THE OHIO STATE UNIVERSITY

Model-Agnostic Meta-Learning from Optimization Perspective

Yingbin Liang

The Ohio State University

TDAI Foundations CoP Deep Learning Summer School

What is Meta-Learning ?

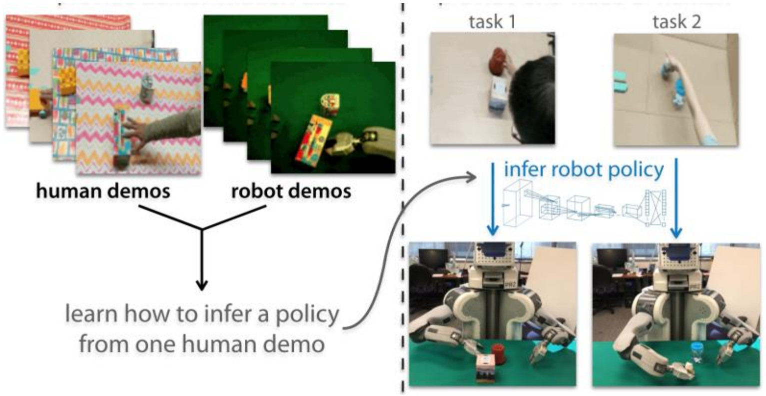


Image source: Yu & Finn, blog, 2018

- A conventional machine learning model often
  1. requires a large number of samples.
  2. needs a long training process.

- Humans, however, learn new concepts much faster.
  1. Kids who have seen trees and flowers only a few times quickly tell them apart.
  2. People who know how to ride bike learn motorcycle fast with little demonstration.
  3. People quickly judge fruit quality based on previous purchase experiences.

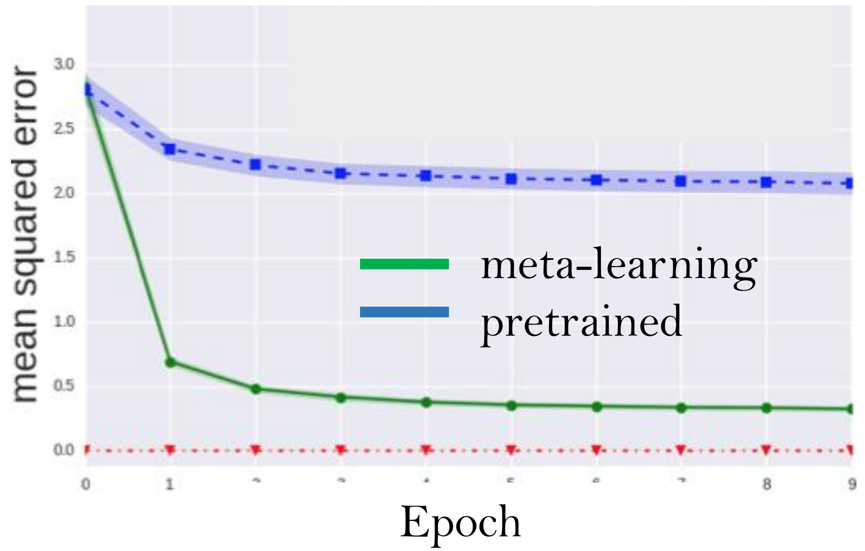
- ***Question:***

Is it possible to have such a learning paradigm which learns new concepts fast with only **a few** data samples?

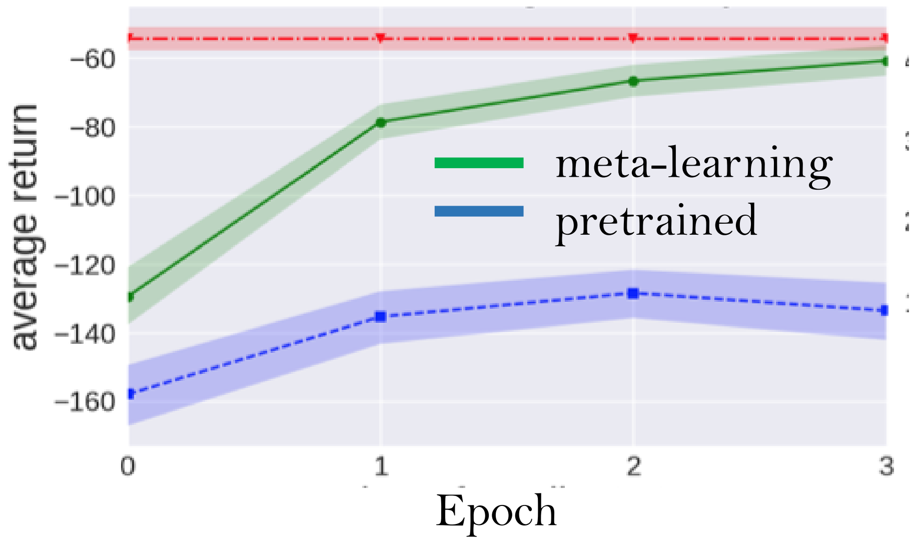
Meta-Learning (Learning to Learn)

- Goal of meta-learning:
  1. Extract prior information or similarity of **observed** learning tasks
  2. Use such information for learning a **new** task (adaptation) with a few samples
     - ➢ Fast training with better prediction accuracy
- It works? (Finn et al, 2017)

Supervised classification



Reinforcement learning



Applications of Meta-Learning:

- Supervised few-shot classification

A classifier trained on cat-bird, flower-bike images (training tasks) can quickly classify a given dog or otter image after seeing a small number of dog-otter pictures (test period).

