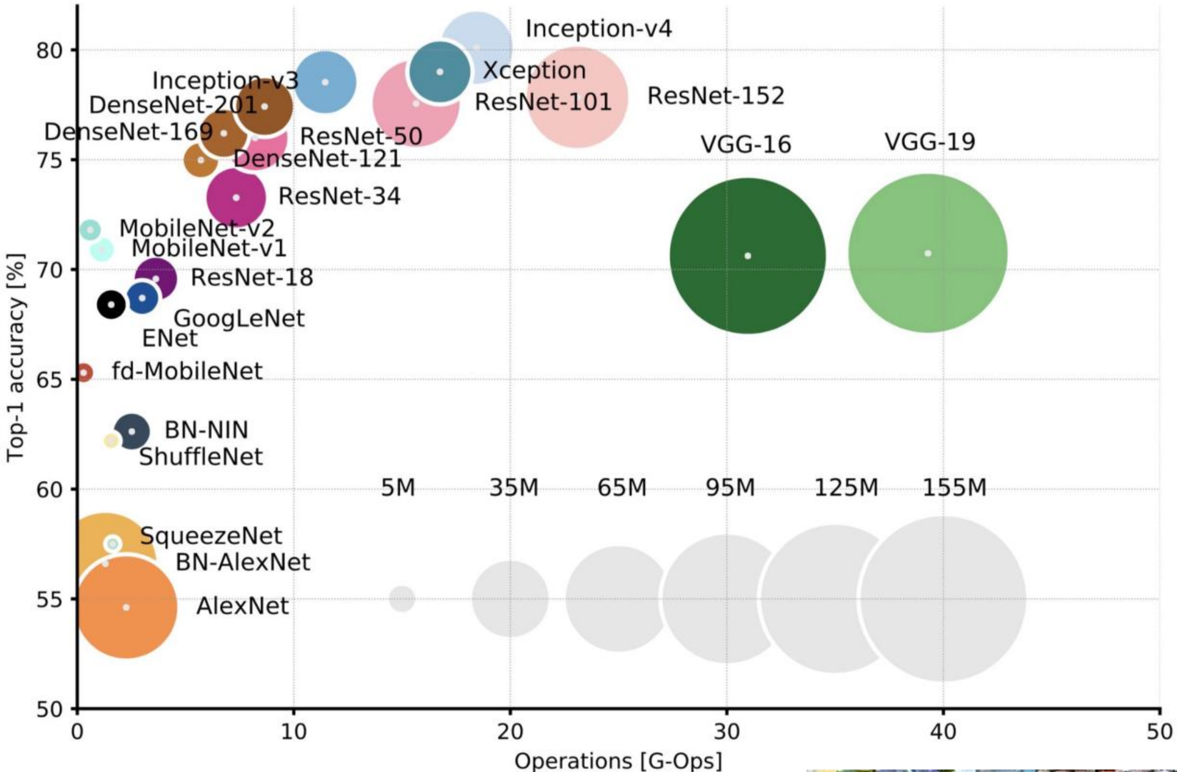
Meta-learning

Zhanyu Wang

PURDUE
UNIVERSITY

Problem in Deep Learning

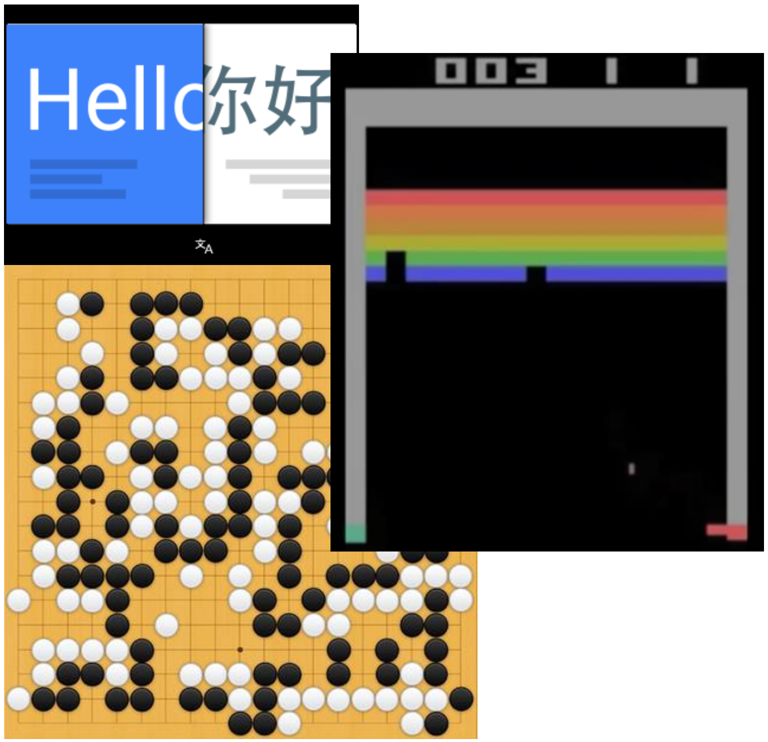


[Source: https://arxiv.org/pdf/1605.07678.pdf](https://arxiv.org/pdf/1605.07678.pdf)

[Source: https://medium.com/zylapp/review-of-deep-learning-algorithms-for-image-classification-5fdbca4a05e2](https://medium.com/zylapp/review-of-deep-learning-algorithms-for-image-classification-5fdbca4a05e2)

Moravec's Paradox

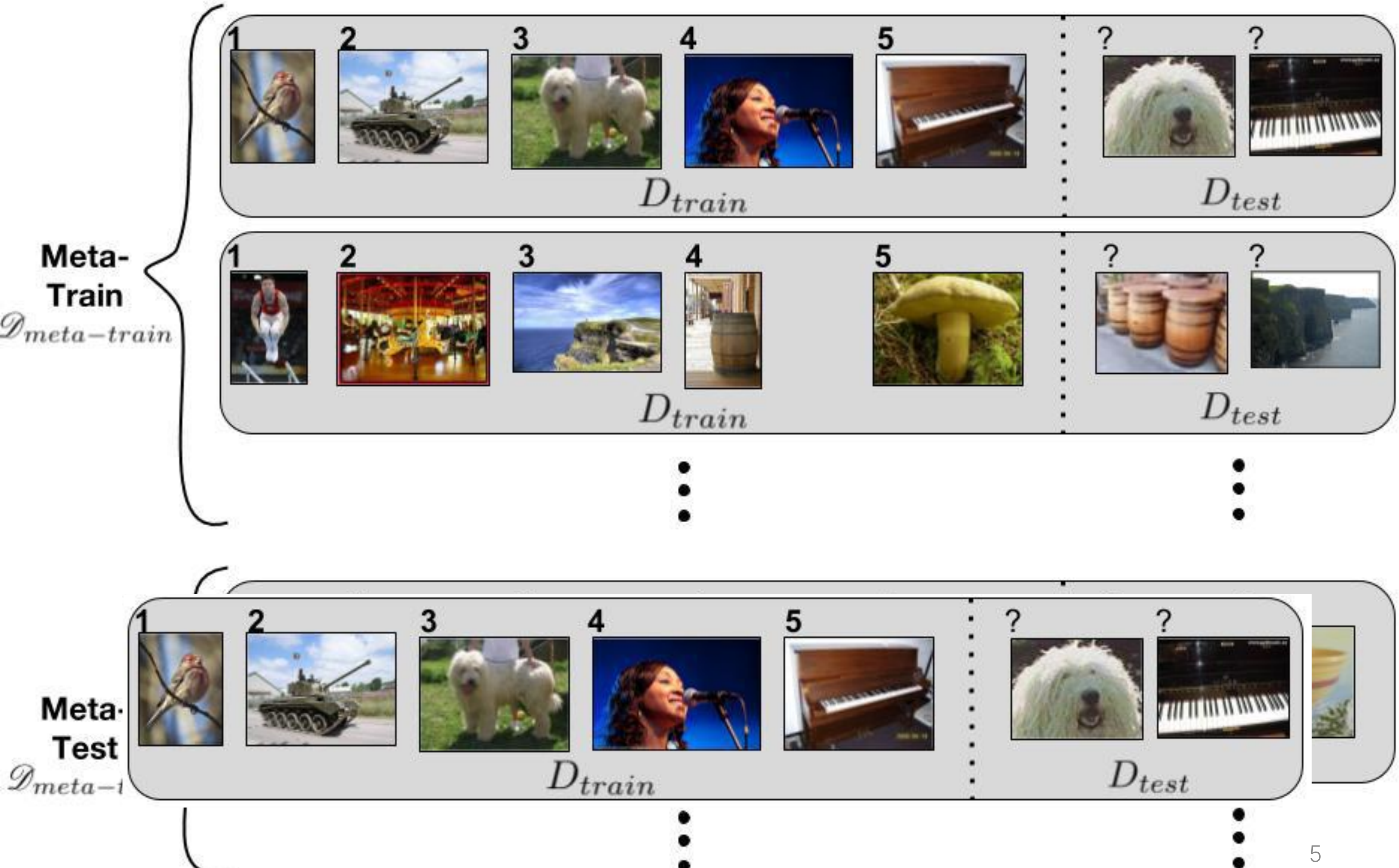
- High-level reasoning requires very little computation, but low-level sensorimotor skills require **enormous** computational resources.



