Learning to Forget for Meta-Learning via Task-and-Layer-Wise Attenuation

Sungyong Baik, *Student Member, IEEE,* Junghoon Oh, *Student Member, IEEE,* Seokil Hong, *Student Member, IEEE,* and Kyoung Mu Lee, *Fellow, IEEE,*

APPENDIX A
COMPARISONS WITH METHODS FROM MULTI-TASK LEARNING

As previously mentioned in the main text, few concurrent works [1], [2] from multi-task learning are similar to our work in that they focus on the gradient conflict phenomenon in the context of multi-task learning. The major difference is that our proposed method L2F addresses the gradient conflicts that occur in meta-learning framework, specifically MAML, and thus utilizes meta-learning to dynamically handle gradient conflicts of varying degrees, depending on each task. To demonstrate that L2F is more effective in handling gradient conflicts for MAML initialization, we apply GradDrop [1] and PCGrad [2] to MAML and evaluate them on miniImageNet and tieredImageNet, as shown in Table A.

APPENDIX B
EXTENDED ANALYSIS ON ATTENUATION

In this section, we provide an extended analysis on attenuation to give a further explanation why attenuation helps control the amount of prior knowledge (encoded in the initialization) that will be used for fast adaptation. Due to the reasons detailed afterwards, the analysis omits the task-adaptive part and focuses on the (task-independent) layer-wise attenuation. Nevertheless, since our interest lies in the attention itself, excluding the task-adaptive part does not constitute any limitations in analyzing the role of the underlying attention.

We claim that controlling the magnitude of the parameters in the initialization via attenuation gives similar effects to controlling the level of ℓ_2 regularization during the meta-learning of the initialization. As such, we start with modifying the MAML meta-update equation (Equation (2) in the main text), rewritten for convenience:

$$\theta \leftarrow \theta - \eta \nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}^{\mathcal{D}_{\mathcal{T}_i}^{\mathcal{Q}}}(f_{\theta_i'}), \tag{A}$$

- *S. Baik, J. Oh, S. Hong, and K. M. Lee are with Automation and Systems Research Institute (ASRI), Department of Electrical and Computer Engineering, Seoul National University, Seoul, South Korea. E-mail: {dsybaik, dh6dh, hongceo96, kyoungmu}@snu.ac.kr*

by adding ℓ_2 regularization, resulting in

$$\theta \leftarrow \theta - \eta[\nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}^{\mathcal{D}_{\mathcal{T}_i}^{\mathcal{Q}}}(f_{\theta_i'}) + \lambda\theta], \tag{B}$$

where λ is a hyperparameter that will control the extent of meta-regularization. λ is similar to attenuation parameter γ in that they both control the magnitude of the parameter values of θ.

Note that in order to achieve similar effects to layer- and task-wise attenuation, λ should also be tuned for each task and layer separately. However, the task-wise control over the meta-regularization hyperparameter λ is not possible to achieve during inference. This is because the meta-update (Equation (B)) is performed only during training and there is no access to labels for query set, which is required for the meta-update of the initialization, during inference.

Therefore, we omit the task-adaptive part and focus on layer-wise control over the meta-regularization hyperparameter, which is similar to meta-learned layer-wise attenuation that is fixed for all tasks (denoted as "layer-wise" attenuation in Table 5 in the main text). In other words, both formulations (layer-wise meta-regularization and layer-wise attenuation) are task-independent, providing a initialization shared among tasks. Neglecting the task-adaptive part does not defeat the purpose of the analysis since the goal is to give a better insight behind the attenuation.

To validate the claim that the meta-regularization hyperparameter λ and attenuation parameter γ play similar roles, we show that finding λ based on γ leads to a large performance boost, similar to meta-learned task-independent layer-wise attenuation. Note that λ and γ are inversely proportional because the smaller γ implies the stronger attenuation and thus the stronger regularization. Thus, λ is set for each layer as follows:

$$\lambda^j = \lambda_0 \times \frac{1}{\gamma^j}, \tag{C}$$

where λ^j and γ^j are the meta-regularization hyperparameter and the meta-learned attenuation parameter for the j-th layer, respectively, and λ_0 is the base hyperparameter value (0.0005 in our case).

