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

Finn, Chelsea, et al.International conference on machine learning. PMLR, 2017.

Department of Software, Sookmyung Women's University
Jisoo Kim

Table of Contents

01 Introduction

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.

1. Proposal : **Model-agnostic** meta-learning algorithm

2. Goal :
   - Produce **generalization** to solve new learning tasks, by training parameters using small number of training samples (few-shot)
   - **“Train the model to be easy to fine-tune!”**

3. Experimental Results & Contributions

01 Introduction

Q. Meta-learning?

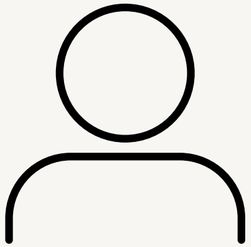
- **Supervised learning**:
  - Gather a large amount of data from a specific task (e.g., cat ↔ dog) and directly maximize performance on that specific task.

- **Transfer Learning / Fine-tuning**:
  - Learn a general representation using a large source dataset (e.g., ImageNet) first, and then retrain (fine-tune) on a single target task.

- **Multi-Task Learning (MTL)**:
  - Jointly train multiple tasks simultaneously using a single model/shared representation, without additional adaptation during deployment.

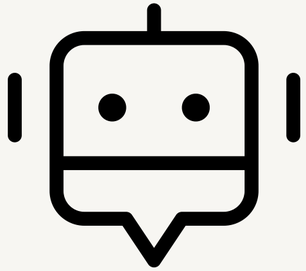
- **Meta-Learning (Learning to Learn)**:
  - Assuming a distribution of multiple tasks $p(T)$, the learning process itself (initialization, update rules, etc.) is optimized to "adapt quickly with fewer shots" when a new task is presented.
  - MAML: One of them is learning “fast-adaptive initialization θ” (inner/outer double loop).

01 Introduction



human

"Ability to learn quickly"



AI agent

<Challenges>

1. Must integrate its prior experience with a small amount of new information, while avoiding overfitting to the new data
2. The form of prior experience and new data depends on the task

<General & model-agnostic meta-learning algorithm : MAML>

- **General & model agnostic** : Can be directly applied to any learning problem and model that is trained with a gradient descent procedure (that can be differentiated.)
- **The Goal of trained model** : quickly learn a new task from a small amount of new data
- **Key Idea** : Train the model's initial parameter θ, to achieve near-maximum performance with only 1 or few gradient steps computed with a small amount of data (K-shot) for a new task.
  - MAML doesn't require additional parameters such as update networks, and doesn't impose any constraints on the model structure.

01 Introduction

- Maximize the **sensitivity** of the loss functions of new tasks with respect to the parameters
  - When the sensitivity is high, **small local changes** to the parameters can lead to **large improvements** in the task loss

The primary contribution of this work is a simple model- and task-agnostic algorithm for meta-learning that trains a model's parameters such that a small number of gradient updates will lead to fast learning on a new task. We demonstrate the algorithm on different model types, including fully connected and convolutional networks, and in several distinct domains, including few-shot regression, image classification, and reinforcement learning. Our evaluation shows that our meta-learning algorithm compares favorably to state-of-the-art one-shot learning methods designed specifically for supervised classification, while using fewer parameters, but that it can also be readily applied to regression and can accelerate reinforcement learning in the presence of task variability, substantially outperforming direct pretraining as initialization.