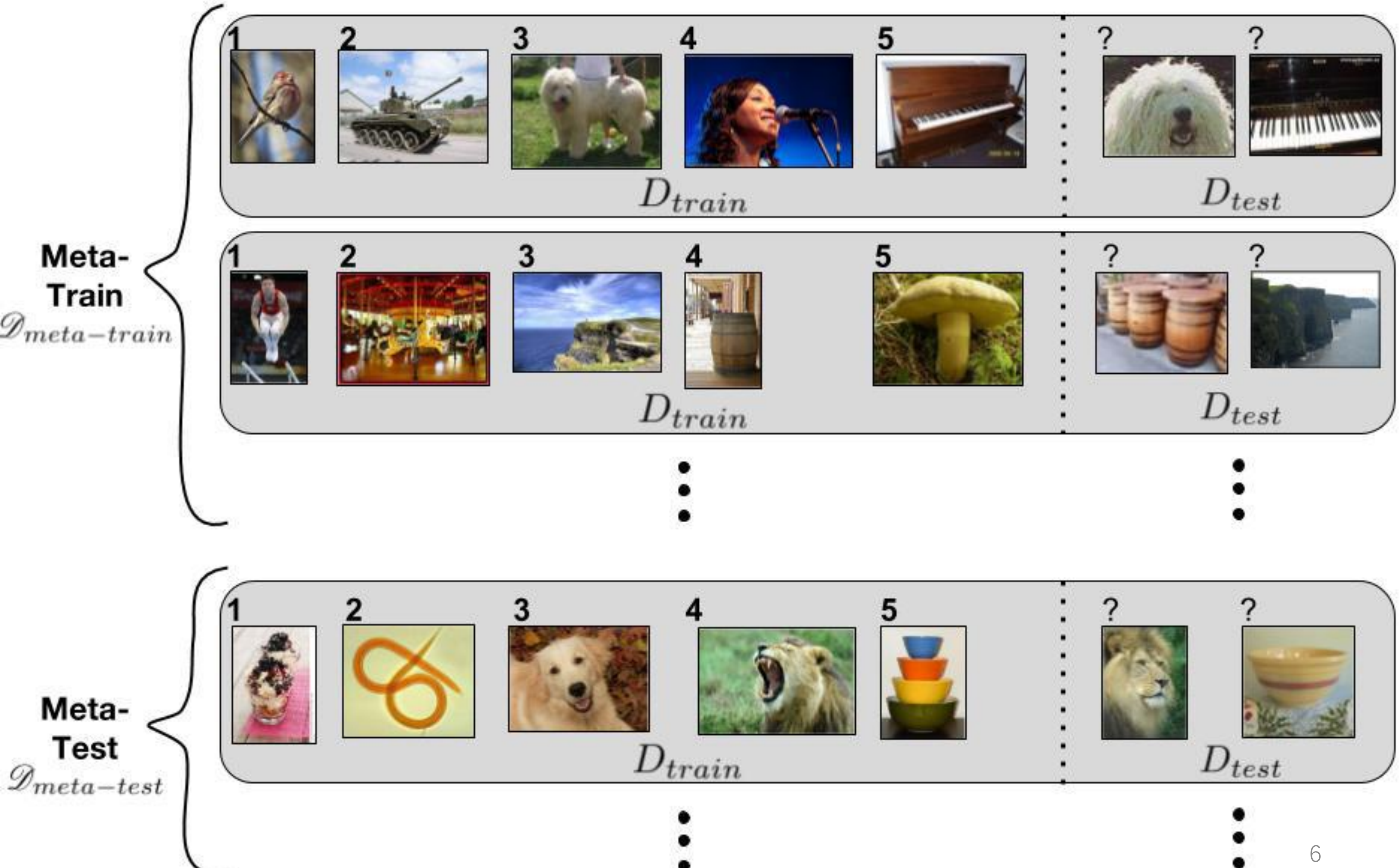
Task-changing Online-learning

Learning from small data

Learning from other tasks



Few-shot learning (1-shot 5-way)



Games vs. Real world

FIFA 18

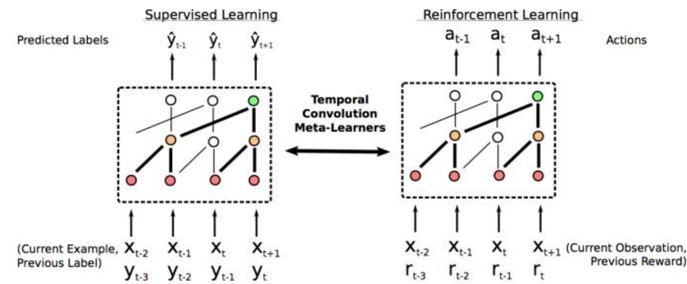
FIFA World Cup 2018

GTA 5

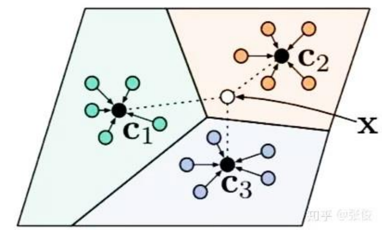
Tesla

Models in Few-shot learning

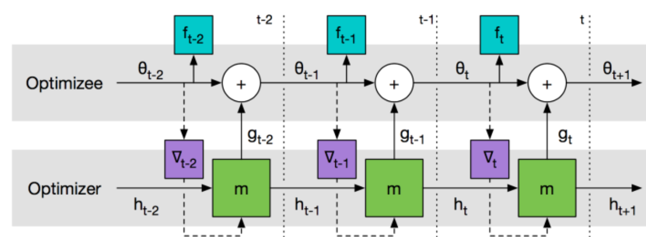
- Model based
  - Meta-learning with memory-augmented neural networks
  - Meta-Learning with Temporal Convolutions
  - Learning to reinforcement learn
  - Rl^2: Fast reinforcement learning via slow RL



- Metric based
  - Siamese neural networks for one-shot image recognition
  - Matching networks for one shot learning
  - Prototypical networks for few-shot learning
  - Learning to compare: Relation network for few-shot learning