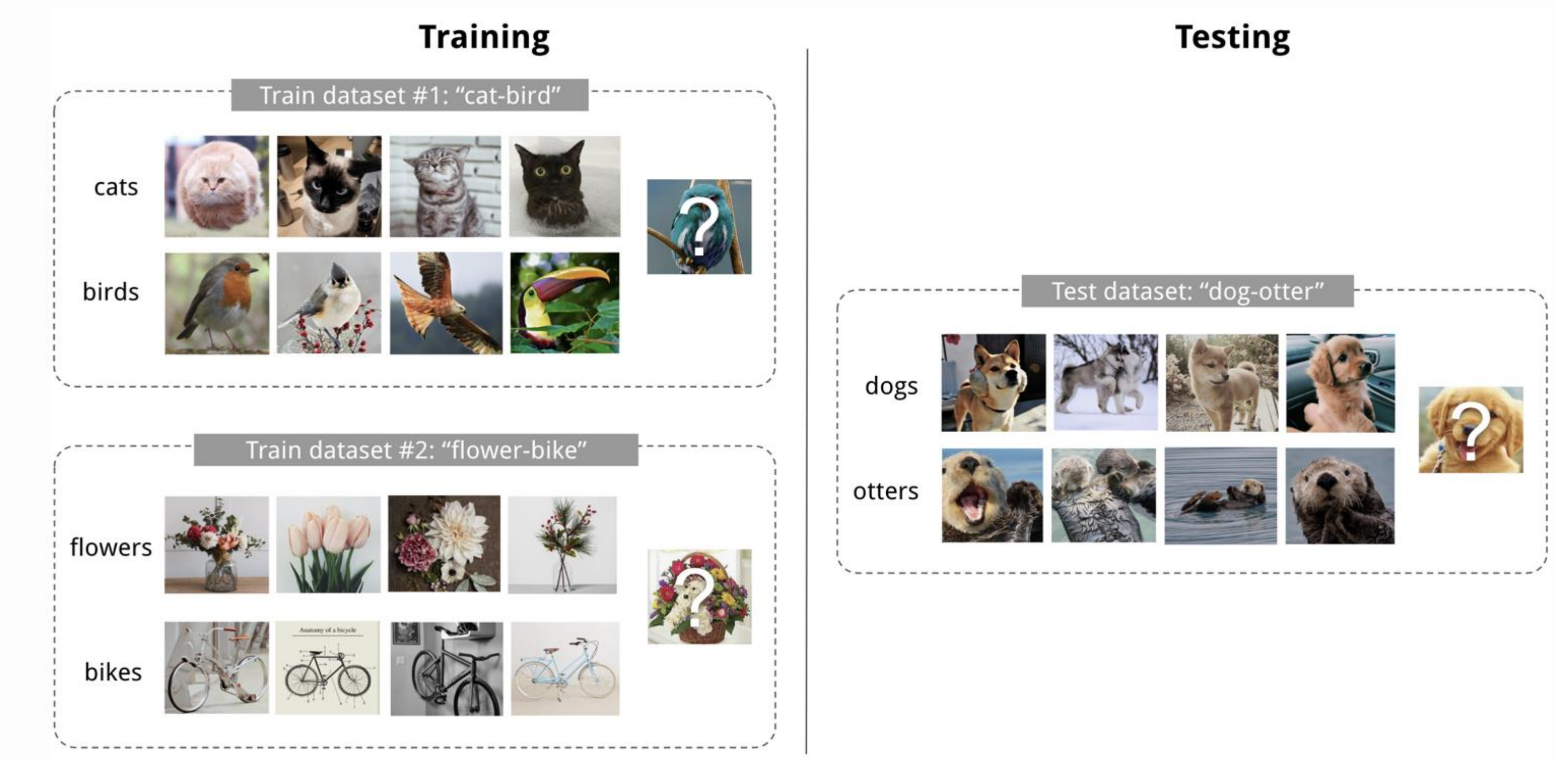


Fig. 1. An example of 4-shot 2-class image classification. (Image thumbnails are from Pinterest)

Applications of Meta-Learning:

- Reinforcement learning

An agent trained on flat surface environment can quickly finish task on a different environment, e.g., uphill surface, during the test.