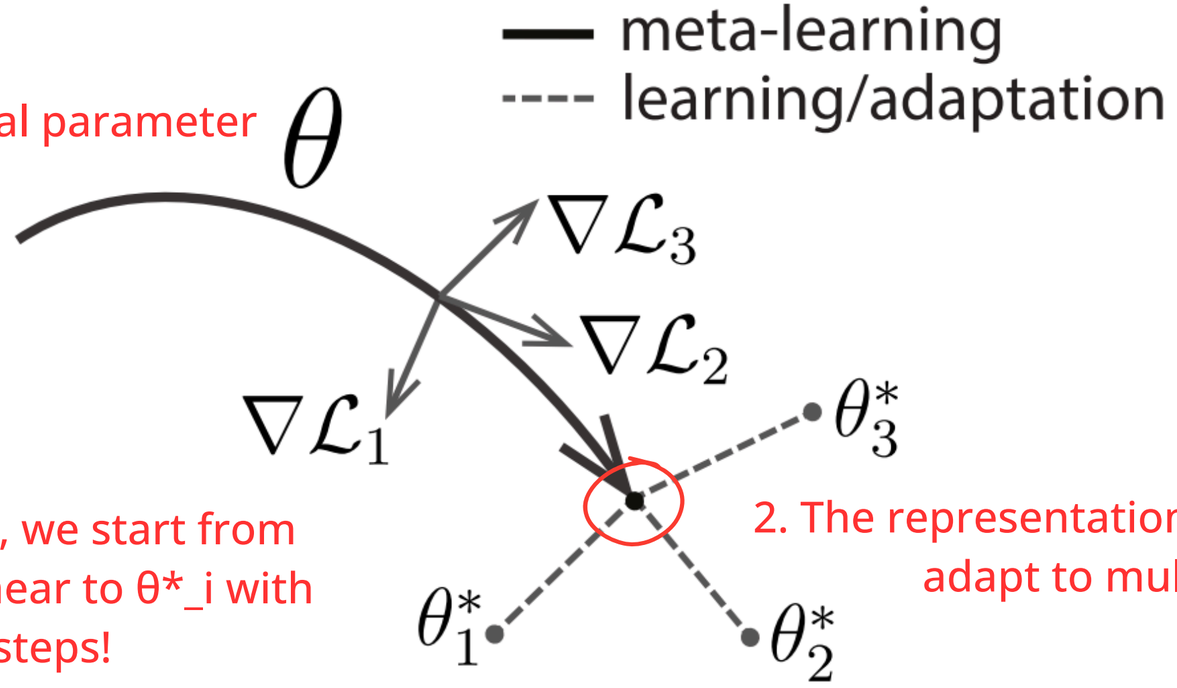
02 Model-Agnostic Meta-Learning - Problem Set-Up

$$T = \{ \underbrace{\mathcal{L}(x_1, a_1, \cdots, x_H, a_H)}_{\text{Loss function in this task (feedback)}}, \underbrace{q(x_1)}_{\text{Distribution that determines where the initial observation (input) } x_1 \text{ comes from}}, q(x_{t+1} \mid x_t, a_t), \overbrace{H}^{\text{Length of the episode}} \}$$

- $H = 1$: i.i.d supervised learning tasks
- $H > 1$: RL / sequential problems
  - ↳ cuz multiple steps are performed.

02 Model-Agnostic Meta-Learning - Algorithm

1. initial parameter



2. The representation θ that can quickly adapt to multiple tasks

3. When a new task is given, we start from meta-learned θ then move near to θ*_i with only few gradient steps!

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

02 Model-Agnostic Meta-Learning - Algorithm

$p(T)$: distribution of task → ex. (cat ↔ dog) (car ↔ bicycle) (lion ↔ tiger)

T_i : each task extracted from the distribution ($p(T) \sim T_i$)

f_θ : The model we are training (e.g., CNN) with parameters θ

$\mathcal{L}_{T_i}(f_\theta)$: Loss function at task T_i → ex. Cross entropy

→ also can be written as $\mathcal{L}_{T_i}(f_\theta; D)$ when specifying a data bundle (batch)

- D_i^{tr} : inner loop support set
- D_i^{val} : outer loop query set

- α : Inner (adaptation) phase learning rate → may be fixed as a hyperparameter or meta-learned
- β : Outer (meta) phase learning rate

K : Number of support shots used in inner updates

02 Model-Agnostic Meta-Learning - Algorithm

*Only 1 gradient update considered for simplicity of notation

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks

Require: α, β: step size hyperparameters

1: randomly initialize θ → Goal: find initial θ that can solve any new task well with just a few micro-updates.

