A MODEL-AGNOSTIC META-LEARNING FRAMEWORK FOR SOLVING FEW-SHOT TASKS IN PARALLEL

Master Thesis Presentation

June 2020

Florian Soulier

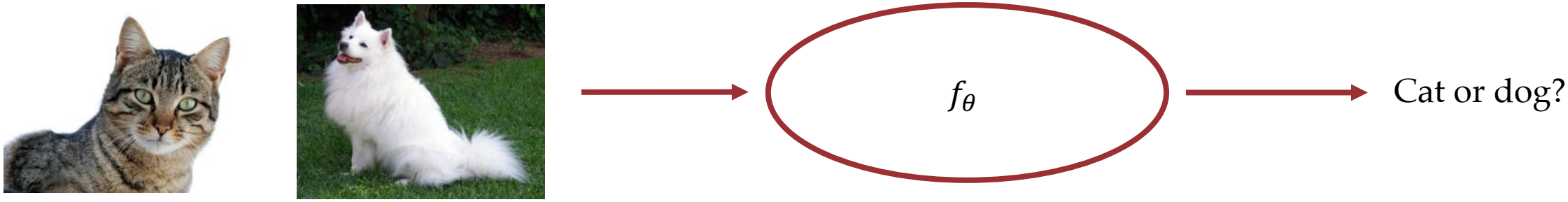
Agenda

- Few-Shot problems
  - Introduction
  - Approaches
- MAML
  - Key ideas
  - Training
- DMAML
  - Idea
  - Schematic
- Experiments
- Conclusion
- Future Work

Few-Shot-Problems

Hard to learn problems for machine learning systems

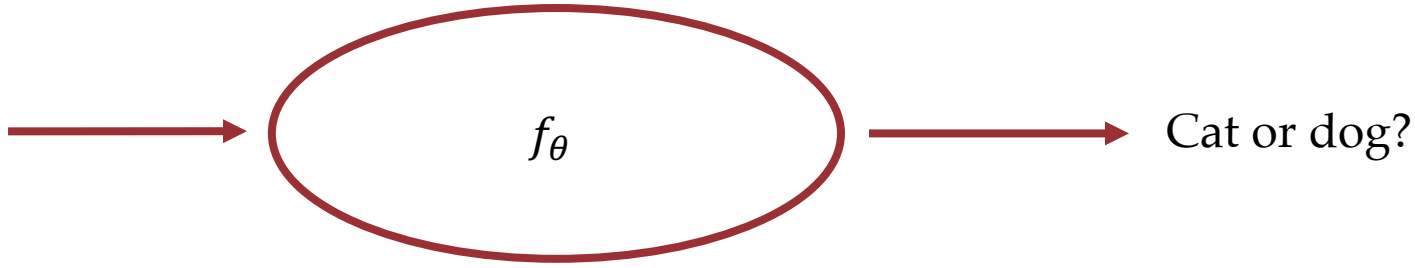
Few-Shot Learning



Few-shot learning

- Learning from only *few* samples for a *task*
- Few means e. g. k = 1 images of each class

Few-Shot Learning



Task

- Entity or objective being learned
- Here: classify images into n = 2 classes “cat” or “dog”

Few-Shot Learning



Cat or dog?

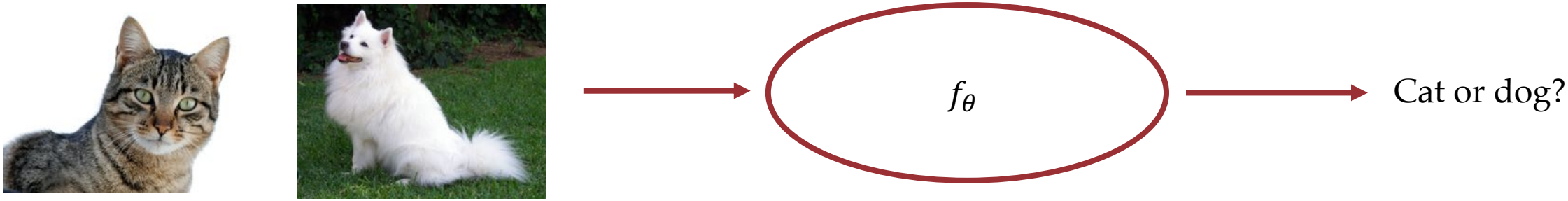
N-way k-shot problems

- N stands for the number of classes to distinguish in 1 task
- K stands for the number of seen samples (e. g. images)



Dog or squirrel?

Meta-Learning



Goal of meta-learning

- Learn to solve *new* tasks by training with a set of training tasks

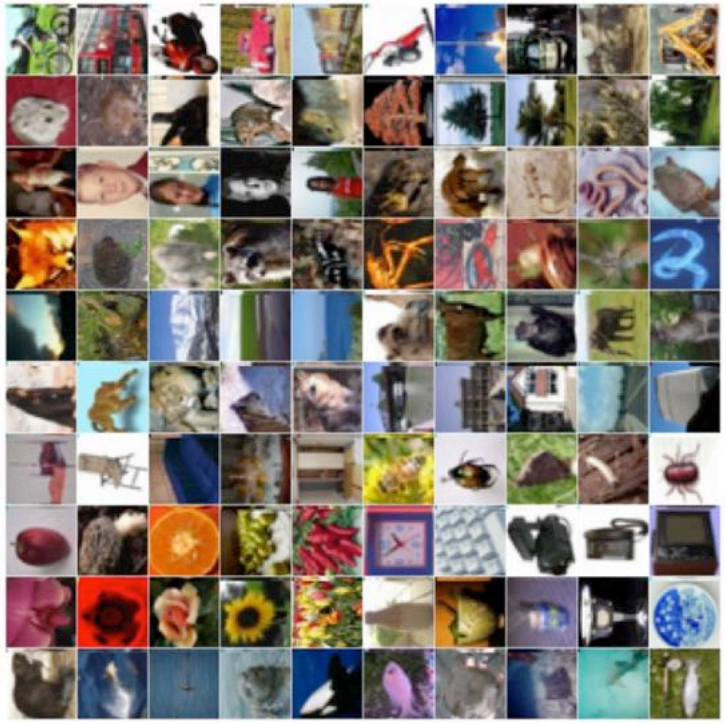
Few-shot problems

- "The problems that are solved by meta-learning"
- Other approaches are there too

Few-Shot-Algorithms

Approaches for solving few-shot problems

Transfer Learning

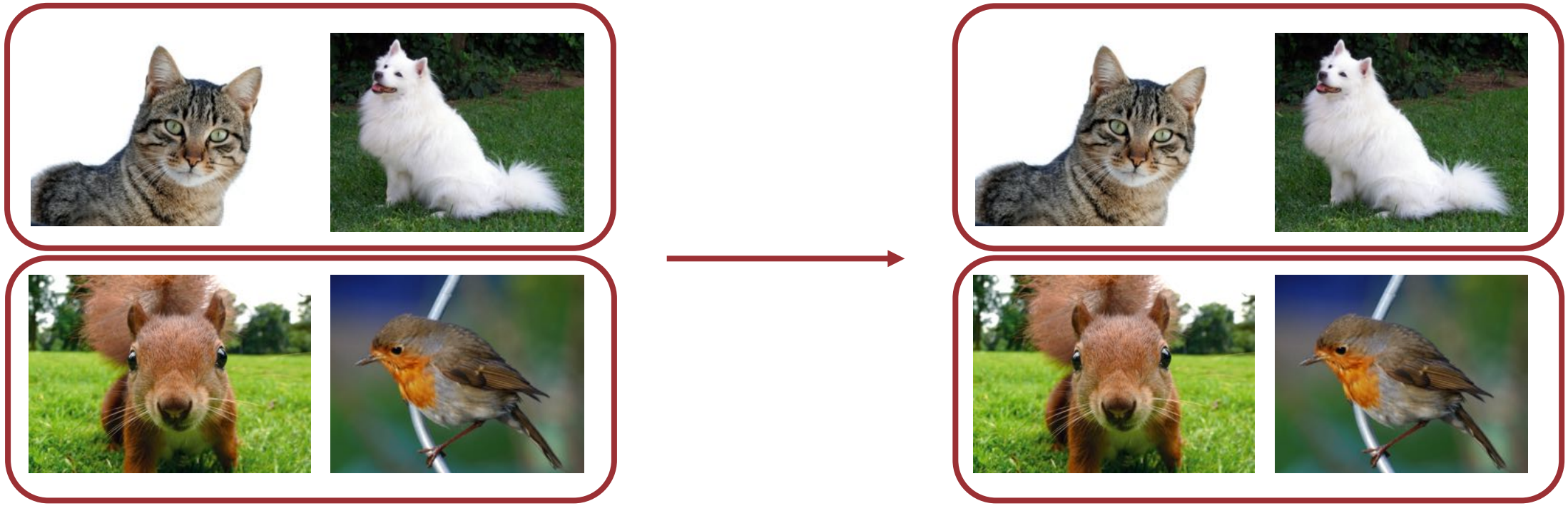


Cifar 100

Cifar 10

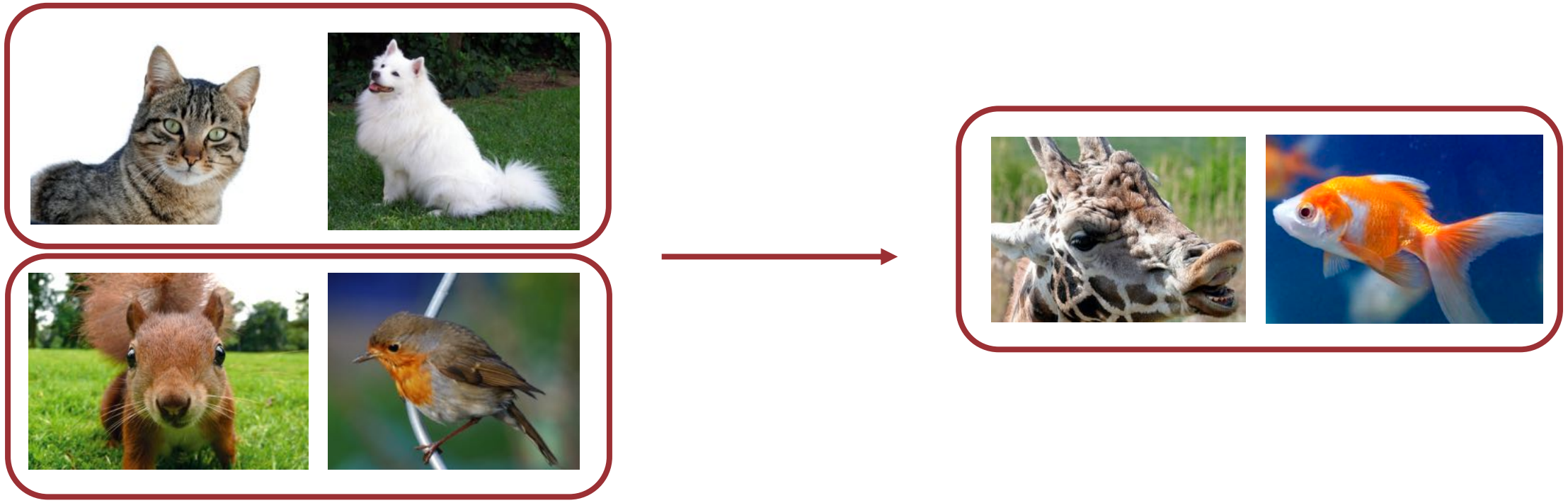
- Same nature of data
- Only train last fully connected layer

