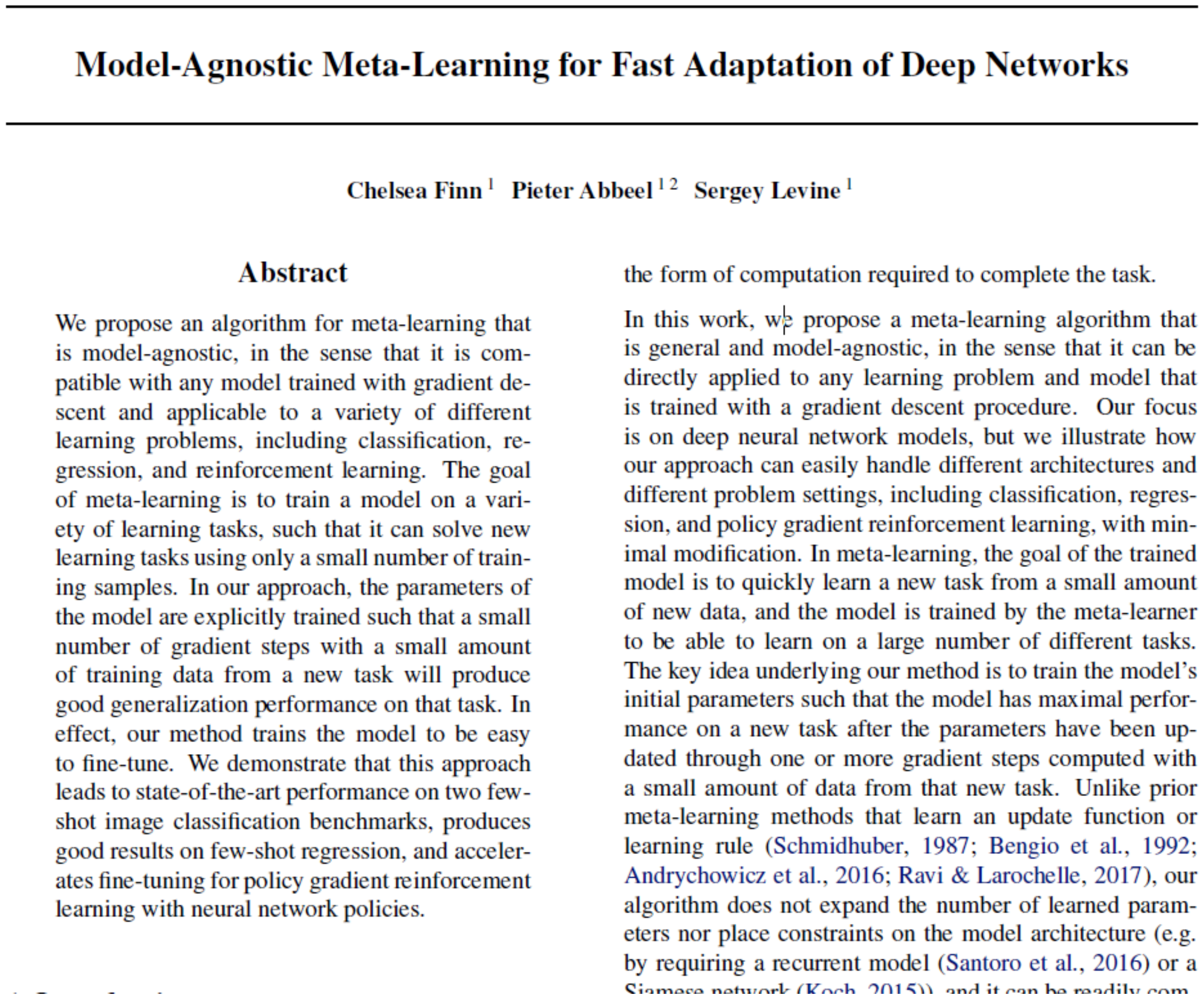
Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Charles

About the paper

- Published on ACM in 2017.
- Cited by 7478 as of today.



Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Chelsea Finn[1] Pieter Abbeel[1 2] Sergey Levine[1]

Abstract

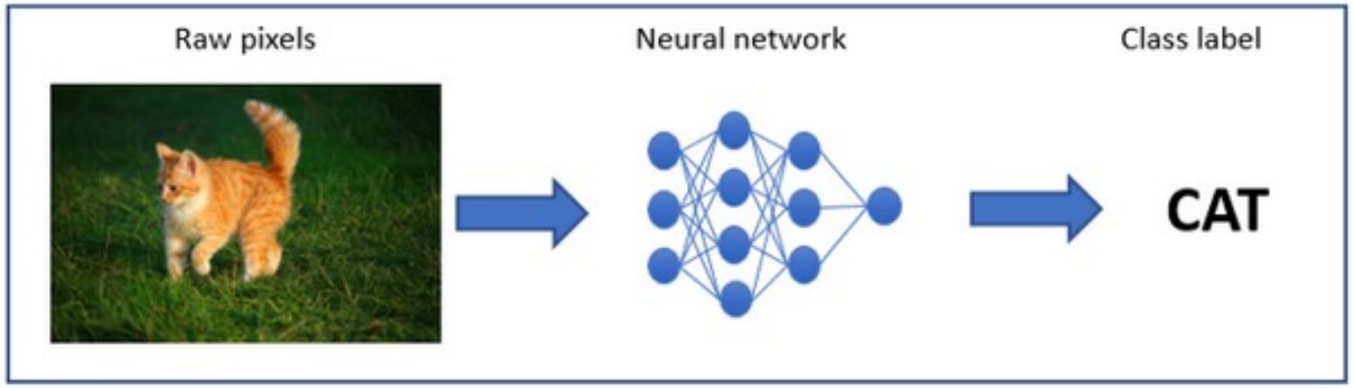
We propose an algorithm for meta-learning that is model-agnostic, in the sense that it is compatible with any model trained with gradient descent and applicable to a variety of different learning problems, including classification, regression, and reinforcement learning. The goal of meta-learning is to train a model on a variety of learning tasks, such that it can solve new learning tasks using only a small number of training samples. In our approach, the parameters of the model are explicitly trained such that a small number of gradient steps with a small amount of training data from a new task will produce good generalization performance on that task. In effect, our method trains the model to be easy to fine-tune. We demonstrate that this approach leads to state-of-the-art performance on two few-shot image classification benchmarks, produces good results on few-shot regression, and accelerates fine-tuning for policy gradient reinforcement learning with neural network policies.

the form of computation required to complete the task.

In this work, we propose a meta-learning algorithm that is general and model-agnostic, in the sense that it can be directly applied to any learning problem and model that is trained with a gradient descent procedure. Our focus is on deep neural network models, but we illustrate how our approach can easily handle different architectures and different problem settings, including classification, regression, and policy gradient reinforcement learning, with minimal modification. In meta-learning, the goal of the trained model is to quickly learn a new task from a small amount of new data, and the model is trained by the meta-learner to be able to learn on a large number of different tasks. The key idea underlying our method is to train the model's initial parameters such that the model has maximal performance on a new task after the parameters have been updated through one or more gradient steps computed with a small amount of data from that new task. Unlike prior meta-learning methods that learn an update function or learning rule (Schmidhuber, 1987; Bengio et al., 1992; Andrychowicz et al., 2016; Ravi & Larochelle, 2017), our algorithm does not expand the number of learned parameters nor place constraints on the model architecture (e.g. by requiring a recurrent model (Santoro et al., 2016) or a Siamese network (Koch, 2015)), and it can be readily com-