2: **while** not done **do** → **1. Inner loop (task adaptation) : minimize L_sup for each task T**

3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$ batch 1: $(\mathcal{T}_1, \ldots, \mathcal{T}_k)$ / batch 2: $(\mathcal{T}_{k+1}, \ldots, \mathcal{T}_{2k})$ / ...

4: **for all** $\mathcal{T}_i$ **do**

5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples → $g_i = \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta; D_i^{tr})$

6: Compute adapted parameters with gradient de- ⇒ Inner update
scent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$

7: **end for** ⇒ For every task in the meta batch, we get θ_i'. ⌋ inner step

8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ → **meta-objective is computed using θ'**

9: **end while** → **2. Outer loop(meta-optimization) is performed over θ**, via SGD

02 Model-Agnostic Meta-Learning - Algorithm

⑤ : Compute the gradient of the loss using support set D_i^{tr} (K shots) in task T_i

ex) T_1 (cat ↔ dog) : calculate the classification loss at the current θ, with 1 cat and 1 dog (if 1-shot) and calculate g_1.

⑥ : ex) θ_1' : parameter quickly adopted to task T_1 (cat ↔ dog)

θ_2' : parameter quickly adopted to task T_2 (car ↔ bicycle)

⋮

⑧ Outer update

Support set loss : $L_{sup}(\theta)$

query set loss : $L_{qry}(\theta') = L_{meta}(\theta)$ ← what meta wants to decrease for real.

$\frac{d}{d\theta}$ → $\nabla_\theta L_{meta}(\theta) = \frac{d}{d\theta} L_{qry}(\theta')$

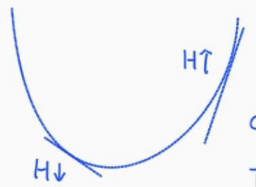
since θ' is from θ, we use chain rule →

$$= \underbrace{\frac{dL_{qry}(\theta')}{d\theta}}_{\text{meta gradient}} = \boxed{\frac{\partial\theta'}{\partial\theta}} \times \underbrace{\frac{\partial L_{qry}(\theta')}{\partial\theta'}}_{\text{query gradient}}$$

$$\theta' = \theta - \alpha \nabla_\theta L_{sup}(\theta)$$

$$\frac{\partial\theta'}{\partial\theta} = 1 - \alpha \frac{\partial}{\partial\theta}(\nabla_\theta L_{sup}(\theta)) = 1 - \alpha \underbrace{\nabla_\theta^2 L_{sup}(\theta)}_{\text{Hessian (second derivative)}} = 1 - \alpha H$$

↳ matrix that contains how quickly the gradient changes (curvature).



If the support loss terrain is steep (curvature ↑) even a small change in θ can significantly change the inner update result θ'.

This "sensitivity" is included in $(I - \alpha H)$.

$$\Rightarrow \nabla_\theta L_{meta}(\theta) = \frac{dL_{qry}(\theta')}{d\theta} = (1 - \alpha \nabla_\theta^2 L_{sup}(\theta)) \times \nabla_{\theta'} L_{qry} = \underline{MAML}$$

03 Species of MAML - "**Supervised**" Regression/Classification

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 $\mathcal{T}_i$ MSE
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) CE
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**

Q. N-way K-shot Classification?

- Extract N classes from a task
- Then use K {input, output} pairs from each class
- → For a total of N * K data points for N-way classification (**N * K** support set samples)

<Example>

- 5-way 1-shot : 5 classes * 1 shot for each class = 5
- 5-way 5-shot : 5 * 5 = 25 (support)
- 2-way 3-shot : 2 * 3 = 6 (support)

- Total number of samples = N×K (support) + N×Q (query)

04 Related Work

1. Prior meta-learning approaches

- Meta-learners that learn the update rule (RNN-based optimizers, learned optimizers)
- Jointly learn weight initialization and optimizer for few-shot tasks
- Often hard to extend beyond classification; require specific architectures
- Some works only studied good random initializations

2. Metric-based methods

- Siamese Nets, Matching Nets, Prototypical Nets → learn to compare embeddings
- Achieve strong results in few-shot classification
- But remain domain-specific and not easily applicable to regression or RL