Multi-Task Learning



- Perform a task from the already trained tasks

Meta-Learning



- Perform a *new, never seen* task

MAML

An algorithm for solving few-shot problems

MAML: Key Ideas

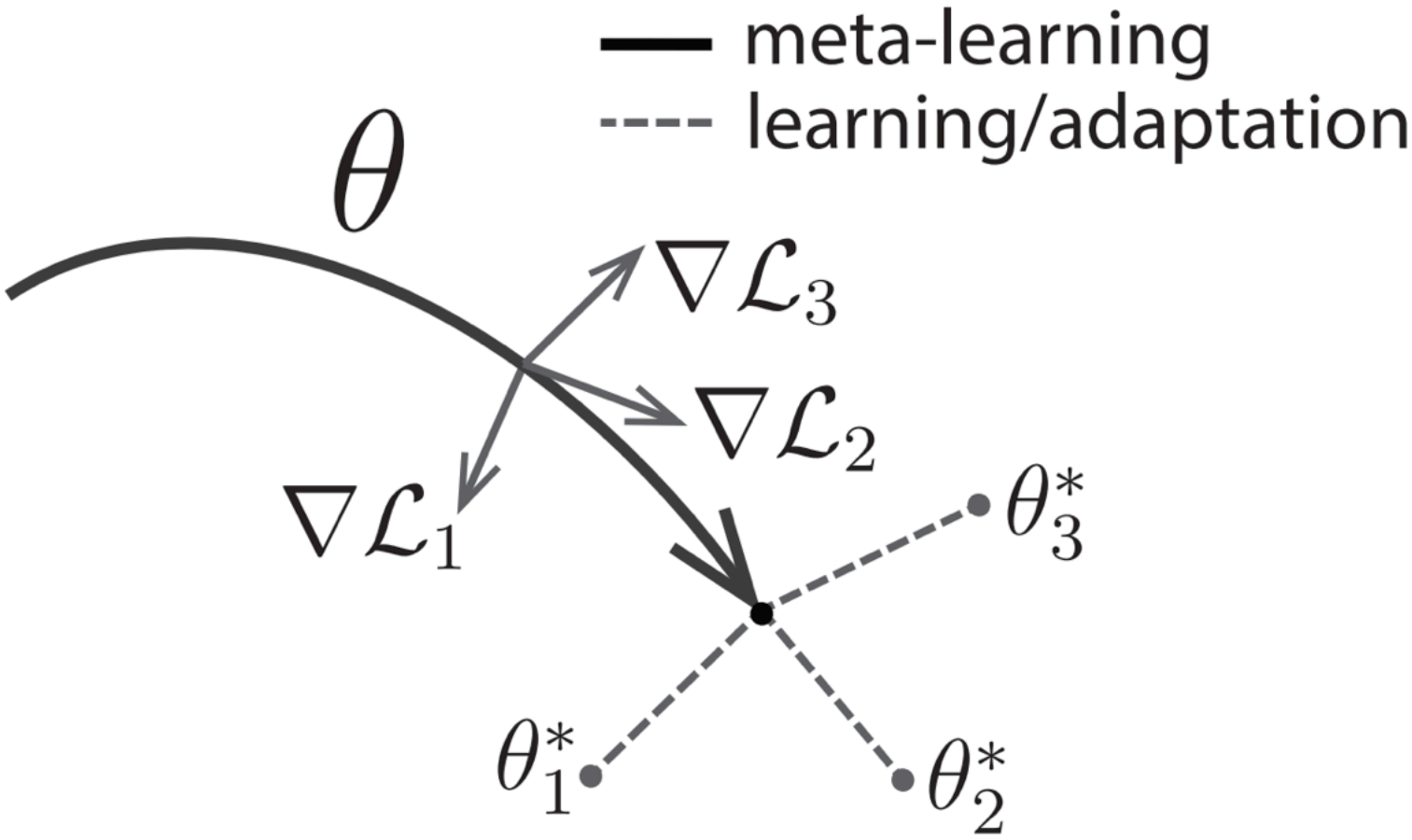
- General to the task
- Model-agnostic
- Train initialization of model for quick adaption
- Train in batches of tasks
- No expansion of parameters needed

MAML: Key Ideas

- **feature learning standpoint**
  - Build internal representation that is broadly suitable
  - Some features are more transferable than others

- **dynamical systems standpoint**
  - Maximizing the sensitivity of loss function
  - Small changes to model can have great impact on loss

MAML: Training



[1]

MAML: Training

- **Fast-training**

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

"Model-copy"

- **Meta-training**

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

"Meta-model"

- **Meta-objective**

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

MAML: Training

Algorithm 1 MAML for Few-Shot Supervised Learning by Finn et al. in [1, 2]

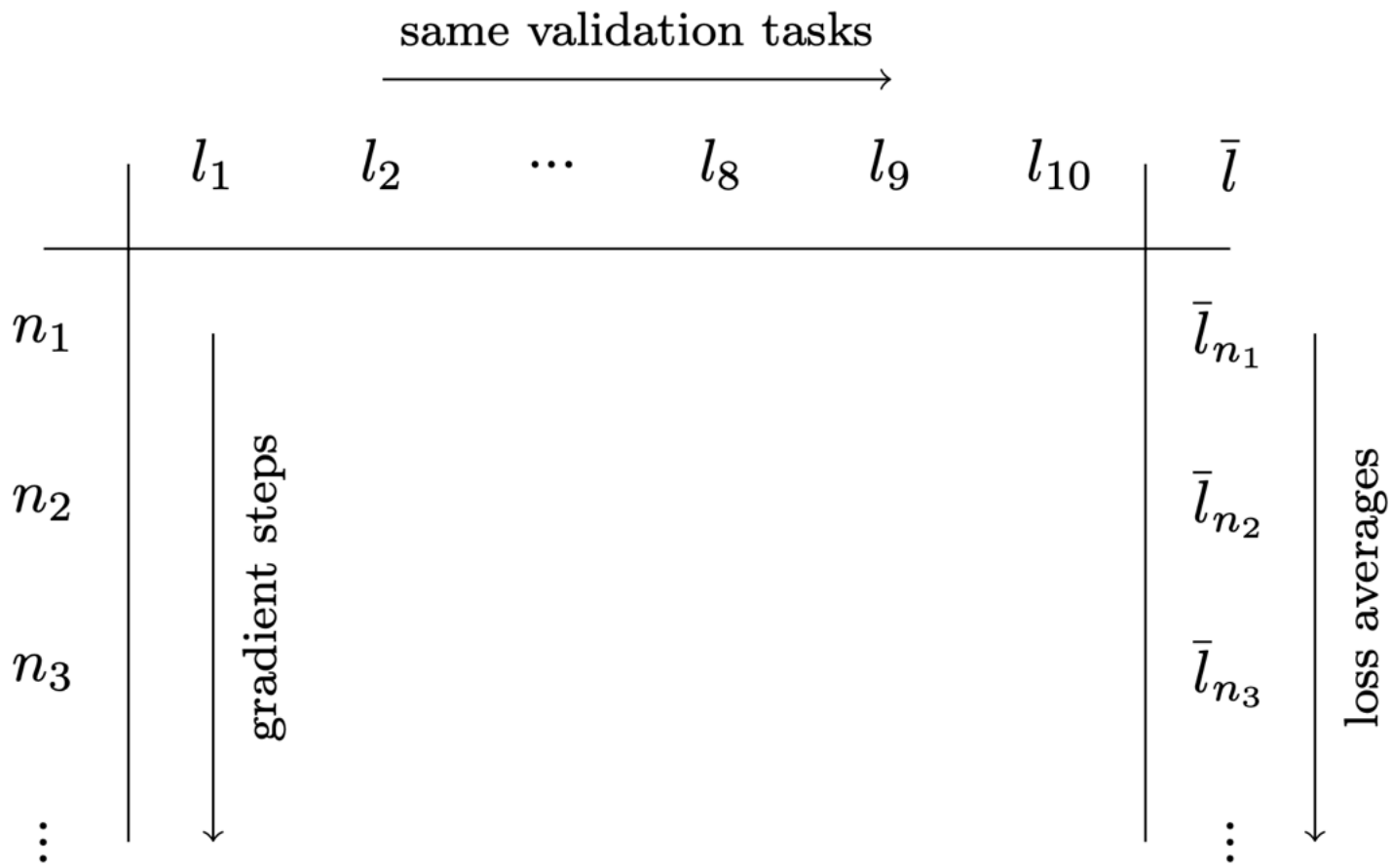
Require: $p(\mathcal{T})$ distribution over tasks
Require: α, β learning rate hyperparameters

1: Randomly initialize θ
2: **while** not done **do**
3: Sample batch of task $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample $k_{support}$ datapoints $\mathcal{D}_{\mathcal{T}_i}^{tr} = \{x^{(i)}, y^{(i)}\}$ from $\mathcal{D}_{\mathcal{T}_i}$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample k_{query} datapoints $\mathcal{D}_{\mathcal{T}_i}^{test} = \{x^{(i)}, y^{(i)}\}$ from $\mathcal{D}_{\mathcal{T}_i}$ depending on $\mathcal{D}_{\mathcal{T}_i}^{tr}$ 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 $\mathcal{D}_{\mathcal{T}_i}^{test}$ and $\mathcal{L}_{\mathcal{T}_i}$
11: **end while**

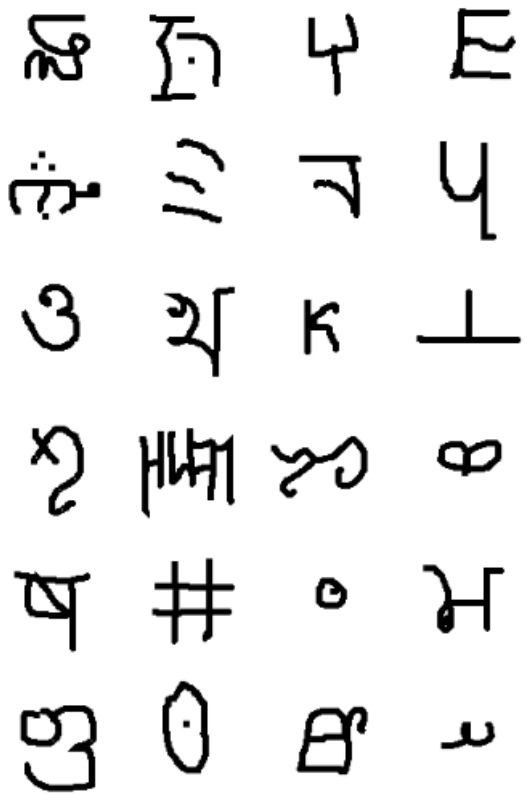
MAML: Validation

- "Validation like training"
- We used 10 validation tasks
- After every run do:
  - Copy meta-model
  - for every of n gradient steps do:
    - Fine-tune meta-model with $k_{support}$ samples
    - Evaluate meta-model with k_query samples

