Meta-Learning

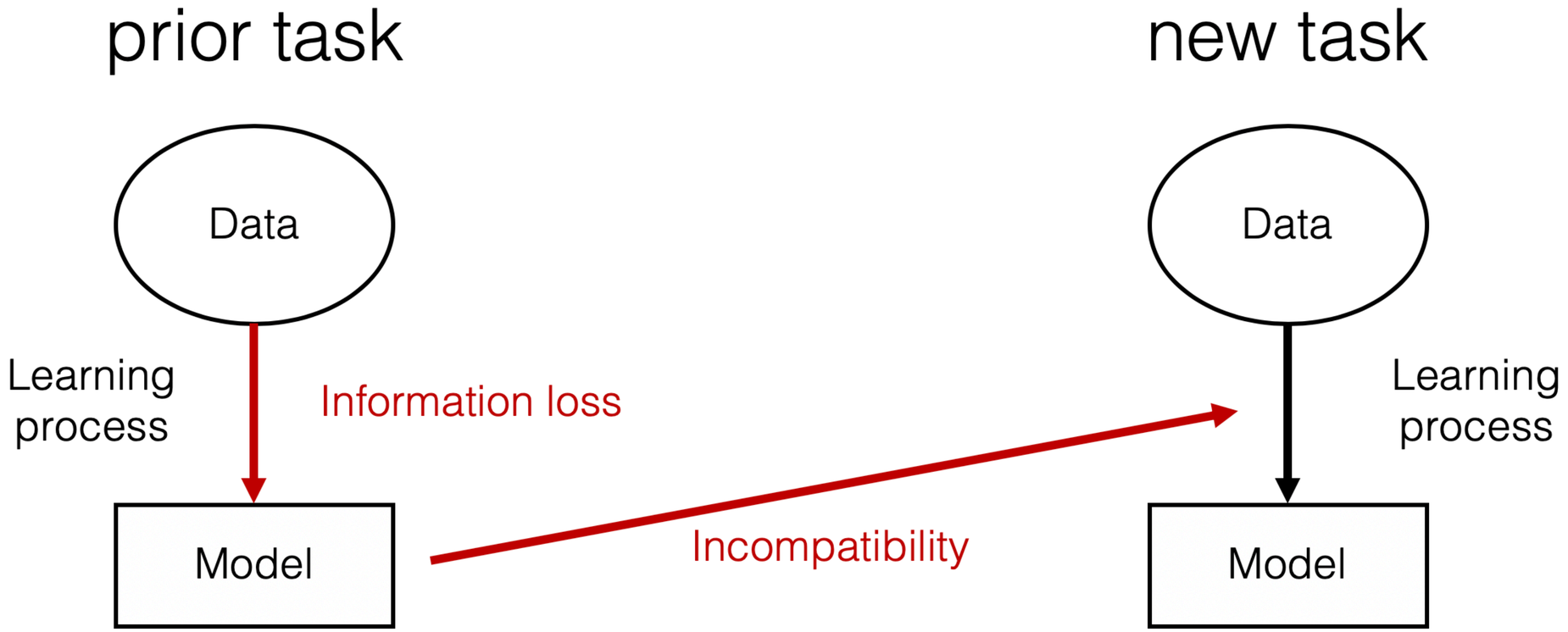
Part I
From MAML to iMAML

Alex Fedorov

June 12, 2020

Fine-tunning

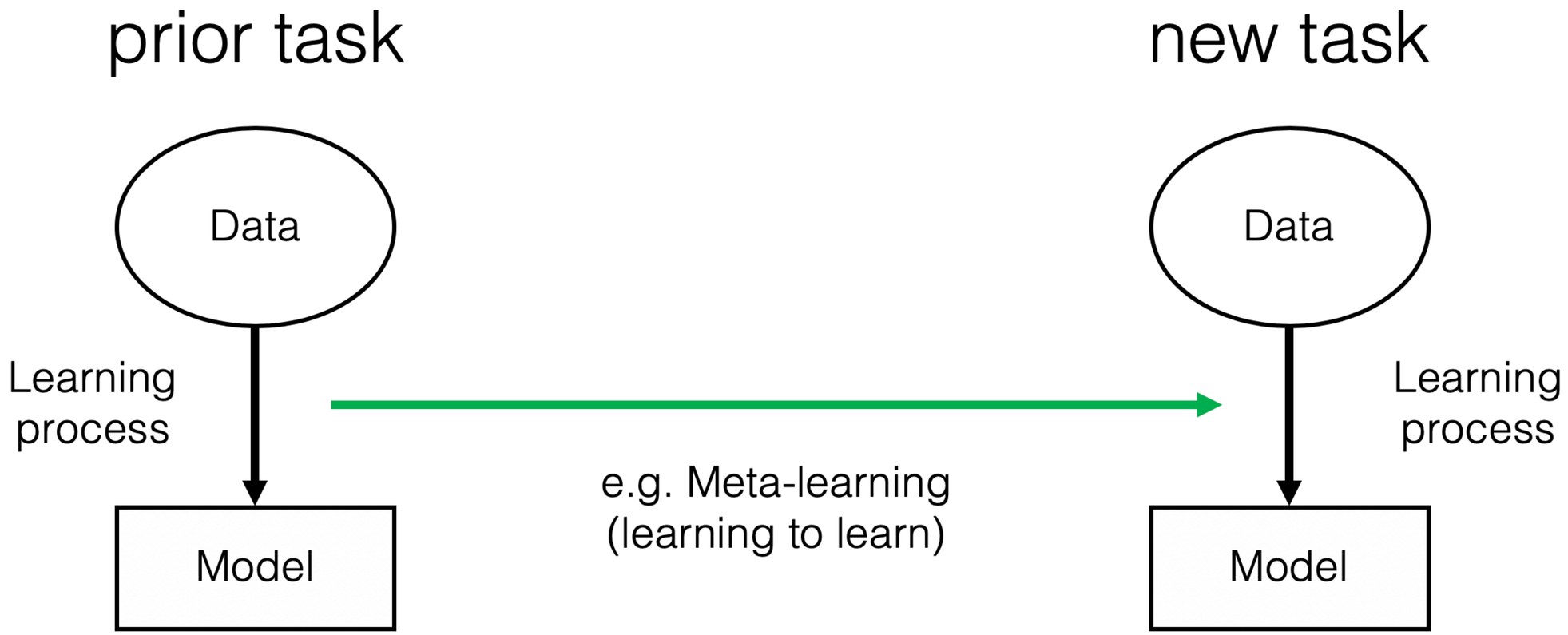
- **Fine-tuning ignores the process of learning when transferring knowledge.**
- **To consistently transfer knowledge, we have to transfer knowledge across across the learning processes themselves.**