3. Memory-augmented methods

- Memory-augmented NNs, Meta-Networks → use RNN + external memory for fast adaptation
- More complex and underperform MAML in few-shot classification

04 Related Work

4. Pretraining-based initialization

- Large-scale supervised pretraining for transfer learning
- MAML generalizes this idea by explicitly optimizing initialization for fast adaptation

5. MAML's key distinction

- No extra meta-parameters or specialized architectures
- Model-agnostic: applicable to classification, regression, and RL
- Learns an initialization that enables rapid adaptation with few gradient steps

05 Experimental Evaluation

1. Can MAML enable **fast learning** of new tasks?
2. Can MAML be used for meta-learning in **multiple different domains**?
3. Can a model learned with MAML continue to improve with **additional gradient updates** and/or examples?

① Task distribution $p(T)$

amplitude : $[0.1, 5.0]$

- each task is sine wave of the form $y = A \cdot \sin(x + \phi)$
  - $x \sim U[-5.0, 5.0]$
  - phase : $[0, \pi]$
- loss : MSE

② Model & meta-learning

- neural network model w/ 2 hidden layers, each 40 units, ReLU

* inner update
- adapt to task T_i with just 1 gradient step
- $\theta' = \theta - \alpha \nabla_\theta \mathcal{L}$ with K-shot datas ($K=10$, $\alpha = 0.01$)

* outer (meta) update
- Adam optimizer

③ 2 baselines for evaluation

1. Pretraining on all tasks

the entire mixed sine wave distribution

- First train a single network on $p(T)$ using general supervised learning
- When test, fine-tune using the given K points
- Limitation : In regression (where inputs are identical and outputs conflict across tasks)
  the model tends to learn to "average" out outputs
  ⇒ Risk of overfitting / misfitting in a small number of shots ↑

2. Oracle (upper bound)

- An ideal model that "additionally" provides the actual amplitude A & phase ϕ as input to the model.
  ↳ the problem-dependent "task identity" info

* oracle : Receive the identity of the task as an additional input, as an upper bound on the performance of the model

05 Experimental Evaluation - Regression

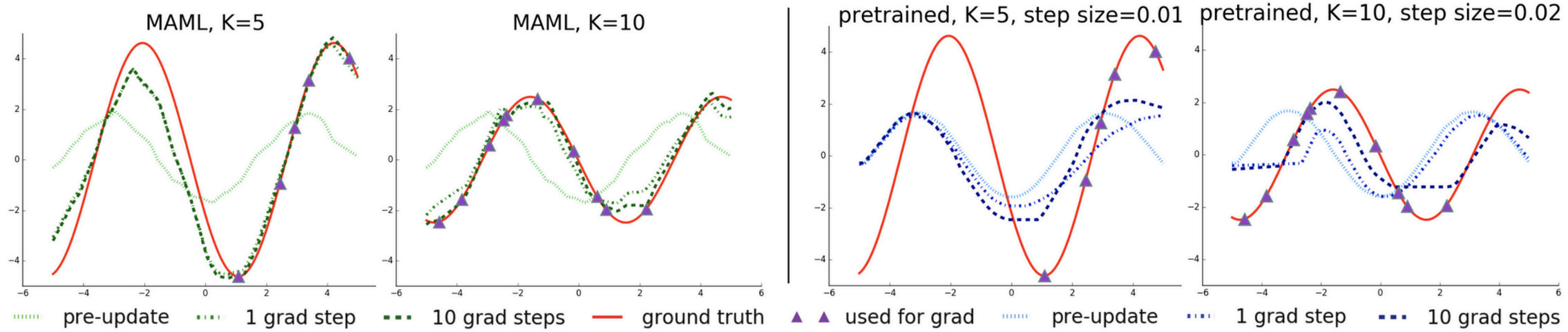


Figure 2. Few-shot adaptation for the simple regression task. Left: Note that MAML is able to estimate parts of the curve where there are no datapoints, indicating that the model has learned about the periodic structure of sine waves. Right: Fine-tuning of a model pretrained on the same distribution of tasks without MAML, with a tuned step size. Due to the often contradictory outputs on the pre-training tasks, this model is unable to recover a suitable representation and fails to extrapolate from the small number of test-time samples.

