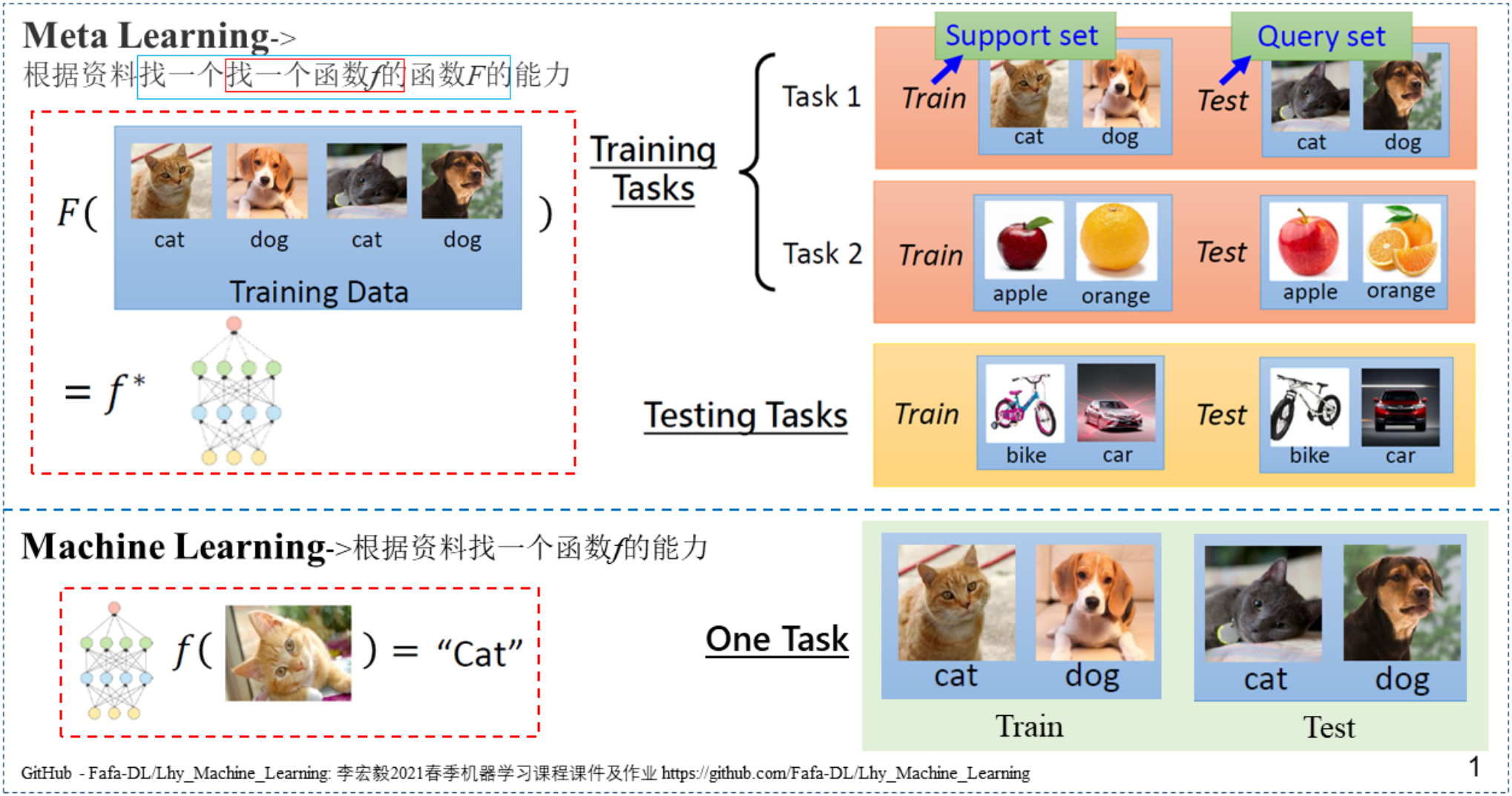
元学习——MAML、Reptile与ANIL

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

之前介绍过元学习——从MAML到MAML++，这次在此基础上进一步探讨，深入了解MAML的本质，引出MAML高效学习的原因究竟是快速学习，学到一个很厉害的初始化参数，还是特征重用，初始化参数与最终结果很接近？因此得到ANIL(Almost No Inner Loop)，随后我们阅读了Reptile——On first-order meta-learning algorithms，另一种元学习方法，并比较了MAML、Reptile与模型预训练之间的区别。

