元学习——从MAML到MAML++

作者：凯鲁嘎吉 - 博客园 http://www.cnblogs.com/kailugaji/

Few-shot learning领域最近有了实质性的进展。这些进步大多来自于将few-shot learning作为元学习问题。Model-Agnostic Meta-Learning (MAML)是目前利用元学习进行few-shot learning的最佳方法之一。MAML简单，优雅，功能强大，但是它有很多问题，比如对神经网络结构非常敏感，经常导致训练时不稳定，需要费力的超参数搜索来稳定训练和实现高泛化，并且在训练和推理时间上都非常昂贵的计算。在文"How to train your MAML"中，对MAML进行了各种改进，不仅稳定了系统，而且大幅度提高了MAML的泛化性能、收敛速度和计算开销。所提方法称之为MAML++。本博文首先介绍什么是元学习，经典的Model-Agnostic Meta-Learning的定义与执行过程，进而说明MAML面临的缺点与挑战，针对这些问题，进行相应改进，从而得到MAML++。

1. Meta Learning (Learn to Learn)

➢ Meta Learning

Supervised Learning:

Inputs: $\mathbf{x}$ Outputs: $\mathbf{y}$

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

Data: $\{(\mathbf{x}, \mathbf{y})_i\}$

Meta-Supervised Learning:

Inputs: $\underbrace{\mathcal{D}_{\text{train}}}_{\{(\mathbf{x}, \mathbf{y})_{1:K}\}}$ $\mathbf{x}_{\text{test}}$ Outputs: $\mathbf{y}_{\text{test}}$

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

Data: $\{\mathcal{D}_i\}$

$$\mathcal{D}_i : \{(\mathbf{x}, \mathbf{y})_j\}$$

➢ Why Learn to Learn?

- effectively **reuse data** on other tasks
- **replace manual engineering** of architecture, hyperparameters, etc.
- learn to **quickly adapt to unexpected scenarios** (inevitable failures, long tail)
- learn how to learn **with weak supervision**

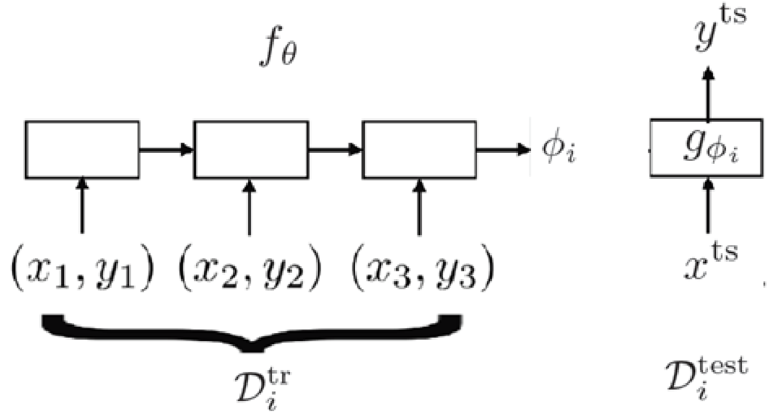
Finn, C., Abbeel, P. & Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. ICML 2017. 1

2. Black-Box Adaption vs Optimization-Based Approach

How to train your MAML

Black-Box Adaptation

$$p(\phi_i|\mathcal{D}_i^{\mathrm{tr}}, \theta)$$



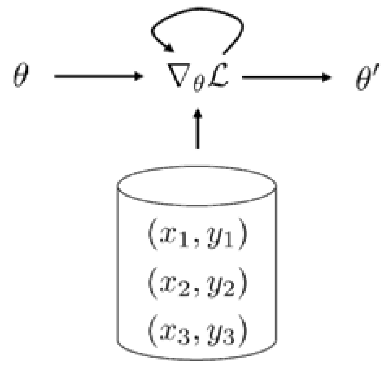
1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\mathrm{tr}}, \mathcal{D}_i^{\mathrm{test}}$ from $\mathcal{D}_i$
3. Compute $\phi_i \leftarrow f_\theta(\mathcal{D}_i^{\mathrm{tr}})$
4. Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\mathrm{test}})$