http://flennerhag.com/2019-05-09-transferring-knowledge-across-learning-processes/

Learning to Learn or Meta-Learning

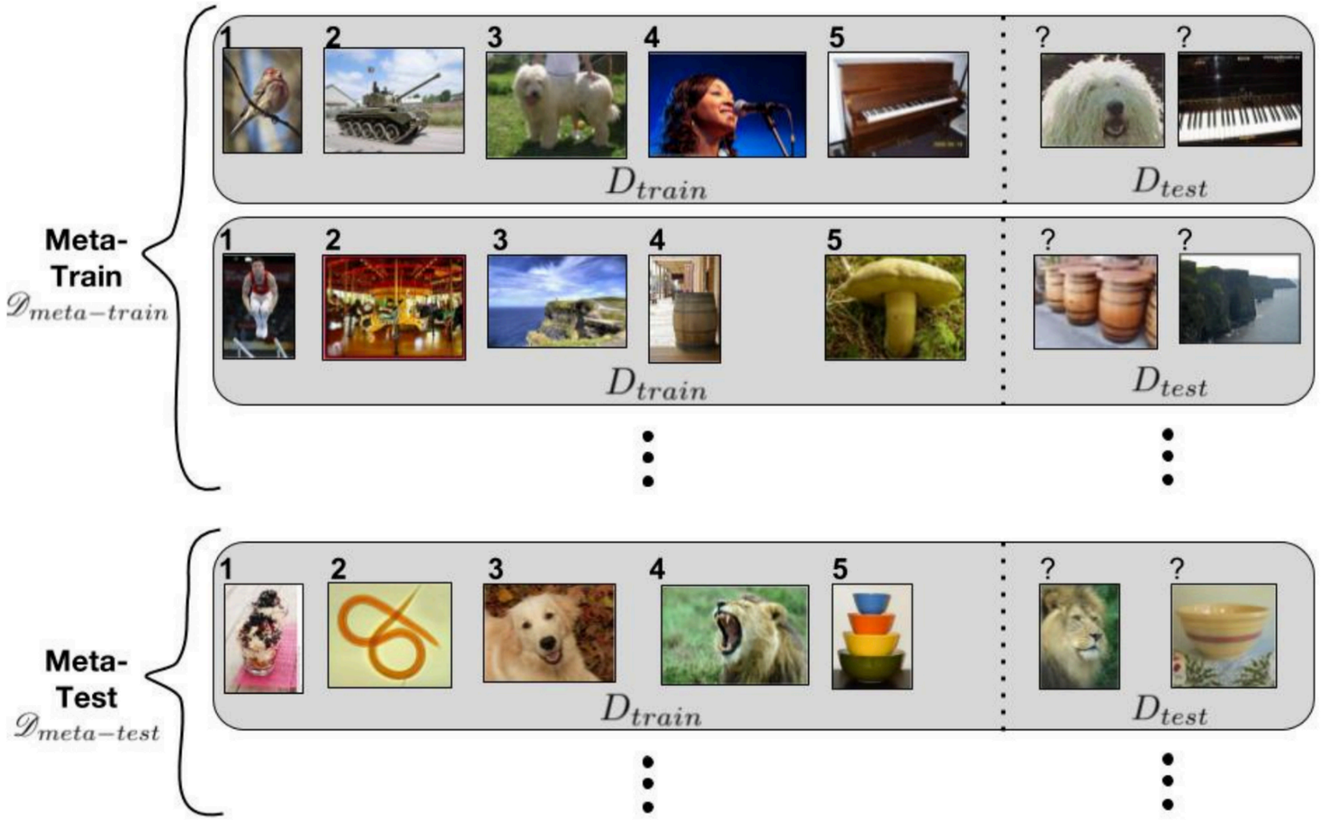
- **Transferring knowledge across learning processes can be seen as a learning problem in its own right, so called learning to learn or meta-learning.**
- **A meta-learner learns how to transfer knowledge when learning a new task.**



http://flennerhag.com/2019-05-09-transferring-knowledge-across-learning-processes/

Meta-Learning

- Meta-learning takes a set of tasks, each with a training and test set, and learns how to adapt a task learner to any one of them.
- The goal is to generalise to unseen tasks, in the sense that the task learner can adapt to them. Credit: Ravi & Larochelle [11].