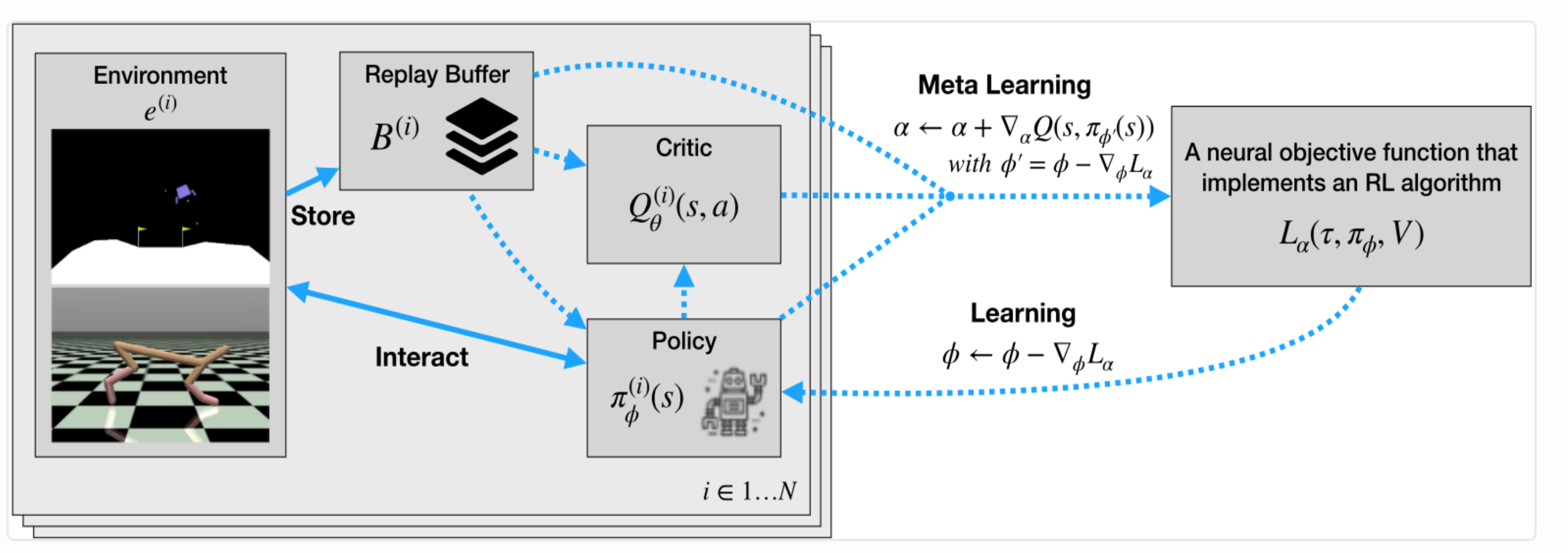


Image source: Louis Kirsch et al., 2020

Meta-Learning Approaches:

- Metric-learning based approach
  - ➢ Prototypical networks
  - ➢ Matching networks

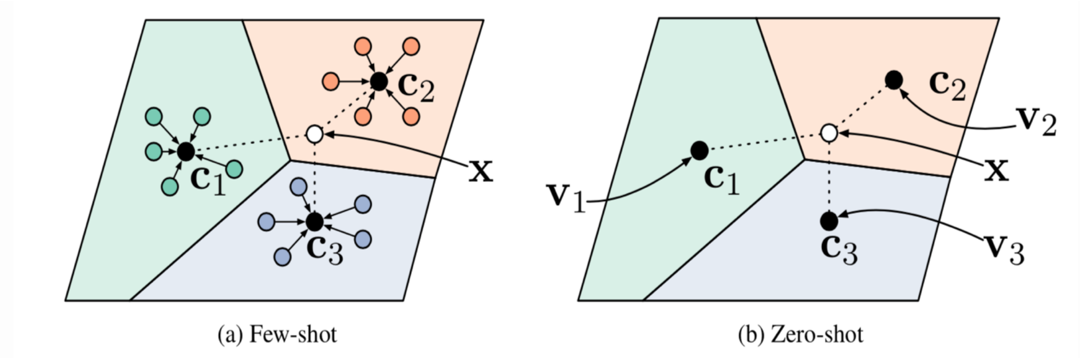


Image source: Snell et al., 2017

- Model-based approach
  - ➢ Memory-augmented neural networks
  - ➢ Meta networks