- Optimization based
  - Learning to learn by gradient descent by gradient descent

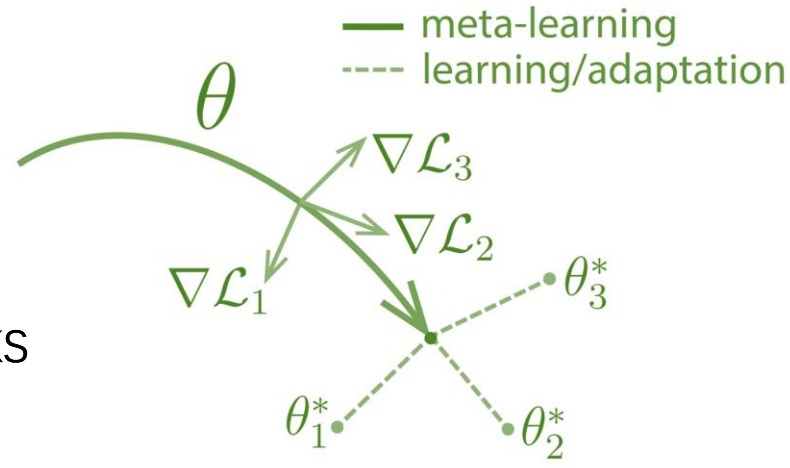


  - Optimization as a model for few-shot learning
  - Learning to Learn: Meta-Critic Networks for Sample Efficient Learning
  - Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks
  - Task-agnostic meta-learning for few-shot learning

Model-Agnostic Meta-Learning

- How to use pretrained model:
  - Fine-tune (by gradient descent)

$$\theta \leftarrow \theta - \alpha \nabla_\theta L_{train}(\theta)$$



- What is our goal:
  - Easy to fine-tune for any tasks
  - A meta loss function

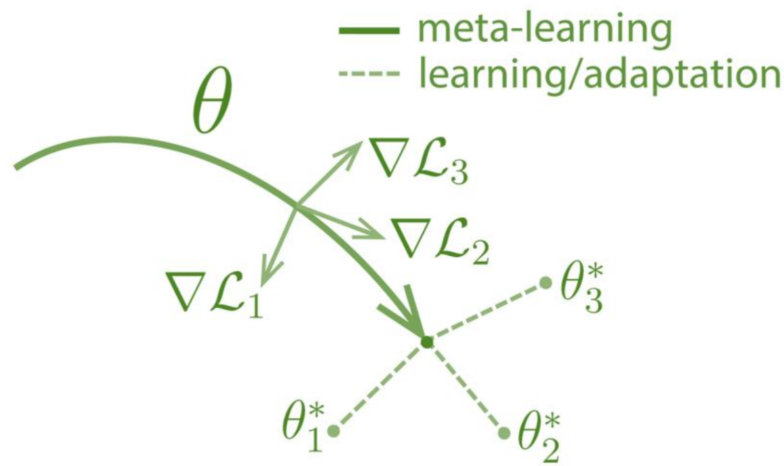
$$\min_\theta \sum_{\text{task } i} L^i_{test}(\theta - \alpha \nabla_\theta L^i_{train}(\theta))$$

Model-Agnostic Meta-Learning

Algorithm 1 Model-Agnostic Meta-Learning

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

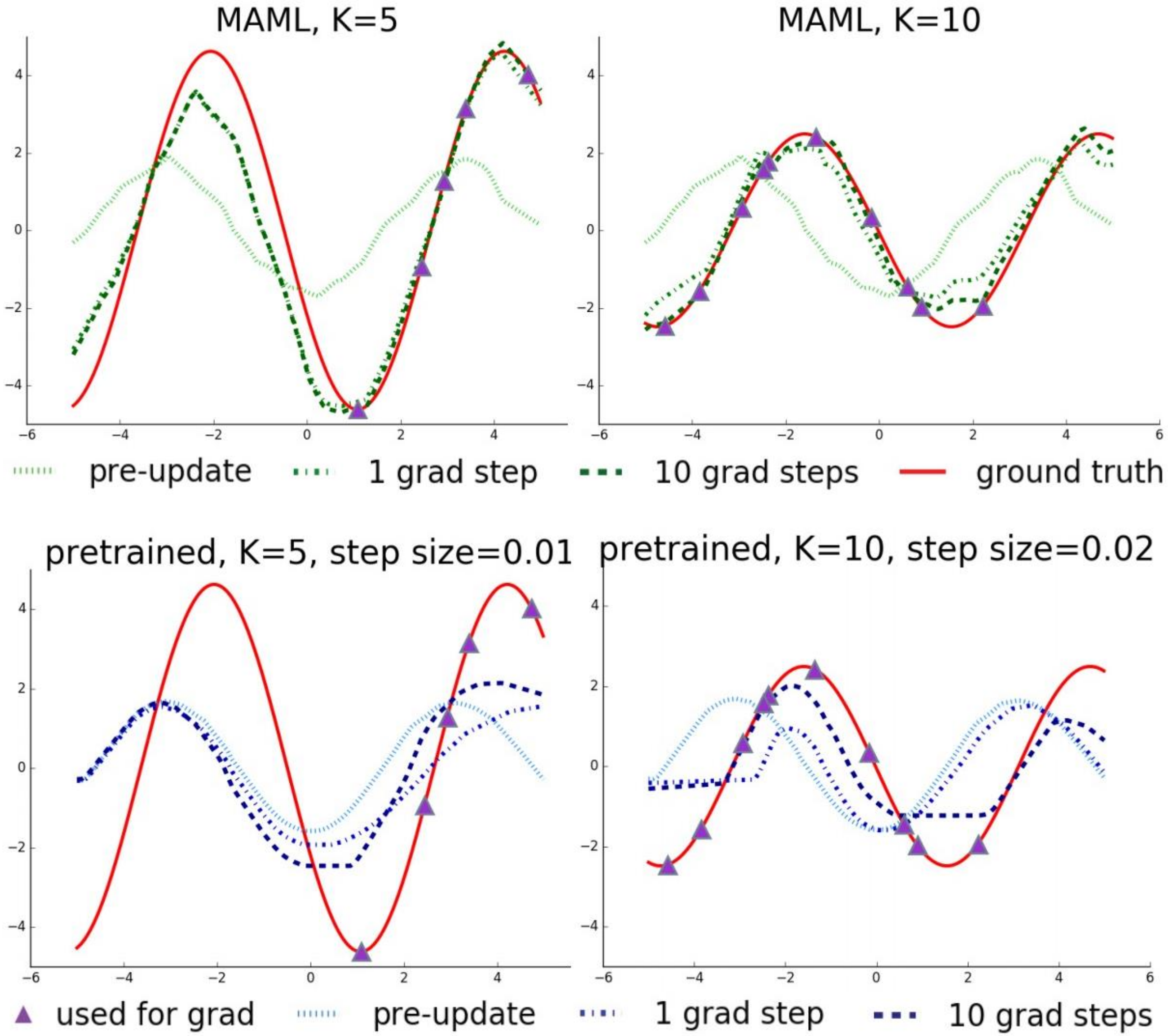
```
randomly initialize θ
while not done do
    Sample batch of tasks T_i ~ p(T)
    for all T_i do
        Evaluate ∇_θ L_{T_i}(f_θ) with respect to K examples
        Compute adapted parameters with gradient descent: θ'_i = θ − α∇_θ L_{T_i}(f_θ)
    end for
    Update θ ← θ − β∇_θ Σ_{T_i~p(T)} L_{T_i}(f_{θ'_i})
end while
```



$$\min_{\theta} \sum_{\text{task } i} L_{test}^{i}(\theta - \alpha\nabla_{\theta} L_{train}^{i}(\theta))$$

MAML applications (regression)

The regressor is a neural network model
with 2 hidden layers of size 40 with ReLU nonlinearities.