1. Meta Learning vs Machine Learning



GitHub - Fafa-DL/Lhy_Machine_Learning: 李宏毅2021春季机器学习课程课件及作业 https://github.com/Fafa-DL/Lhy_Machine_Learning

2. MAML vs Model Pre-Training

MAML vs Model Pre-Training

MAML

$$L(\phi) = \sum_{n=1}^{N} l^n(\hat{\theta}^n)$$

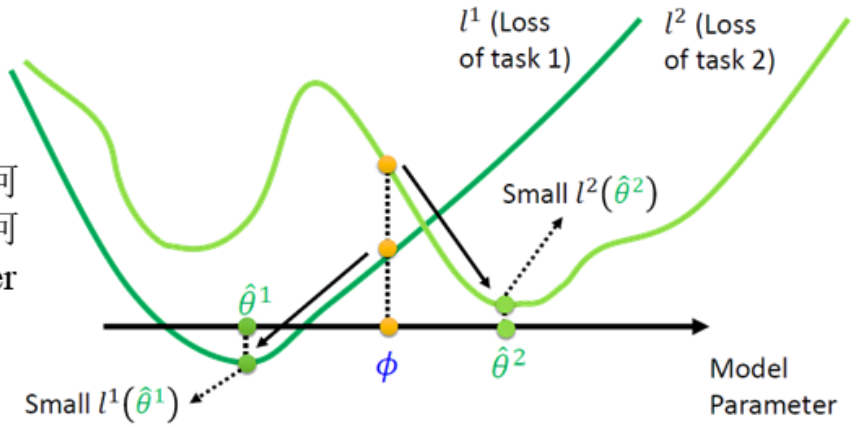
N个任务 — model learned from task n — loss of task n on the testing set of task n

$$\phi = \phi - \eta \nabla_{\phi} L(\phi)$$

Considering one-step training:

$$\hat{\theta} = \phi - \varepsilon \nabla_{\phi} l(\phi)$$

- 我们不在意Φ在训练任务上的表现如何
- 我们在意用Φ训练出来的 $\hat{\theta}^n$ 表现如何
- Find Φ achieving good performance after training (潜力)

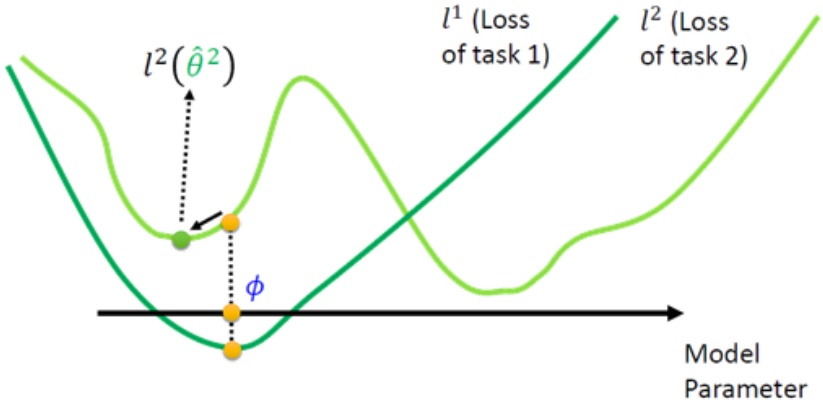


目标：学习到的**Meta Model经过每个Task的Adaption之后最好**

Model Pre-Training

$$L(\phi) = \sum_{n=1}^{N} l^n(\phi)$$

- 找寻在所有任务上都最好的Φ
- 并不能保证拿Φ去训练后能得到好的 $\hat{\theta}^n$
- Find Φ achieving good performance (现在表现如何)



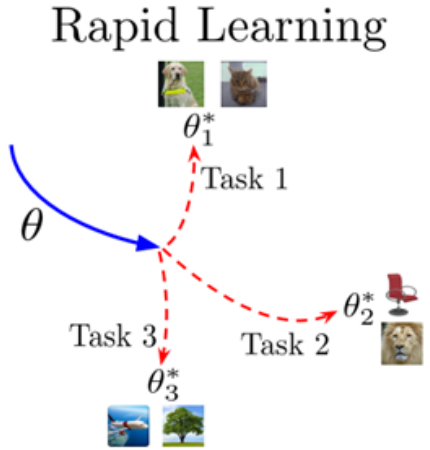
目标：学习到的**Model本身**在各个Task上最好，**而在Pretraining的过程中是不会考虑Fine-tuning的**

