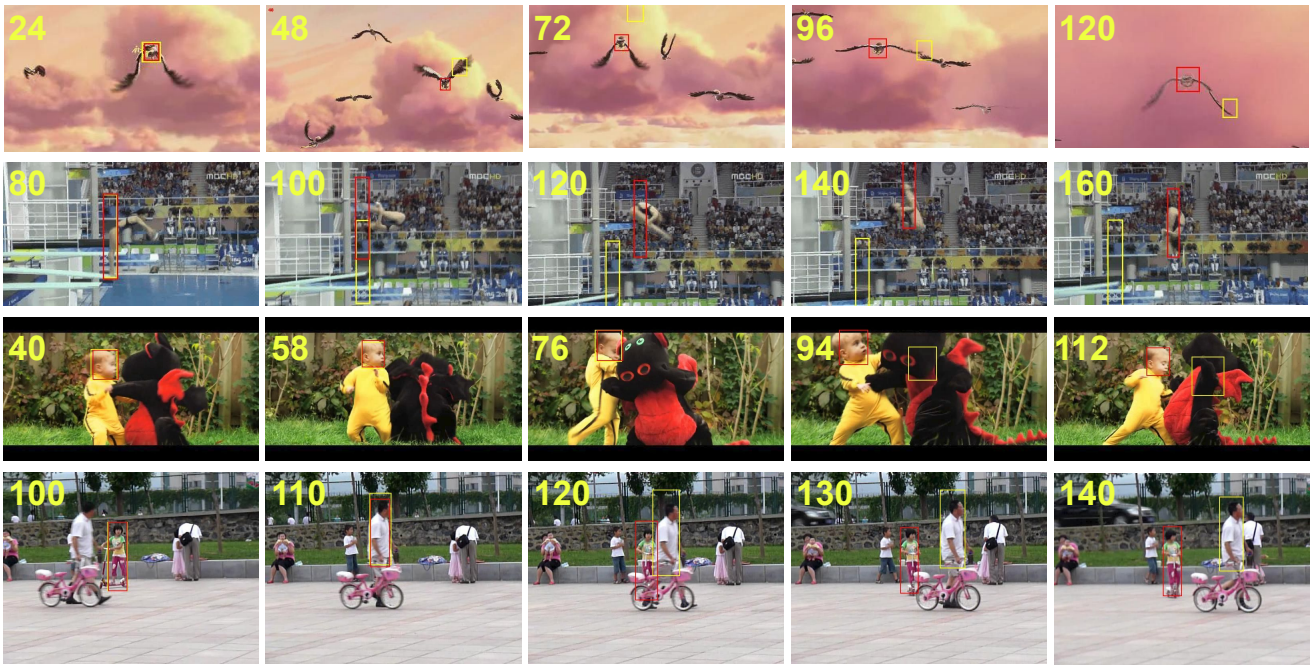



Figure B: Examples of meta-tracking results. **Yellow box** denotes MetaCREST and **red box** denotes MetaCREST+MeTAL. Results are shown for sequences in the OTB2015 dataset where each row shows selected frames from *bird1*, *diving*, *dragonBaby*, and *girl2* sequences.

## F. Implementation Details

For experiments on $N$-way $k$-shot classification in this work, we follow the typical settings that are similar to [6] when training and reporting results for our baselines (MAML [6], MAML++ [1], ALFA [3]) and our method MeTAL. During both meta-training and evaluation, inner-loop optimization (*a.k.a.* fast adaptation) is performed with a fixed number (5 in this work) of inner-loop steps with an inner-loop learning rate of $\alpha = 0.1$. Meta-training for each reproduced baseline and our method is performed with second-order gradients and a meta-learning rate of $\eta = 0.001$ for 100 epochs, each of which has 500 iterations. As with the typical settings [6, 1, 3], each task consists of 15 query examples (15 *shots*) per class (hence 75 in total for 5-way classification: $|\mathcal{D}_i^Q| = 75$) for both meta-training and evaluation. Again, as with previous works, models are trained with a meta-batch size of 2 for 5-shot and 4 for 1-shot classification. Similar to [1, 3], all results reported in this work are obtained by an ensemble of 5 top-validation-performance models from the same run, the whole process of which is repeated 3 times with different random seeds.

