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

Scaramuzzino Giovanna

September 6, 2021

Overview

1. Introduction
   - What is Meta Learning?
   - Formalizing Meta-Learning
   - Few Shot Learning
2. Model-Agnostic Meta-Learning
   - Algorithm
   - Gradient of Gradient
   - First order approximation
3. Experimets and results
   - Regression problem
   - Classification problem
4. Conclusion

What is Meta Learning?

Deep Learning technics have seen great success in a variaty of fields but there are limitations:

- quantities of data (large training set to train our model)
- compute resources

Meta Learning is an alternative paradigm where machine learning models gains experience over multiple learning process and uses this experience to improve its future learning performance.
Meta learning produces a versatile AI model that can learn to perform various tasks without having to train them from scratch.

Goal: learner quickly learn a new task from a small amount of new data

Meta learning

Two stages to update the model:

- **base learning**: inner learning algorithm solve a task
- **meta learning**: an outer algorithm updates the inner learning algorithm such that the model it learns improves an outer objective.

Conventional Machine Learning

In supervised learning, we are given a training dataset $\mathcal{D} = \{$(x1,y1),...,(xn,yn)$\}$.

We can train a predictive model $\hat{y} = f_\theta(x)$ by solving:

$$\theta^* = \arg\min_\theta \mathcal{L}(\mathcal{D}; \theta, \omega) \tag{1}$$

Where:

- $\mathcal{L}$: loss function
- θ: parameter of model
- ω: the optimizer chosed

Meta Learning: Task-Distribiution View

Define:

- $\mathcal{T} = \{\mathcal{D}, \mathcal{L}\}$ task, with $\mathcal{D}$:dataset, $\mathcal{L}$:loss function
- p($\mathcal{T}$): distribution of tasks
- ω: meta-knowledge
- M source tasks used in meta-training stage as $\mathcal{D}_{source} = \{(\mathcal{D}^{train}_{source}, \mathcal{D}^{val}_{source})^{(i)}\}_{i=1}^{M}$
- Q target tasks used in meta-testing stage as $\mathcal{D}_{target} = \{(\mathcal{D}^{train}_{target}, \mathcal{D}^{test}_{target})^{(i)}\}_{i=1}^{Q}$

We have to solve:

$$\min_{\omega} \mathop{\mathbb{E}}_{\mathcal{T} \sim p(\mathcal{T})} \mathcal{L}(D; \omega) \tag{2}$$

Assume access to a set of source tasks sampled from a distribution of taks $p(\mathcal{T})$, **Meta-training** step can be written as:

$$\omega^* = arg \max_{\omega} \log p(\omega | \mathcal{D}_{source}) \tag{3}$$

Meta Learning: Task-Distribiution and Bilevel Optimization Views

How to solve the meta-training step?
Using **bilevel optimization**, can be formalised as follows:

$$\omega^* = \arg\min_{\omega} \sum_{i=1}^{M} \mathcal{L}^{meta}(\theta^{*(i)}(\omega), \omega, \mathcal{D}_{source}^{val}{}^{(i)}) \tag{4}$$

$$s.t. \quad \theta^{*(i)}(\omega) = \arg\min_{\theta} \mathcal{L}^{task}(\theta, \omega, \mathcal{D}_{source}^{train}{}^{(i)}) \tag{5}$$

where $\mathcal{L}^{meta}$ and $\mathcal{L}^{task}$ refer to the outer and inner objectives

Meta-testing use the meta-knowledge to train the base model on each previously unseen target task i:

$$\theta^{*(i)} = arg \max_{\theta} \log p(\theta | \omega^*, \mathcal{D}_{target}^{train}{}^{(i)}) \tag{6}$$

$\mathcal{D}_{target}^{test}{}^{(i)}$ can be used to evaluate the accuracy of the meta-learner.