Optimization-Based Approach

$$\max_{\phi_i} \log p(\mathcal{D}_i^{\mathrm{tr}}|\phi_i) + \log p(\phi_i|\theta)$$

$$y^{\mathrm{ts}} = f_{\mathrm{MAML}}(\mathcal{D}_i^{\mathrm{tr}}, x^{\mathrm{ts}}) = f_{\phi_i}(x^{\mathrm{ts}})$$

where $\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\mathrm{tr}})$



1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\mathrm{tr}}, \mathcal{D}_i^{\mathrm{test}}$ from $\mathcal{D}_i$
3. ~~Compute $\phi_i \leftarrow f_\theta(\mathcal{D}_i^{\mathrm{tr}})$~~ Optimize $\phi_i \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\mathrm{tr}})$
4. Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\mathrm{test}})$

2

3. MAML

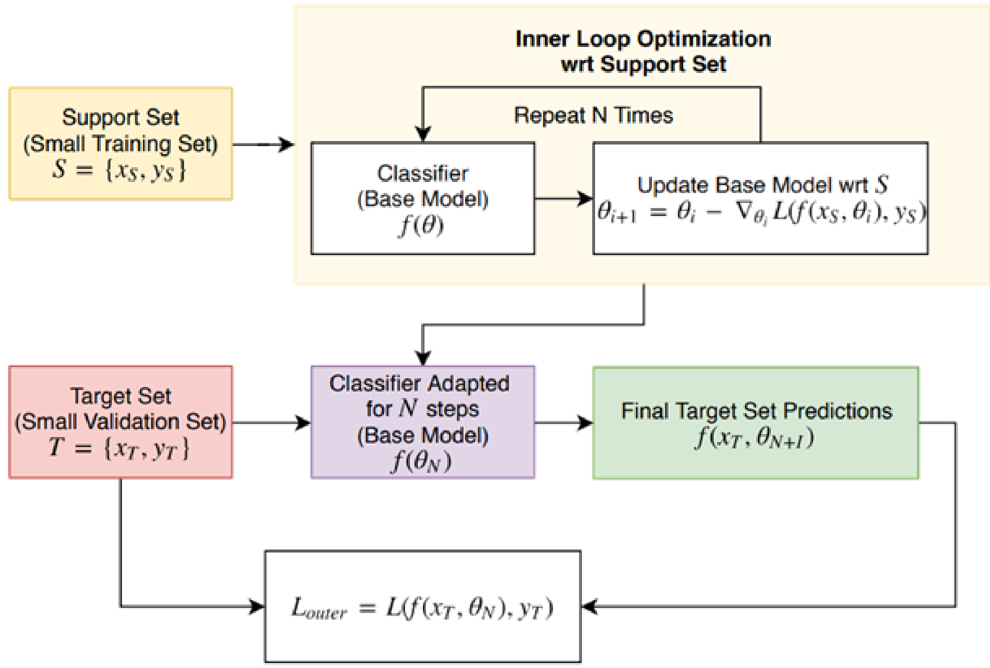
➤ MAML

Meta-Train

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

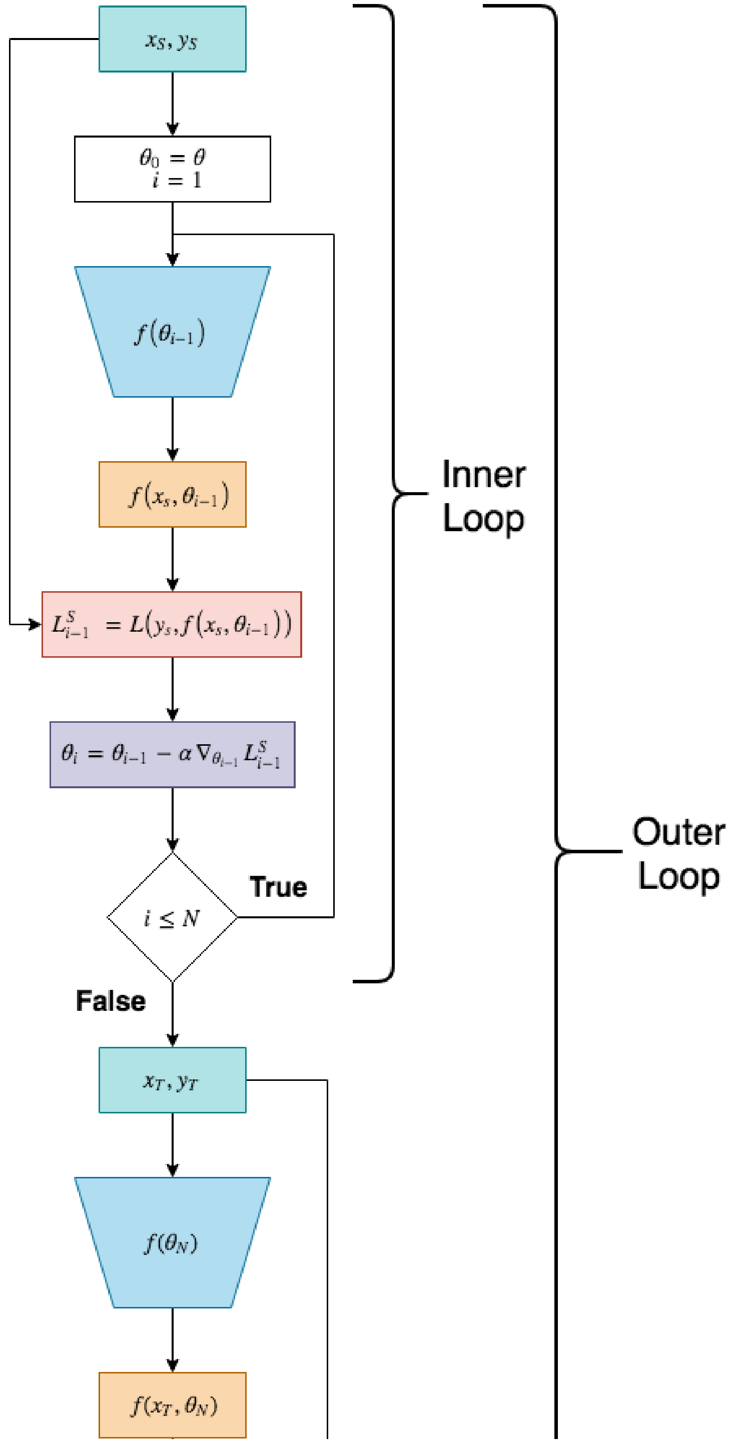
randomly initialize θ