MAML: Validation



MAML: Classification Datasets



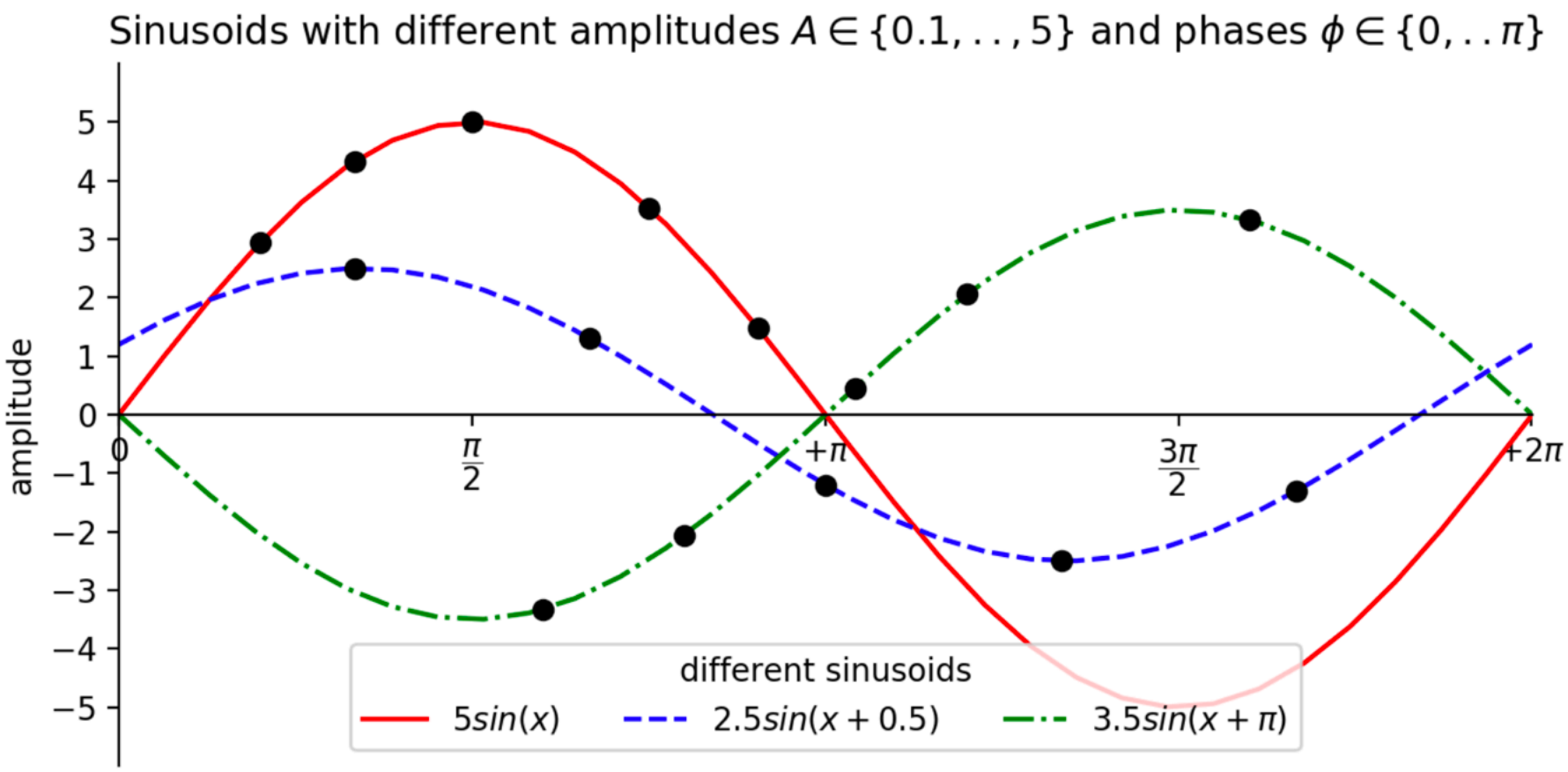
Omniglot

MiniImageNet

MAML: Classification Task

$$\begin{bmatrix}0\\0\\1\\0\\0\end{bmatrix} \quad \begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix} \quad \begin{bmatrix}0\\0\\0\\0\\1\end{bmatrix} \quad \begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix} \quad \begin{bmatrix}0\\0\\0\\1\\0\end{bmatrix}$$

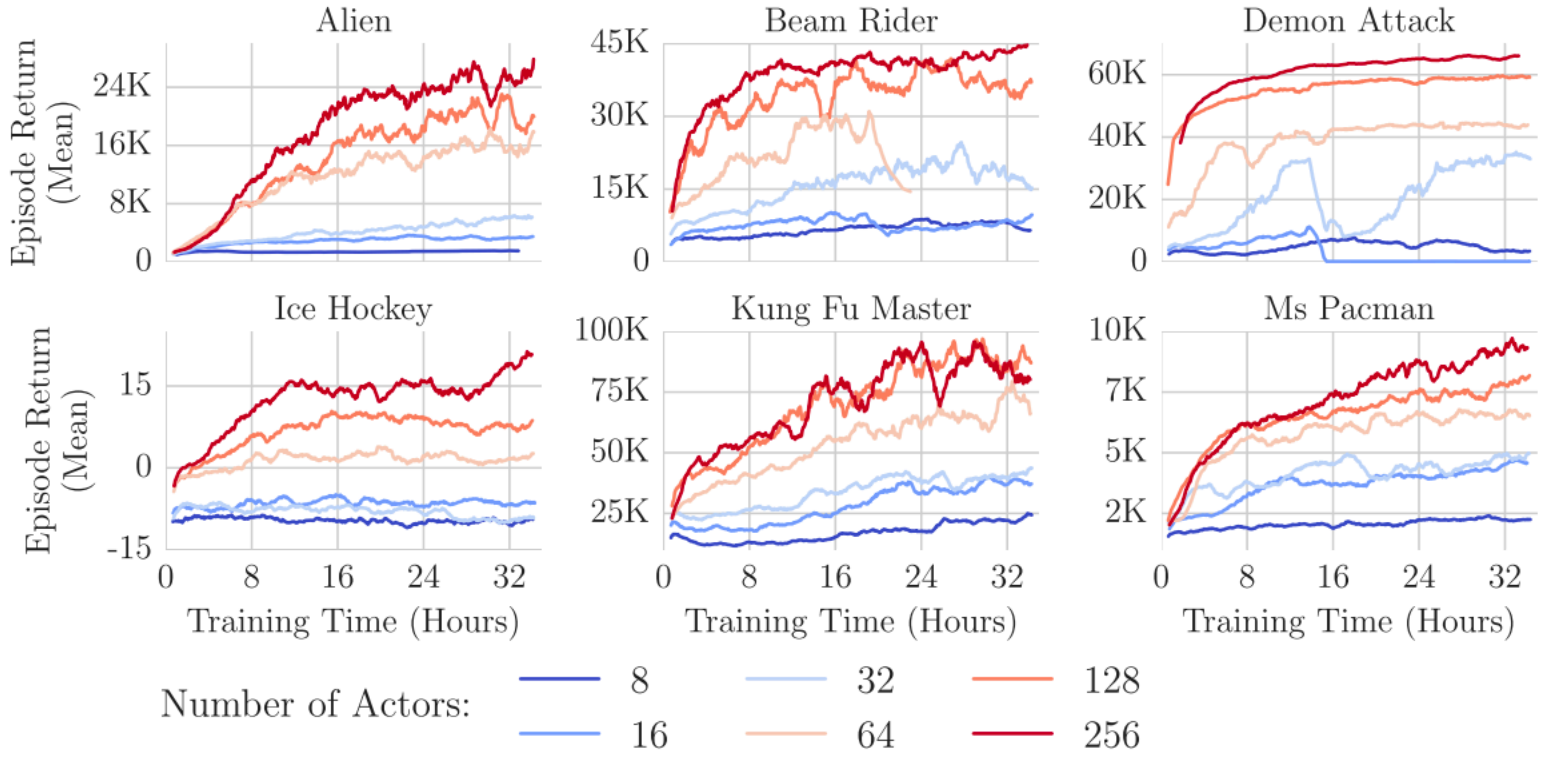
MAML: Regression Task



DMAML

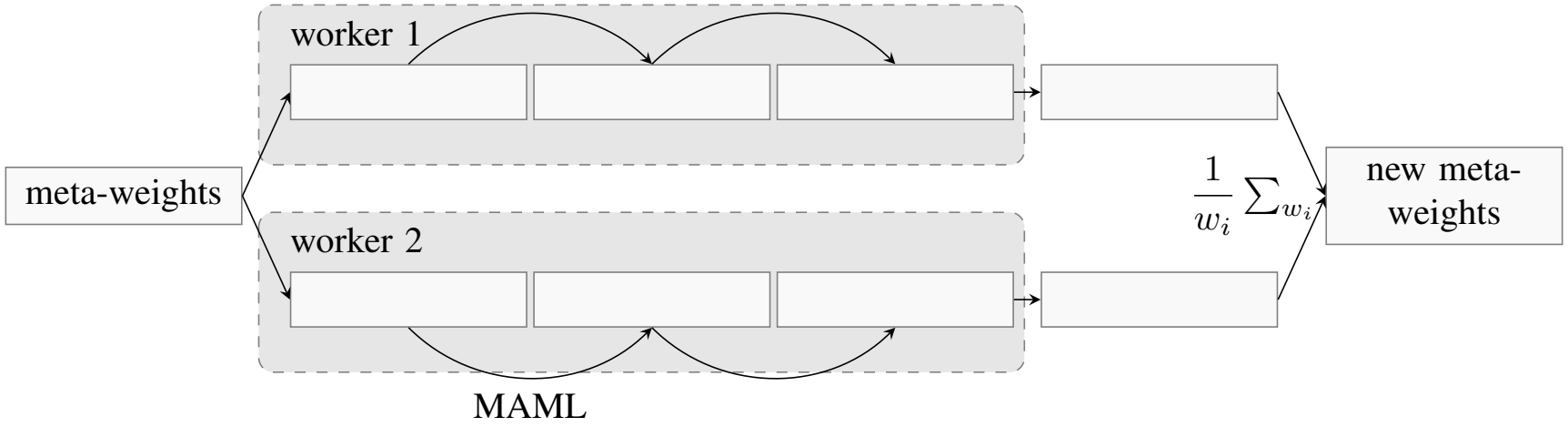
A parallelization of MAML

DMAML: Idea



From [4]: Scaling the number of actors in Reinforcement Learning Area with Ape-X.

DMAML: Schematic



Experiments

Experiments

- Only selected experiments
- Mainly Omniglot experiments
- Except implementation test experiments always ci95 with polyfit function with grade n = 10

Implementation Test: Omniglot

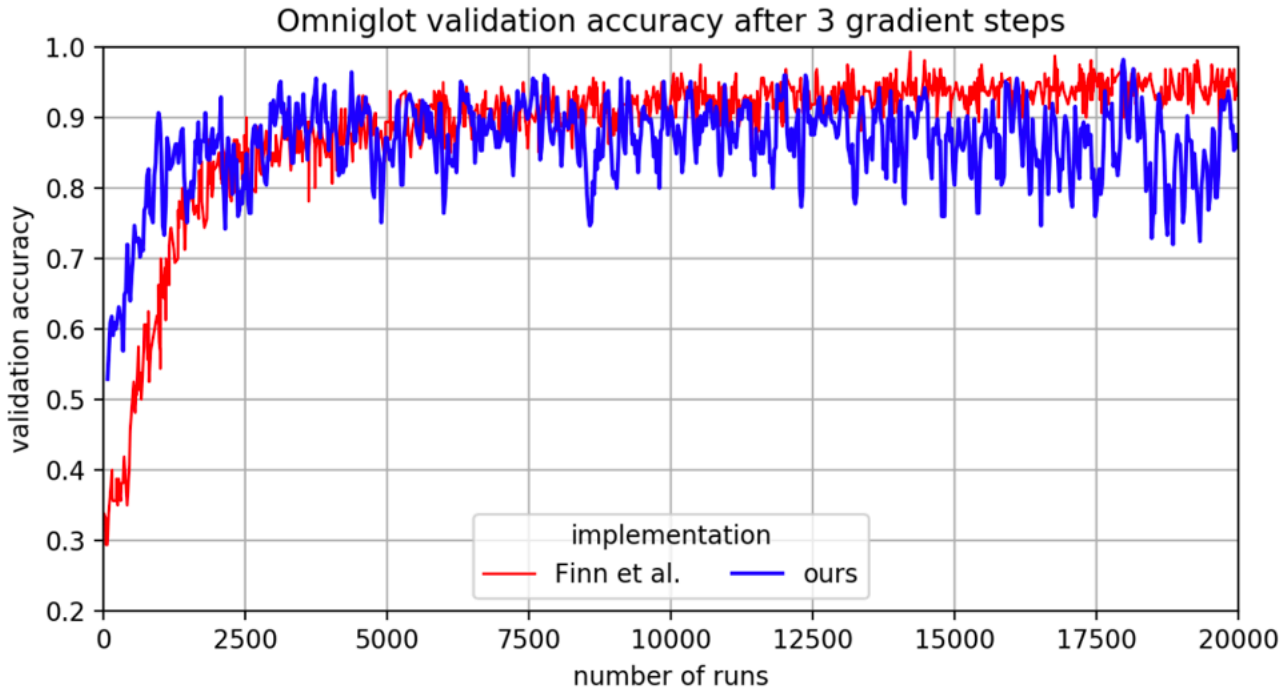


Figure 6.1: Omniglot implementation test validation accuracy after 3 gradient steps.