Meta-Learning focuses on Few-Shot Learning

http://flennerhag.com/2019-05-09-transferring-knowledge-across-learning-processes/

MAML

MAML

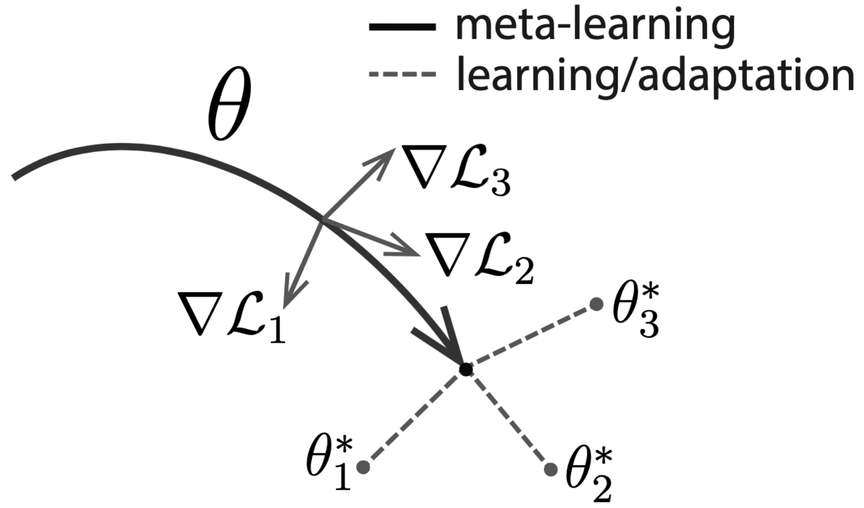


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

Find θ s.t. for $t_i \sim p(t)$ with $\mathcal{L}_{t_i}$

$$\min_\theta \mathbb{E}_t \left[L_t \left(U_t^k(\theta) \right) \right]$$

U_t^k: operator that updates θ k times

$\Updownarrow$

perform gradient descent

MAML $\Rightarrow$ $\min_\theta \mathbb{E}_t \left[L_{t,B} \left(U_{t,A}(\theta) \right) \right]$

$$g_{MAML} = \frac{\partial}{\partial \theta} L_{t,B}\left(U_{t,A}(\theta)\right) = \underbrace{U'_{t,A}(\theta)}_{\text{Jacobian of } U_{t,A}(\theta)} L'_{t,B}(\tilde{\theta}), \quad \tilde{\theta} = U_{t,A}(\theta)$$

$U_{t,A}(\theta) = \theta + g_1 + \ldots + g_k$

FO-MAML: $U'_{t,A}(\theta) = I$

$$\Rightarrow g_{FO\text{-}MAML} = L'_{t,B}(\tilde{\theta})$$

Algorithm

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

Algorithm 2 MAML for Few-Shot Supervised Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample datapoints $\mathcal{D}_i' = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

Related work

Reptile

Algorithm 1 Reptile (serial version)

Initialize ϕ, the vector of initial parameters
for iteration $= 1, 2, \ldots$ **do**
  Sample task τ, corresponding to loss L_τ on weight vectors $\widetilde{\phi}$
  Compute $\widetilde{\phi} = U_\tau^k(\phi)$, denoting k steps of SGD or Adam
  Update $\phi \leftarrow \phi + \epsilon(\widetilde{\phi} - \phi)$
end for

$$g_{\text{MAML}} = \overline{g}_k - \alpha \overline{H}_k \sum_{j=1}^{k-1} \overline{g}_j - \alpha \sum_{j=1}^{k-1} \overline{H}_j \overline{g}_k + O(\alpha^2)$$

$$\mathbb{E}\left[g_{\text{MAML}}\right] = (1)\text{AvgGrad} - (2(k-1)\alpha)\text{AvgGradInner}$$

$$g_{\text{FOMAML}} = g_k = \overline{g}_k - \alpha \overline{H}_k \sum_{j=1}^{k-1} \overline{g}_j + O(\alpha^2)$$

$$\mathbb{E}\left[g_{\text{FOMAML}}\right] = (1)\text{AvgGrad} - ((k-1)\alpha)\text{AvgGradInner}$$

$$g_{\text{Reptile}} = -(\phi_{k+1} - \phi_1)/\alpha = \sum_{i=1}^{k} g_i = \sum_{i=1}^{k} \overline{g}_i - \alpha \sum_{i=1}^{k} \sum_{j=1}^{i-1} \overline{H}_i \overline{g}_j + O(\alpha^2)$$

$$\mathbb{E}\left[g_{\text{Reptile}}\right] = (k)\text{AvgGrad} - \left(\tfrac{1}{2}k(k-1)\alpha\right)\text{AvgGradInner}$$

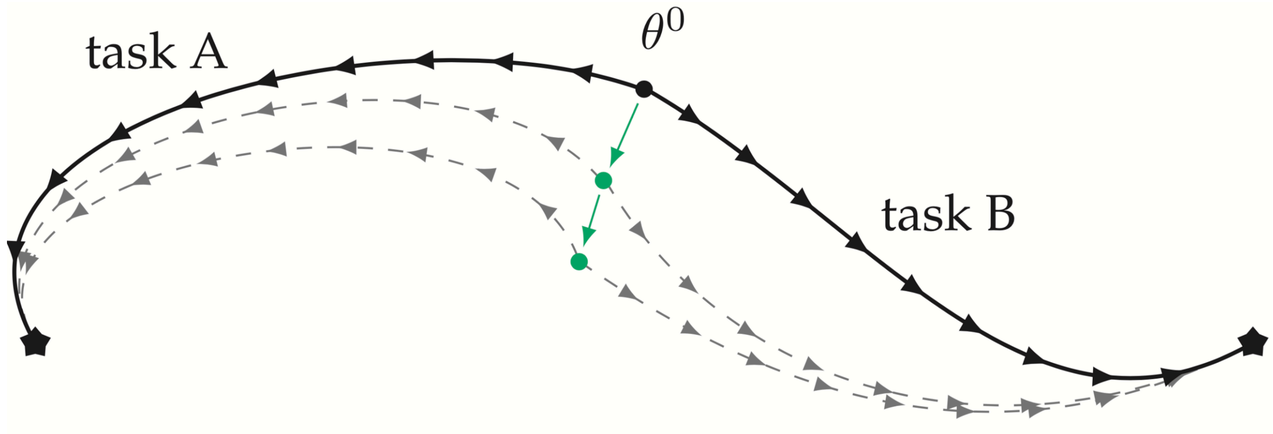
$g_i = L_i'(\phi_i)$ (gradient obtained during SGD)
$\phi_{i+1} = \phi_i - \alpha g_i$ (sequence of parameter vectors)
$\overline{g}_i = L_i'(\phi_1)$ (gradient at initial point)
$\overline{H}_i = L_i''(\phi_1)$ (Hessian at initial point)