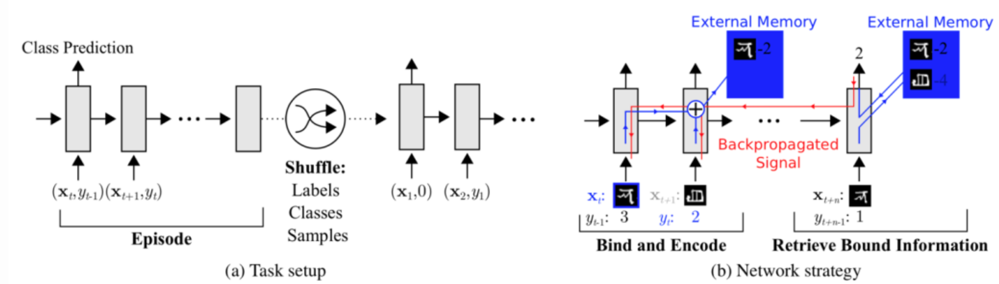


Image source: Santoro et al., 2016

- Optimization-based approach
  - ➢ Model-agnostic meta-learning (MAML)
  - ➢ Almost no inner loop (ANIL)
  - ➢ Bilevel meta-learning

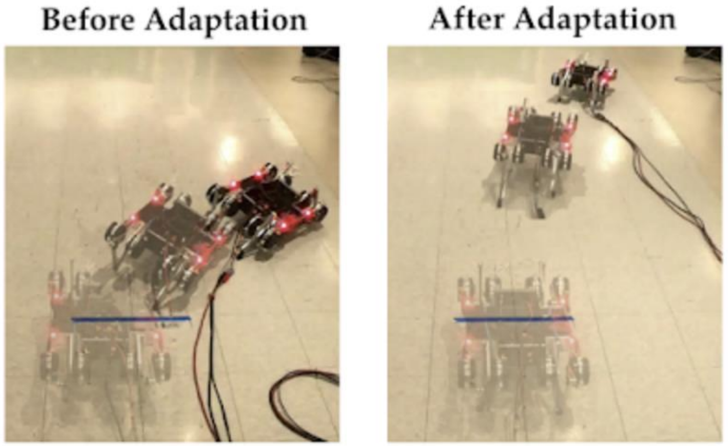
✓ Simpler
✓ Efficient
✓ Effective

Model-Agnostic Meta-Learning (MAML)

(Finn et al, 2017)   A pioneering optimization-based method.

Generalizability: applicable to any model trained with gradients

- Recommendation
- Computer vision
- Meta-reinforcement learning
- Data mining
- Personalized federate learning
- …………



Reinforcement learning, Google AI

As most essential framework:

- Inspire numerous follow-up variants

How MAML Works

- ✓ Based on observed tasks, find good initial w;
- ✓ For a new task, starting from w, a **small** number of gradient steps find good parameter.

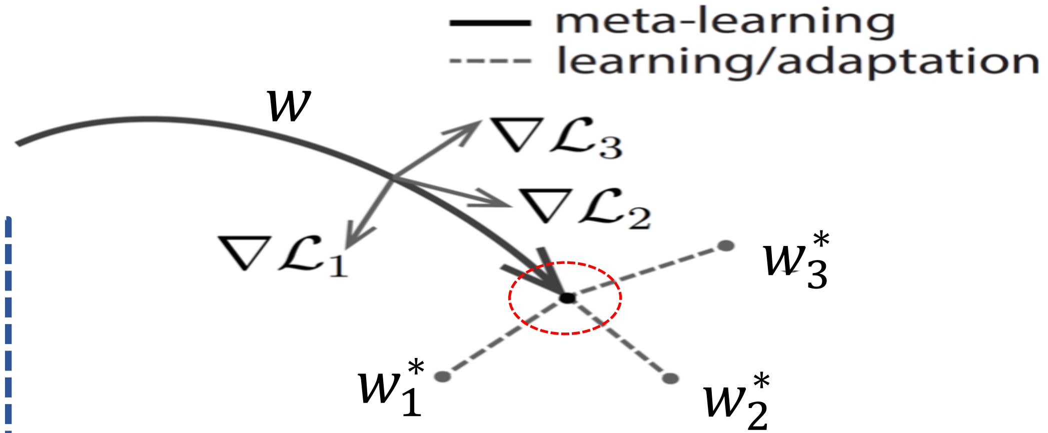


Diagram of the MAML approach.

Image sources: Finn et al, 2017

Inner loop: task-specific adaptation

- Each task i:

Loss in expectation: $l_i(w) = \mathbb{E}_{\tau \sim \mathcal{D}} L_i(w, \tau)$

Same initial w:

$$w_0^i = w \text{ for all tasks } i$$

For $j = 0, \ldots, N-1$

$$w_{j+1}^i = w_j^i - \alpha \nabla l_i(w_j^i)$$

Outer loop: meta objective

- Loss averaged over all tasks:

$$\min_{w \in \mathbb{R}^d} \mathcal{L}(w) := \mathbb{E}_{i \sim p_T}\, l_i(w_N^i)$$

where p_T is task distribution

- w_N^i depends on w

$$\nabla \mathcal{L}(w) = \mathbb{E}_{i \sim p_T} \prod_{j=0}^{N-1} (I - \alpha \nabla^2 l_i(w_j^i)) \nabla l_i(w_N^i)$$

MAML Algorithm

Data sampling required!