Few Shot Learning

Few-Shot learning is learning from fewer data points

N-way K-shot Learning:

- N is the number of the classes
- K is the number of data points in each of the classes in the dataset

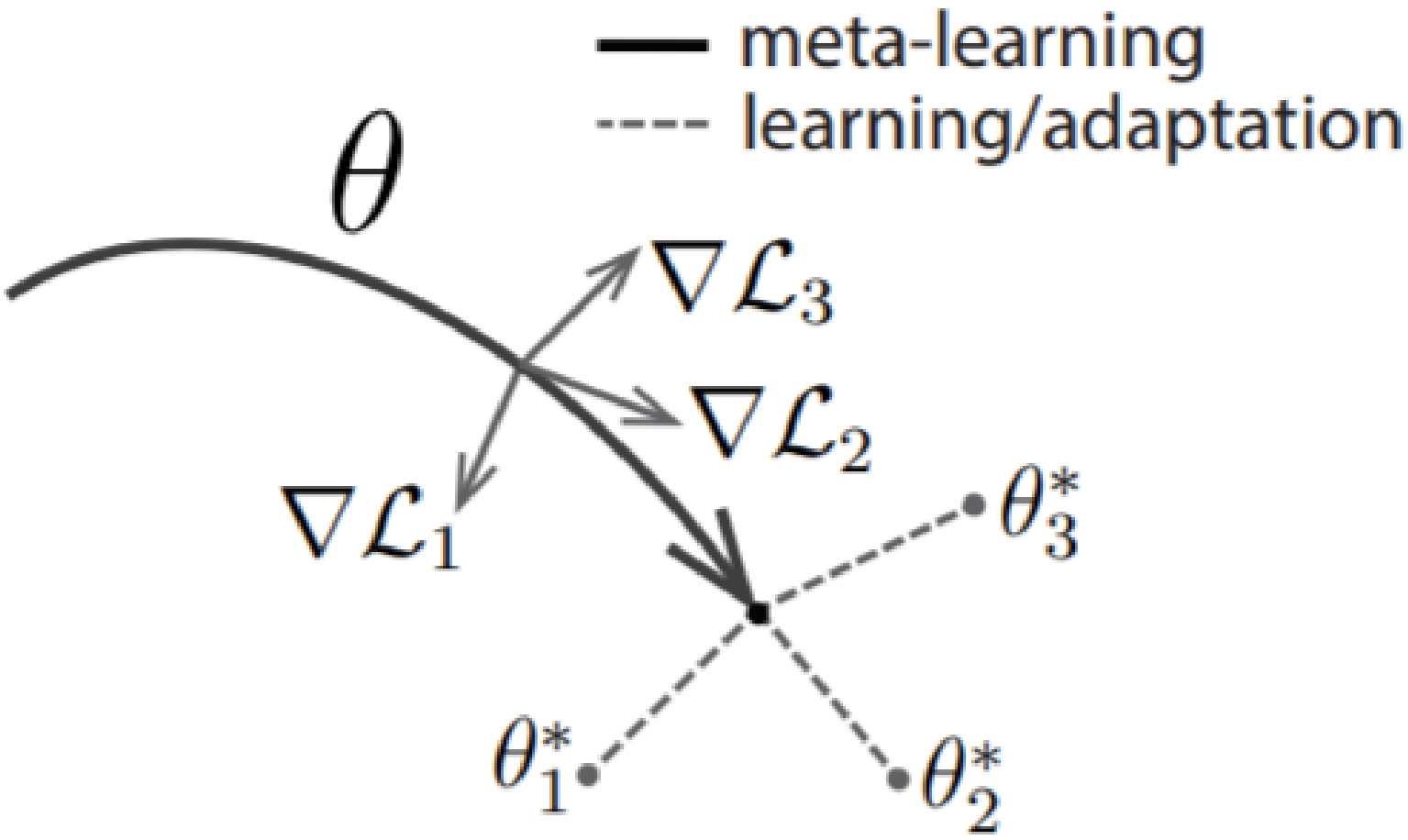
Figure: Example of 5way-2shot - MiniImagenet Dataset

Model-Agnostic Meta-Learning

- The authors proposed an algorithm **compatible with any model** trained with gradient descent and applicable to a variety of different learning problems: classification, regression, reinforcement learning.

- **Key idea**: train the model's initial parameters such that a **small number of gradient updates** will lead to fast learning on a new task using a **small amount of data**.