Nichol, Alex, Joshua Achiam, and John Schulman. "On first-order meta-learning algorithms." *arXiv preprint arXiv:1803.02999* (2018).

Leap

$$\theta^{k+1} = \theta^k + u(L(x; \theta^k), y).$$

$$d(\theta^0, \theta^K) = \sum_{k=0}^{K-1} \|\theta^{k+1} - \theta^k\|.$$



Consequently, we can learn to transfer knowledge across learning processes by learning an initialisation such that the expected distance we have to travel when learning a similar task is as short as possible.

Leap learns an initialisation that induces faster learning on tasks from the given task distribution. By minimising the distance we need to travel, we make tasks as ‘easy’ as possible to learn.

Flennerhag, Sebastian, et al. "Transferring knowledge across learning processes." *arXiv preprint arXiv:1812.01054* (2018).

http://flennerhag.com/2019-05-09-transferring-knowledge-across-learning-processes/

Omniglot

The Omniglot data set contains 50 alphabets.

Omniglot Comparison

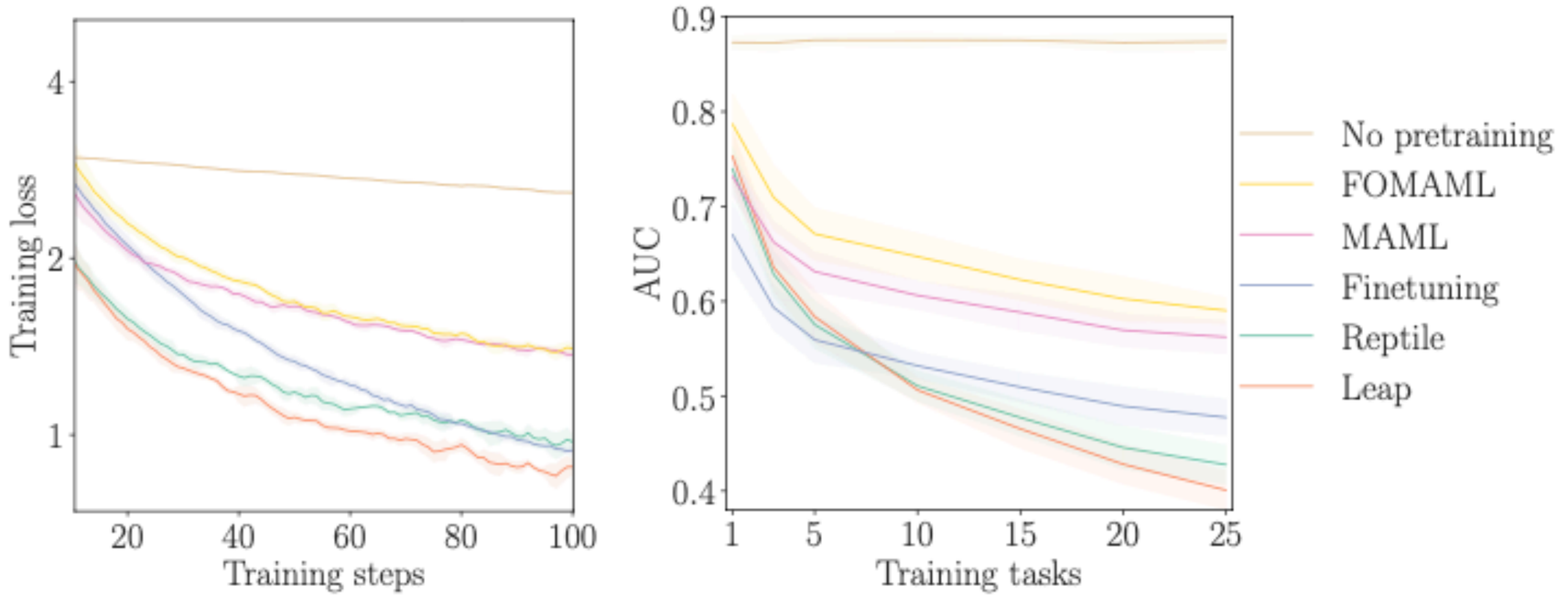


Figure 3: Results on Omniglot. *Left:* Comparison of average learning curves on held-out tasks (across 10 seeds) for 25 tasks in the meta-training set. Curves are moving averages with window size 5. Shading: standard deviation within window. *Right:* AUC across number of tasks in the meta-training set. Shading: standard deviation across 10 seeds.