To show that the effectiveness of the regularization hyperparameter values modeled by Equation (C) (Model 1), we compare it with the models that use the mean value

TABLE A: 5-way classification test accuracy on miniImageNet and tieredImageNet

	Backbone	miniImageNet		tieredImageNet	
		1-shot	5-shot	1-shot	5-shot
MAML+GradDrop	4 conv	$50.26 \pm 0.27\%$	$66.94 \pm 0.25\%$	$49.84 \pm 0.28\%$	$68.36 \pm 0.29\%$
MAML+PCGrad	4 conv	$50.42 \pm 0.23\%$	$65.26 \pm 0.37\%$	$51.30 \pm 0.37\%$	$67.21 \pm 0.43\%$
MAML+L2F (Ours)	4 conv	$52.10 \pm 0.50\%$	$69.38 \pm 0.46\%$	$54.40 \pm 0.50\%$	$73.34 \pm 0.44\%$

TABLE B: Ablation studies on regularization hyperparameters λ for Equation (B).

Model	λ^j	Accuracy
1	Eq. C	$68.41 \pm 0.37\%$
2	mean of Eq. C	$65.26 \pm 0.34\%$
3	min. of Eq. C	$64.27 \pm 0.31\%$
4	max. of Eq. C	$61.24 \pm 0.36\%$

of λ^j (Model 2), the minimum of λ^j (Model 3), and the maximum of λ^j (Model 4).

Table B demonstrates that meta-training with regularization hyperpamreters found by Equation (C) provides almost the similar performance to meta-learned task-independent layer-wise attenuation (from Table 5 in the main text) while other variants perform worse, sometimes even worse than MAML due to heavy regularization (Model 4). The results suggest that the layer-wise attenuation on the learned initialization and meta-training with layer-wise ℓ_2 meta-regularization play similar roles, thus supporting the claim that the attenuation helps control the amount of prior knowledge encoded in the initialization.

APPENDIX C
IMPLEMENTATION DETAILS

C.1 Classification

Unless specified otherwise, the number of inner-loop steps is 5 with a fixed learning rate of $\alpha = 0.01$. The number of query examples $\mathcal{D}^Q$ is 15 at each iteration. Meta-models are trained for 50000 iterations in miniImageNet and CIFAR-based datasets while 125000 in tieredImageNet to account for the relatively large number of examples as in [3]. The meta batch size of the tasks is set to 2 for 5-shot and 4 for 1-shot. For LEO and Meta-Dataset experiments, we use the given hyperparameters from LEO [4] and Meta-Dataset [5], respectively.

f_θ 4 conv architecture. As with MAML++ [6], we use 4 layers, each containing 48-filter (128 filters for WarpGrad variants as with [7]) 3×3 convolution filters, a batch normalization [8], a Leaky ReLU nonlinearity, and a 2×2 max pooling. The classification linear layer and softmax are placed at the end of the network.

f_θ ResNet12 architecture. Following the settings in [9], the network consists of 4 residual blocks, with each residual block consisting of three 3×3 convolution layers. The first two convolution layers are followed by a batch normalization and a Leaky ReLU nonlinearity. The last convolution layer is followed by a batch normalization and a skip connection containing a 1×1 convolution layer and a batch normalization. After a skip connection, a Leaky ReLU nonlinearity and a max 2×2 are placed at the end of each residual block. The number of convolution filters for the 4 residual blocks is set to 64, 128, 256, and 512 in increasing order of depth.

g_ϕ architecture. g_ϕ is a 3-layer MLP, with each layer of l hidden units, where l is the number of layers of f_θ. The activation functions are ReLU in between layers and a sigmoid at the end. The input is layer-wise mean of gradients.