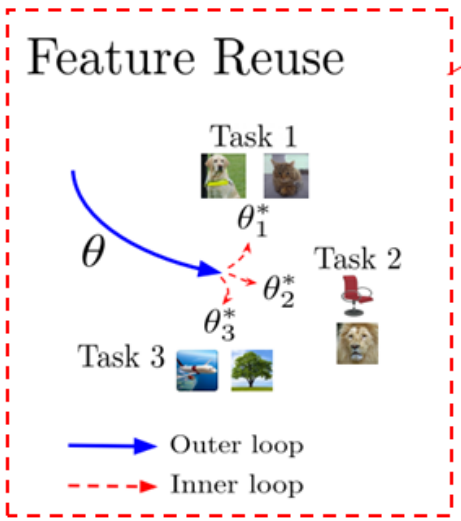
GitHub - Fafa-DL/Lhy_Machine_Learning: 李宏毅2021春季机器学习课程课件及作业 https://github.com/Fafa-DL/Lhy_Machine_Learning

3. MAML——Feature Reuse

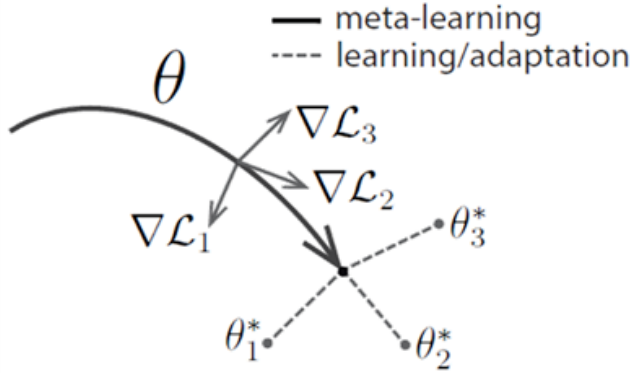
MAML

- 尽管MAML很流行，一个基本的开放问题仍然存在——MAML的有效性是由于为快速学习准备的元初始化(表现形式的大而有效的变化)，还是由于特性重用，而元初始化已经包含了高质量的特性?
- 通过进行消融实验和潜在表征分析，发现MAML即使不进行adaptation也能有很好的效果，说明MAML学到了一个更好的特征而不是能够快速适应，因此特征重用是MAML有效学习的主要原因。





是MAML高效学习的主导因素



In **Rapid Learning**, outer loop training leads to a parameter setting that is well-conditioned for fast learning, and inner loop updates result in significant task specialization.

In **Feature Reuse**, the outer loop leads to parameter values corresponding to reusable features, from which the parameters do not move significantly in the inner loop.

图中灰色分支线代表不同task的更新方向，而黑色的主轴线代表模型参数的最终走向，这可以看作是一种“中庸”的思想，可以防止参数不过拟合到某一个task上。最后的虚线代表的是对新task的adaption，也就是对模型参数的fin-tine。adaption的过程**可以是简单的几步梯度更新就可以使模型适应到新的task上**，这也是文章的主要目的。

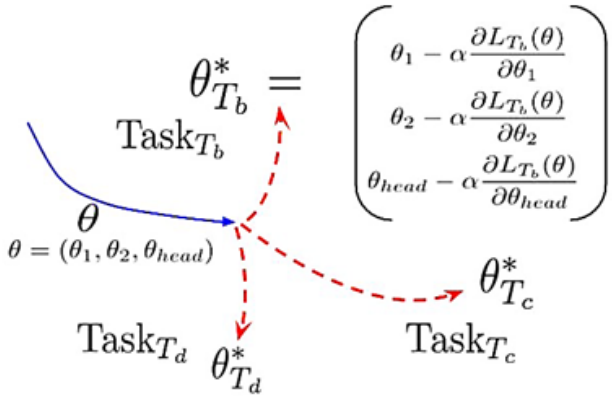
Finn, C., Abbeel, P. & Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. ICML 2017.
Aniruddh Raghu, Maithra Raghu, Samy Bengio, Oriol Vinyals, Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML, ICLR, 2020.