Implicit MAML

https://homes.cs.washington.edu/~aravraj/assets/research/imaml_poster.pdf

Problem Setting

i - task index
θ - meta parameter (init, lr, steps)
$\mathcal{L}$ - Loss function

ϕ - model weights
A - learning algorithm (SGD, Adam)
$\mathcal{D}$ - task dataset

$$\min_{\theta} \left\{ F(\theta) := \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i \left(\phi_i = A(\mathcal{D}_i, \theta) \right) \right\}$$

$$\nabla_\theta \mathcal{L}_i(\theta) = \underbrace{\frac{d\phi_i}{d\theta}}_{\text{hard}} \underbrace{\nabla_\phi \mathcal{L}_i(\varphi_i)}_{\text{easy}}$$

- Idea:
  - Optimize F through gradient-based iterative algorithms
- Requirements:
  - Efficient computation of task meta-gradients

Why not MAML?

- Restricts algorithms:
  - Each atomic operation need to be first order and differentiable
    - No line-search, trust-region, randomization
- Memory complexity
  - Linear in the length of Algorithm
- Vanishing gradients
  - Due possible long paths

iMAML

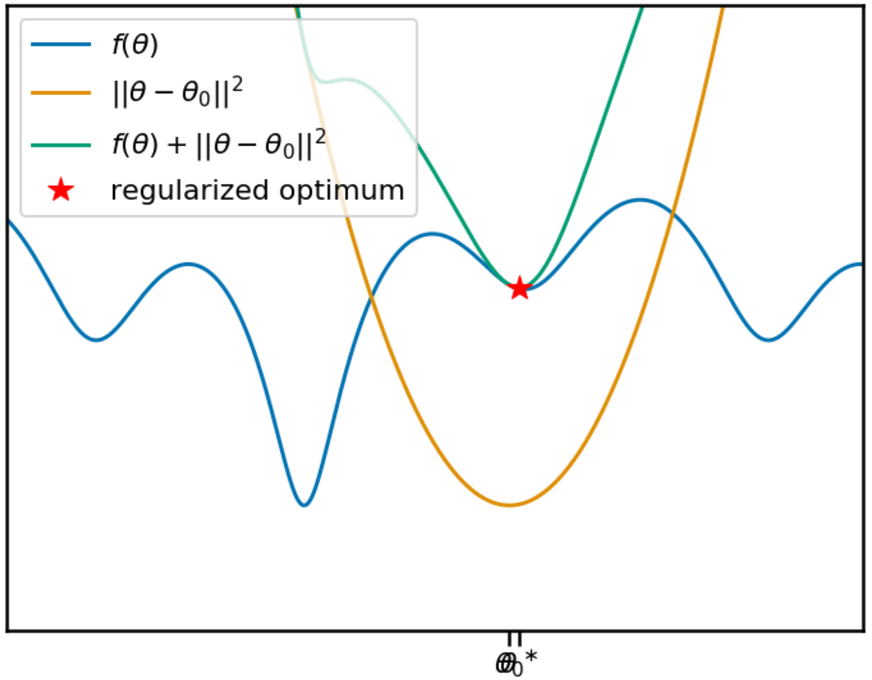
Bi-Level optimization interpretation

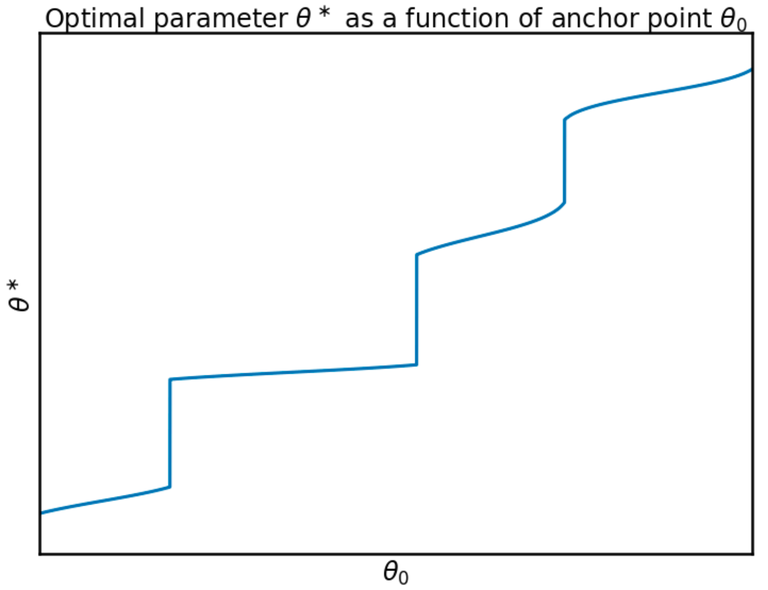
$$\mathcal{L}_i(\phi) \equiv \mathcal{L}_i(\phi, \mathcal{D}_i^{test})$$

$$\hat{\mathcal{L}}_i(\varphi) \equiv \mathcal{L}_i(\phi, \mathcal{D}_i^{train})$$

$$\min_{\theta} f(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i(A_i^*(\theta))$$

$$A_i^*(\theta) = \arg\min_{\phi} G_i(\varphi, \theta) = \hat{\mathcal{L}}_i(\varphi) + \frac{\lambda}{2} \|\phi - \theta\|_2^2$$





Implicit Function Theorem

$$\underset{\lambda}{\operatorname{argmin}} \; \mathcal{L}_V\left(\underset{\theta}{\operatorname{argmin}} \; \mathcal{L}_T(\theta, \lambda)\right)$$