Inner loop: task-specific adaptation

- Each task i:

For $j = 0, \ldots, N-1$

Draw samples S_i

$$w_{j+1}^i = w_j^i - \alpha \nabla l_i(w_j^i; S_i)$$

$\nabla L_i(w_j^i; S_i)$: unbiased estimate of $\nabla l_i(w_j^i)$

➢ RL example: REINFORCE (Williams, 1992)

Outer loop: model updates

- Meta-gradient estimate:

$$\widehat{G}_i = \prod_{j=0}^{N-1} \left(I - \alpha \nabla^2 L_i(w_j^i; D_j^i)\right) \nabla L_i(w_N^i; T_i)$$

➢ T_i, D_j^i: samples at outer loop

- Model update with sampled tasks

$$w \leftarrow w - \frac{\beta}{B} \sum_{i \in \mathcal{B}} \widehat{G}_i$$

Convergence for MAML

Theorem (Ji, Yang, Liang, JMLR 2021)

- If inner stepsize $\alpha \leq \Theta(\frac{1}{NL})$,

(Guarantee to stationary point:) $\mathbb{E}\|\nabla\mathcal{L}(w)\| \leq \mathcal{O}\Big(\sqrt{\frac{1}{K} + \frac{1}{S} + \frac{1}{B} + \frac{1}{TB}}\Big).$

N: # of inner-loop steps; *K*: # of out-loop steps; *S*: Inner-loop batch size;
B: outer-loop task batch size; *T*: outer-loop sample batch size

- To achieve ϵ-accurate stationary point, it requires

Gradient computations: $\mathcal{O}(\frac{N}{\epsilon^4} + \frac{1}{\epsilon^2})$

Hessian-vector computations: $\mathcal{O}\big(\frac{N}{\epsilon^2}\big)$

- Inner stepsize should not be too large
- Error reduces when increasing batch sizes

$\Big\}$ Empirically verified in **Antonio et al., 2019**

Almost No Inner Loop (ANIL)

(Raghu et al., 2019)   A simple and efficient variant of MAML

Interesting findings in Raghu et al., 2019:

- ❑ During inner-loop adaptation, several layers of MAML model nearly do not change

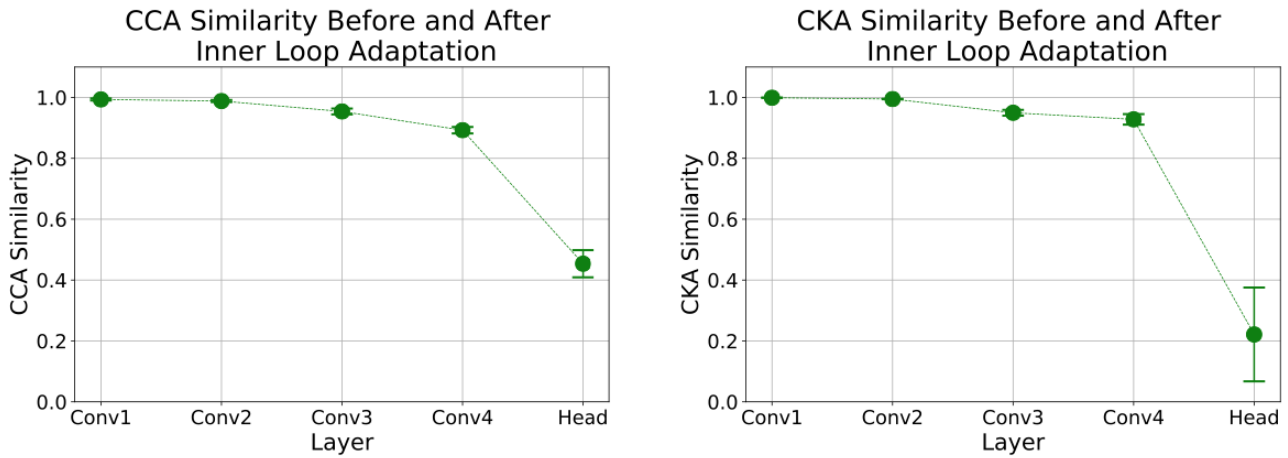


Image source: Raghu et al., 2020

This motivates ANIL as an efficient simplification of MAML by

- ❑ Adapting or updating only partial parameters (e.g., head of NN) in inner loop

ANIL Training

Each task i's parameters split into $w^i = (u^i, \phi)$

- ϕ: common parameters shared by all tasks

Task adaptation on partial parameters u^i

- Each task i:

  For $j = 0, \ldots, N-1$

  $$u^i_{j+1} = u^i_j - \alpha \nabla_u l_{S_i}(u^i_j, \phi)$$

- Same initial u:

  $u^i_0 = u$ for all tasks i

MAML: adapt all parameters w^i

Meta optimization:

Loss averaged over all tasks:

$$\min_{u,\phi} \mathcal{L}(u, \phi) := \mathbb{E}_{i \sim p_T} \, l_{T_i}(u^i_N, \phi)$$

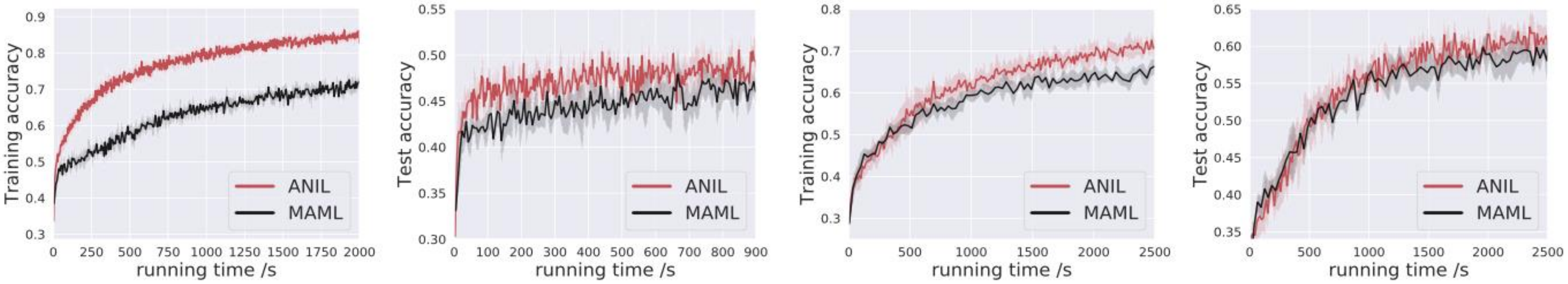
MAML:

$$\min_{w \in \mathbb{R}^d} \mathcal{L}(w) := \mathbb{E}_{i \sim p_T} \, l_{T_i}(w^i_N)$$

Importance of ANIL

- Converge much faster than MAML (Ji, Lee, Liang, Poor NeurIPS 2020):

Backbone: 4-layer CNN



(a) dataset: FC100

(b) dataset: miniImageNet

Loss Geometries of ANIL Training

Geometries of inner-loop loss $l_{S_i}(u, \phi)$:

- ❑ **Strongly-convex** w.r.t. u
  - ➢ u : head (last layer) of neural networks (e.g., last row of table)
- ❑ **Nonconvex** w.r.t. u
  - ➢ u : more than one layers (e.g., first 4 rows of table)

Different geometries lead to different convergence behaviors

Strongly-Convex ANIL

Theorem (Ji. et al., NeurIPS 2020)

Convergence rate: (N is number of inner-loop steps)

$$\text{(Rate w.r.t. } u) \quad \mathbb{E}\left\|\frac{\partial L(u,\phi)}{\partial u}\right\|^2 \leq \mathcal{O}\left(\frac{(1-\kappa^2)^{\frac{N}{2}}}{K} + \frac{(1-\kappa^2)^{\frac{N}{2}}}{B}\right),$$

$$\text{(Rate w.r.t. } \phi) \quad \mathbb{E}\left\|\frac{\partial L(u,\phi)}{\partial \phi}\right\|^2 \leq \mathcal{O}\left(\frac{(1-\kappa^2)^{\frac{N}{2}} + b_1}{K} + \frac{(1-\kappa^2)^{\frac{3N}{2}} + b_2}{B}\right)$$

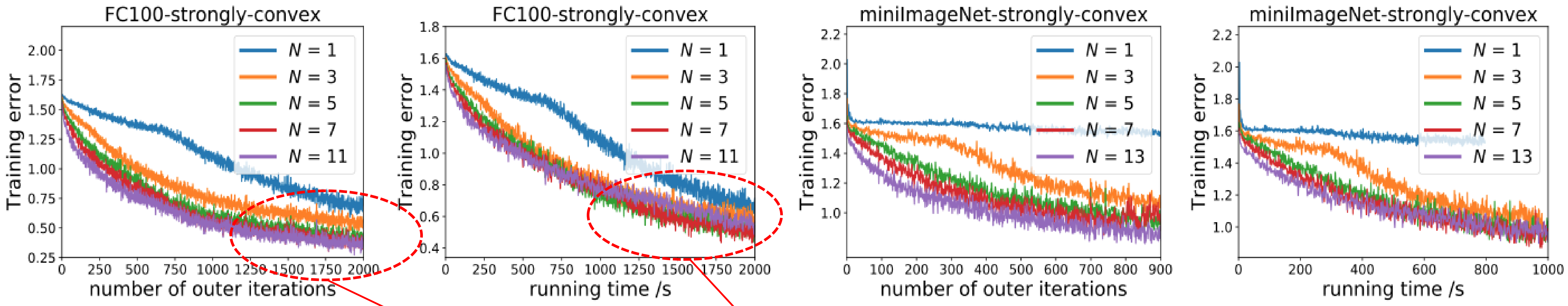
Computational complexity : $\mathcal{O}(CN(1-\kappa^2)^{N/2} + N)\epsilon^{-2}$, constant $C \gg 1$