For a base learner network $f$, we adopt an architecture design from [1, 21, 6, 3] for 4-CONV and [18, 4, 3] for ResNet12. In particular, 4-CONV has 4 convolution layers with a fully-connected layer and softmax at the end for classification. Each convolution layer is composed of 48 convolution filters of size $3 \times 3$, a batch normalization [10] unit, a Leaky ReLU non-linear activation unit, and a max pooling layer of size $2 \times 2$. ResNet12 has 4 residual blocks with, again, a fully-connected layer and softmax at the end for classification. Each residual block is composed of three convolution operations, each with a filter size of $3 \times 3$. In between convolution operations, a batch normalization unit and a ReLU non-linear activation unit are placed. At the end of each residual block has a sequence of a batch normalization unit, a skip connection, a ReLU non-linear

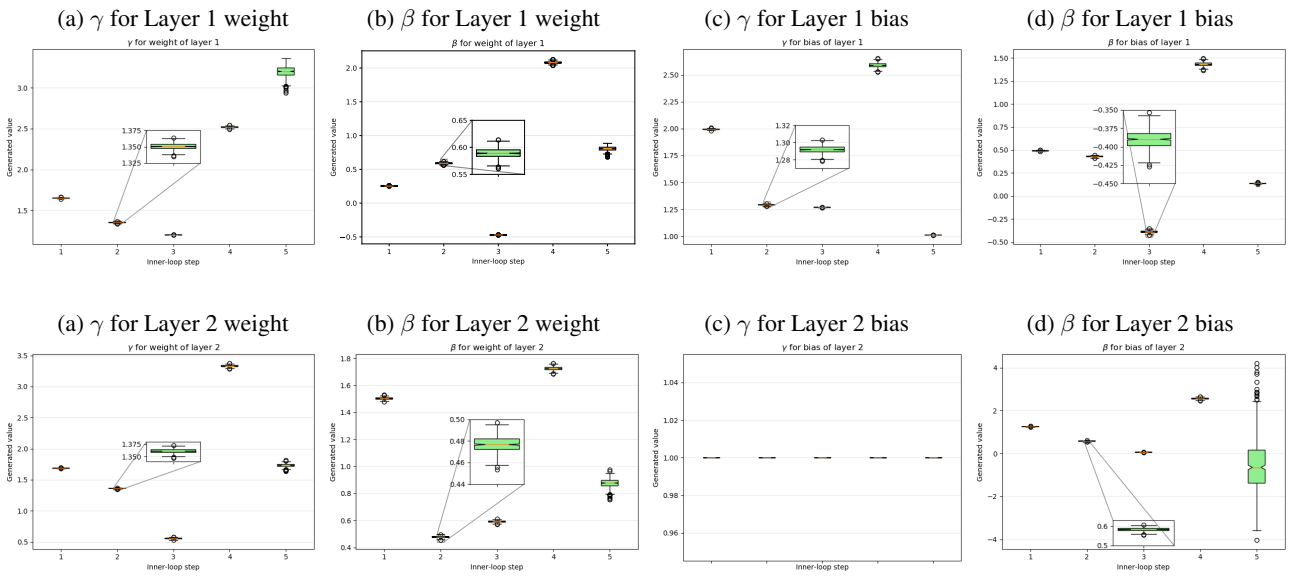



Figure C: Visualization of affine transformation parameters generated by the meta-network $g_\psi$ across different layers of loss learner network $\mathcal{L}_\phi$, different inner-loop steps, and different validation tasks. Visualization is performed on 5-way 5-shot miniImageNet validation set.

activation unit, and a max pooling unit of size $2 \times 2$. A skip connection itself has a batch normalization unit and a ReLU non-linear activation unit. The first residual block has 64 filters for each convolution operation, with each successive residual block having double the number of filters from a preceding block. Each experiment with 4-CONV base learner backbone is performed on a single NVIDIA GeForce GTX 2080Ti GPU while ResNet12 base learner backbone on a single NVIDIA Quadro RTX 8000 GPU.