Implementation Test: MiniImageNet

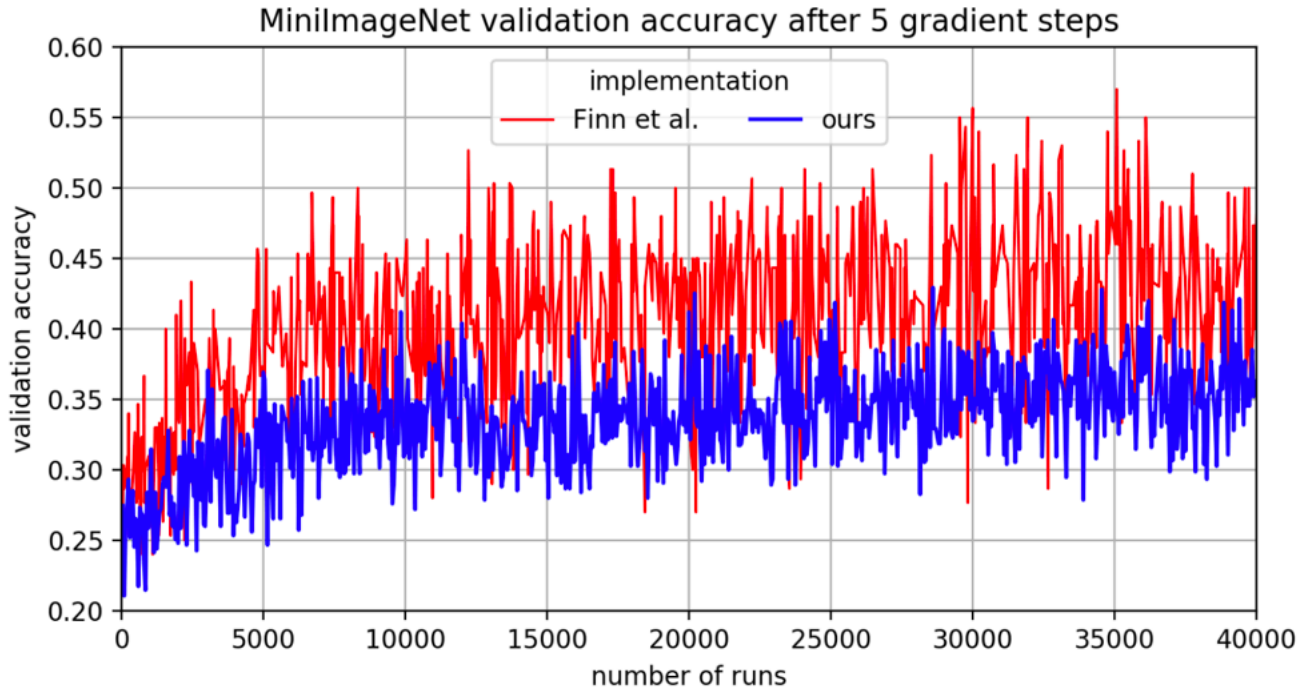


Figure 6.2: MiniImageNet implementation test validation accuracy after 5 gradient steps.

Implementation Test: Regression

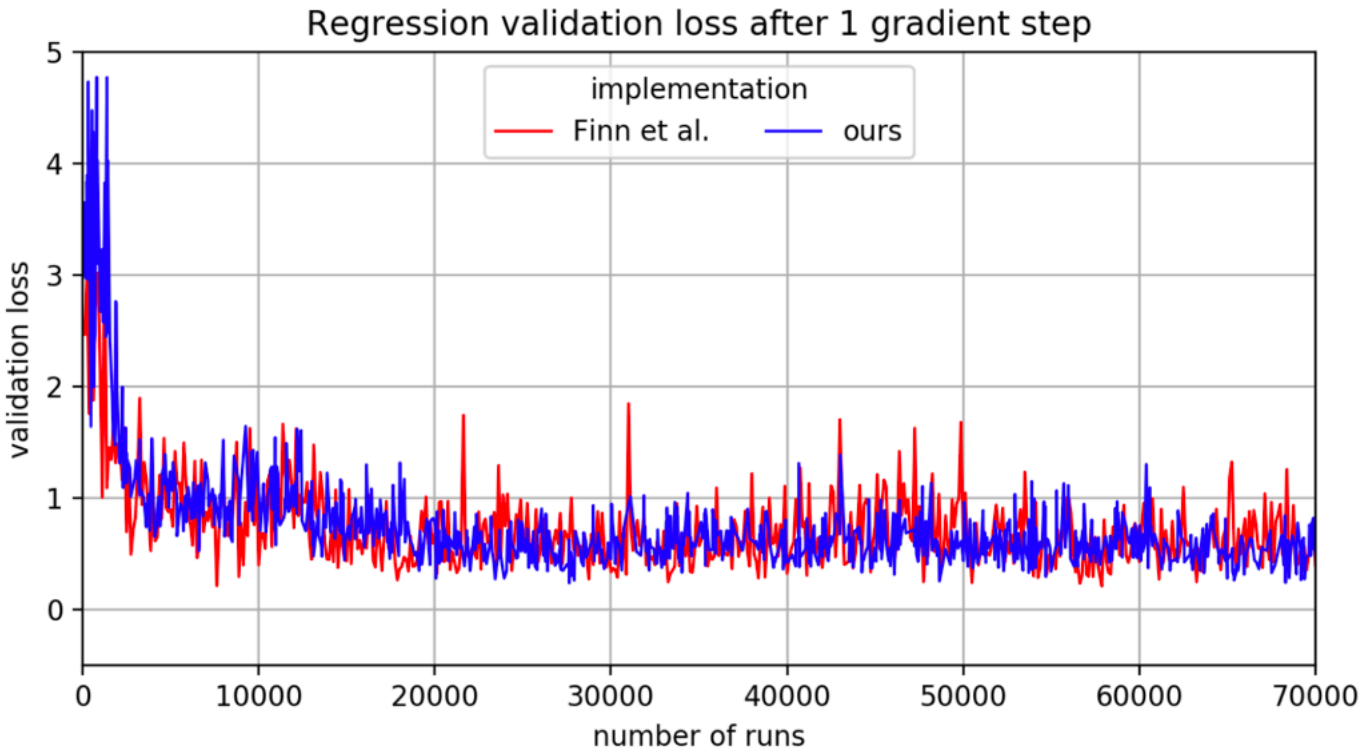


Figure 6.3: Regression implementation validation loss after one gradient step.

First Parallelization Experiment

Figure 6.4: Omniglot validation accuracy with different number of workers. Parameters of [5] have been used.

β-Hyperparameter Search Omniglot

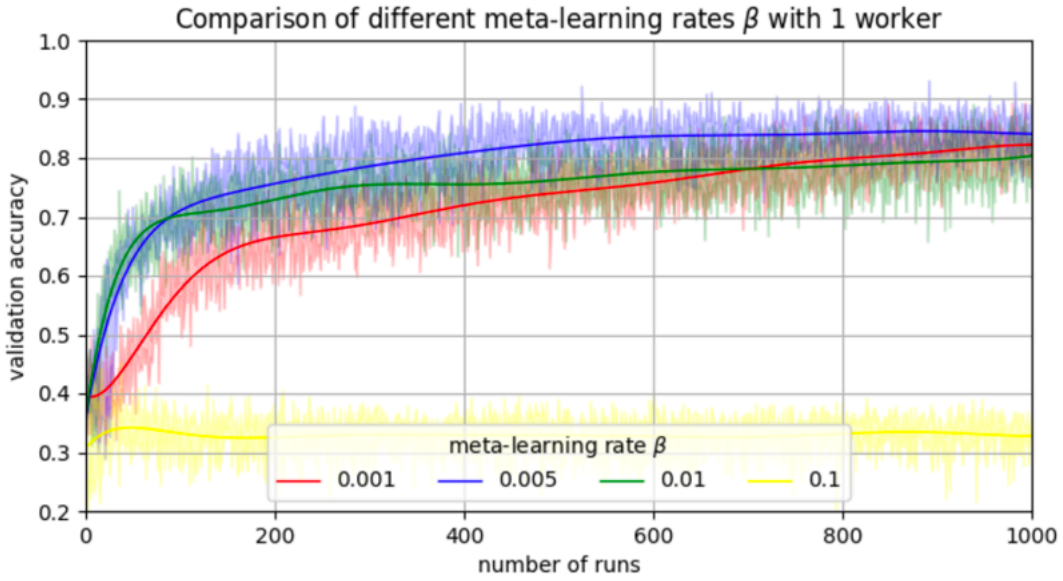


Figure 6.5: Omniglot validation accuracy with different values for meta-learning rate β after 3 validation steps and 1 worker

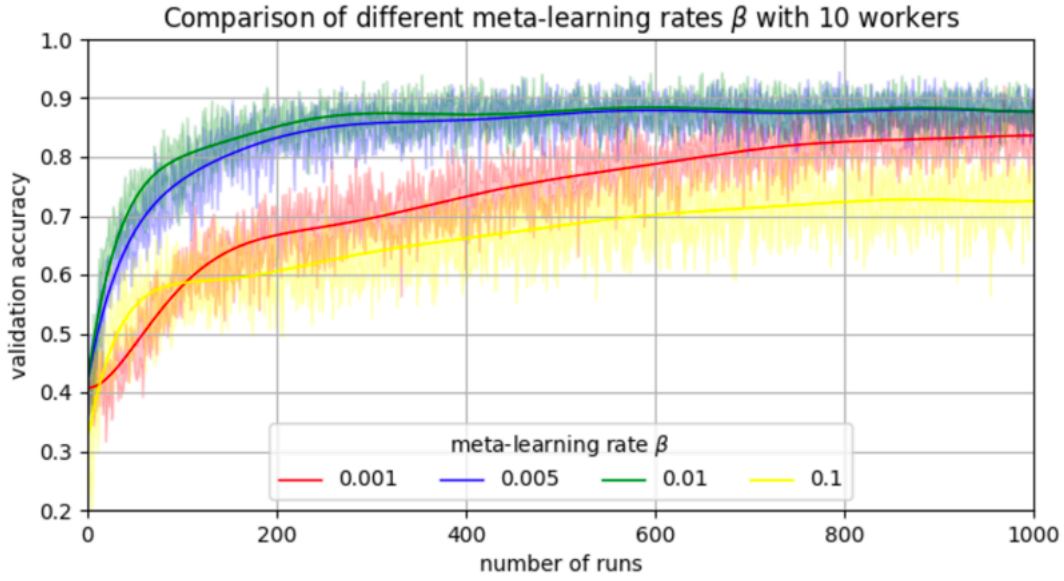


Figure 6.6: Omniglot validation accuracy with different values for meta-learning rate β after 3 validation steps and 10 workers