MAML applications (regression)

The regressor is a neural network model
with 2 hidden layers of size 40 with ReLU nonlinearities.

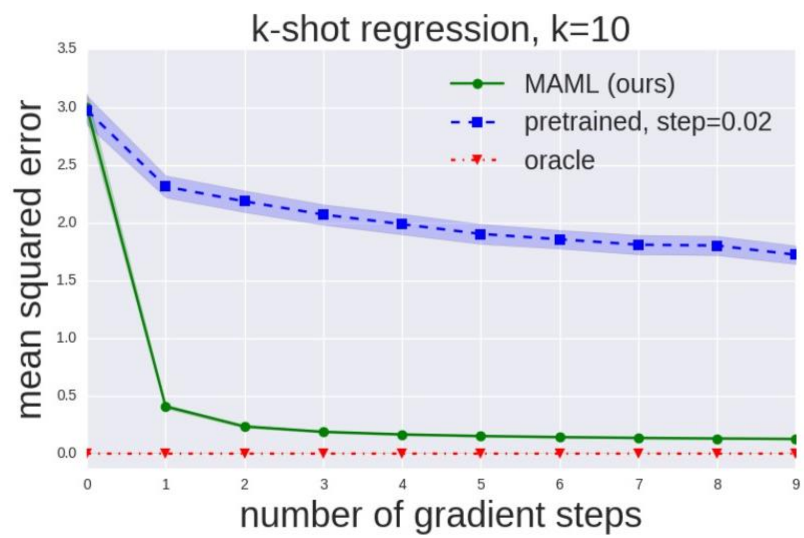


Figure 3. Quantitative sinusoid regression results showing the learning curve at meta test-time. Note that MAML continues to improve with additional gradient steps without overfitting to the extremely small dataset during meta-testing, achieving a loss that is substantially lower than the baseline fine-tuning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

MAML for Image Classification

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 $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    Sample datapoints $\mathcal{D}_i' = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
  end for
  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
end while

MAML applications (miniImageNet)

https://github.com/y2l/mini-imagenet-tools

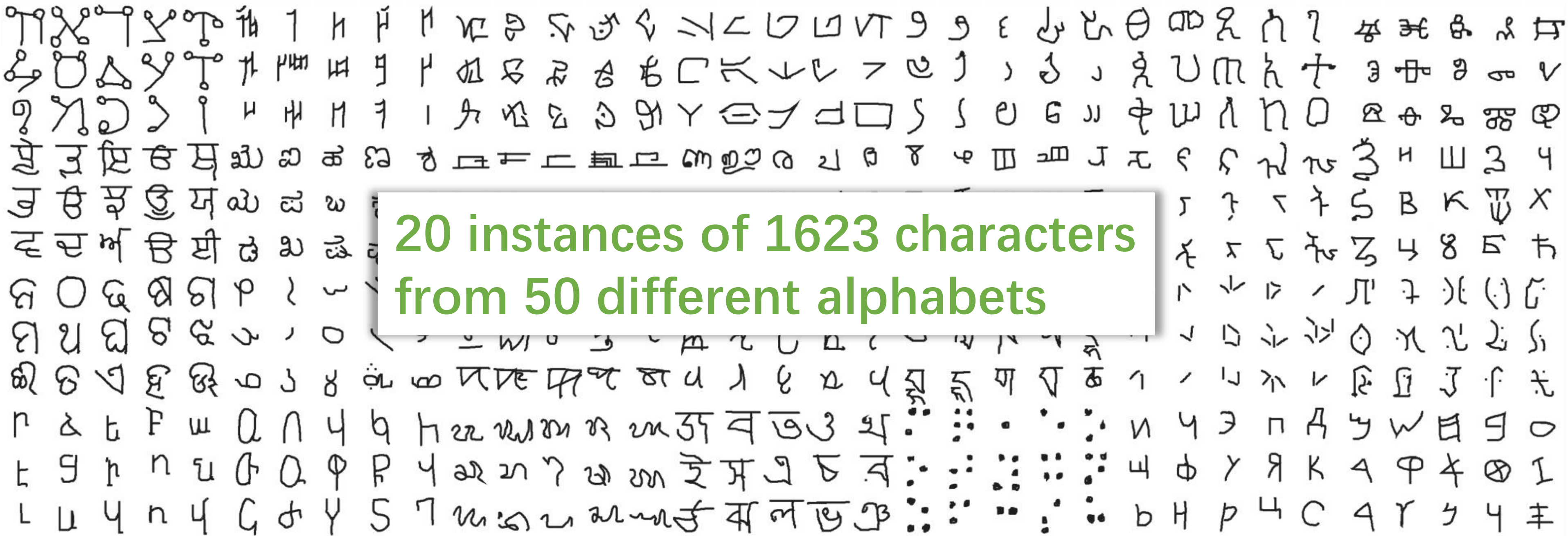
64 training classes, 12 validation classes, and 24 test classes.
Follow the experimental protocol proposed by Vinyals et al. (2016), which involves fast learning of N-way classification with K (1 or 5) shots.

Select N unseen classes,
provide the model with K different instances of each of the N classes,
evaluate the model's ability to classify new instances within the N classes.

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

MAML applications (Omniglot)

https://github.com/brendenlake/omniglot



Omniglot (Lake et al., 2011)	5-way Accuracy 1-shot	5-way Accuracy 5-shot	20-way Accuracy 1-shot	20-way Accuracy 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%**