λ: hyper parameters

$\mathcal{L}_V$: validation Loss

θ: parameters

$\mathcal{L}_T$: training Loss

$$\theta^*(\lambda) = \arg\min_{\theta} f(\theta, \lambda)$$

$$\left.\frac{\partial \theta^*}{\partial \lambda}\right|_{\lambda_0} = -\left[\frac{\partial^2 \mathcal{L}_T}{\partial \theta \partial \theta}\right]^{-1} \left.\frac{\partial^2 \mathcal{L}_T}{\partial \theta \partial \lambda}\right|_{\lambda_0, \theta^*(\lambda_0)}$$

I in : MAML

since $\frac{1}{2}\|\theta - \lambda\|^2$

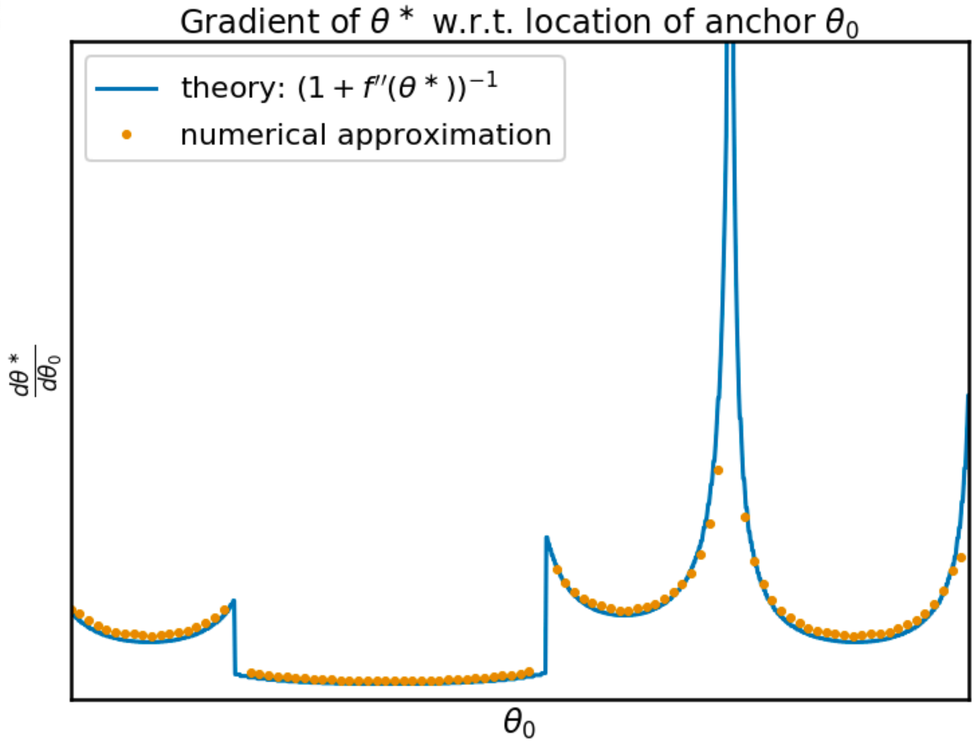
Lorraine, Jonathan, Paul Vicol, and David Duvenaud. "**Optimizing Millions of Hyperparameters by Implicit Differentiation**." *arXiv preprint arXiv:1911.02590* (2019)

https://www.inference.vc/meta-learning-with-the-implicit-function-theorem/

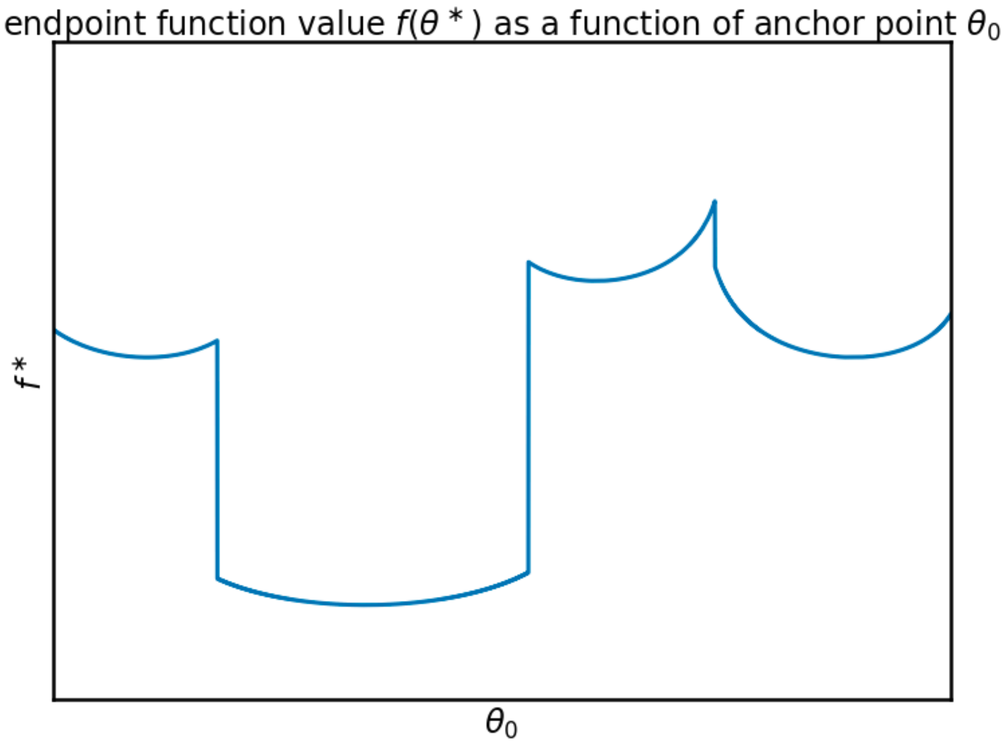
Implicit Gradients

$$\frac{d\theta^*}{d\theta^0} = \frac{1}{1 + f''(\theta^*)}$$

closed form



Optimizing the meta-objective



$f(\theta^*(\theta_0))$

ugly

We optimize

$\sum_i f_i(\theta_i^*(\theta_0))$

meta-objective

More about Loss ⇒

Garipov, Timur, et al. "Loss surfaces, mode connectivity, and fast ensembling of dnns." *Advances in Neural Information Processing Systems*. 2018.