4. MAML vs ANIL

➢ MAML vs ANIL

- 由上述可知，特征重用是MAML有效学习的主要原因。基于这些结果，提出了ANIL(几乎没有内环)算法，这是对MAML的一个显著简化，它在训练和推理过程中删除了除头部(最后一层)以外的所有神经网络的内环更新。

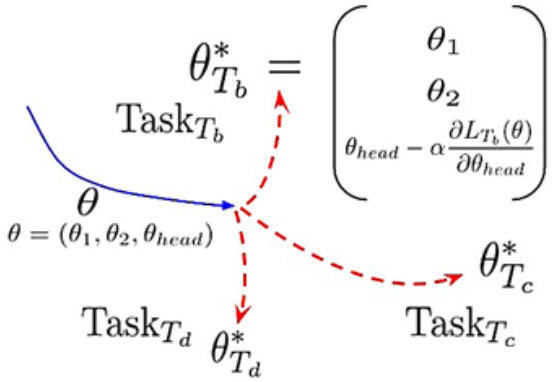
MAML



- 在MAML(左)中，内环(task-specific)梯度更新被应用到所有参数θ，这些参数是用外环的元初始化初始化的。
- 为神经网络找到一个初始化，这样就可以通过两个优化循环，用很少的样例(每个类的k个例子进行k-shot学习)学习新任务。
- 外层循环:更新神经网络参数的初始化(通常称为元初始化)，使其能够快速适应新任务。
- 内部循环:执行自适应:接受外部循环初始化，并对每个任务分别对为自适应提供的k个标记样例(支持集)执行一些梯度更新。

$$\theta_m^{(b)} = \theta_{m-1}^{(b)} - \alpha \nabla_{\theta_{m-1}^{(b)}} \mathcal{L}_{S_{T_b}}(f_{\theta_{m-1}^{(b)}(\theta)})$$

ANIL



- 在ANIL(右)中，训练和测试时内环只更新网络头对应的参数。
- 只有最后一层获得内部循环更新。

$$\theta_m^{(b)} = \left(\theta_1, \ldots, (\theta_l)_{m-1}^{(b)} - \alpha \nabla_{(\theta_l)_{m-1}^{(b)}} \mathcal{L}_{S_b}(f_{\theta_{m-1}^{(b)}})\right)$$

- 在Omniglot和Mini-ImageNet实验结果表明，内环更新只更新最后一层与MAML的效果相当。说明头部(最后一层)在学习网络体良好特征的训练中非常重要。
- The inner loop adaptation of the body is unnecessary for learning good features. Removing the inner loop during training does not hinder performance.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, Oriol Vinyals, Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML, ICLR, 2020

MAML的目标是学习一个参数θ使得其经过一个梯度迭代就可以在新任务上达到最好的性能。

内循环：与具体任务相关的任务适配参数的更新(自适应具体任务)

外循环：整体任务空间上的模型参数的更新(元初始化)

5. Reptile: On First-Order Meta-Learning Algorithms

Reptile: On First-Order Meta-Learning Algorithms

➢ Reptile

- 工作原理是重复对任务进行采样，对其进行随机梯度下降，并将初始参数更新为在该任务中学习到的最终参数。
- Reptile是最短下降算法在元学习设置中的应用，它在数学上类似于一阶MAML，只需要黑盒访问优化器(如SGD或Adam)，具有类似的计算效率和性能。
- 与MAML一样，Reptile寻求神经网络参数的初始化，以便可以使用来自新任务的少量数据对网络进行微调。
- 但当MAML通过梯度下降算法的计算图展开和微分时，Reptile只是以标准方式对每个任务执行随机梯度下降（SGD）——它不展开计算图或计算任何二阶导数。这使得Reptile比MAML需要更少的计算和内存。
- 当k=1时，该算法对应于“联合训练”——对混合的所有任务执行SGD。虽然联合训练在某些情况下可以学到有用的初始化，但当无法进行zero-shot学习时(例如，当输出标签是随机排列的)，它学到的很少。Reptile要求k>1，其中更新依赖于损失函数的高阶导数；正如我们在文中所示，这与k=1(联合训练)非常不同。

Reptile

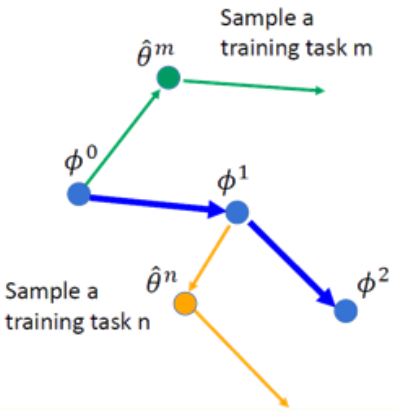
Initialize Φ, the initial parameter vector
for iteration 1,2,3,… **do**
  Randomly sample a task T
  Perform $k>1$ steps of SGD on task T, starting with parameters Φ, resulting in parameters W
  Update: $\Phi \leftarrow \Phi + \epsilon(W - \Phi)$
end for
Return Φ