05 Experimental Evaluation - Regression

1. Pretraining

- In multi-task regression, environments where different y's are possible for the same x are trained as a single unit → The network converges to an "average y" or loses good representations due to conflicting signals
  - In this state, few-shot fine-tuning is extremely unstable (e.g., overfitting quickly or failing to find structure)

2. MAML

- From the beginning, the initial values are optimized to "quickly move along the amplitude/phase axis with one or two steps of gradient descent."
  - As a result, θ is located near the "sine wave space (amplitude · phase manifold)" and quickly aligns to the correct task even with small data sets

05 Experimental Evaluation - Regression

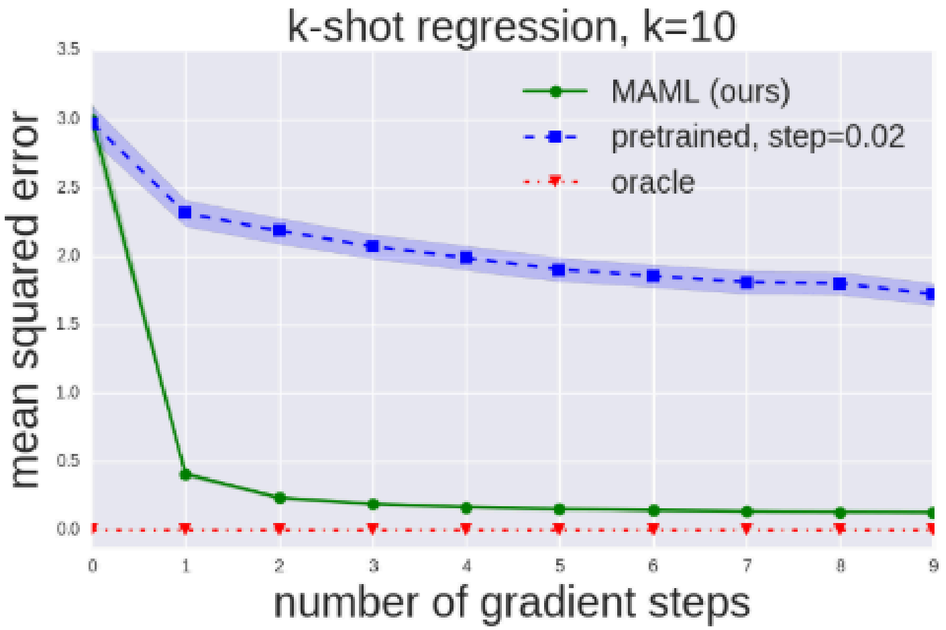


Figure 3. Quantitative sinusoid regression results showing the learning curve at meta test-time. Note that MAML continues to improve with additional gradient steps without overfitting to the extremely small dataset during meta-testing, achieving a loss that is substantially lower than the baseline fine-tuning approach.

MAML optimizes θ to lie in a region that is amenable to fast adaptation and sensitive to loss functions from p(T)

05 Experimental Evaluation - Classification

- Few-shot learning benchmark : Omniglot, MiniImageNet
- Protocol **(N-way, K-shot)**
  - Internal learning : select **N classes** (have not been seen before), adapt to **K samples** from each class
  - Then classify new samples within the same **N classes** to measure accuracy

5.2. Classification <Dataset Infos>

To evaluate MAML in comparison to prior meta-learning and few-shot learning algorithms, we applied our method to few-shot image recognition on the Omniglot (Lake et al., 2011) and MiniImagenet datasets. The Omniglot dataset consists of 20 instances of 1623 characters from 50 different alphabets. Each instance was drawn by a different person. The MiniImagenet dataset was proposed by Ravi & Larochelle (2017), and involves 64 training classes, 12 validation classes, and 24 test classes. The Omniglot and MiniImagenet image recognition tasks are the most common recently used few-shot learning benchmarks (Vinyals et al., 2016; Santoro et al., 2016; Ravi & Larochelle, 2017).