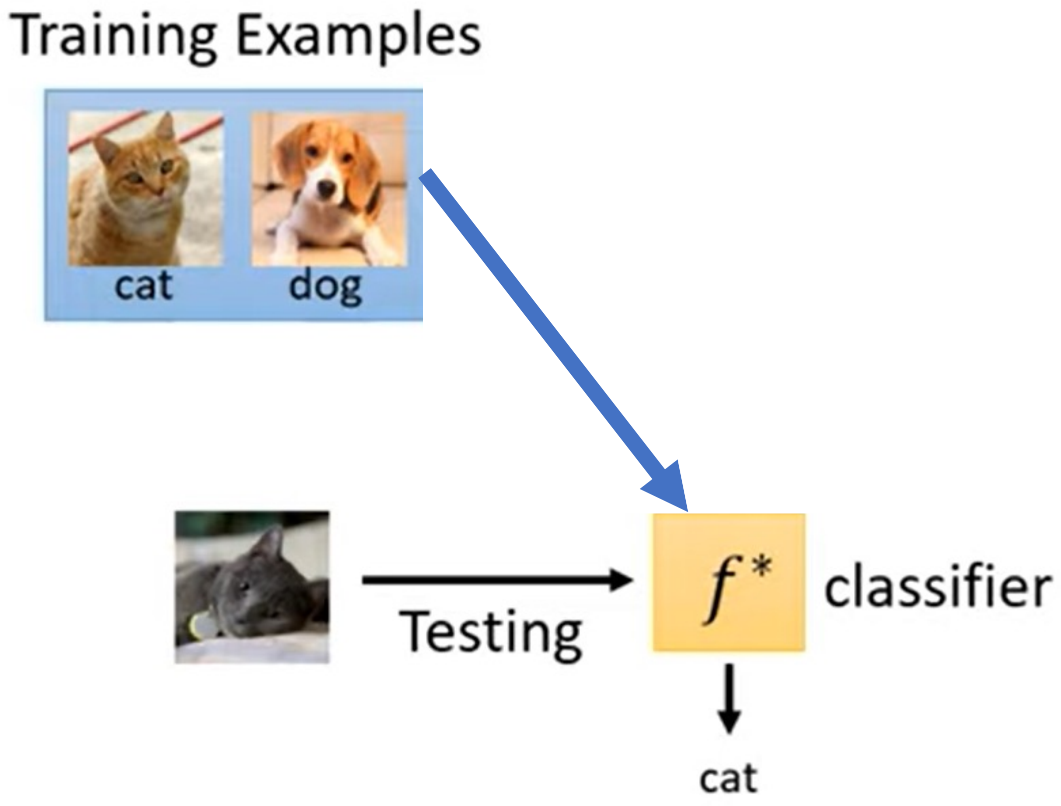
Meta Learning

- Learning to learn
- Machine learning :