MAML for Reinforcement Learning

Algorithm 3 MAML for Reinforcement Learning

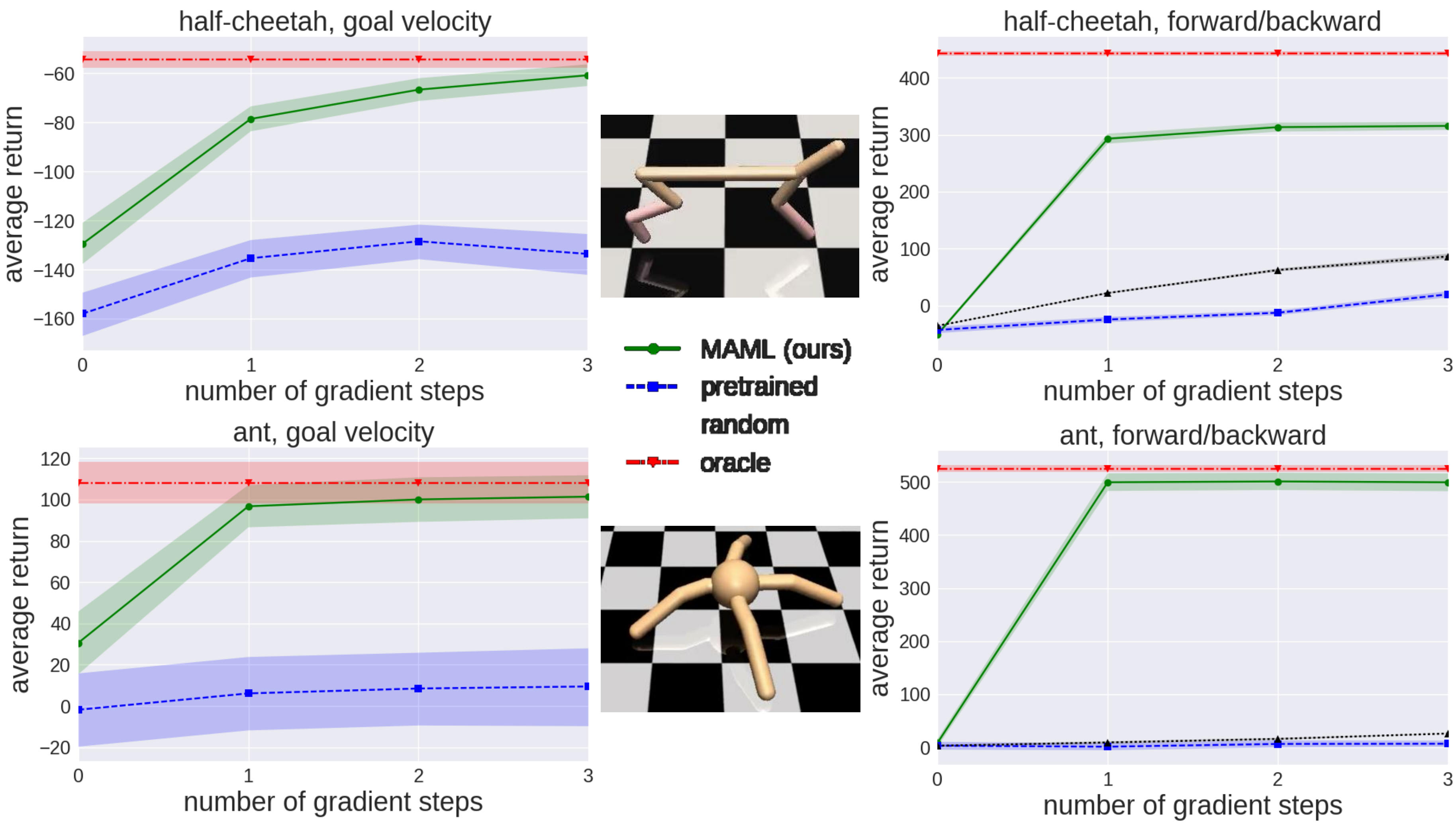
Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

randomly initialize θ
while not done **do**
    Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
    for all $\mathcal{T}_i$ **do**
        Sample K trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using f_θ in $\mathcal{T}_i$
        Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
        Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
        Sample trajectories $\mathcal{D}_i' = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using $f_{\theta_i'}$ in $\mathcal{T}_i$
    end for
    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 4
end while

MAML applications (RL locomotion)

https://github.com/rll/rllab

https://sites.google.com/view/maml



Meta-Learning and Universality

https://arxiv.org/pdf/1710.11622.pdf

- For a sufficiently deep learner model, MAML has the same theoretical representational power as recurrent meta-learners.

- Universal function approximation (UFA) theorem
  - A neural network with one hidden layer of finite width can approximate any continuous function on compact subsets of R^n up to arbitrary precision.
- Universal learning procedure approximator
  - UFA with input (D, x^*) and output y^*.
  - D is training dataset, (x^*, y^*) is test input and desired output.

Meta-Learning and Universality

- First (Model based)
  - Meta-learning with memory-augmented neural networks
  - Rl^2: Fast reinforcement learning via slow RL
  - Learning to reinforcement learn
  - A simple neural attentive meta-learner

$$\hat{\mathbf{y}}^\star = g(\mathcal{D}_\mathcal{T}, \mathbf{x}^\star; \phi) = g((\mathbf{x}, \mathbf{y})_1, \dots, (\mathbf{x}, \mathbf{y})_K, \mathbf{x}^\star; \phi)$$

- Second (Optimization based)
  - Learning to optimize neural nets.
  - Optimization as a model for few-shot learning
  - Hypernetworks.
  - Learning to learn by gradient descent by gradient descent

$$\hat{\mathbf{y}}^\star = f(\mathbf{x}^\star; \theta'_\mathcal{T}) = f(\mathbf{x}^\star; g(\mathcal{D}_\mathcal{T}; \phi)) = f(\mathbf{x}^\star; g((\mathbf{x}, \mathbf{y})_{1:K}; \phi))$$

Meta-Learning and Universality

- MAML

$$\hat{\mathbf{y}}^\star = f_{\text{MAML}}(\mathcal{D}_\mathcal{T}, \mathbf{x}^\star; \theta)$$
$$= f(\mathbf{x}^\star; \theta'_\mathcal{T}) = f(\mathbf{x}^\star; \theta - \alpha\nabla_\theta \mathcal{L}(\mathcal{D}_\mathcal{T}, \theta)) = f\left(\mathbf{x}^\star; \theta - \alpha\nabla_\theta \frac{1}{K}\sum_{k=1}^{K} \ell(\mathbf{y}_k, f(\mathbf{x}_k; \theta))\right)$$

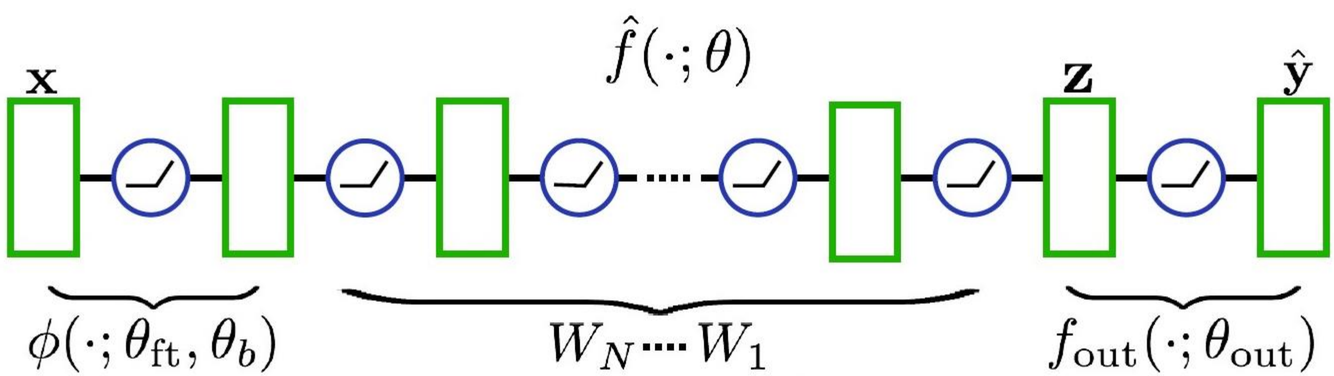


Figure 1: A deep fully-connected neural network with N+2 layers and ReLU nonlinearities. With this generic fully connected network, we prove that, with a single step of gradient descent, the model can approximate any function of the dataset and test input.