iMAML

Implicit function theorem

Let $\phi_i^* := \mathcal{A}_i^*(\theta)$

$$\nabla_\theta \mathcal{L}_i(\theta) = \left(I + \frac{l}{\mu} \nabla_\phi^2 \hat{\mathcal{L}}_i(\phi_i^*)\right)^{-1} \nabla_\phi \mathcal{L}_i(\phi_i^*)$$

Compare to previous slide

$\Rightarrow$ Gradients depends only on result of $\mathcal{A}$ and not the path.

iMAML

Practical algorithm

- Solver inner optimization approximately to find
  - $\| \phi_i - \phi_i^* \| \leq \delta$
- Approximately find meta-gradient using conjugate gradient algorithm that requires only Hessian-vector products to get
  - $\| g_i - \left(I + \left(\frac{1}{\lambda}\right) \nabla_\phi^2 \hat{\mathcal{L}}_i(\phi_i) \right)^{-1} \nabla_\phi \mathcal{L}_i(\phi_i) \| \leq \delta$

Theorem : If $\ell_i(\varphi, \theta)$ is strongly convex in ϕ, for above algorithm, we have bounded error

$$\| g_i - \nabla_\theta \mathcal{L}_i(\theta) \| \leq O(\delta)$$

Vanilla vs FO vs implicit MAML

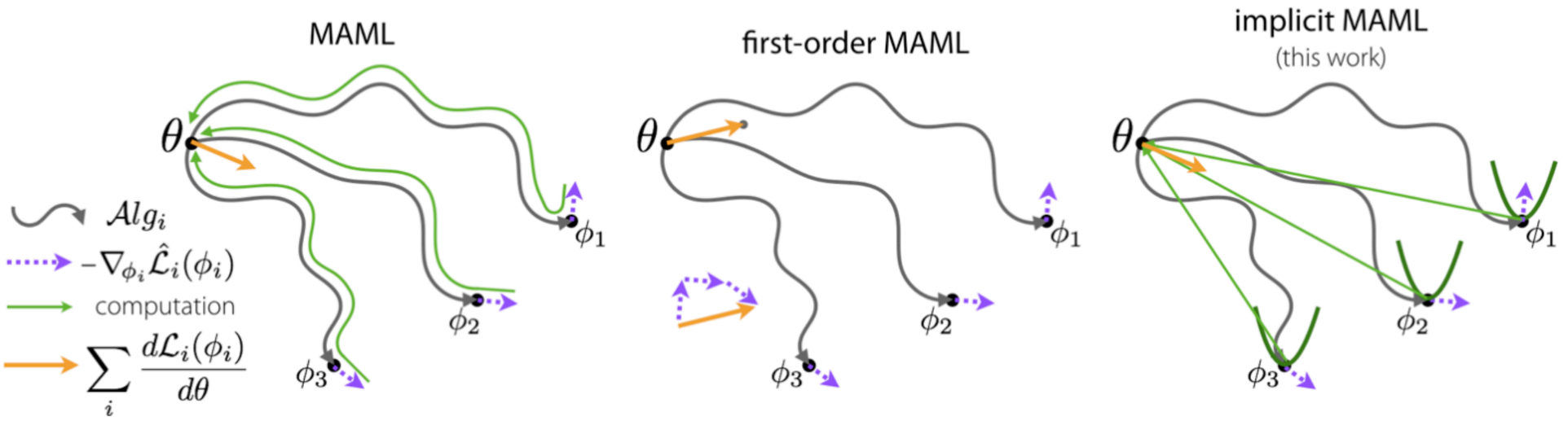


Figure 1: To compute the meta-gradient $\sum_i \frac{d\mathcal{L}_i(\phi_i)}{d\theta}$, the MAML algorithm differentiates through the optimization path, as shown in green, while first-order MAML computes the meta-gradient by approximating $\frac{d\boldsymbol{\phi}_i}{d\boldsymbol{\theta}}$ as $\boldsymbol{I}$. Our implicit MAML approach derives an analytic expression for the exact meta-gradient without differentiating through the optimization path by estimating local curvature.

iMAML

Complexity

Table: *Compute and memory complexity. D = diameter, κ = condition number of inner level. † compares with $\mathscr{A}$, while * compares with $\mathscr{A}^*$*

Algorithm	Compute	Memory	Error
MAML (GD + full back-prop)	$\kappa \log\left(\frac{D}{\delta}\right)$	$\text{Mem}(\nabla\hat{\mathcal{L}}_i) \cdot \kappa \log\left(\frac{D}{\delta}\right)$	$0^\dagger$
MAML (Nesterov's AGD + full back-prop)	$\sqrt{\kappa} \log\left(\frac{D}{\delta}\right)$	$\text{Mem}(\nabla\hat{\mathcal{L}}_i) \cdot \sqrt{\kappa} \log\left(\frac{D}{\delta}\right)$	$0^\dagger$
Truncated back-prop (GD) [2]	$\kappa \log\left(\frac{D}{\delta}\right)$	$\text{Mem}(\nabla\hat{\mathcal{L}}_i) \cdot \kappa \log\left(\frac{1}{\epsilon}\right)$	$\epsilon^\dagger$
Implicit MAML (this work)	$\sqrt{\kappa} \log\left(\frac{D}{\delta}\right)$	$\text{Mem}(\nabla\hat{\mathcal{L}}_i)$	δ^*