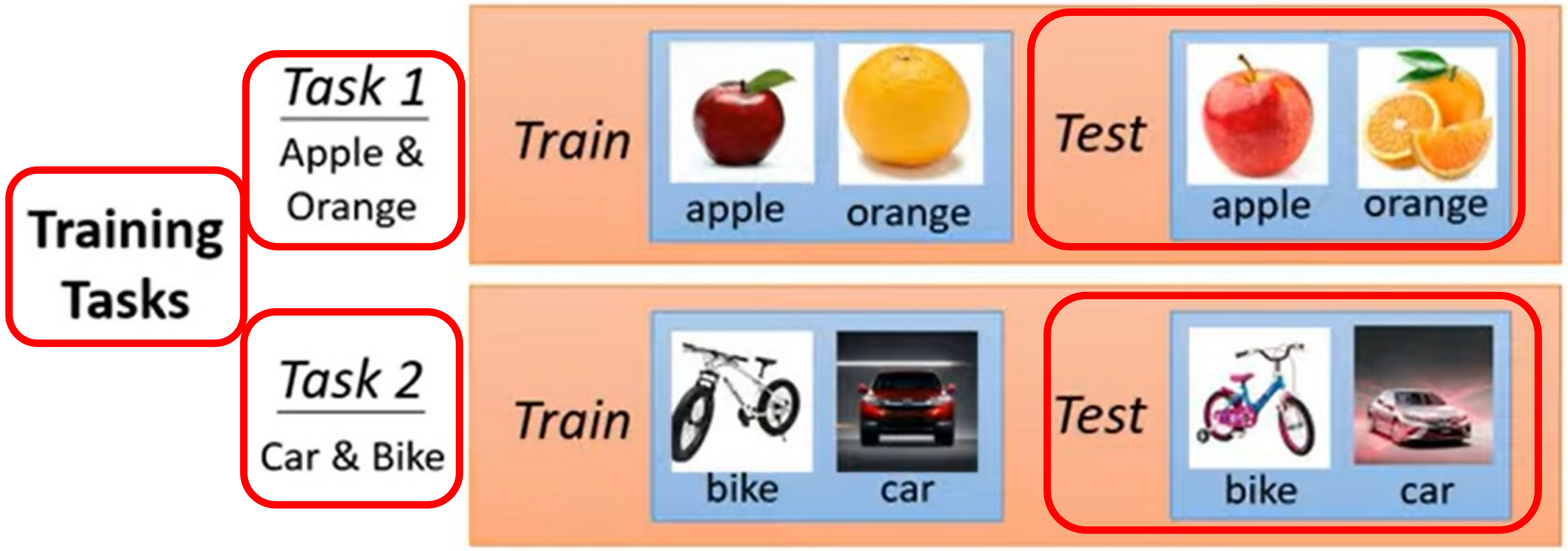


  - We need to hand craft the parameters of the model (Learning rate…)
- Meta learning: Model try to learn the parameters.
  - Why? learn new concepts and skills fast with a few training examples.