C.2 Regression

The base network is a 2-layer MLP, with each layer of 40 hidden units with ReLU activation function in between. The inner-loop optimization is performed with a vanilla SGD with a learning rate of 0.01. The outer-loop optimziation is performed with an Adam [10] optimizer and a meta-batch size of 4. The attenuator network g_ϕ architecture is the same as in classification.

C.3 Reinforcement Learning

For all RL experiments, we follow the protocol described in [11]. The linear feature baseline introduced in [12] is used to extract features that are fed into the policy network. The inner-loop optimization is performed with a vanilla policy gradient with a learning rate of 0.01. The outer-loop optimization is performed with TRPO optimizer for 500 epochs. Then, the evaluation is performed with the model that has the highest training average reward. The other experimental settings vary depending on environments. For MuJoCo experiments, the horizon is set to be 200 while the meta-batch size is 40. The evaluation is performed after fast adaptation with 4 gradient steps and 40 samples for each task. As for 2D navigation, the horizon is 100, the meta-batch size is 20, and the evaluation is performed with 4 gradient steps and 20 samples for each task. RLBench few-shot benchmark has diverse tasks with observations of different length. To use the same policy network across tasks, only features common among tasks are fed into the policy network. The horizon is 200, the meta-batch size is 15, and the evaluation is performed with 1 gradient step and 5 samples for each task. For all RL experiments, the attenuator network g_ϕ architecture is the same as in classification.

C.4 Visual Tracking

We fix and do not change all the hyperparameter values and settings set in the official Meta-Tracker [13] public code[1]. As for the attenuator network g_ϕ, the same architecture design from classification experiments is used.

1. https://github.com/silverbottlep/meta_trackers

(a) MetaCREST

(b) MetaSDNet

Fig. A: **Qualitative results** on video sequences in OTB2015 dataset. L2F is shown to improve the tracking performance of baselines: (a) MetaCREST and (b) MetaSDNet.

APPENDIX D
QUALITATIVE RESULTS FOR VISUAL TRACKING

The qualitative results for Meta-Tracker [13] (denoted as Baseline with a red box), L2F (denoted with a green box), DropGrad [14] (denoted with a blue box), and L2F+DropGrad (denoted with a cyan box) in OTB2015 dataset [15] are displayed in Figure A. L2F is shown to successfully track the target under diverse challenging scenarios, such as occlusion, fast motion, and drastic appearance changes, when Meta-Tracker (Baseline) and DropGrad fail to keep track of the target and result in drifts.

REFERENCES

[1] Z. Chen, J. Ngiam, Y. Huang, T. Luong, H. Kretzschmar, Y. Chai, and D. Anguelov, "Just pick a sign: Optimizing deep multitask models with gradient sign dropout," in *NeurIPS*, 2020.

[2] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," in *NeurIPS*, 2020.

[3] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. J. Hwang, and Y. Yang, "Learning to propagate labels: Transductive propagation network for few-shot learning," in *ICLR*, 2019.

[4] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *ICLR*, 2019.

[5] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, and H. Larochelle, "Meta-dataset: A dataset of datasets for learning to learn from few examples," in *ICLR*, 2020.

[6] A. Antoniou, H. Edwards, and A. Storkey, "How to train your maml," in *ICLR*, 2019.

[7] S. Flennerhag, A. A. Rusu, R. Pascanu, F. Visin, H. Yin, and R. Hadsell, "Meta-learning with warped gradient descent," in *ICLR*, 2020.

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[9] B. N. Oreshkin, P. Rodriguez, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," in *NeurIPS*, 2018.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[11] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.

[12] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *ICML*, 2016.

[13] E. Park and A. C. Berg, "Meta-tracker: Fast and robust online adaptation for visual object trackers," in *ECCV*, 2018.

[14] H.-Y. Tseng, Y.-W. Chen, Y.-H. Tsai, S. Liu, Y.-Y. Lin, and M.-H. Yang, "Regularizing meta-learning via gradient dropout," *arXiv:2004.05859*, 2020.

[15] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015.