β-Hyperparameter Search MiniImageNet

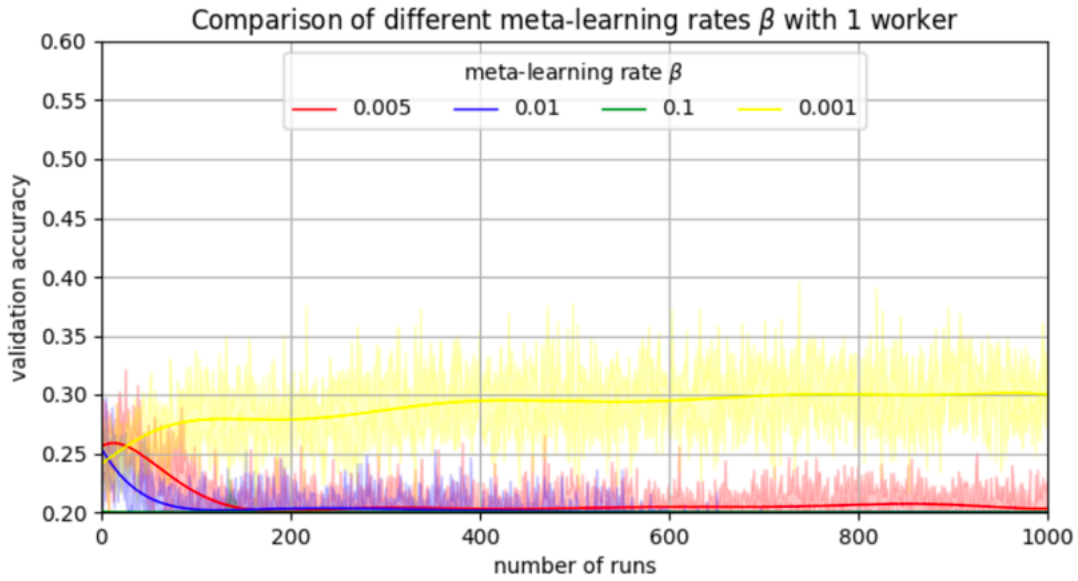


Figure 6.7: MiniImageNet validation accuracy with different values for meta-learning rate β after 10 validation steps for 1 worker

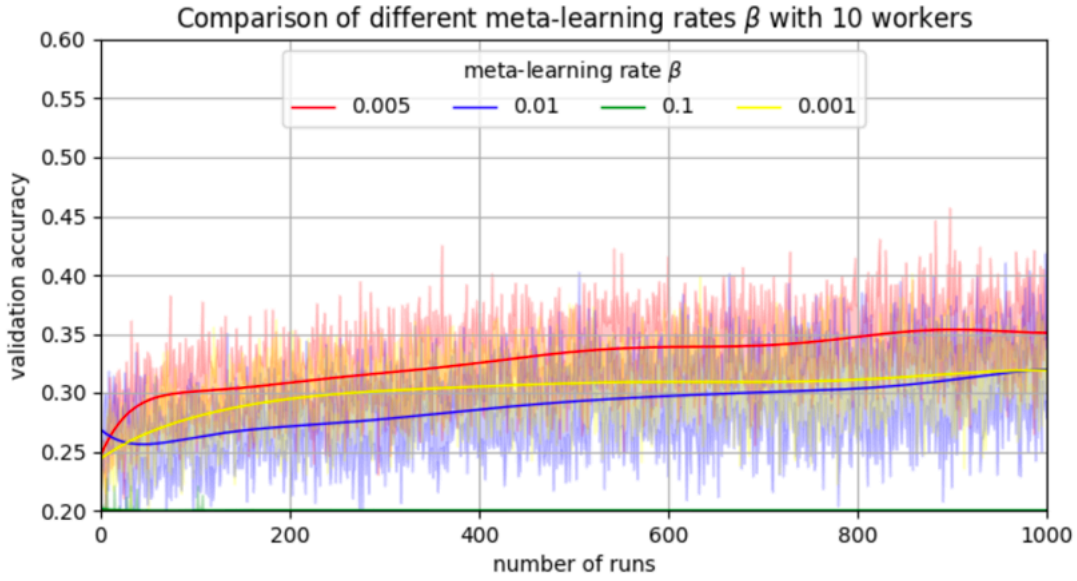


Figure 6.8: MiniImageNet validation accuracy with different values for meta-learning rate β after 10 validation steps for 10 workers

β-Hyperparameter Search Regression

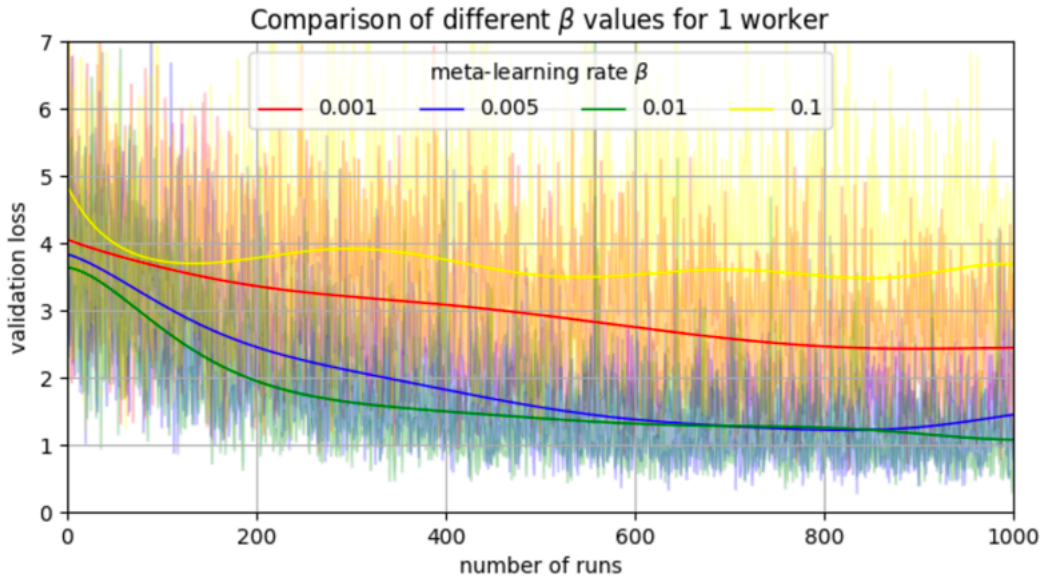


Figure 6.9: Sinusoid validation loss with different values for meta-learning rate β after 1 validation step for 1 worker

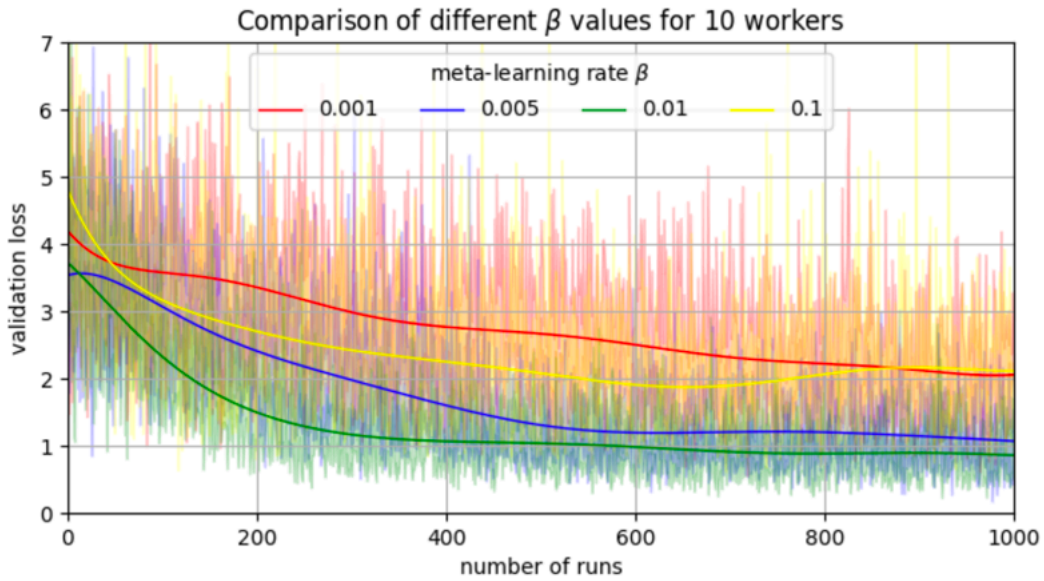


Figure 6.10: Sinusoid validation loss with different values for meta-learning rate β after 1 validation step for 10 workers

Second Parallelization Experiment

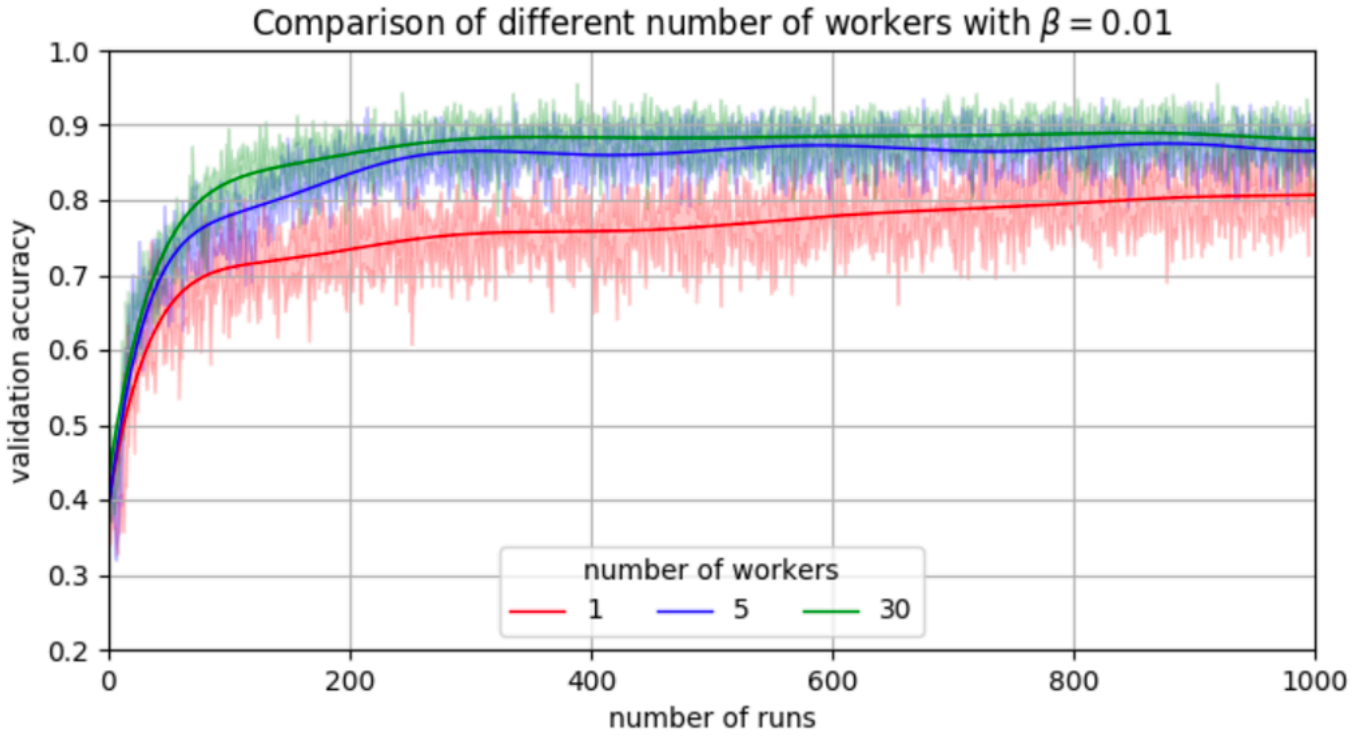


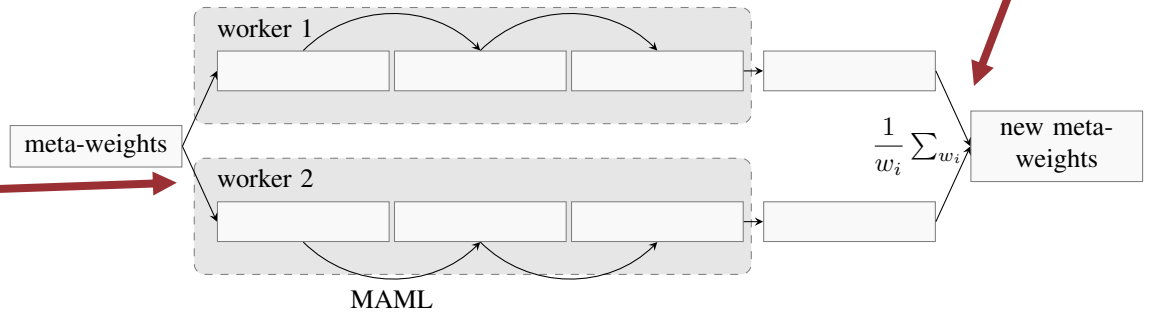
Figure 6.11: Omniglot validation accuracy after 3 validation steps with small number of workers