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ Support Set
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ Query Set
end while



- 无论是meta-train还是fine-tune阶段，每个task都包括两部分，support set和query set。
- 不同的是，在meta-train阶段，support set参与第一次参数更新。这里的参数更新并没有直接作用于原模型，我们可以理解为先copy了一下模型，用来计算新参数。利用第一轮更新后的参数，通过query set计算第二轮梯度，这一轮的梯度才是模型真正用于更新参数的梯度；
- 在fine-tune阶段，support set参与第一次参数更新，更新结果直接作用于原模型，此时没有第二次参数更新，因为query set相当于测试集。
- 作者：徐不知，链接：https://www.zhihu.com/question/292959709/answer/605504088

Finn, C., Abbeel, P. & Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. ICML 2017.

MAML Computation Graph



x_S, y_S
$\theta_0 = \theta$
$i = 1$
$f(\theta_{i-1})$
$f(x_s, \theta_{i-1})$
$L^S_{i-1} = L(y_s, f(x_s, \theta_{i-1}))$
$\theta_i = \theta_{i-1} - \alpha \nabla_{\theta_{i-1}} L^S_{i-1}$
$i \leq N$
True
False
x_T, y_T
$f(\theta_N)$
$f(x_T, \theta_N)$
Inner Loop
Outer Loop