Meta-Learning and Universality

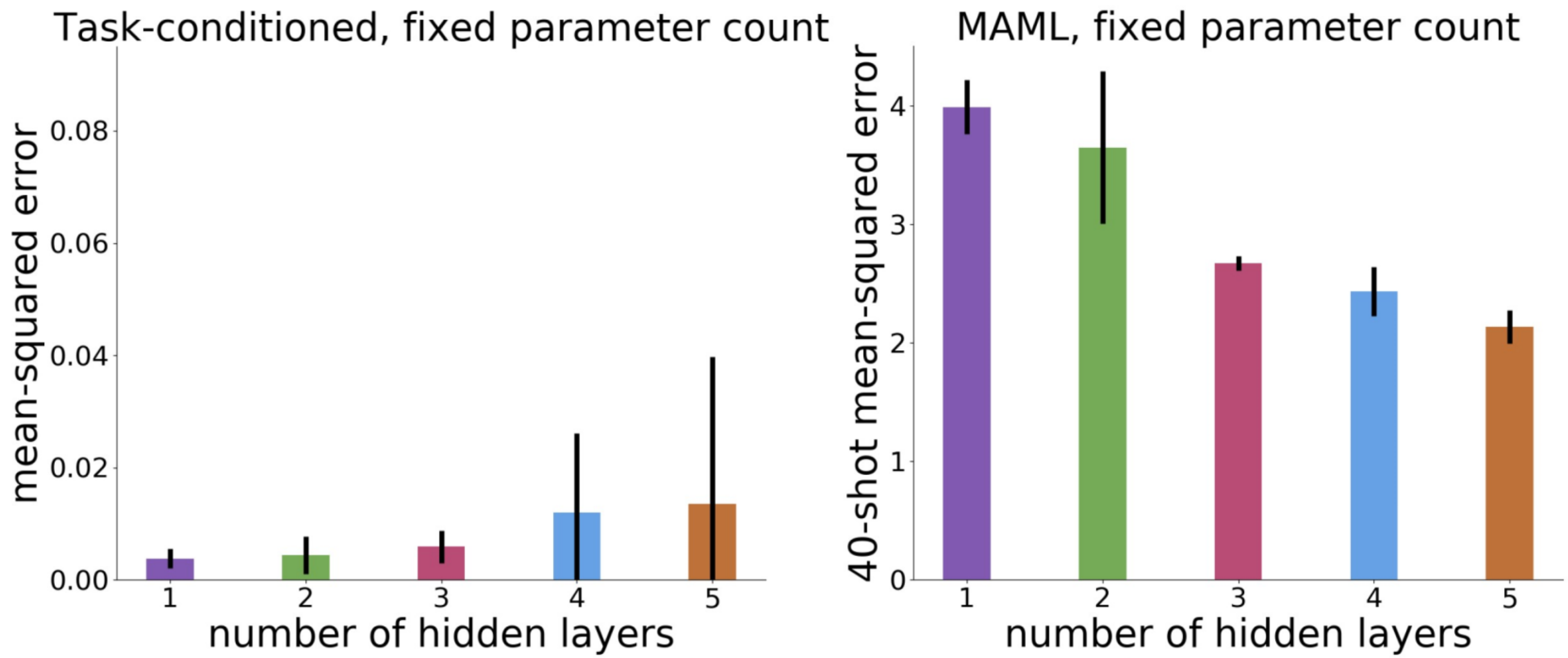


Figure 5: Comparison of depth while keeping the number of parameters constant. Task-conditioned models do not need more than one hidden layer, whereas meta-learning with MAML clearly benefits from additional depth. Error bars show standard deviation over three training runs.

Meta-Learning and Universality

- MAML can be further improved from additional gradient steps.

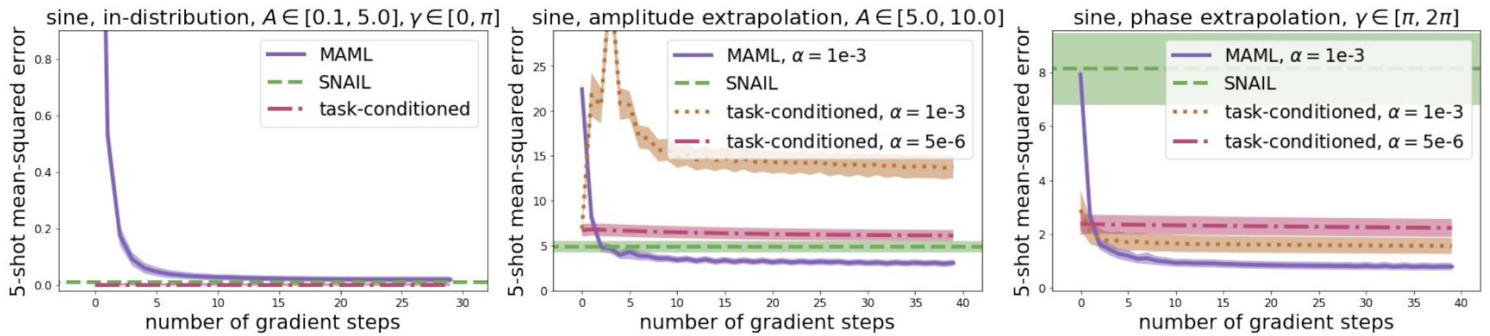


Figure 2: The effect of additional gradient steps at test time when attempting to solve new tasks. The MAML model, trained with 5 inner gradient steps, can further improve with more steps. All methods are provided with the same data – 5 examples – where each gradient step is computed using the same 5 datapoints.

- MAML initialization is substantially better suited for extrapolation beyond the distribution of tasks seen at meta-training time.

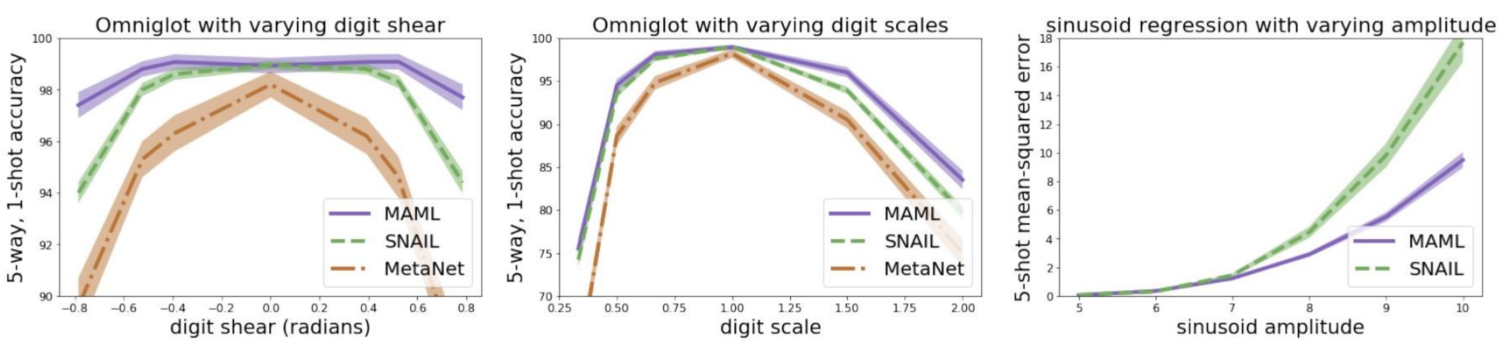


Figure 3: Learning performance on out-of-distribution tasks as a function of the task variability. Recurrent meta-learners such as SNAIL and MetaNet acquire learning strategies that are less generalizable than those learned with gradient-based meta-learning.

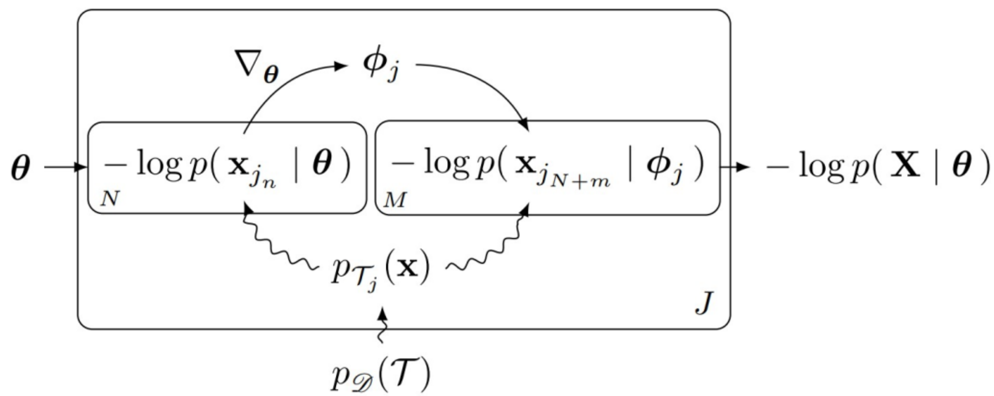
Recasting gradient-based meta-learning as hierarchical Bayes