Idea

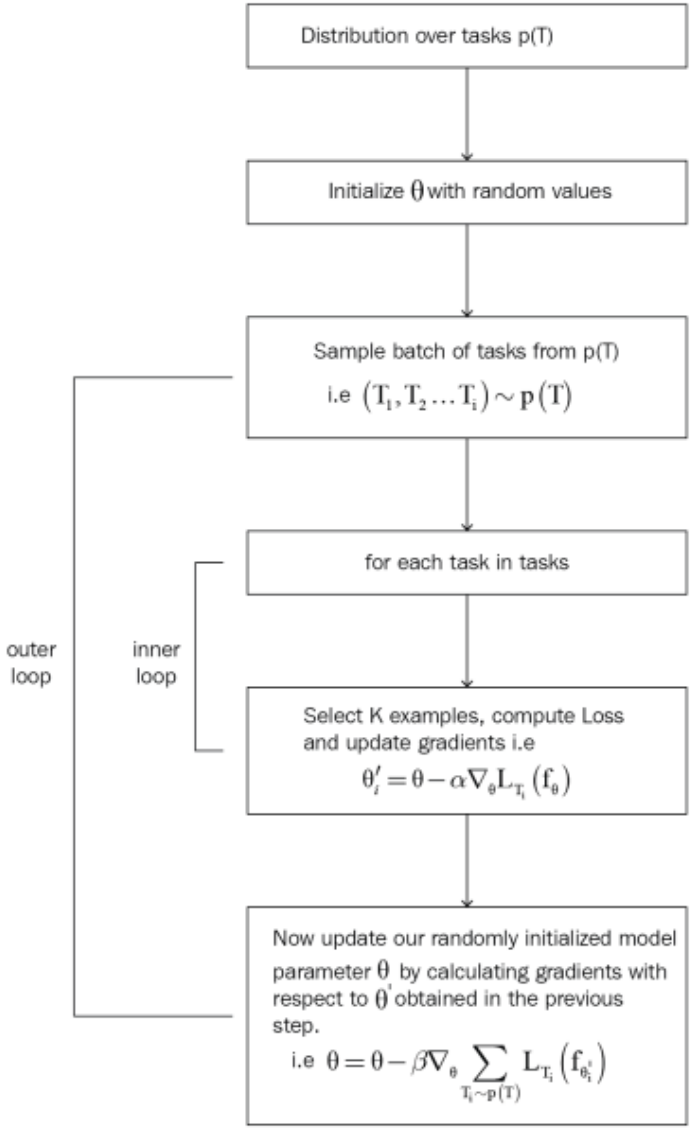

meta-learning
learning/adaptation
θ
$\nabla\mathcal{L}_3$
$\nabla\mathcal{L}_2$
$\nabla\mathcal{L}_1$
θ_3^*
θ_1^*
θ_2^*

Algorithm

Algorithm 2 MAML for Few-Shot Supervised Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

```
randomly initialize θ
while not done do
    Sample batch of tasks T_i ~ p(T)
    for all T_i do
        Sample K datapoints D = {x^(j), y^(j)} from T_i
        Evaluate ∇_θ L_{T_i}(f_θ) using D and L_{T_i} in Equation (2) or (3)
        Compute adapted parameters with gradient descent: θ'_i = θ − α∇_θ L_{T_i}(f_θ)
        Sample datapoints D'_i = {x^(j), y^(j)} from T_i for the meta-update
    end for
    Update θ ← θ − β∇_θ Σ_{T_i ~ p(T)} L_{T_i}(f_{θ'_i}) using each D'_i and L_{T_i} in Equation 2 or 3
end while
```



Mean-Squared-Error for regression:

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

Cross-Entropy for classification:

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{x^{(j)}, y^{(j)} \sim \mathcal{T}_i} y^{(j)} log f_\phi(x^{(j)}) + (1 - y^{(j)}) log(1 - f_\phi(x^{(j)})) \quad (3)$$

Gradient of Gradient

From line 10 in Algorithm 2:

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

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

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

($\mathcal{L}$ is differentiable)

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

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \boxed{(\mathrm{I} - \alpha \nabla^2_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta))} \nabla_{\theta'_i} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}))$$

$\Downarrow$

Calculation of Hessian matrix is required.
It Makes MAML be slow when backpropagating.