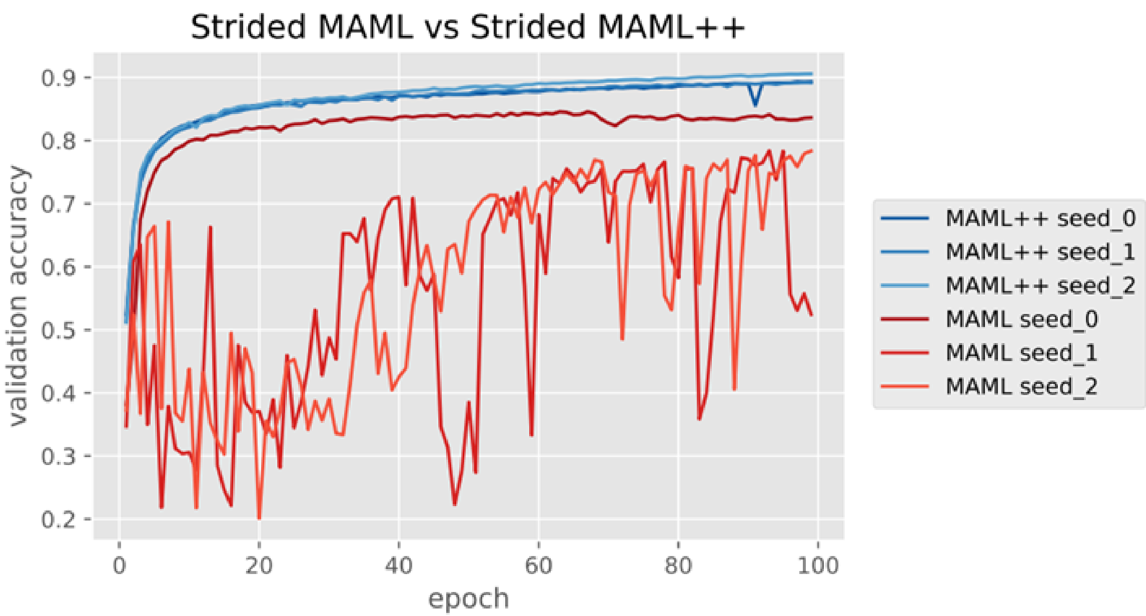
4. MAML Problems

$L_N^T = L(y_T, f(x_s, \theta_N))$

How to train your MAML

➢ MAML->MAML++

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_\theta \mathcal{L}(\mathcal{D}_{\text{train}}))$$



MAML Problems:

- ➢ Training Instability
- ➢ Second Order Derivative Cost
- ➢ Absence of Batch Normalization Statistic Accumulation
- ➢ Shared (across step) Batch Normalization Bias
- ➢ Shared Inner Loop (across step and across parameter) Learning Rate
- ➢ Fixed Outer Loop Learning Rate

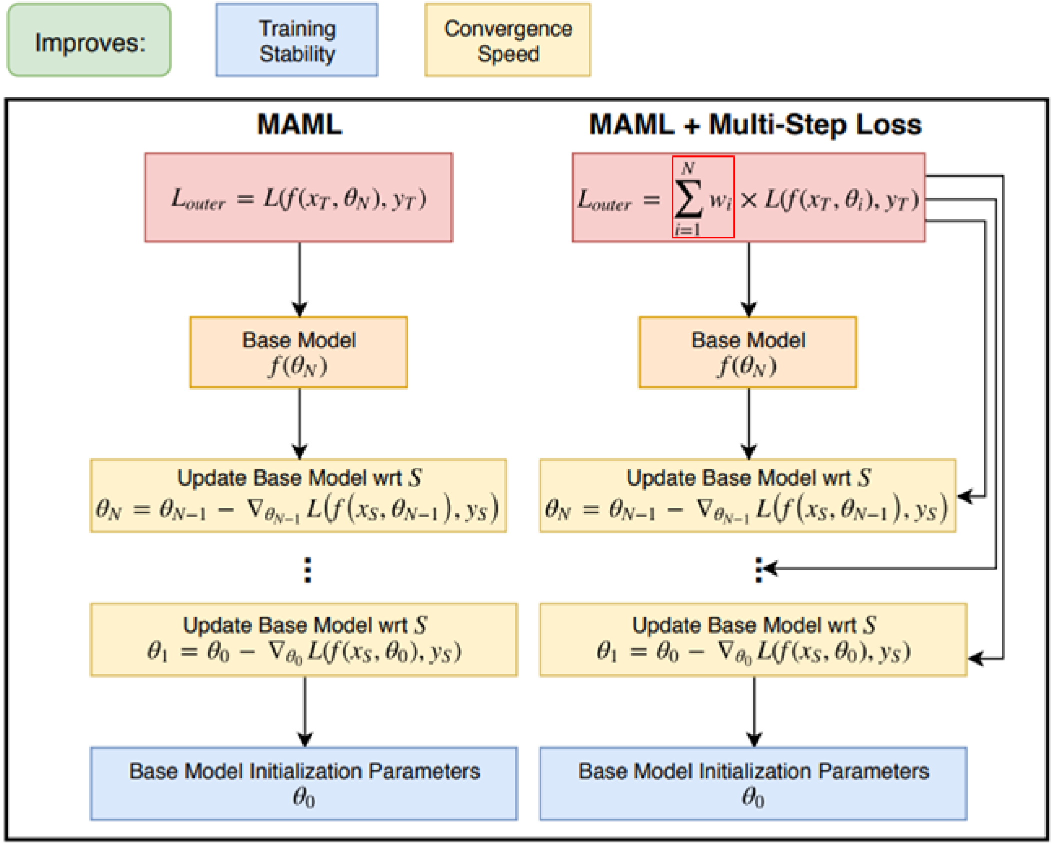
Antoniou, A., Edwards, H., & Storkey, A. How to train your MAML. ICLR 2019.