05 Experimental Evaluation - Classification

Table 1. Few-shot classification on held-out Omniglot characters (top) and the MiniImagenet test set (bottom). MAML achieves results that are comparable to or outperform state-of-the-art convolutional and recurrent models. Siamese nets, matching nets, and the memory module approaches are all specific to classification, and are not directly applicable to regression or RL scenarios. The ± shows 95% confidence intervals over tasks. Note that the Omniglot results may not be strictly comparable since the train/test splits used in the prior work were not available. The MiniImagenet evaluation of baseline methods and matching networks is from Ravi & Larochelle (2017).

Omniglot (Lake et al., 2011)	5-way Accuracy		20-way Accuracy	
	1-shot	5-shot	1-shot	5-shot
MANN, no conv (Santoro et al., 2016)	82.8%	94.9%	–	–
MAML, no conv (ours)	**89.7 ± 1.1%**	**97.5 ± 0.6%**	–	–
Siamese nets (Koch, 2015)	97.3%	98.4%	88.2%	97.0%
matching nets (Vinyals et al., 2016)	98.1%	98.9%	93.8%	98.5%
neural statistician (Edwards & Storkey, 2017)	98.1%	99.5%	93.2%	98.1%
memory mod. (Kaiser et al., 2017)	98.4%	99.6%	95.0%	98.6%
MAML (ours)	**98.7 ± 0.4%**	**99.9 ± 0.1%**	**95.8 ± 0.3%**	**98.9 ± 0.2%**

MiniImagenet (Ravi & Larochelle, 2017)	5-way Accuracy	
	1-shot	5-shot
fine-tuning baseline	28.86 ± 0.54%	49.79 ± 0.79%
nearest neighbor baseline	41.08 ± 0.70%	51.04 ± 0.65%
matching nets (Vinyals et al., 2016)	43.56 ± 0.84%	55.31 ± 0.73%
meta-learner LSTM (Ravi & Larochelle, 2017)	43.44 ± 0.77%	60.60 ± 0.71%
MAML, first order approx. (ours)	**48.07 ± 1.75%**	**63.15 ± 0.91%**
MAML (ours)	**48.70 ± 1.84%**	**63.11 ± 0.92%**

05 Experimental Evaluation - Classification

<Value of FOMAML>

the meta-objective (see Equation (1)). On MiniImagenet, we show a comparison to a first-order approximation of MAML, where these second derivatives are omitted. Note that the resulting method still computes the meta-gradient at the post-update parameter values θ_i', which provides for effective meta-learning. Surprisingly however, the performance of this method is nearly the same as that obtained with full second derivatives, suggesting that most of the improvement in MAML comes from the gradients of the objective at the post-update parameter values, rather than the second order updates from differentiating through the gradient update. Past work has observed that ReLU neu-

- ReLU neural networks are locally almost linear → second derivatives may be close to 0
- Led to roughly 33% speed-up in network computation

06 Strengths & Limitations

We introduced a meta-learning method based on learning easily adaptable model parameters through gradient descent. Our approach has a number of benefits. It is simple and does not introduce any learned parameters for meta-learning. It can be combined with any model representation that is amenable to gradient-based training, and any differentiable objective, including classification, regression, and reinforcement learning. Lastly, since our method merely produces a weight initialization, adaptation can be performed with any amount of data and any number of gradient steps, though we demonstrate state-of-the-art results on classification with only one or five examples per class. We also show that our method can adapt an RL agent using policy gradients and a very modest amount of experience.

Reusing knowledge from past tasks may be a crucial ingredient in making high-capacity scalable models, such as deep neural networks, amenable to fast training with small datasets. We believe that this work is one step toward a simple and general-purpose meta-learning technique that can be applied to any problem and any model. Further research in this area can make multitask initialization a standard ingredient in deep learning and reinforcement learning.