First order approximation

The Authors to reduce the cost try to compute the 1 order approximation
Ignoring the second-order term that appears in the update of MAML

$$\theta = \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \boxed{(\mathrm{I} - \alpha \nabla^2_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta))} \nabla_{\theta'_i} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}))$$

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

Regression problem

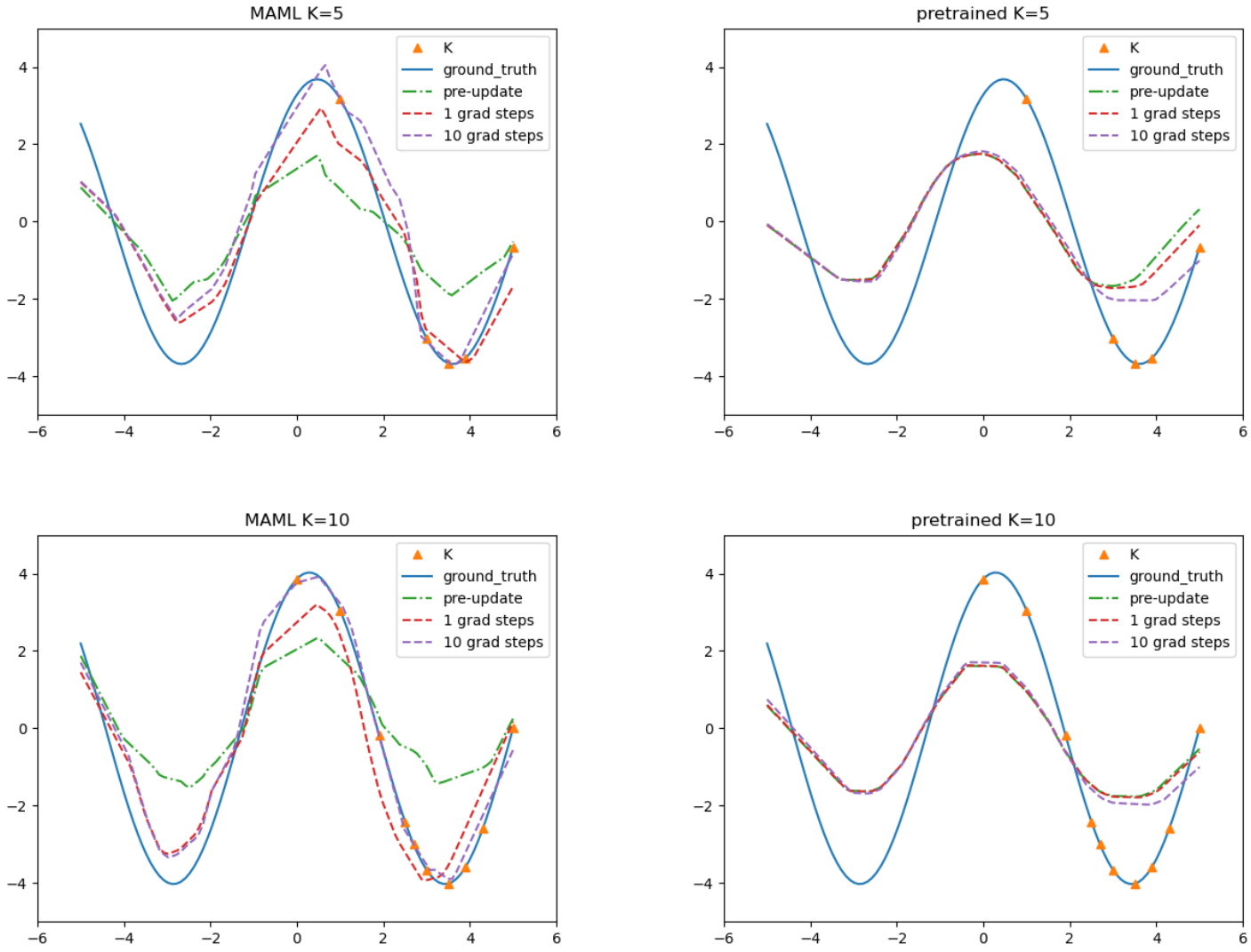
- Sine wave function with:
  - Amplitude in [0.1,5.0]
  - Phase in [0, π]