4

5. MAML++

➢ MAML->MAML++

- Gradient Instability->**Multi-Step** Loss Optimization (MSL)



- ☐ Minimizing the target set loss computed by the base-network after every step towards a support set task.
- ☐ More specifically, the loss minimized is a weighted sum of the target set losses after every support set loss update.

$$\theta = \theta - \beta \nabla_\theta \sum_{b=1}^{B} \boxed{\sum_{i=0}^{N}} v_i \mathcal{L}_{T_b}\left(f_{\theta_i^b}\right)$$

task b, step i

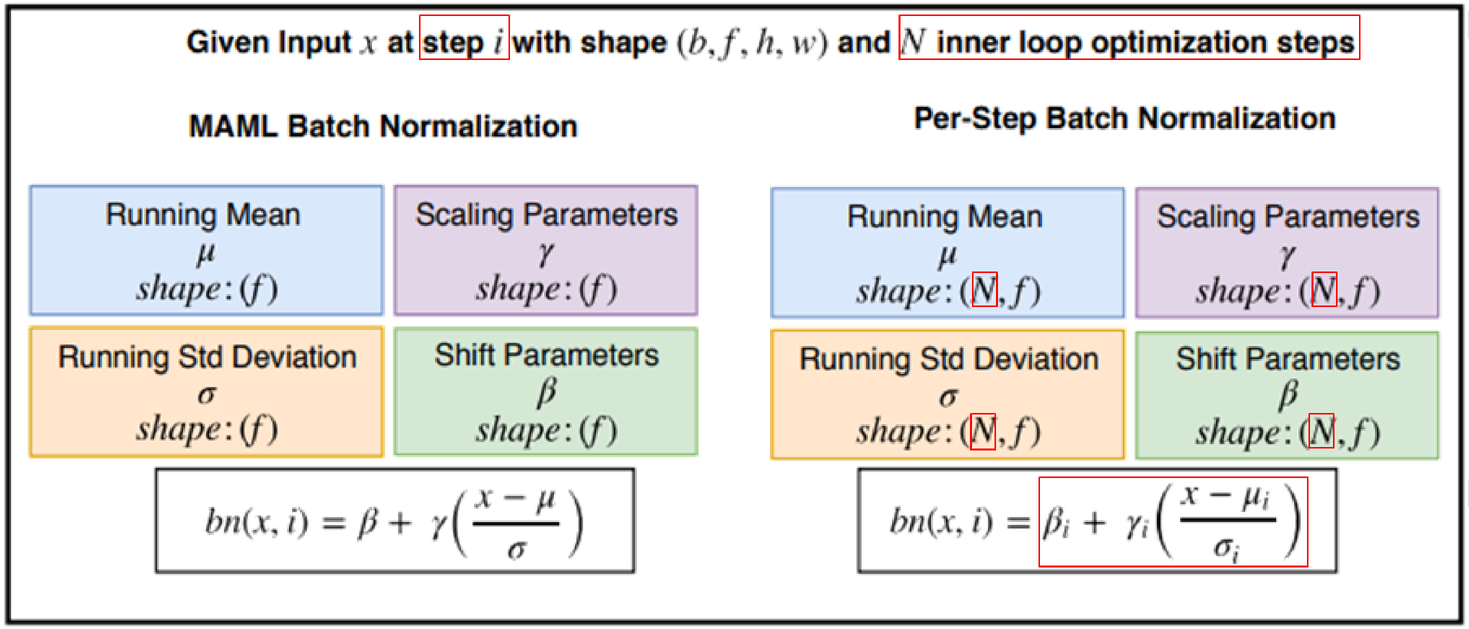
Antoniou, A., Edwards, H., & Storkey, A. How to train your MAML. ICLR 2019.

