(Online) Meta-learning

By Chelsea Finn et al.

Presented by Dalin Guo, 5/13/2021

Meta-Learning: Learning to Learn

- Fast and flexible learning: learning and adapting quickly from only a few examples, and continuing to adapt as more data becomes available
  - integrate prior experience + avoid overfitting on new data
- Goal:
  - learn on a large number of different tasks (tasks must have share structure)
  - quickly learn a new task from a small amount of new data
- Key idea:
  - train the model initial parameters
    - s.t. maximal performance on a new task (after 1+ gradient steps with small data)

Example Meta-Learning Problem

5-way, 1-shot image classification (MiniImagenet)

Given 1 example of 5 classes:

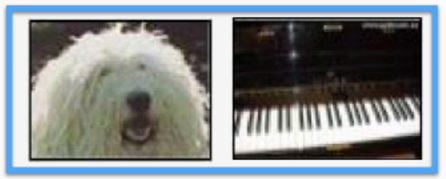
Classify new examples

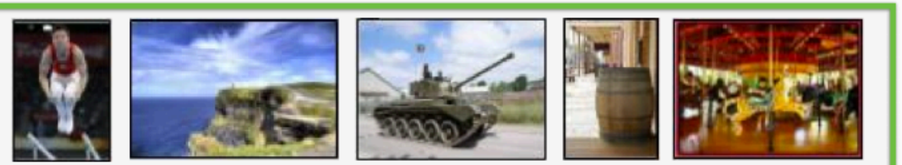
held-out classes

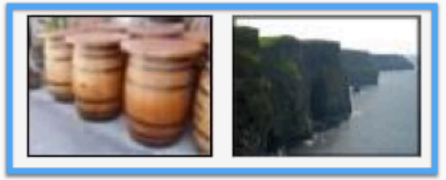
meta-training

$\mathcal{T}_1$

$\mathcal{T}_2$







training classes

Can replace image classification with: **regression, language generation, skill learning,** **any ML problem**

From Finn CS 330 course slide

Outline

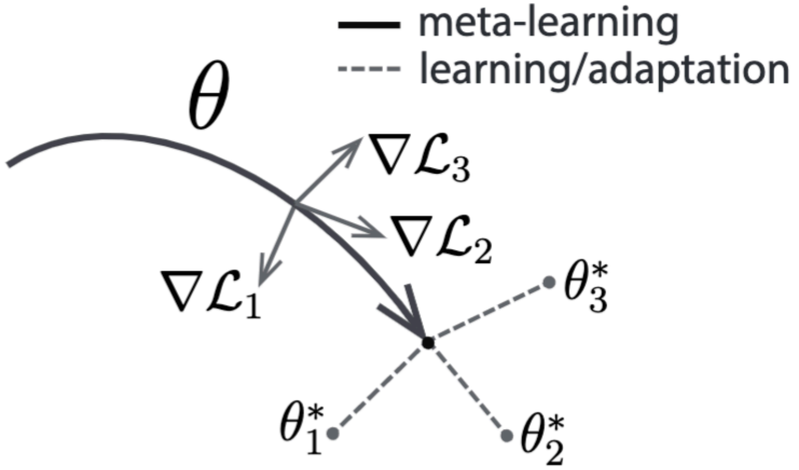
- **MAML**: Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks (2017) by Chelsea Finn, Pieter Abbeel, Sergey Levine
  - idea + algorithm
  - 2/3 experiment
- **FTML:** Online Meta-learning (2019) by Chelsea Finn, Aravind Rajeswaran, Sham Kakade, Sergey Levine
  - idea + algorithm
  - theoretical results (without proof)
  - 1/3 experiment

Model-Agnostic Meta-Learning (MAML)

- Model-agnostic:
  - any learning problem: classification, regression, policy gradient RL
  - any model (trained with a gradient descent): e.g. deep neural network
- No additional learned parameters (for meta-learning)/constraints on the model architecture (e.g. RNN/Siamese) & combine with a variety of loss functions

Key Idea

- Optimize for models easy and fast to fine-tune: build representation suitable for many tasks (more transferable, general-purpose)
  - ~ fine-tuning
- maximizing the sensitivity of the loss functions of new tasks w.r.t. parameters (small changes -> large improvements)
  - few-shot learning