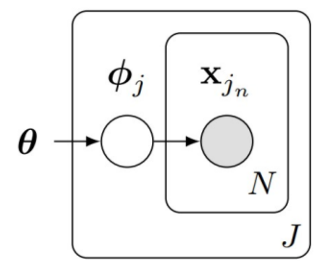
https://arxiv.org/pdf/1801.08930.pdf

- MAML objective in a Maximum Likelihood setting:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{J}\sum_{j}\left[\frac{1}{M}\sum_{m} -\log p\Big(\mathbf{x}_{j_{N+m}} \mid \underbrace{\boldsymbol{\theta} - \alpha\nabla_{\boldsymbol{\theta}}\frac{1}{N}\sum_{n} -\log p\left(\mathbf{x}_{j_n} \mid \boldsymbol{\theta}\right)\Big)}_{\boldsymbol{\phi}_j}\right]$$

- MAML as Hierarchical Bayesian Inference:





$$p\left(\mathbf{X} \mid \boldsymbol{\theta}\right) = \prod_{j}\left(\int p\left(\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_N} \mid \boldsymbol{\phi}_j\right) p\left(\boldsymbol{\phi}_j \mid \boldsymbol{\theta}\right) \mathrm{d}\boldsymbol{\phi}_j\right)$$

$$-\log p\left(\mathbf{X} \mid \boldsymbol{\theta}\right) \approx \sum_{j}\left[-\log p\left(\mathbf{x}_{j_{N+1}}, \ldots \mathbf{x}_{j_{N+M}} \mid \hat{\boldsymbol{\phi}}_j\right)\right]$$

Recasting gradient-based meta-learning as hierarchical Bayes

Algorithm `MAML-HB` ($\mathscr{D}$)

Initialize $\boldsymbol{\theta}$ randomly

while *not converged* **do**

  Draw J samples $\mathcal{T}_1, \ldots, \mathcal{T}_J \sim p_{\mathscr{D}}(\mathcal{T})$

  Estimate $\mathbb{E}_{\mathbf{x} \sim p_{\mathcal{T}_1}(\mathbf{x})}[-\log p(\mathbf{x} \mid \boldsymbol{\theta})], \ldots, \mathbb{E}_{\mathbf{x} \sim p_{\mathcal{T}_J}(\mathbf{x})}[-\log p(\mathbf{x} \mid \boldsymbol{\theta})]$ using `ML-`$\cdots$

  Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \, \nabla_{\boldsymbol{\theta}} \sum_j \mathbb{E}_{\mathbf{x} \sim p_{\mathcal{T}_j}(\mathbf{x})}[-\log p(\mathbf{x} \mid \boldsymbol{\theta})]$

end

Algorithm 2: Model-agnostic meta-learning as hierarchical Bayesian inference. The choices of the subroutine `ML-`$\cdots$ that we consider are defined in Subroutine 3 and Subroutine 4.

Subroutine `ML-POINT` ($\boldsymbol{\theta}, \mathcal{T}$)

Draw N samples $\mathbf{x}_1, \ldots, \mathbf{x}_N \sim p_{\mathcal{T}}(\mathbf{x})$

Initialize $\boldsymbol{\phi} \leftarrow \boldsymbol{\theta}$

for k *in* $1, \ldots, K$ **do**

  Update $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha \, \nabla_{\boldsymbol{\phi}} \log p(\mathbf{x}_1, \ldots, \mathbf{x}_N \mid \boldsymbol{\phi})$

end

Draw M samples $\mathbf{x}_{N+1}, \ldots, \mathbf{x}_{N+M} \sim p_{\mathcal{T}}(\mathbf{x})$

return $-\log p(\mathbf{x}_{N+1}, \ldots, \mathbf{x}_{N+M} \mid \boldsymbol{\phi})$

Subroutine 3: Subroutine for computing a point estimate $\hat{\boldsymbol{\phi}}$ using truncated gradient descent to approximate the marginal negative log likelihood (NLL).

Recasting gradient-based meta-learning as hierarchical Bayes

$$\begin{aligned}\boldsymbol{\phi}_{(k)} &= \boldsymbol{\phi}_{(k-1)} - \alpha \nabla_{\boldsymbol{\phi}} \left[\|\mathbf{y} - \mathbf{X}\boldsymbol{\phi}\|_2^2 \right]_{\boldsymbol{\phi}=\boldsymbol{\phi}_{(k-1)}} \\ &= \boldsymbol{\phi}_{(k-1)} - \alpha \mathbf{X}^{\mathrm{T}} \left(\mathbf{X}\boldsymbol{\phi}_{(k-1)} - \mathbf{y} \right) \end{aligned} \tag{4}$$

for iteration index k and learning rate $\alpha \in \mathbb{R}^+$. Santos (1996) shows that, starting from $\boldsymbol{\phi}_{(0)} = \boldsymbol{\theta}$, $\boldsymbol{\phi}_{(k)}$ in (4) solves the regularized linear least squares problem

$$\min \left(\|\mathbf{y} - \mathbf{X}\boldsymbol{\phi}\|_2^2 + \|\boldsymbol{\theta} - \boldsymbol{\phi}\|_{\mathbf{Q}}^2 \right) \tag{5}$$

$$p\,(\,\boldsymbol{\phi} \mid \mathbf{X}, \mathbf{y}, \boldsymbol{\theta}\,) \propto \mathcal{N}(\mathbf{y}\,;\,\mathbf{X}\boldsymbol{\phi}, \mathbb{I})\,\mathcal{N}(\boldsymbol{\phi}\,;\,\boldsymbol{\theta}, \mathbf{Q})$$

$$\ell(\boldsymbol{\phi}) = -\log p(\,\mathbf{x}_1 \ldots, \mathbf{x}_N \mid \boldsymbol{\phi}\,)$$

$$\ell(\boldsymbol{\phi}) \approx \tilde{\ell}(\boldsymbol{\phi}) := \tfrac{1}{2}\|\boldsymbol{\phi} - \boldsymbol{\phi}^*\|_{\mathbf{H}^{-1}}^2 + \ell(\boldsymbol{\phi}^*) \qquad \mathbf{H} = \nabla_{\boldsymbol{\phi}}^2\, \ell(\boldsymbol{\phi}^*)$$

$$\boldsymbol{\phi}_{(k)} = \boldsymbol{\phi}_{(k-1)} - \mathcal{B}\,\nabla_{\boldsymbol{\phi}}\,\tilde{\ell}(\boldsymbol{\phi}_{(k-1)})$$

$$\min \left(\|\boldsymbol{\phi} - \boldsymbol{\phi}^*\|_{\mathbf{H}^{-1}}^2 + \|\boldsymbol{\phi}_{(0)} - \boldsymbol{\phi}\|_{\mathbf{Q}}^2 \right)$$