As N increaes:
- Convergence rate first increases, then saturates
- Complexity first decreases, then increases

Training guideline: choose a moderate but not too large N

Experimental Verification

- Few-shot meta-learning on FC100 and miniImageNet:



(a) dataset: FC100

(b) dataset: miniImageNet

As N increaes:

- ➢ Rate w.r.t. iterations (left plot) first increases, then saturates
- ➢ Rate w.r.t. running time (right plot) first increases, then decreases

Nonconvex ANIL

Theorem (Ji. et al., NeurIPS 2020)

Convergence rate:

$$\text{(Rate w.r.t. } u) \quad \mathbb{E}\left\|\frac{\partial L(u,\phi)}{\partial u}\right\|^2 \leq \mathcal{O}\left(\frac{N}{K}+\frac{N}{B}\right),$$

$$\text{(Rate w.r.t. } \phi) \quad \mathbb{E}\left\|\frac{\partial L(u,\phi)}{\partial \phi}\right\|^2 \leq \mathcal{O}\left(\frac{N}{K}+\frac{N}{B}\right).$$

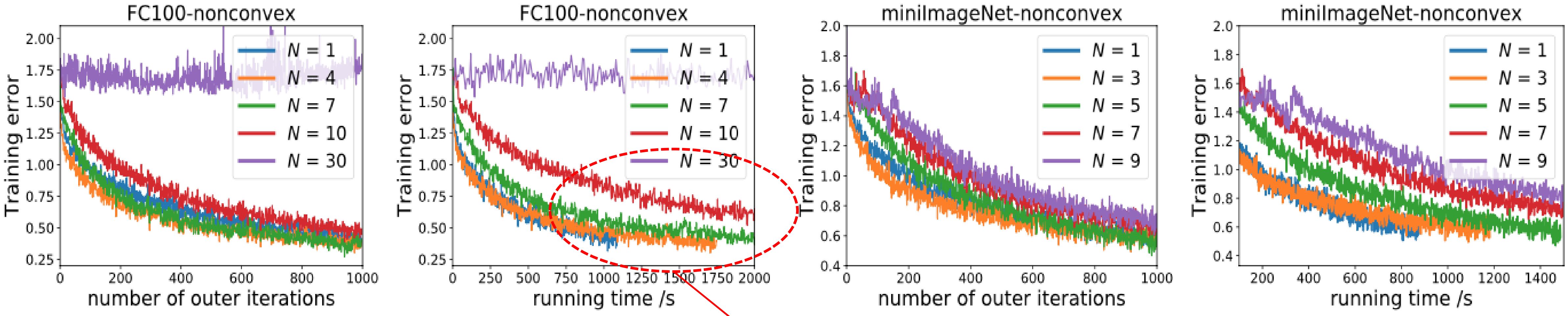
Computational complexity : $\mathcal{O}(N^2\epsilon^{-2})$

As N increases: (opposite to strongly-convex ANIL)

Convergence rate and complexity both become worse

Experimental Verification

- Few-shot meta-learning:



(a) dataset: FC100 (b) dataset: miniImageNet

➢ Rate w.r.t. iterations & running time become worse when N increases

Training guideline: choose a small N

Bilevel Meta-Learning with Shared Embedding

- ϕ: parameters of feature embedding model

$$\min_{\phi} \mathcal{L}(\phi) = \frac{1}{B} \sum_{i \in \mathcal{B}} l(u_i^*, \phi; T_i)$$

For each task i: $u_i^* = \arg\min_{u} l(u, \phi; S_i)$

In contrast to MAML and ANIL:

- ❑ No common initial u to train
- ❑ Only updates embedding model parameters ϕ

- Generic bilevel optimization: $\min_{x \in \mathbb{R}^p} \Phi(x) := f(x, y^*(x)) \quad \text{s.t.} \quad y^*(x) = \arg\min_{y \in \mathbb{R}^q} g(x, y).$

- Challenges of bilevel methods: hypergradient computation

$$\nabla_\phi f(\phi, u^*) = \nabla_\phi f(\phi, u^*) - \nabla_\phi \nabla_u g(\phi, u^*) \underbrace{(\nabla_u^2 g(\phi, u^*))^{-1} \nabla_u f}(\phi, u^*)$$

Two efficient approximations:

- ❑ 1. approximated implicit differentiation (AID). 2. iterative differentiation (ITD)