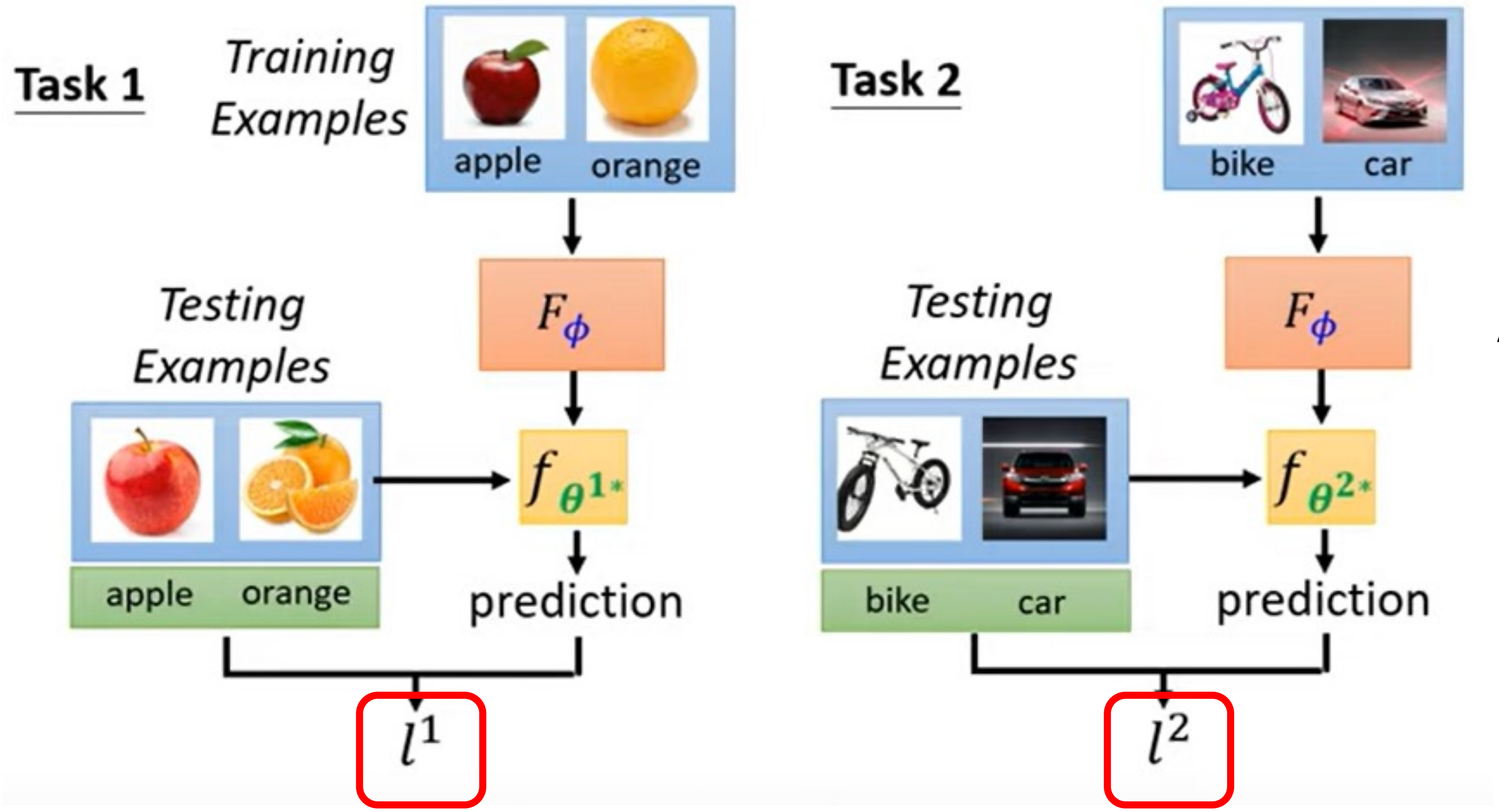
Meta Learning



Input of meta learning: tasks



Training in meta learning (Loss of meta training)