Nichol A, Achiam J, Schulman J. On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999, 2018.
Nichol A, Schulman J. Reptile: a scalable metalearning algorithm. arXiv preprint arXiv:1803.02999, 2018, 2(3): 4.
Reptile: A Scalable Meta-Learning Algorithm. OpenAI. https://openai.com/blog/reptile/

6. MAML, Model Pre-Training, and Reptile

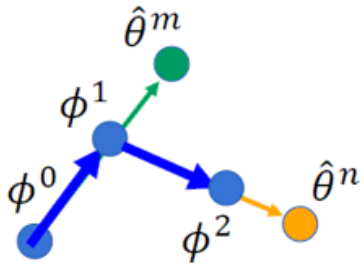
MAML, Model Pre-Training, and Reptile

MAML



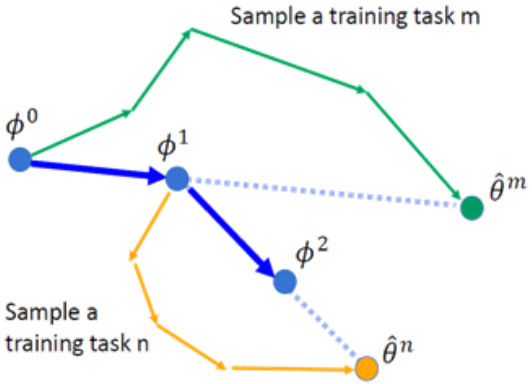
- MAML是使用子任务的参数，**第二次更新**的gradient的方向来更新参数(所以第一个蓝色箭头的方向与第二个绿色箭头的方向平行；第二个蓝色箭头的方向与第二个橘色箭头的方向平行)。
- 每次base model参数的更新，是在每个task的二阶更新的 基础上做的。

Model Pre-Training



- Model Pre-training是使用子任务第一步更新的gradient的方向来更新参数(子任务的梯度往哪个方向走，model的参数就往哪个方向走)。
- 每次在一阶上做，每次都要求当前的loss的最小化方向做参数更新。

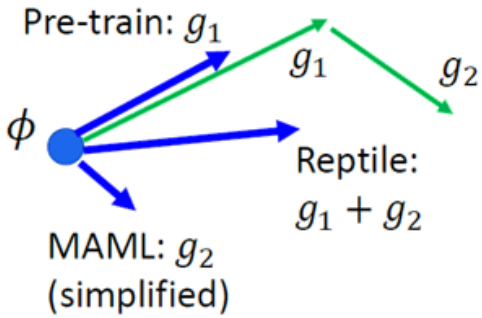
Reptile



- Reptile训练任务的网络可以更新多次，不再像MAML一样计算梯度(因此带来了工程性能的提升)，而是直接用一个参数ε乘以meta网络与训练任务的网络参数的差来更新meta网络参数。
- 一个一阶的基于梯度的元学习算法。每次base model的参数更新是在每个task的一阶梯度上做的，只不过是做了很多次。

$$\Phi \leftarrow \Phi + \epsilon(W - \Phi)$$

三者总结



- Pre-train: θ一次训练以后的方向g1就是Φ更新的方向
- MAML: θ第二次更新的方向，我们将其作为更新Φ的方向
- Reptile: 两次方向的向量和，当然，我们不仅仅可以更新两次，可以更新很多次，之后很多次的和方向就是最后的方向。

一文入门元学习（Meta-Learning）（附代码） - 知乎 https://zhuanlan.zhihu.com/p/136975128
GitHub - Fafa-DL/Lhy_Machine_Learning: 李宏毅2021春季机器学习课程课件及作业 https://github.com/Fafa-DL/Lhy_Machine_Learning

6

7. 参考文献

[1] GitHub - Fafa-DL/Lhy_Machine_Learning: 李宏毅2021春季机器学习课程课件及作业 https://github.com/Fafa-DL/Lhy_Machine_Learning

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

[3] Aniruddh Raghu, Maithra Raghu, Samy Bengio, Oriol Vinyals, Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML, ICLR, 2020.

[4] Nichol A, Achiam J, Schulman J. On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999, 2018.

[5] Nichol A, Schulman J. Reptile: a scalable metalearning algorithm. arXiv preprint arXiv:1803.02999, 2018, 2(3): 4.

[6] Reptile: A Scalable Meta-Learning Algorithm. OpenAI. https://openai.com/blog/reptile/

[7] 一文入门元学习（Meta-Learning）（附代码） - 知乎 https://zhuanlan.zhihu.com/p/136975128