How to train your MAML

➢ MAML->MAML++

- Absence of Batch Normalization Statistic Accumulation->**Per-Step** Batch Normalization Running Statistics (BNRS)
- Shared (across step) Batch Normalization Bias->**Per-Step** Batch Normalization Weights and Biases (BNWB)

Improves: Training Stability | Convergence Speed | Generalization Performance



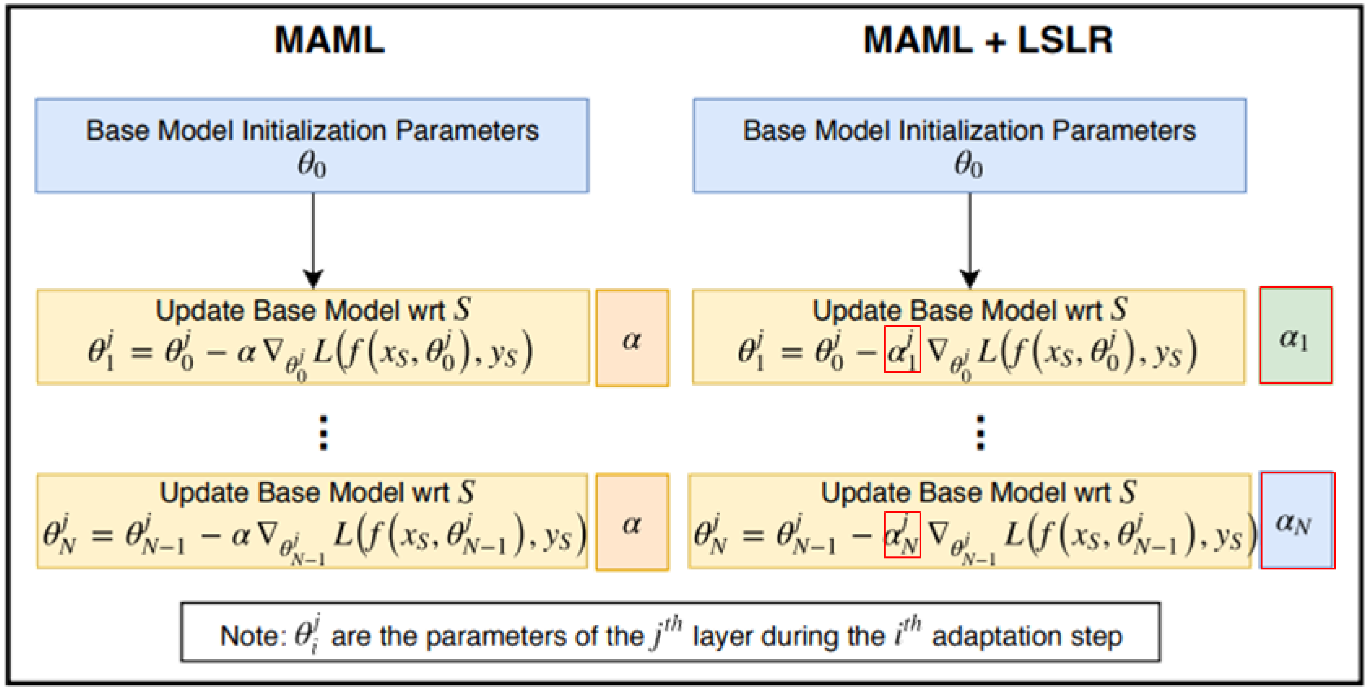
- ☐ Collecting statistics in a per-step regime. Instantiating N sets of running mean and running standard deviation for each batch normalization layer in the network and update the running statistics respectively with the steps being taken during the optimization.
- ☐ Learning a set of biases per-step within the inner-loop update process.