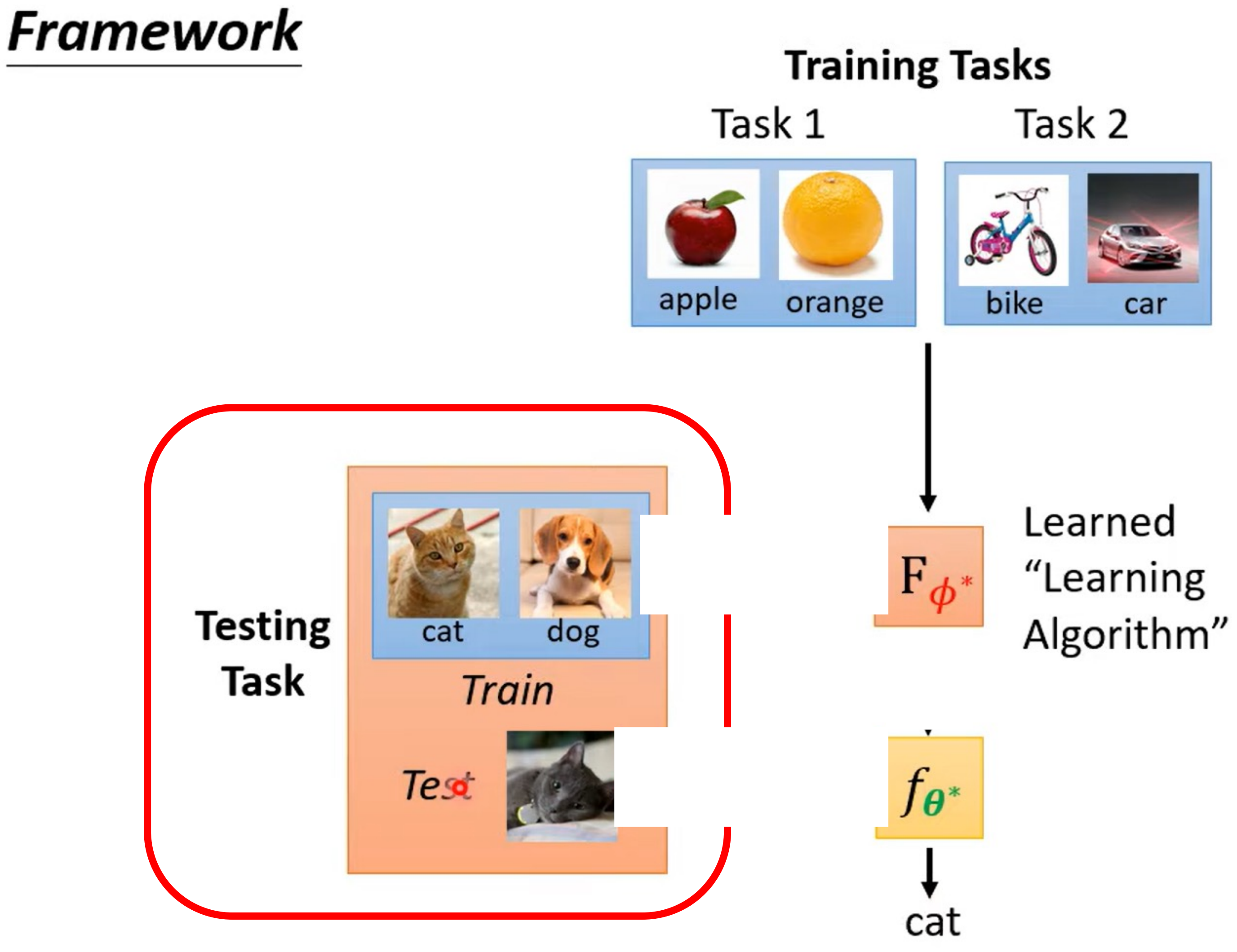


$$\boxed{L(\phi) = \sum_{n=1}^{N} l^n}$$

After defining the loss function of meta learning, we can optimize the meta learning model, to minimize loss. (Gradient descent, reinforcement learning)

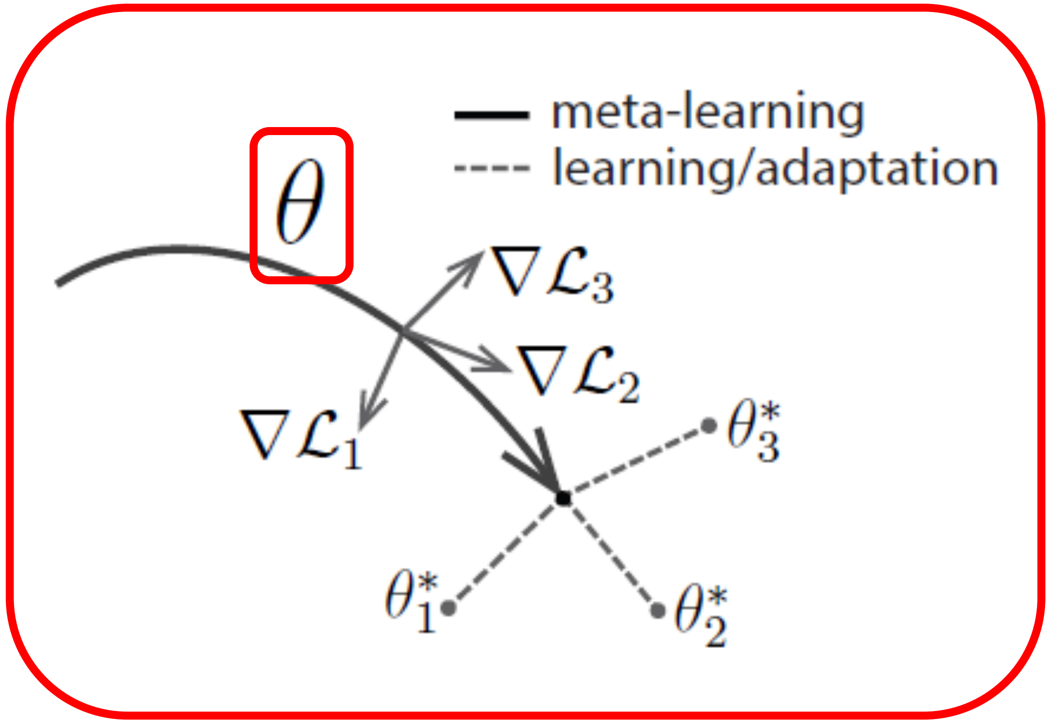
Testing in meta learning

- Don’t need a lot of training dataset for the testing task in meta learning.