Evolution Strategies (ES) for Bilevel Optimization

- Why compute second-order information

$$\nabla_\phi f(\phi, u^*) = \nabla_\phi f(\phi, u^*) - \underbrace{\nabla_\phi \nabla_u g(\phi, u^*)(\nabla_u^2 g(\phi, u^*))^{-1}}_{\mathcal{J}_*(\phi) = \frac{\partial u^*(\phi)}{\partial \phi}\text{ : Response Jacobian matrix}} \nabla_u f(\phi, u^*)$$

- Our ES-based Jacobian (ESJ) estimator $\mathcal{J}_*(\phi) = \frac{\partial u^*(\phi)}{\partial \phi}$

$$\hat{\mathcal{J}}_N(\phi; z) = \frac{u^N(\phi + \beta z) - u^N(\phi)}{\beta} z^\top$$

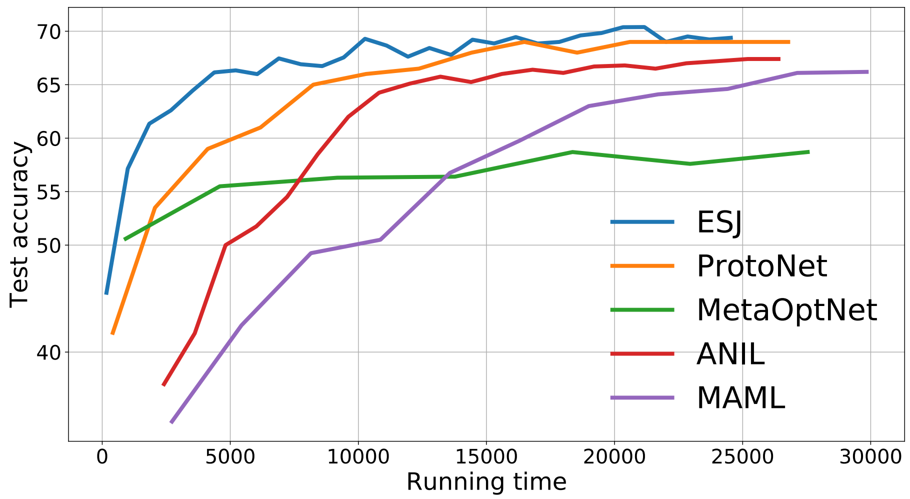
  - $u^N(\phi)$: output of N gradient descent steps of inner problem

ESJ: Bilevel Optimizer with ES-based Jacobians

- Features

  - ❑ No second-order information involved (Hessian-free)
  - ❑ With convergence guarantee: $\mathcal{O}(\frac{p}{\epsilon^2}\log\frac{1}{\epsilon})$
  - ❑ A **stochastic** version also provided for large datasets

- Scalability to deep models (ResNet-12, 12 millions parameters)



Algorithm	67%	69%
ESJ	**2.0**	**2.8**
MetaOptNet	20+	20+
ProtNet	3.5	4.4
ANIL	6.3	-
MAML	9.7	-

Conclusion & Takeaways

- Convergence Theory for General Multi-Step MAML
  - Convergence & complexity analysis for MAML framework
  - A general tool for analyzing other meta-learning algorithms
- Convergence Theory for ANIL
  - Theoretical guideline on parameter selection
  - Characterization of impact of loss geometries on convergence behaviors
- Bilevel Optimization for Meta-Learning
  - Faster and scalable bilevel optimizers with higher efficiency & improved performance guarantee

Future Directions

- Scalable and efficient meta-learning
  - Accelerate Hessian-free meta-learning algorithms
- Distributed meta-learning for edge networks
- Meta-learning for other areas
  - Design better “learning to optimization” (L2O) algorithms
  - Meta-learning application in offline reinforcement learning
  - Design fast hyperparameter optimization algorithms

References

1. K. Ji, J. Yang, Y. Liang. "Theoretical convergence of multi-step model-agnostic meta-learning", JMLR 2021.
2. K. Ji, J. D. Lee, Y. Liang, H. V. Poor. "Convergence of meta-learning with task-specific adaptation over partial parameters", NeurIPS 2020.
3. K. Ji, J. Yang, Y. Liang. "Bilevel optimization: Convergence analysis and enhanced design", ICML 2021
4. J. Yang, K. Ji, Y. Liang. "Provably faster algorithms for bilevel optimization", NeurIPS 2021, **spotlight**.
5. K. Ji, Y. Liang. "Lower bounds and accelerated algorithms for bilevel optimization." submitted for publication.
6. D. Sow, K. Ji, Y. Liang. "ES-based Jacobian enables faster bilevel optimization." submitted for publication.

Acknowledgement

Joint work with Dr. Kaiyi Ji (U. Michigan), Junjie Yang (OSU), Davis Sow (OSU), Dr. Jason Lee (Princeton), and Dr. Vincent Poor (Princeton)

Questions!