Antoniou, A., Edwards, H., & Storkey, A. How to train your MAML. ICLR 2019.

➢ MAML->MAML++

- Shared Inner Loop (across step and across parameter) Learning Rate->Learning **Per-Layer Per-Step** Learning Rates and Gradient Directions (LSLR)

Improves: Training Stability | Convergence Speed | Generalization Performance



- ☐ Learning a learning rate and direction for each layer in the network.
- ☐ Learning different learning rates for each adaptation of the base-network.

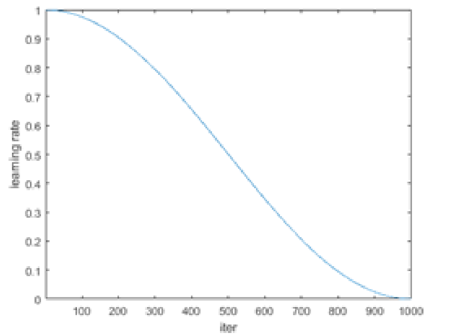
Antoniou, A., Edwards, H., & Storkey, A. How to train your MAML. ICLR 2019.

➢ MAML->MAML++

- Second Order Derivative Cost->Derivative-Order Annealing (DA)
  - ☐ Using first-order gradients for the first 50 epochs of the training phase, then switching to second-order gradients for the remainder of the training phase.
- Fixed Outer Loop Learning Rate->Cosine Annealing of Meta-Optimizer Learning Rate (CA)
  - ☐ Applying the cosine annealing scheduling on the meta-model's optimizer (i.e. the meta-optimizer).

$$\theta \leftarrow \theta - \boxed{\beta} \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$$