Recasting gradient-based meta-learning as hierarchical Bayes

- Laplace approximation

$$\int p\left(\mathbf{X}_j \mid \boldsymbol{\phi}_j\right) p\left(\boldsymbol{\phi}_j \mid \boldsymbol{\theta}\right) \mathrm{d}\boldsymbol{\phi}_j \approx p\left(\mathbf{X}_j \mid \boldsymbol{\phi}_j^*\right) p\left(\boldsymbol{\phi}_j^* \mid \boldsymbol{\theta}\right) \det(\mathbf{H}_j/2\pi)^{-\frac{1}{2}}$$

$$\mathbf{H}_j = \nabla^2_{\boldsymbol{\phi}_j}\left[-\log p\left(\mathbf{X}_j \mid \boldsymbol{\phi}_j\right)\right] + \nabla^2_{\boldsymbol{\phi}_j}\left[-\log p\left(\boldsymbol{\phi}_j \mid \boldsymbol{\theta}\right)\right]$$

$$-\log p\left(\mathbf{X} \mid \boldsymbol{\theta}\right) \approx \sum_j \left[-\log p\left(\mathbf{X}_j \mid \hat{\boldsymbol{\phi}}_j\right) - \log p\left(\hat{\boldsymbol{\phi}}_j \mid \boldsymbol{\theta}\right) + \tfrac{1}{2}\log\det(\mathbf{H}_j)\right]$$

Subroutine `ML-LAPLACE` $(\boldsymbol{\theta}, \mathcal{T})$
  Draw N samples $\mathbf{x}_1, \ldots, \mathbf{x}_N \sim p_{\mathcal{T}}(\mathbf{x})$
  Initialize $\boldsymbol{\phi} \leftarrow \boldsymbol{\theta}$
  for k *in* $1, \ldots, K$ **do**
    Update $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha \nabla_{\boldsymbol{\phi}} \log p(\mathbf{x}_1, \ldots, \mathbf{x}_N \mid \boldsymbol{\phi})$
  end
  Draw M samples $\mathbf{x}_{N+1}, \ldots, \mathbf{x}_{N+M} \sim p_{\mathcal{T}}(\mathbf{x})$
  Estimate quadratic curvature $\hat{\mathbf{H}}$
  return $-\log p(\mathbf{x}_{N+1}, \ldots, \mathbf{x}_{N+M} \mid \boldsymbol{\phi}) + \eta \log\det(\hat{\mathbf{H}})$

Subroutine 4: Subroutine for computing a Laplace approximation of the marginal likelihood.

Recasting gradient-based meta-learning as hierarchical Bayes

Algorithm `MAML-HB` ($\mathscr{D}$)

Initialize $\boldsymbol{\theta}$ randomly
while *not converged* **do**
  Draw J samples $\mathcal{T}_1, \ldots, \mathcal{T}_J \sim p_{\mathscr{D}}(\mathcal{T})$
  Estimate $\mathbb{E}_{\mathbf{x} \sim p_{\mathcal{T}_1}(\mathbf{x})}[-\log p(\, \mathbf{x} \mid \boldsymbol{\theta}\,)], \ldots, \mathbb{E}_{\mathbf{x} \sim p_{\mathcal{T}_J}(\mathbf{x})}[-\log p(\, \mathbf{x} \mid \boldsymbol{\theta}\,)]$ using `ML-···`
  Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \, \nabla_{\boldsymbol{\theta}} \sum_j \mathbb{E}_{\mathbf{x} \sim p_{\mathcal{T}_j}(\mathbf{x})}[-\log p(\, \mathbf{x} \mid \boldsymbol{\theta}\,)]$
end

Algorithm 2: Model-agnostic meta-learning as hierarchical Bayesian inference. The choices of the subroutine `ML-···` that we consider are defined in Subroutine 3 and Subroutine 4.

Subroutine `ML-LAPLACE` ($\boldsymbol{\theta}, \mathcal{T}$)

Draw N samples $\mathbf{x}_1, \ldots, \mathbf{x}_N \sim p_{\mathcal{T}}(\mathbf{x})$
Initialize $\boldsymbol{\phi} \leftarrow \boldsymbol{\theta}$
for k *in* $1, \ldots, K$ **do**
  Update $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha \, \nabla_{\boldsymbol{\phi}} \log p(\, \mathbf{x}_1, \ldots, \mathbf{x}_N \mid \boldsymbol{\phi}\,)$
end
Draw M samples $\mathbf{x}_{N+1}, \ldots, \mathbf{x}_{N+M} \sim p_{\mathcal{T}}(\mathbf{x})$
Estimate quadratic curvature $\hat{\mathbf{H}}$
return $-\log p(\, \mathbf{x}_{N+1}, \ldots, \mathbf{x}_{N+M} \mid \boldsymbol{\phi}\,) + \eta \log \det(\hat{\mathbf{H}})$

Subroutine 4: Subroutine for computing a Laplace approximation of the marginal likelihood.

Probabilistic MAML

https://arxiv.org/pdf/1806.02817.pdf

$$p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}}) = \int p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \phi_i) p(\phi_i|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \theta) d\phi_i \approx p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \phi_i^\star)$$

$$\log p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}) \geq E_{\theta \sim q_\psi} \left[\log p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \phi_i^\star) + \log p(\theta)\right] + \mathcal{H}(q_\psi(\theta|\mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}}))$$

$$q_\psi(\theta|\mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}}) = \mathcal{N}(\boldsymbol{\mu}_\theta + \boldsymbol{\gamma}_q \nabla \log p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \boldsymbol{\mu}_\theta); \mathbf{v}_q)$$

Algorithm 1 Meta-training, differences from MAML in red

Require: $p(\mathcal{T})$: distribution over tasks
- initialize $\Theta := \{\boldsymbol{\mu}_\theta, \boldsymbol{\sigma}_\theta^2, \mathbf{v}_q, \boldsymbol{\gamma}_p, \boldsymbol{\gamma}_q\}$
- **while** not done **do**
  - Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  - **for all** $\mathcal{T}_i$ **do**
    - $\mathcal{D}^{\text{tr}}, \mathcal{D}^{\text{test}} = \mathcal{T}_i$
    - Evaluate $\nabla_{\boldsymbol{\mu}_\theta} \mathcal{L}(\boldsymbol{\mu}_\theta, \mathcal{D}^{\text{test}})$
    - Sample $\theta \sim q = \mathcal{N}(\boldsymbol{\mu}_\theta - \boldsymbol{\gamma}_q \nabla_{\boldsymbol{\mu}_\theta} \mathcal{L}(\boldsymbol{\mu}_\theta, \mathcal{D}^{\text{test}}), \mathbf{v}_q)$
    - Evaluate $\nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$
    - Compute adapted parameters with gradient descent: $\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$
  - Let $p(\theta|\mathcal{D}^{\text{tr}}) = \mathcal{N}(\boldsymbol{\mu}_\theta - \boldsymbol{\gamma}_p \nabla_{\boldsymbol{\mu}_\theta} \mathcal{L}(\boldsymbol{\mu}_\theta, \mathcal{D}^{\text{tr}}), \boldsymbol{\sigma}_\theta^2))$
  - Compute $\nabla_\Theta \big(\sum_{\mathcal{T}_i} \mathcal{L}(\phi_i, \mathcal{D}^{\text{test}}) + D_{\text{KL}}(q(\theta|\mathcal{D}^{\text{test}}) \mid\mid p(\theta|\mathcal{D}^{\text{tr}}))\big)$
  - Update Θ using Adam

Algorithm 2 Meta-testing

Require: training data $\mathcal{D}_\mathcal{T}^{\text{tr}}$ for new task $\mathcal{T}$
Require: learned Θ
- Sample θ from the prior $p(\theta|\mathcal{D}^{\text{tr}})$
- Evaluate $\nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$
- Compute adapted parameters with gradient descent: $\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$

Future topics

- Training
  - How many meta-samples (tasks) do we need for meta-learning?
  - What if some meta-samples are wrong?
- Testing
  - How many samples do we need for a new task?
  - What if we know the new task beforehand?
  - Can we get better robustness and less uncertainty by meta-learning?
- Model
  - What should a good meta-loss function be like?
  - How to measure, store and use the meta-knowledge?
  - How to incorporate tasks on different domains?