Influence of parameters

Gradient steps, workers, epochs, tasks

Schematic of parameters

(Generator)



# of parallel workers

Algorithm 1 MAML for Few-Shot Supervised Learning by Finn et al. in [1, 2]

Require: $p(\mathcal{T})$ distribution over tasks
Require: α, β learning rate hyperparameters
1: Randomly initialize θ
2: **while** not done **do**
3: Sample batch of task $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample $k_{support}$ datapoints $\mathcal{D}^{tr}_{\mathcal{T}_i} = \{x^{(i)}, y^{(i)}\}$ from $\mathcal{D}_{\mathcal{T}_i}$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$
7: Compute adapted parameters with gradient descent: $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample k_{query} datapoints $\mathcal{D}^{test}_{\mathcal{T}_i} = \{x^{(i)}, y^{(i)}\}$ from $\mathcal{D}_{\mathcal{T}_i}$ depending on $\mathcal{D}^{tr}_{\mathcal{T}_i}$ 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 $\mathcal{D}^{test}_{\mathcal{T}_i}$ and $\mathcal{L}_{\mathcal{T}_i}$
11: **end while**

# of epochs / MAML Steps

# of tasks

gradient steps

Influence of *gradient steps*

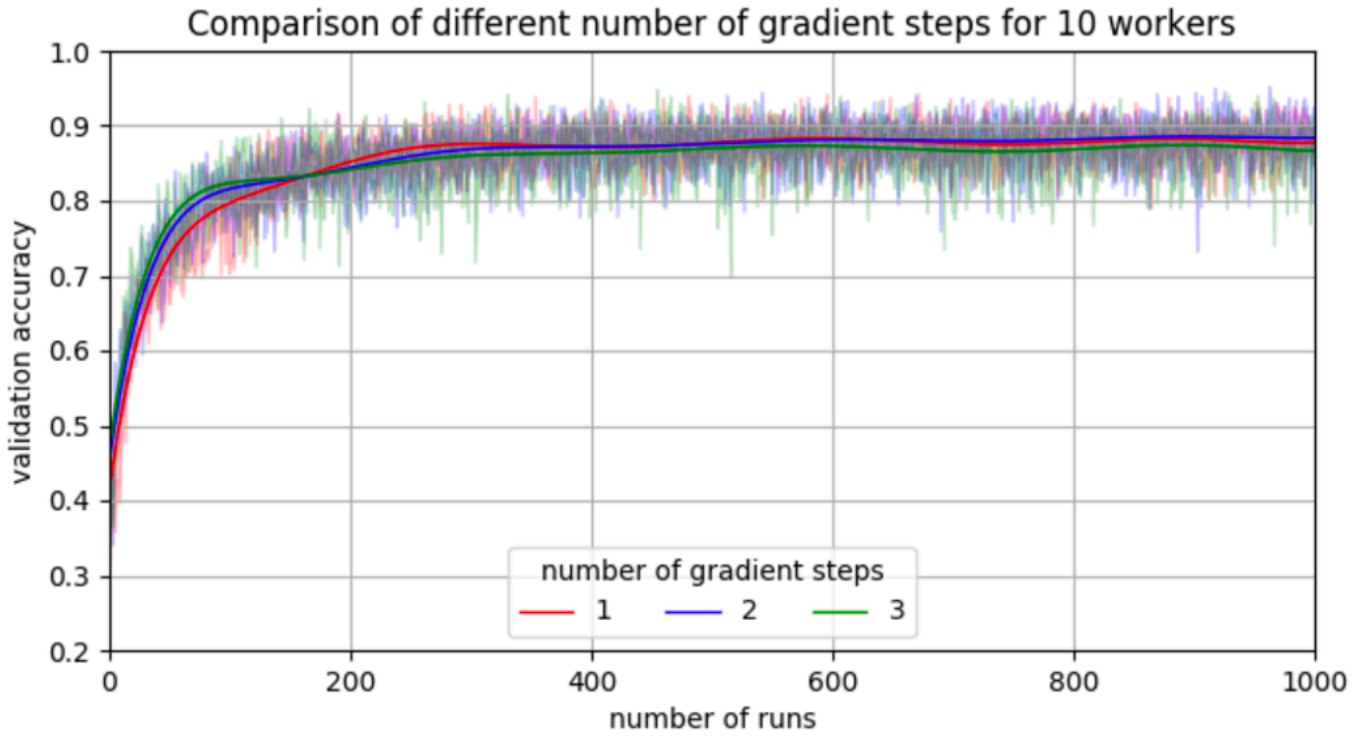


Figure 6.16: Omniglot validation accuracy with different number of gradient descent steps for 10 workers

Criteria

- What happens to the slope of the accuracy graph?
- What are the time costs? (wall time!)

Comparison

Table 6.2: Values of the Parameters With Different Scaling Ratios

scaling ratio	**1**	**2**	**4**	**8**	**16**
number of workers	1	2	4	8	16
nuber of epochs	1	2	4	8	16
number of tasks	32	64	128	256	512
seen tasks per run	32	64	128	256	512

MAML

Influence of *workers*

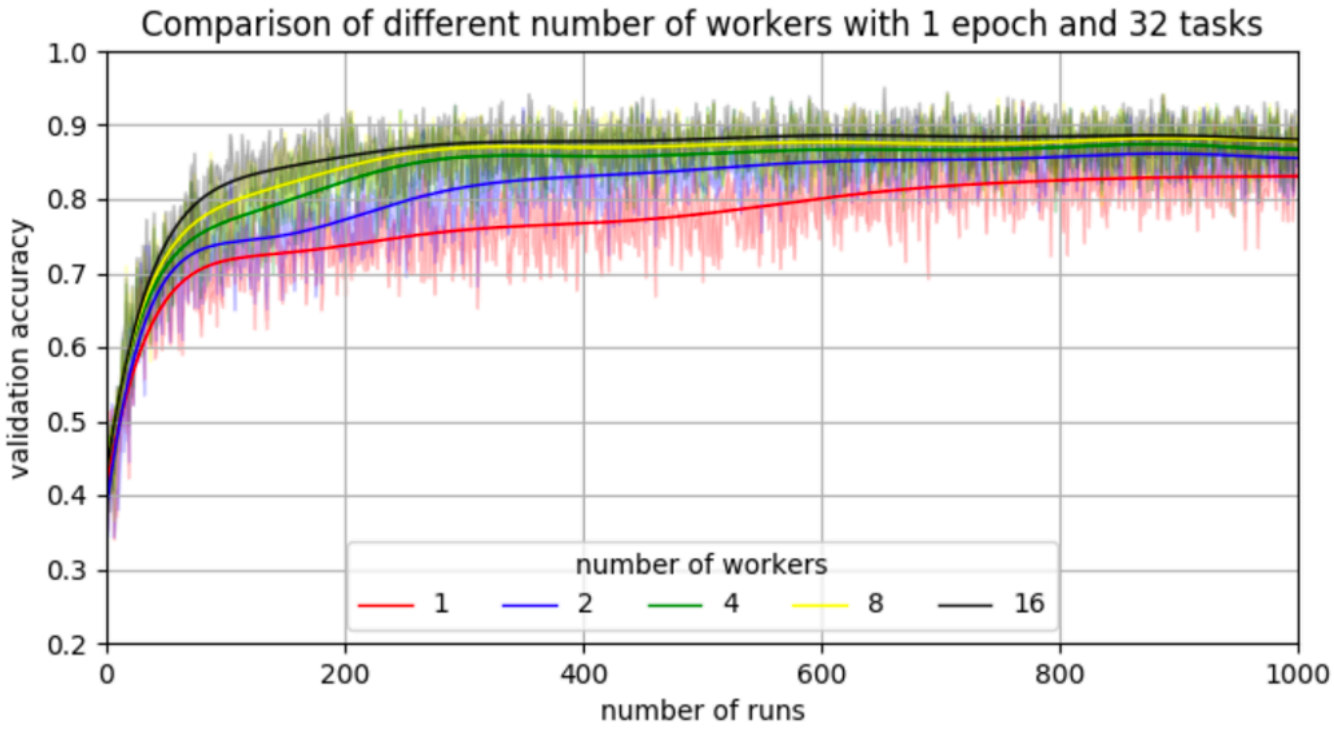


Figure 6.17: Comparison of the validation accuracy after 3 gradient steps with different number of workers.

Influence of *epochs*

Figure 6.18: Comparison of the validation accuracy after 3 gradient steps with different number of epochs.

Influence of *tasks*

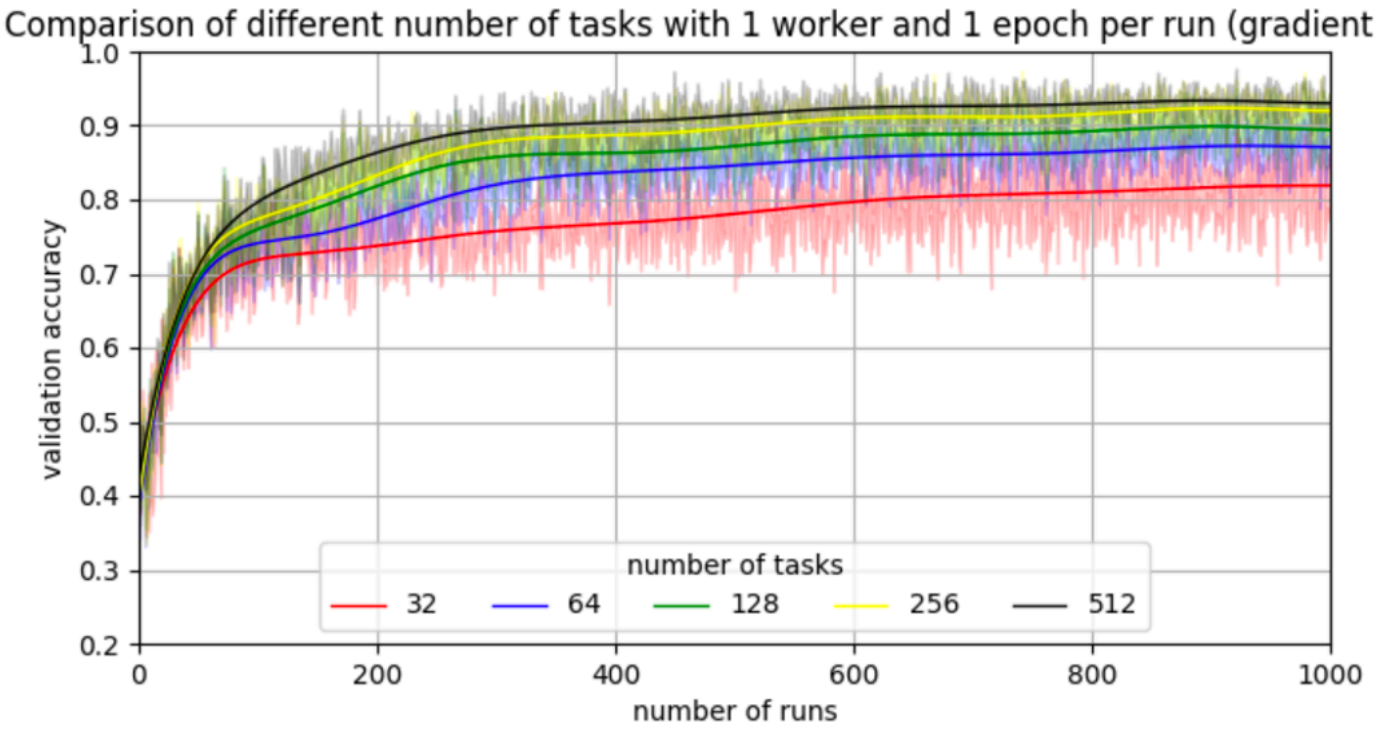


Figure 6.19: Comparison of the validation accuracy after 3 gradient steps with different number of tasks and usage of gradient sum