  } Varied between tasks

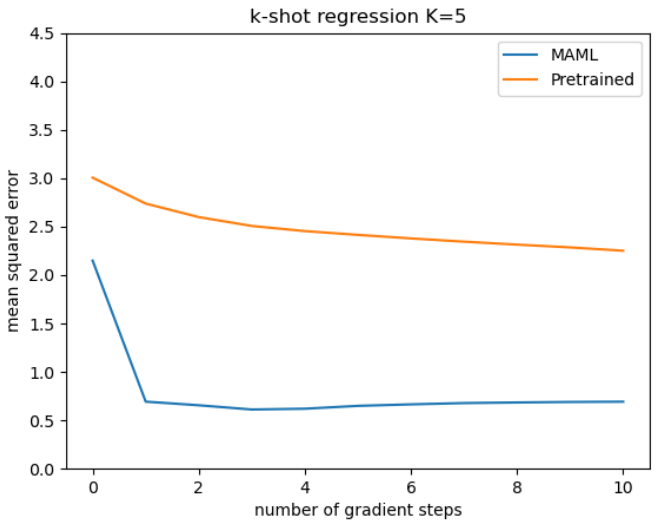
  - Datapoints sampled uniformly from [-5.0, 5.0]
- Loss function: Mean Squared Error
- Neural Network: 2 hidden layers with 40 units and ReLU
- Training:
  - 1 gradient step in the inner loop
  - Examples K: 5 or 10
  - Optimizer: Adam with $\alpha = 0.01$
- Evaluation (finetuning the model learned by MAML):
  - 10 gradient steps in the inner loop
  - K: 5 and 10 datapoints

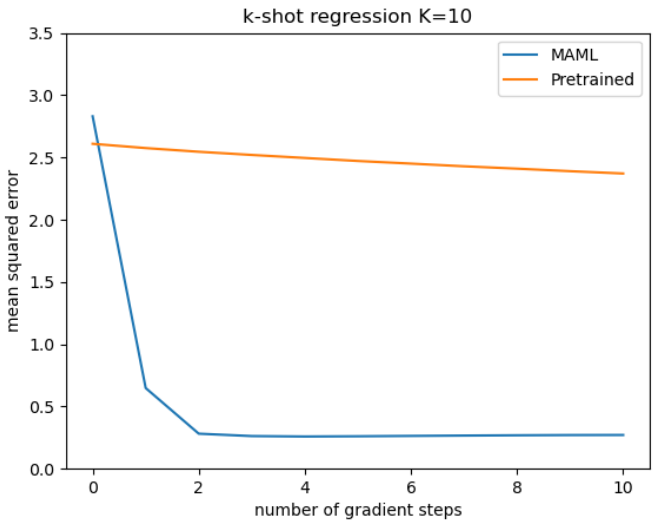
Regression problem: results

- To evaluate performance MAML model was compared with model pretraining on all of the tasks



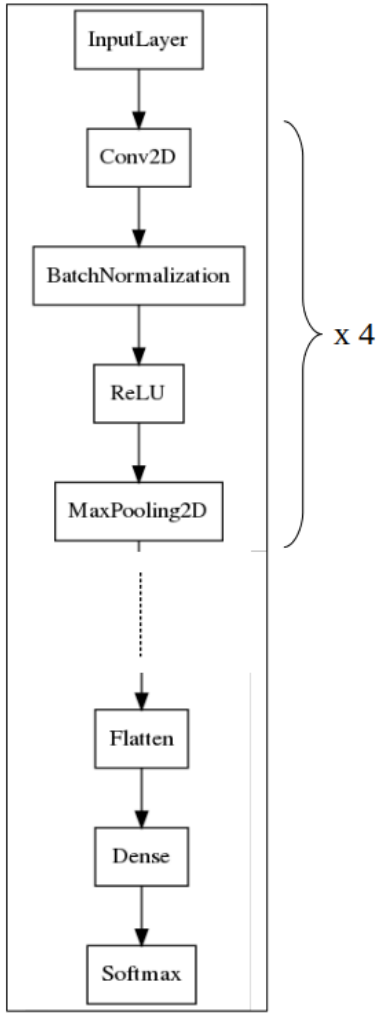
Quantitative sinusoide regression results





Classification problem

- Datasets:
  - Omniglot:
    - 50 different alphabets of 1623 characters
    - each character have 20 instances drawn by 20 different people
    - 1200 characters for training, 423 for test
    - images downsampled to 28x28 and augmented with rotations by multiples of 90 degrees
  - MiniImagenet:
    - 64 training classes, 12 validation classes, 24 test classes
- Loss: cross-entropy



Classification experiments

- N-way K-shot classification
- Outer Learning rate (lr) (β): 0.001
- models trained for 60000 iterations