iMAML

Complexity

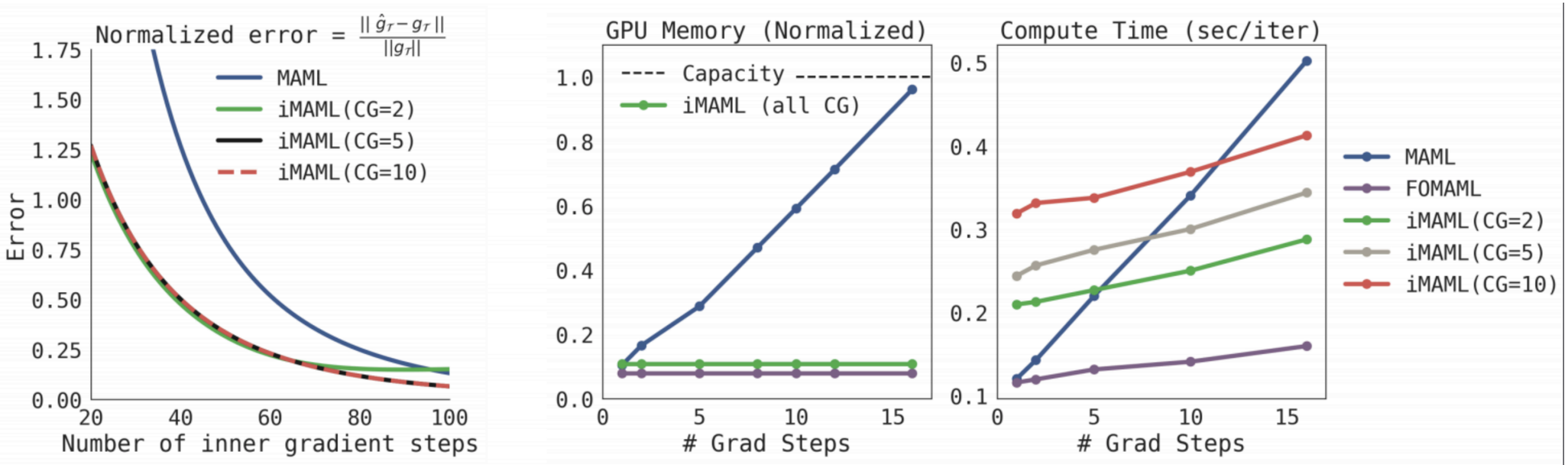


Figure: *(left) MAML and iMAML computation vs exact meta-gradient on a synthetic example. (right) Compute and memory on 20-way-5-shot Omniglot*

iMAML

Omniglot

Table: *Comparison of algorithms on Omniglot. Gradient descent (GD) and Hessian-Free (w/ line-search) algorithms considered for $\mathscr{A}$. $\lambda = 2.0$ and CG=5*

Algorithm	5-way 1-shot	5-way 5-shot	20-way 1-shot	20-way 5-shot
MAML [15]	$98.7 \pm 0.4\%$	$\mathbf{99.9 \pm 0.1\%}$	$95.8 \pm 0.3\%$	$98.9 \pm 0.2\%$
first-order MAML [15]	$98.3 \pm 0.5\%$	$99.2 \pm 0.2\%$	$89.4 \pm 0.5\%$	$97.9 \pm 0.1\%$
Reptile [43]	$97.68 \pm 0.04\%$	$99.48 \pm 0.06\%$	$89.43 \pm 0.14\%$	$97.12 \pm 0.32\%$
iMAML, GD (ours)	$99.16 \pm 0.35\%$	$99.67 \pm 0.12\%$	$94.46 \pm 0.42\%$	$98.69 \pm 0.1\%$
iMAML, Hessian-Free (ours)	$\mathbf{99.50 \pm 0.26\%}$	$99.74 \pm 0.11\%$	$\mathbf{96.18 \pm 0.36\%}$	$\mathbf{99.14 \pm 0.1\%}$

iMAML

Summary

- No vanishing meta-gradients due to regularization
- Meta-Gradient depends only on final result of algorithm not path
- Wider class of algorithms is supported by implicit MAML
- Efficient, Convergent, Gains
- FOMAML and Reptile (CG=0) of iMAML

- However, iMAML just one piece of whole cake:
  - Chen, Yutian, et al. "Modular meta-learning with shrinkage." arXiv preprint arXiv:1909.05557 (2019).
  - Lorraine, Jonathan, Paul Vicol, and David Duvenaud. "Optimizing Millions of Hyperparameters by Implicit Differentiation." arXiv preprint arXiv:1911.02590 (2019)

Stochasticity

$$\mathcal{M}_{\text{stochastic}}(\theta) = \sum_i \mathbb{E}_{\theta \sim \text{Alg}(f_i, \theta_0)} g_i(\theta)$$

Bayesian interpretation of iMAML

$$\mathcal{M}_{\text{variational}}(\theta_0, Q_i) = \sum_i \left(\underbrace{KL\left[Q_i \,|\, N_{\theta_0} \right]}_{\Updownarrow} + \mathbb{E}_{\theta \sim Q_i} f_i(\theta) \right)$$

$$\Updownarrow$$

$$\| \theta_i - \theta_0 \|^2$$

$$\Updownarrow$$

in MAML $\quad \| Alg(f_i, \theta_0) - \theta_0 \|^2$

ANIL

To be continued...