Comparison by accuracy

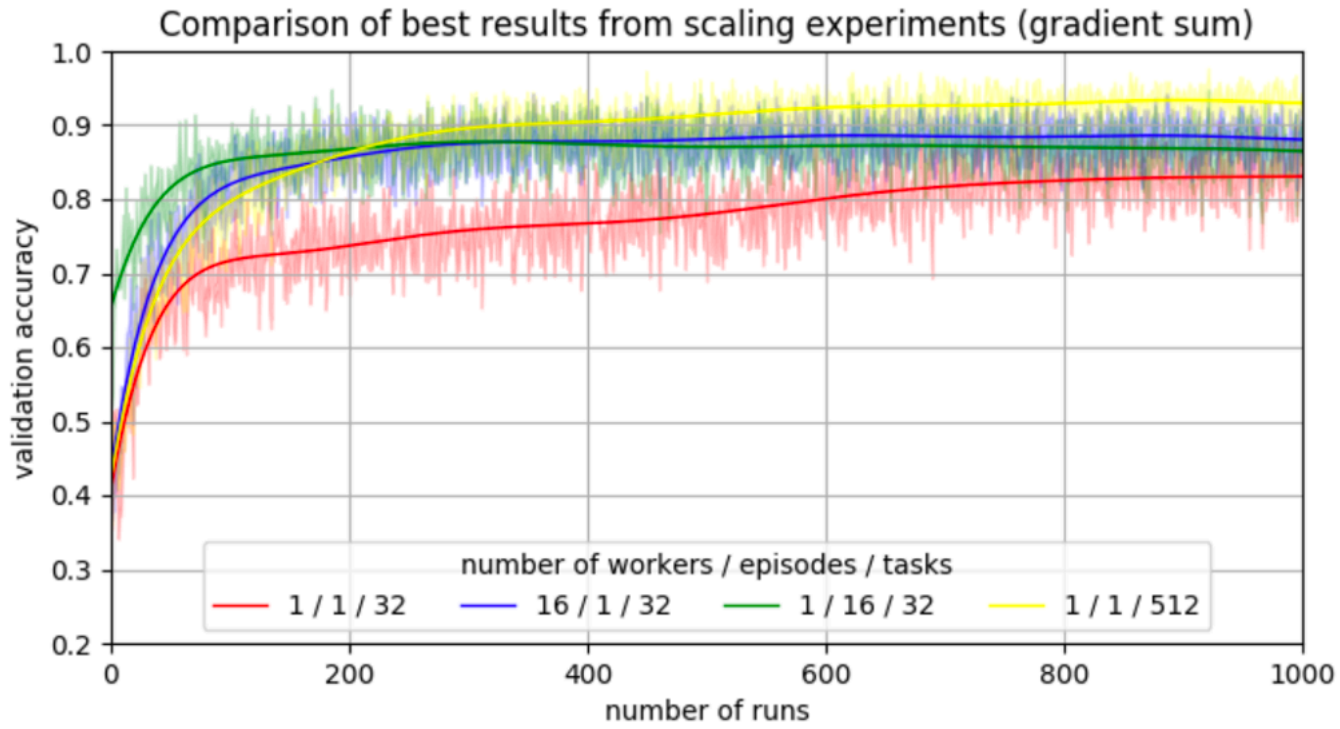


Figure 6.23: Comparison of validation accuracies with gradient sum: The graphs with one parameter each increased by ratio 16 are shown.

Comparison by accuracy

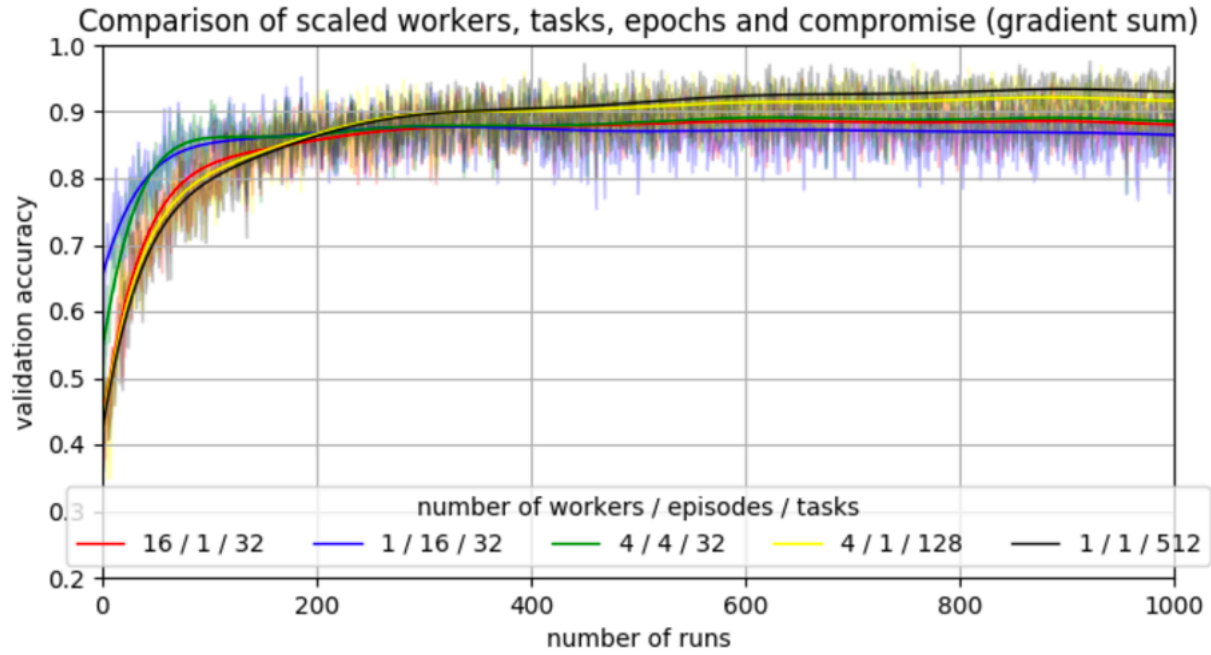


Figure 6.24: Comparison of validation accuracies with gradient sum: The graphs with one parameter each increased by ratio 16 are shown together with those, where two parameters have been increased with the same proportion.

Comparison by wall time

Table 6.4: Relative Increase in Average Duration in Percent of a Run With Different Scaling Ratio

scaling ratio	**1**	**2**	**4**	**8**	**16**
number of workers	100	97.67	124.96	131.01	143.03
number of epochs	100	194.26	374.03	742.42	1498.17
number of tasks	100	195.8	376.96	745.58	1520.85
seen tasks per run	32	64	128	256	512

Comparison by wall time

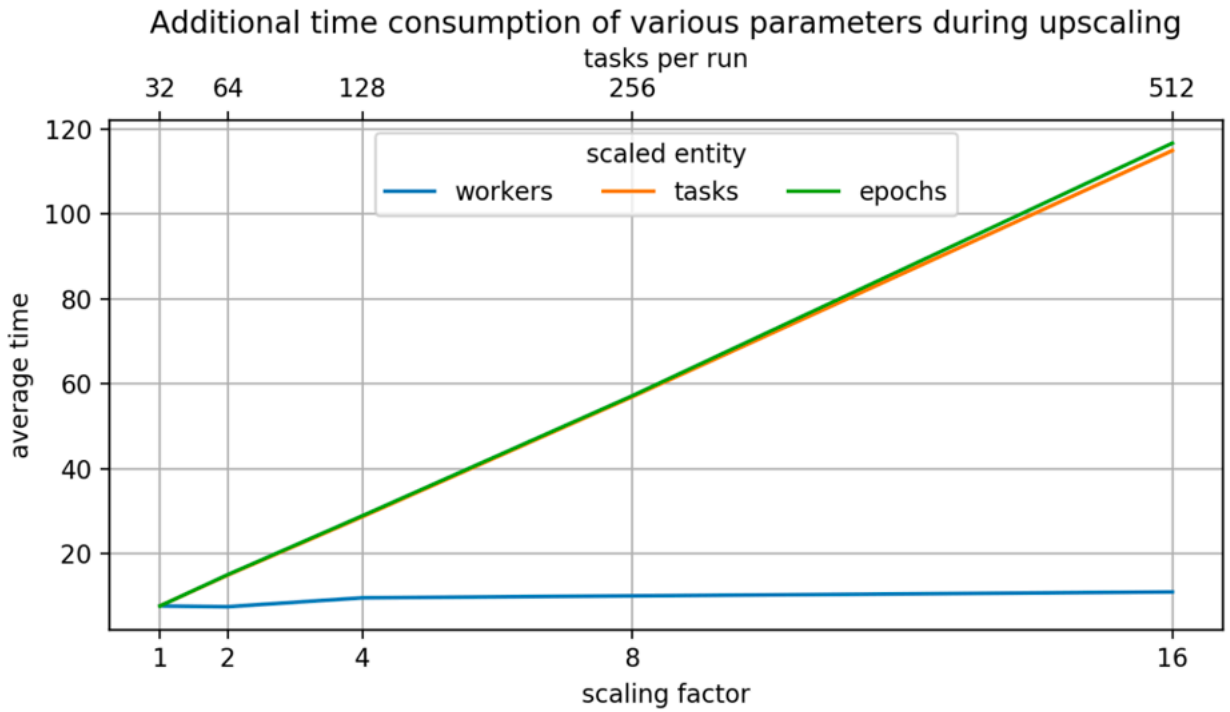


Figure 6.25: Comparison of the average wall time of a single run with different scaling factors for experiments with increased parameters workers, tasks, epochs.