MAML

- Aim to find model parameters sensitive to changes in tasks.
- parameterized function fθ (model), with parameter=θ.
- L -> loss



$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta).$$

Gradient descent update

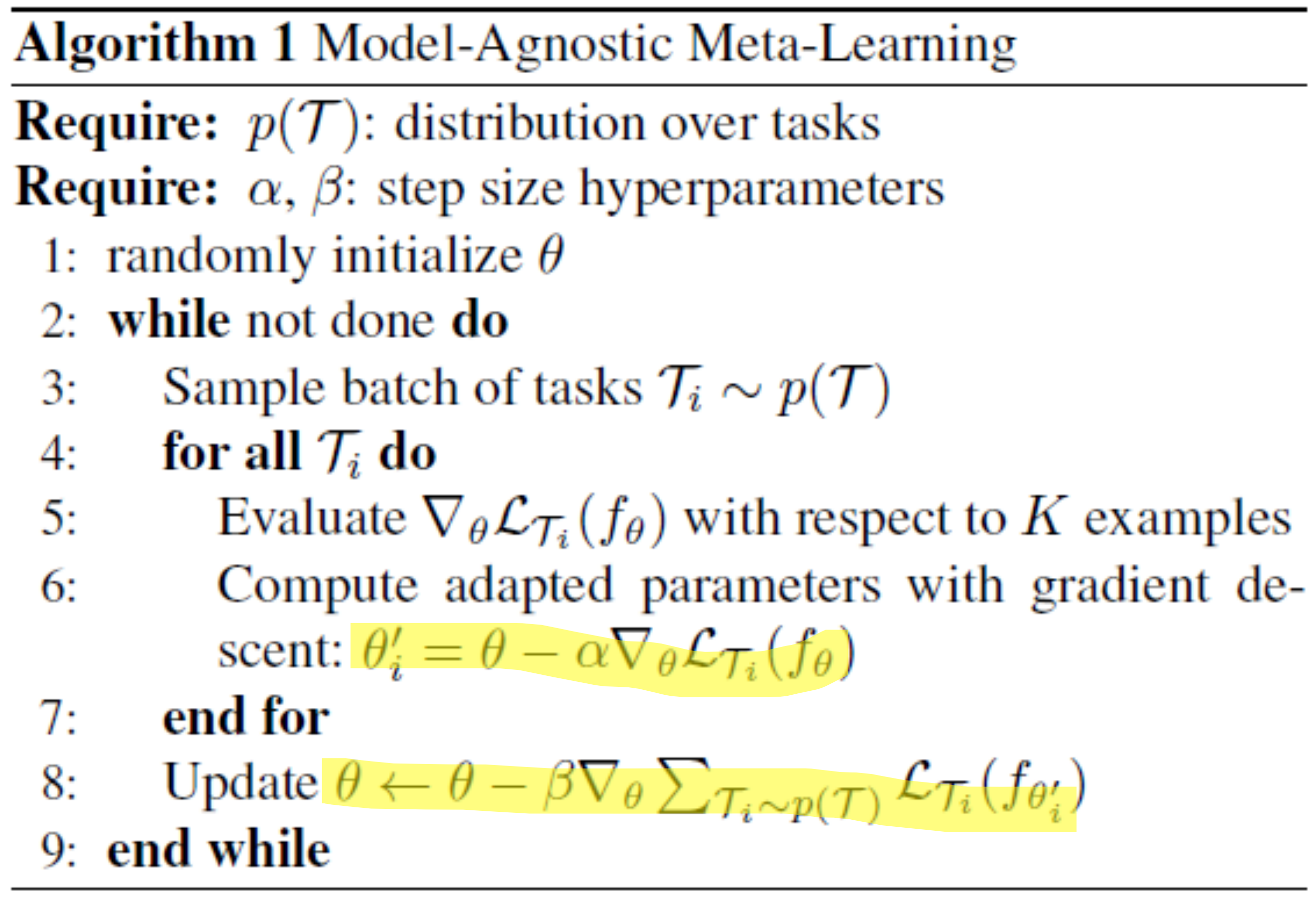
$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(\boxed{f_{\theta_i'}}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)})$$

Meta-objective

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

Update model parameters θ

1) get θ’ for each task
2) Minimize sum of loss of tasks
3) Update θ



Algorithm 1 Model-Agnostic Meta-Learning

```
Require: $p(\mathcal{T})$: distribution over tasks
Require: $\alpha$, $\beta$: step size hyperparameters
1: randomly initialize $\theta$
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 de-
           scent: $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
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
```