$$\lambda(t) = \frac{1}{2}\left(1 + \cos\left(\frac{t\pi}{T}\right)\right)$$



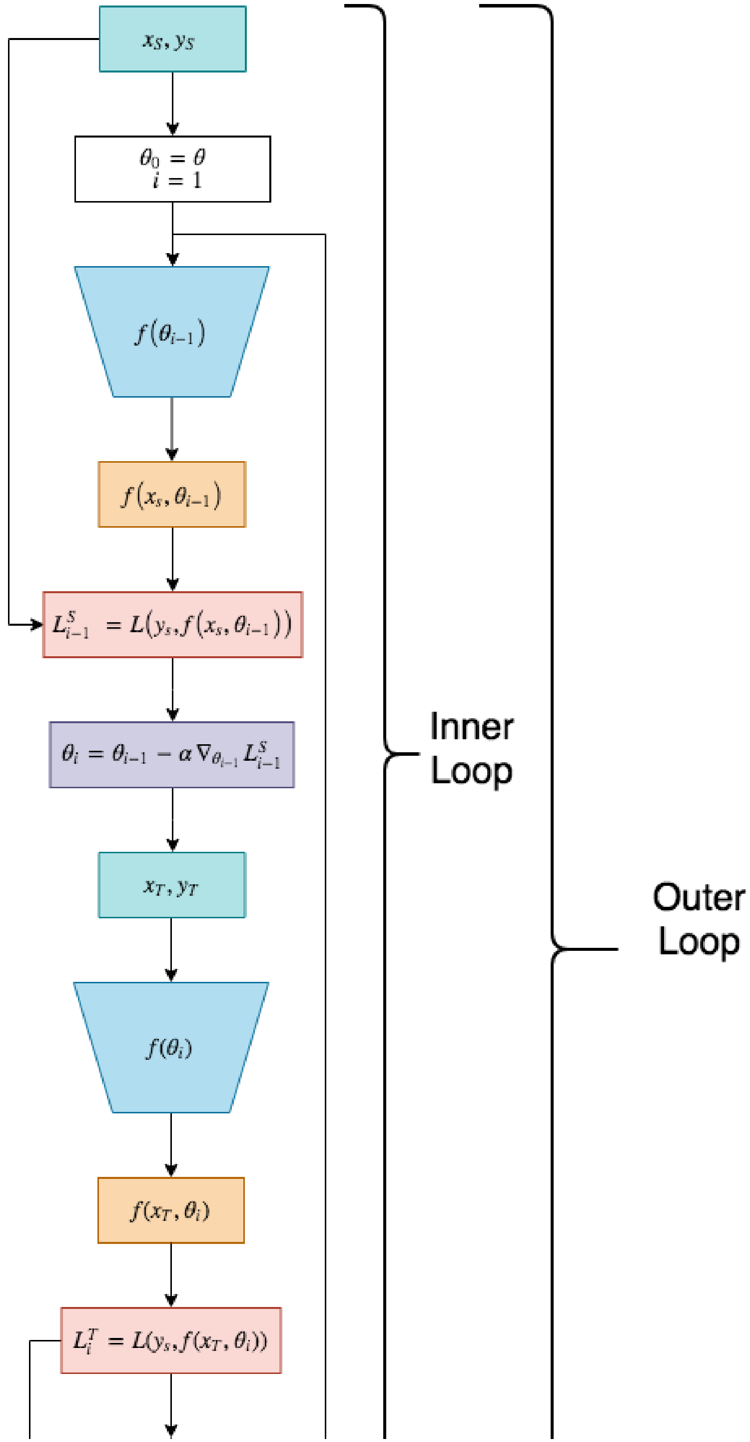
	Accuracy			
	Omniglot 20-way		Mini-ImageNet 5-way	
Approach	**1-shot**	**5-shot**	**1-shot**	**5-shot**
Siamese Nets	88.2%	97.0%	-	-
Matching Nets	93.8%	98.5%	43.56%	55.31%
Neural Statistician	93.2%	98.1%	-	-
Memory Mod.	95.0%	98.6%	-	-
Meta-SGD	95.93±0.38%	98.97±0.19%	50.47±1.87%	64.03±0.94%
Meta-Networks	97.00%	—	49.21%	-
MAML (original)	95.8±0.3%	98.9±0.2%	48.70±1.84%	63.11±0.92%
MAML (local replication)	91.27±1.07%	98.78%	48.25±0.62%	64.39±0.31%
MAML++	**97.65±0.05%**	**99.33±0.03%**	**52.15±0.26%**	**68.32±0.44%**

MAML++ Few-Shot Results

☐ The results of the approach indicate that learning per-step learning rates, batch normalization parameters and optimizing on per-step target losses appears to be key for fast, highly automatic and strongly generalizable few-shot learning.

Antoniou, A., Edwards, H., & Storkey, A. How to train your MAML. ICLR 2019.

MAML with Multi-Step Loss Computation Graph



x_S, y_S
$\theta_0 = \theta$
$i = 1$
$f(\theta_{i-1})$
$f(x_s, \theta_{i-1})$
$L_{i-1}^S = L(y_s, f(x_s, \theta_{i-1}))$
$\theta_i = \theta_{i-1} - \alpha \nabla_{\theta_{i-1}} L_{i-1}^S$
x_T, y_T
$f(\theta_i)$
$f(x_T, \theta_i)$
$L_i^T = L(y_s, f(x_T, \theta_i))$
Inner Loop
Outer Loop

6．参考文献

[1] Finn, C., Abbeel, P. & Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. ICML 2017.
Code: https://github.com/cbfinn/maml, https://github.com/dragen1860/MAML-Pytorch

Finn个人主页：https://ai.stanford.edu/~cbfinn/

[2] Antoniou, A., Edwards, H., & Storkey, A. How to train your MAML. ICLR 2019. Code: https://github.com/AntreasAntoniou/HowToTrainYourMAMLPytorch

[3] How to train your MAML: A step by step approach · BayesWatch https://www.bayeswatch.com/2018/11/30/HTYM/

[4] CS 330: Deep Multi-Task and Meta Learning http://web.stanford.edu/class/cs330/

[5] Meta-Learning: Learning to Learn Fast https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.html