For the proposed loss adapter meta-network $g_{\boldsymbol{\psi}}$, we employ a 2-layer MLP with ReLU activation unit between layers, as described in the main text. The dimensions of input and hidden units are the same $(1 + L + N)$, where $L$ is the number of layers of a base learner backbone network $f$ and $N$ is the dimension of the output of a base learner $f$. The input dimension is $1 + L + N$ as the meta-network takes in a typical classical loss value $\mathcal{L}$ (cross entropy for classification), the layer-wise mean of base learner network weights, and the output of base learner $f$. For semi-supervised settings, when feeding unlabeled query information into the meta-network, it should match the input dimension of $1 + L + N$. Because unlabeled query examples lack ground-truth, cross entropy loss cannot be obtained. To replace the cross entropy loss, we calculate entropy of the output of base learner to replace cross entropy loss in the case of unlabeled examples. The output dimension of $g_{\boldsymbol{\psi}}$ is $4L_{\boldsymbol{\phi}} = 8$, generating affine transformation parameters $\boldsymbol{\gamma}$, $\boldsymbol{\beta}$ for weight and bias of each layer of loss leaner meta-network $\mathcal{L}_{\boldsymbol{\phi}}$ that has 2 layers ($L_{\boldsymbol{\phi}} = 2$). As we desire to produce one set of affine transformation parameters for the whole task, not for each example, we take a batch-wise mean of the input such that the output has a batch dimension of 1 (one set of affine transformation parameters). A similar architecture design is employed for loss learner meta-network $\mathcal{L}_{\boldsymbol{\phi}}$: a 2-layer MLP with ReLU activation unit between layers. The network input and hidden unit dimension is $1 + L + N$ while the output dimension is 1. As the network needs to produce 1-dimensional scalar value for backpropagation operation (using autograd package in PyTorch library [20]), we took a batch-wise mean of its output to reduce the dimension to 1 (Note this is in contrast to the loss adapter meta-network $g_{\boldsymbol{\psi}}\psi$ that takes a batch-wise mean of the input). For the best performance, each pair of meta-networks $g_{\boldsymbol{\psi}}\psi$ and $\mathcal{L}_{\boldsymbol{\phi}}$ has different meta-parameters for support and query examples and for each inner-loop step. This does not increase the number of parameters significantly and boosts the performance by $1 \sim 2\%$. As for regression, similar network designs are used for the two meta-networks. For more details, please refer to the released code[1].

## References

[1] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. In *ICLR*, 2019. 5

[2] Antreas Antoniou and Amos Storkey. Learning to learn via self-critique. In *NeurIPS*, 2019. 2

[1]The code is available at https://github.com/baiksung/MeTAL

[3] Sungyong Baik, Myungsub Choi, Janghoon Choi, Heewon Kim, and Kyoung Mu Lee. Meta-learning with adaptive hyperparameters. In *NeurIPS*, 2020. 2, 3, 4, 5

[4] Sungyong Baik, Seokil Hong, and Kyoung Mu Lee. Learning to forget for meta-learning. In *CVPR*, 2020. 1, 4, 5

[5] Luca Bertinetto, Joao F. Henriques, Philip H.S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2019. 1

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 1, 3, 5

[7] Sebastian Flennerhag, Andrei A. Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. In *ICLR*, 2020. 4

[8] Erin Grant, Ghassen Jerfel, Katherine Heller, and Thomas L.Griffiths. Modulating transfer between tasks in gradient-based meta-learning. In *ECCV*, 2020. 4

[9] Shell Xu Hu, Pablo Garcia Moreno, Yang Xiao, Xi Shen, Guillaume Obozinski, Neil Lawrence, and Andreas Damianou. Empirical bayes transductive meta-learning with synthetic gradients. In *ICLR*, 2020. 2, 3, 4

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[11] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. University of Toronto, 2009. 1

[12] Hae Beom Lee, Taewook Nam, Eunho Yang, and Sung Ju Hwang. Meta dropout: Learning to perturb features for generalization. In *ICLR*, 2020. 4

[13] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019. 3, 4

[14] Chenghao Liu, Zhihao Wang, Doyen Sahoo, Yuan Fang, Kun Zhang, and Steven C.H. Hoi. Adaptive task sampling for meta-learning. In *ECCV*, 2020. 3, 4

[15] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sungju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. In *ICLR*, 2019. 2

[16] Yaoyao Liu, Bernt Schiele, and Qianru Sun. An ensemble of epoch-wise empirical bayes for few-shot learning. In *ECCV*, 2020. 2, 3, 4

[17] Jaehoon Oh, Hyungjun Yoo, ChangHwan Kim, and Se-Young Yun. Boil: Towards representation change for few-shot learning. In *ICLR*, 2021. 3, 4

[18] Boris N. Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, 2018. 1, 5

[19] Eunbyung Park and Alexander C. Berg. Meta-tracker: Fast and robust online adaptation for visual object trackers. In *ECCV*, 2018. 3, 5

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 6

[21] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017. 1, 5

[22] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018. 1, 2

[23] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *ICLR*, 2019. 4

[24] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *NIPS*, 2018. 1

[25] Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson W. H. Lau, and Ming-Hsuan Yang. Crest: Convolutional residual learning for visual tracking. In *ICCV*, 2017. 3

[26] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015. 5

[27] Zhongqi Yue, Hanwang Zhang, Qianru Sun, and Xian-Sheng Hua. Interventional few-shot learning. In *NeurIPS*, 2020. 4

[28] Yufan Zhou, Zhenyi Wang, Jiayi Xian, Changyou Chen, and Jinhui Xu. Meta-learning with neural tangent kernels. In *ICLR*, 2021. 3, 4