1. **Simplicity & Generality**
- Can be applied to any model which is able to perform GD based learning
2. Experimental validation across **diverse domains**
3. **Achieved SOTA** in a few-shot scenario
- Especially on challenging benchmarks like MiniImageNet, it clearly outperformed Matching Nets and Meta-LSTM
4. **Adaptability**
- Adapts quickly with just 1-2 gradient steps and a few samples, and additional improvements are possible with more data/steps
5. **Efficiency and Practicality** of First-Order Approximation (FOMAML)

06 Strengths & Limitations

pretraining : 1 * (forward + backward) → O(C)
FOMAML : inner(1) + outer(1) → O(2C)
Full MAML : inner(1) + outer(1) → O(2C + H)

1. **High computational cost**
   - Although the paper complements this with FOMAML, meta-learning itself is still heavier than pretraining
2. **Training stability issues**
   - Simultaneously trains the inner loop (task-specific gradient update) and outer loop (meta-update)
     - → sensitive to hyperparameters (learning rate, step size etc.)
   - The paper appendix also states that the step size/number of steps are set differently for each task, making tuning difficult
3. **Model Simplicity**
   - The models used in the paper are relatively simple CNN/MLP and RL policy nets → lacking validation against large & complex models.
   - Subsequent studies have extended this model to Transformers and large-scale models, but the original paper lacks scaling results.

06 Automotive Ethernet IDS w/ MAML (thoughts)

1. **The Need for Few-shot learning**
   - In real-world IVN, new attack types can emerge and labeled data is scarce....
   - Using MAML, we can meta-train the model using a variety of existing attack tasks (e.g., DoS, Fuzzy, etc.) and quickly adapt to new attack types with just a few samples
2. **Model-Agnostic Characteristics**
   - MAML can be applied regardless of the classifier architecture used for IDS, such as CNN, MLP, or RNN
3. **Experimental Expansion Ideas**
   - **Meta-Training** : Define 4-5 existing attack types as tasks
   - **Meta-Test** : Adapt to attack types unseen during training (e.g., new flooding) with just few shots
   - **Comparison** : Regular fine-tuning vs MAML vs FOMAML
   - **Expectation** : MAML initialization will show significantly fasterr detection performance

The End.

07 Appendix

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{\mathbf{x}^{(j)},\mathbf{y}^{(j)}\sim\mathcal{T}_i} \| f_\phi(\mathbf{x}^{(j)}) - \mathbf{y}^{(j)} \|_2^2, \tag{2}$$

$$\begin{aligned}\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{\mathbf{x}^{(j)},\mathbf{y}^{(j)}\sim\mathcal{T}_i} & \mathbf{y}^{(j)} \log f_\phi(\mathbf{x}^{(j)}) \\ & + (1 - \mathbf{y}^{(j)}) \log(1 - f_\phi(\mathbf{x}^{(j)}))\end{aligned} \tag{3}$$

07 Appendix

Model Architecture

Our model follows the same architecture as the embedding function used by Vinyals et al. (2016), which has 4 modules with a 3×3 convolutions and 64 filters, followed by batch normalization (Ioffe & Szegedy, 2015), a ReLU nonlinearity, and 2×2 max-pooling. The Omniglot images are downsampled to 28×28, so the dimensionality of the last hidden layer is 64. As in the baseline classifier used by Vinyals et al. (2016), the last layer is fed into a softmax. For Omniglot, we used strided convolutions instead of max-pooling. For MiniImagenet, we used 32 filters per layer to reduce overfitting, as done by (Ravi & Larochelle, 2017). In order to also provide a fair comparison against memory-augmented neural networks (Santoro et al., 2016) and to test the flexibility of MAML, we also provide results for a non-convolutional network. For this, we use a network with 4 hidden layers with sizes 256, 128, 64, 64, each including batch normalization and ReLU nonlinearities, followed by a linear layer and softmax. For all models, the loss function is the cross-entropy error between the predicted and true class. Additional hyperparameter details are included in Appendix A.1.