Comparison by wall time

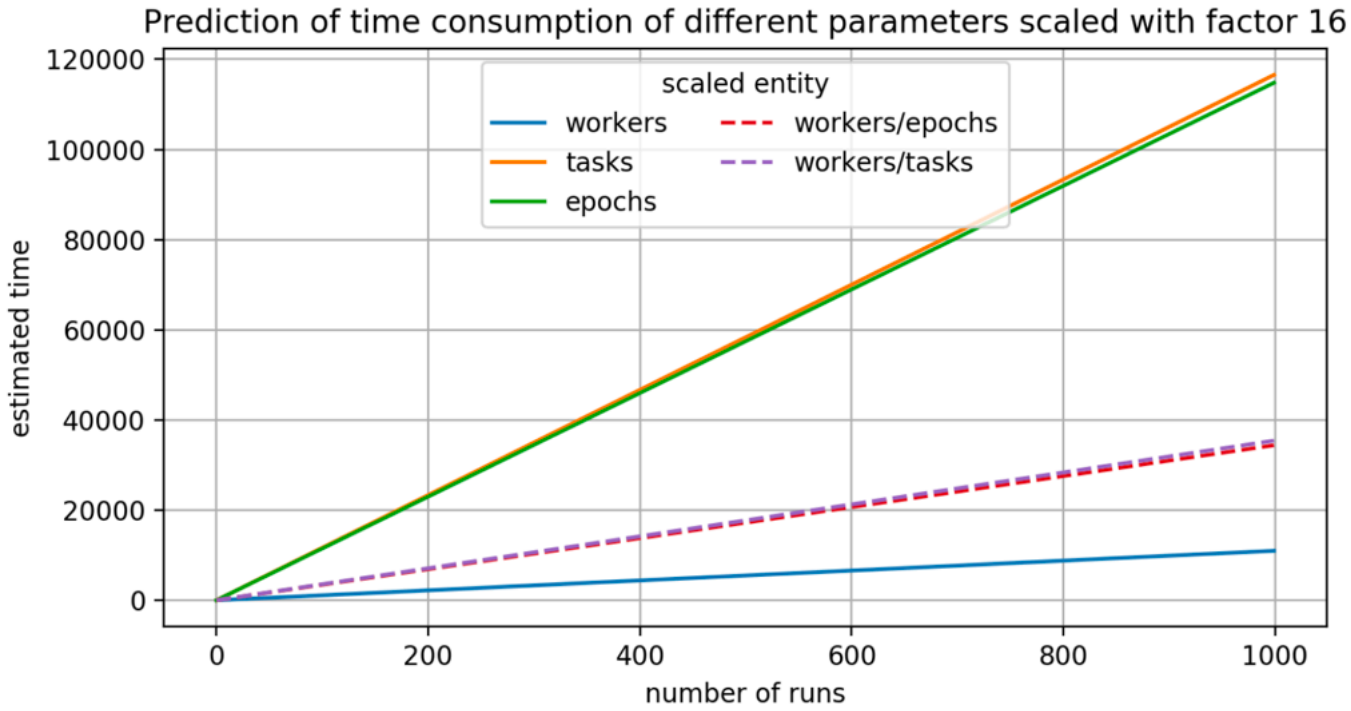


Figure 6.26: Estimated wall times for 1000 runs with different parameters scaled with a ratio of 16.

Conclusions

Conclusions

- Parallelization of MAML does work
- β needs hyperparameter search
- Increasing workers does not result in proportional speed up in accuracy
- Increase of gradient steps does not speed up learning

Conclusions

- Increase of *workers, epochs and tasks* lead to acceleration of accuracy **per run**
- Highest increase **per run** with increase of **tasks**
- => MAML and DMAML mainly dependent on number of seen tasks

Conclusions

- Increase of workers perform worse than increase of tasks but gives a massive clock time advantage
- => **implementation**: mix increase of tasks and workers
- => **general: parallelization of tasks would be ideal**
  - **Reasons:**
    - **MAML mainly dependent on number of seen tasks**
    - **Overall accuracy (after first 200 runs) way higher with higher number of tasks**
    - **In complicated setting (e. g. RL setting) the computation of tasks is the ″thing″ that consumes the computation time**

Future Work

- Implementation of task parallelization
- Reinforcement learning experiments
- Weight generator experiments
- MetaWorld framework investigations

Bibliography

- [1] Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. *34th International Conference on Machine Learning, ICML 2017*, *3*, 1856–1868.
- [2] Finn, C. (2018). Learning to Learn with Gradients (University of California, Berkeley). Retrieved from http://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-105.html
- [3] Soulier, F. (2020). A model-agnostic meta-learning framework for solving few-shot tasks in parallel (TH Köln), *Master Thesis*

Bibliography

- [4] Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H., & Silver, D. (2018). *Distributed Prioritized Experience Replay*. 1–19. Retrieved from http://arxiv.org/abs/1803.00933
- [6] https://de.wikipedia.org/wiki/Konfidenzintervall#/media/Datei:Confidence_intervall_normal_dist.svg

Loss Functions

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

Mean-Squared Error

$$\mathcal{L}_{\mathcal{T}_i}(f_\theta) = -\sum_{i=1}^{C} y^{(i)} \log(\hat{y}^{(i)})$$

Categorical cross-entropy loss

$$\mathcal{L}_{\mathcal{T}_i}(f_\theta) = -\mathbb{E}_{x_t, y_t \sim f_\theta, \rho_{\mathcal{T}_i}} \left[\sum_{t=1}^{H} R_i(x_t, y_t) \right]$$

Estimated Reward

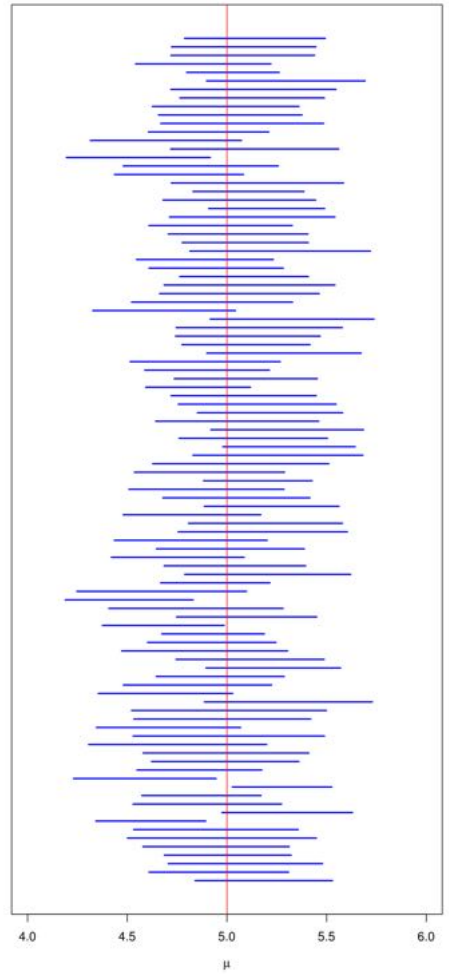
Confidence Intervals

$$t_1, t_2 = [\bar{x} \pm \frac{z(\gamma, n) \cdot \gamma}{\sqrt{n}}]$$

Gaussian distribution

experiments n = 3

Confidence level $\gamma = 0.95$



[6]

n = 100

96 Intervalle überdecken

4 Intervalle überdecken nicht

"The calculated confidence interval is with a probability of 95%
One of the intervals that contains the desired expected value."

Polyfit with 10 polynomials

MAML: Reinforcement Learning Task

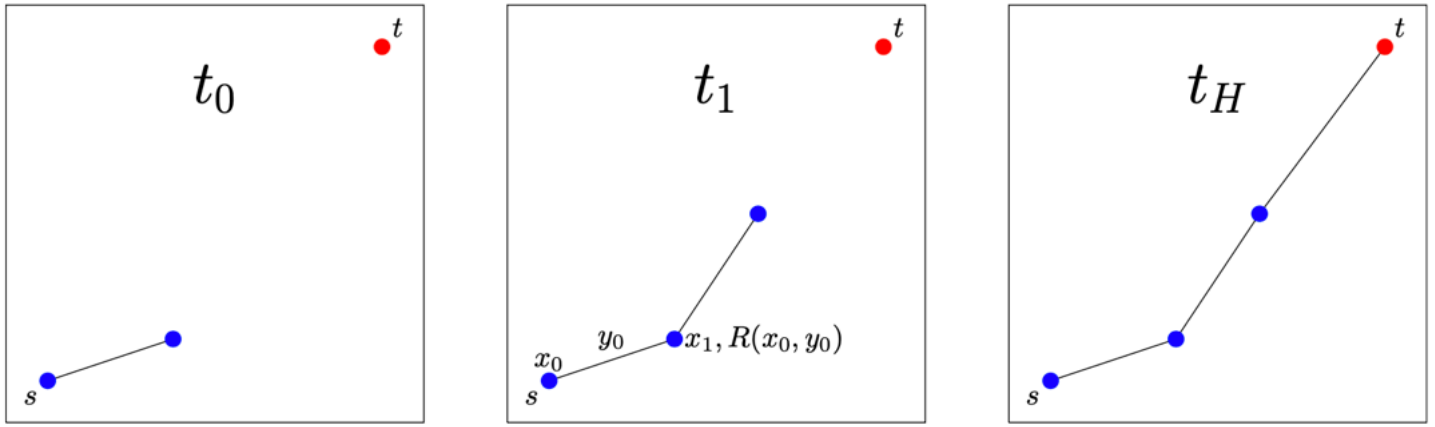
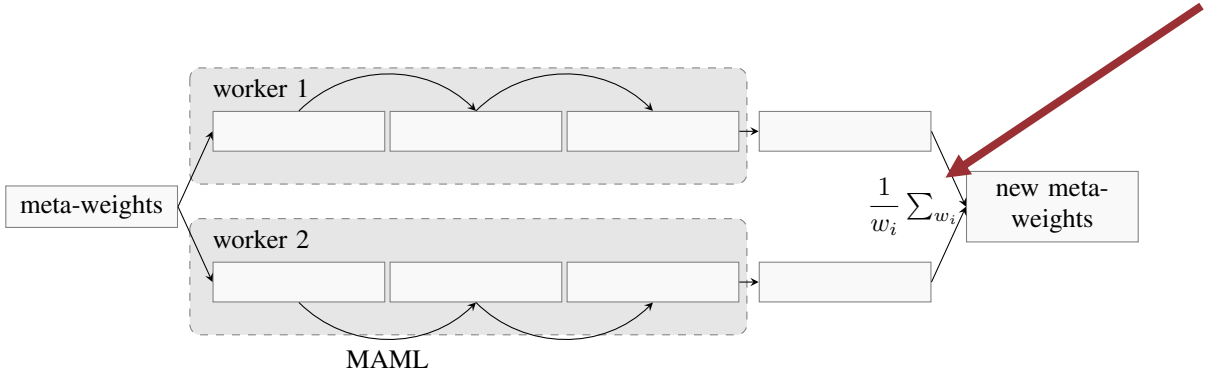


Figure 3.7: Sampling of a trajectory in a navigation environment: In this example is $k_{support} = 1$, which means that 1 trajectory is sampled for support set. A trajectory consists of $\mathcal{H} = 3$ transitions. In every transition a tuple $\{x_t, y_t\}$ with observation x_t and action y_t is recorded, as can be seen at t_1. For every transition tuple an imediate reward $R(x_t, y_t)$ is collected for the loss function. When the environment ends or $\mathcal{H}$ many samples are gathered, the trajectory is complete.

Weight Generators



- Arithmetic mean
- Loss median
- Weighted
- Loss moving average

Take the average of the last 5 losses per worker

Worker	**1**	**2**	**3**	**4**	**5**
loss	2	3	8	19	20
acc	0.89	0.88	0.60	0.43	0.40

Parameters

Table 5.1: Omniglot Classification Experiment Parameters.

Experiment parameters	Symbol	Value
Number of support samples	$k_{support}$	1
Number of query samples	k_{query}	1
Inner-learning rate	α	0.4
Meta-learning rate	β	0.01
Number of tasks	n	32
Number of epochs		1
Number of gradient steps		1
Number of validation steps		3
Number of validation tasks		10
Number of validation samples		1

Table 5.1: Omniglot Classification Experiment Parameters.

Experiment parameters	Symbol	Value
Number of support samples	$k_{support}$	1
Number of query samples	k_{query}	1
Inner-learning rate	α	0.4
Meta-learning rate	β	0.01
Number of tasks	n	32
Number of epochs		1
Number of gradient steps		1
Number of validation steps		3
Number of validation tasks		10
Number of validation samples		1

Table 5.1: Omniglot Classification Experiment Parameters.

Experiment parameters	Symbol	Value
Number of support samples	$k_{support}$	1
Number of query samples	k_{query}	1
Inner-learning rate	α	0.4
Meta-learning rate	β	0.01
Number of tasks	n	32
Number of epochs		1
Number of gradient steps		1
Number of validation steps		3
Number of validation tasks		10
Number of validation samples		1

Arithmetic MAML Variant

- "Fair" comparison of parameters
- Step size = learning rate * sum of gradient
- If number of tasks higher, sum bigger!
- Therefore arithmetic mean

$$\theta \leftarrow \theta - \beta \frac{1}{|\{\mathcal{T}_i\}|} \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$