MiniImagenet		**Train**		**Test**	
	meta-batch-size	gradient steps	inner lr	gradient steps	inner lr
5-way 1-shot	4 tasks	5	0.01	10	0.01
5-way 5-shot	2 tasks	5	0.01	10	0.01

Omniglot		**Train**		**Test**	
	meta-batch-size	gradient steps	inner lr	gradient steps	inner lr
5-way 1-shot **5-way 5-shot**	32 tasks	1	0.4	3	0.4

Classification results

MiniImagenet	5-way Accuracy	
	1-shot	5-shot
author's model	48.70±1.84%	63.11±0.92%
my result	45.55±0.4%	63.04±0.48%
author's model first order approx.	48.07 ±1.75%	63.15±0.91%
my result first order approx.	43.5±0.4%	59.4±0.49%

Omniglot	5-way Accuracy	
	1-shot	5-shot
author's model	98.7±0.4%	99.9 ±0.1%
my result	97.41±0.1%	99.09±0.3%

Metrics

- Accuracy $= \frac{\text{number of correct prediction}}{\text{total number of prediction}}$
- $\pm$ shows 95% confidence intervals over tasks:

$$\text{average} \pm 1.96 * \frac{\text{std}}{\sqrt{\text{number of observations}}}$$

Classification results - approximation

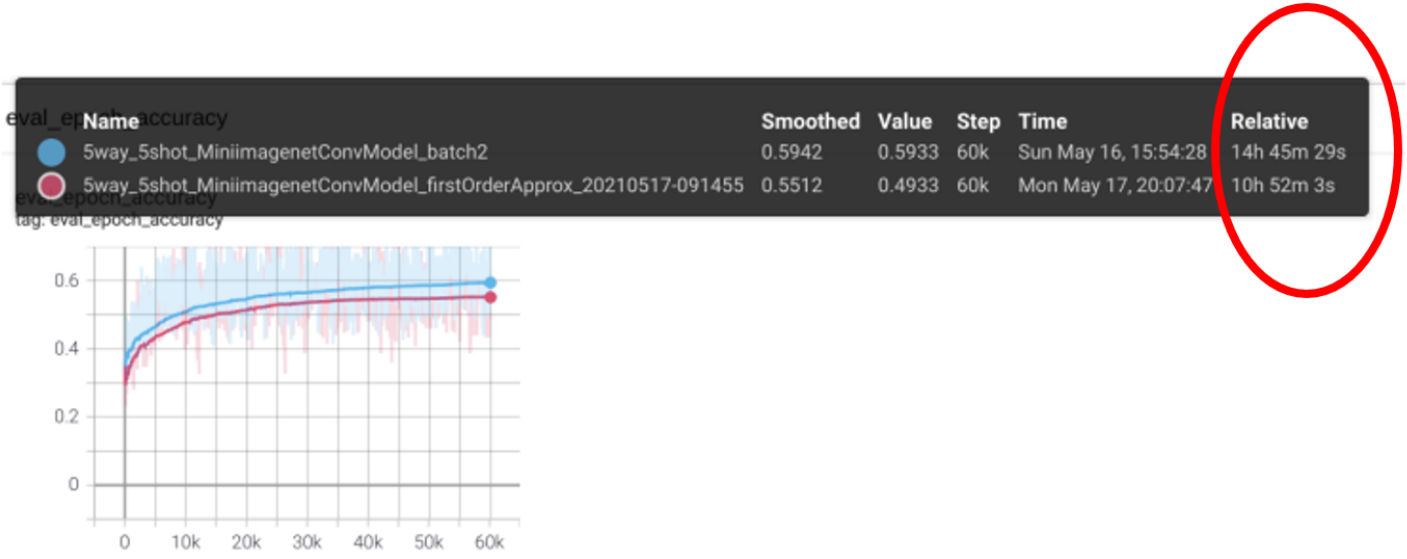


Figure: Accuracy on validation set: pink line rapresent model with 1 order approximation

Approximation speed up network computation by 33% while maintaining approximately the same performance

Conclusion

Goal

Implement Model-Agnostic Meta learning algorithm

Final considerations

- My models reach comparables results with reference to the proposed one
- differences may be due to:
  - different choice of some parameters not indicated in the paper
  - different split of the dataset
  - use of early stopping
  - the authors could train the model several times and then choose the best one