θ: parameter vector being meta-learned

θ_i^*: optimal parameter vector for task i

Problem setup

- Notations
  - model: $f_\theta(\mathbf{x}) = \mathbf{a}$, parameters: θ
  - each task:
    $\mathcal{T} = \{\mathcal{L}(\mathbf{x}_1, \mathbf{a}_1, \ldots, \mathbf{x}_H, \mathbf{a}_H), q(\mathbf{x}_1), q(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{a}_t), H\}$
  - distribution of tasks: $p(\mathcal{T})$
- K-shot learning
  - a new task: $\mathcal{T}_i \sim p(\mathcal{T})$,
  - samples: $\mathbf{x}_j \sim q_i(\mathbf{x}), j = 1, \ldots, K$,
  - feedback: $\mathcal{L}_{\mathcal{T}_i}$
- θ: parameter vector being meta-learned
- adapting to task $\mathcal{T}_i$: $f_{\theta_i'}$
  - update function: $\theta_i' = U_i(\theta)$
  - one gradient update for simplicity
- Meta-objective:
  - $\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(\theta_i')$

MAML Algorithm

Algorithm 1 Model-Agnostic Meta-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: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
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**

- Meta-learning phase
  - sample a task $\mathcal{T}_i$
  - train with K samples $\mathcal{D}_i^{tr}$: e.g. one gradient step $\theta_i' = \theta - \alpha \nabla \mathcal{L}(\theta, \mathcal{D}_i^{tr})$
  - test on new samples $\mathcal{D}_i^{val}$ from $\mathcal{T}_i$
  - improve model: how the test/validation error changes w.r.t. parameters $\theta = \theta - \beta \nabla_\theta \mathcal{L}(\theta_i, D_i^{val})$,
    - i.e. test/validation error on $\mathcal{T}_i$ serves as training error of meta-learning
- Evaluation: model performance after training with K samples on new tasks (held-out)

Second-order derivative

- meta-update: $\theta = \theta - \beta \nabla_\theta \mathscr{L}(\theta_i, \mathscr{D}_i^{val}) = \theta - \beta \nabla_\theta \mathscr{L}(U_i(\theta), \mathscr{D}_i^{val})$
- $U_i(\theta) = \theta - \alpha \nabla \mathscr{L}(\theta, \mathscr{D}_i^{tr})$
- a gradient through a gradient
- full Hessian? No, vector-Hessian product
- higher-order derivatives with more inner gradient steps? No

Second-order derivative

- meta-update: $\theta = \theta - \beta \nabla_\theta \mathscr{L}(\theta_i, \mathscr{D}_i^{val}) = \theta - \beta \nabla_\theta \mathscr{L}(U_i(\theta), \mathscr{D}_i^{val})$
- $U_i(\theta) = \theta - \alpha \nabla \mathscr{L}(\theta, \mathscr{D}_i^{tr})$
- full Hessian?
- $$\frac{d}{d_\theta} \mathscr{L}(\theta_i, \mathscr{D}_i^{val}) = \frac{d}{d_{\theta_i}} \mathscr{L}(\theta_i, \mathscr{D}_i^{val}) \Big|_{\theta_i = U_i(\theta)} \cdot \frac{d}{d_\theta} U(\theta, \mathscr{D}_i^{tr})$$
- $$\frac{d}{d_\theta} U(\theta, \mathscr{D}_i^{tr}) = I - \alpha \nabla^2 \mathscr{L}(\theta, \mathscr{D}_i^{tr})$$

Second-order derivative

- meta-update: $\theta = \theta - \beta \nabla_\theta \mathscr{L}(\theta_i, \mathscr{D}_i^{val}) = \theta - \beta \nabla_\theta \mathscr{L}(U_i(\theta), \mathscr{D}_i^{val})$
- $U_i(\theta) = \theta - \alpha \nabla_\theta \mathscr{L}(\theta, \mathscr{D}_i^{tr}) - \alpha \nabla_{\theta'} \mathscr{L}(\theta', \mathscr{D}_i^{tr}), \theta' = U_i(\theta)$
- higher-order derivatives with more inner gradient steps?
- $\frac{d}{d_\theta} U(\theta, \mathscr{D}_i^{tr}) = I - \alpha \nabla^2 \mathscr{L}(\theta, \mathscr{D}_i^{tr}) - \alpha \nabla^2_{\theta'} \mathscr{L}(\theta', \mathscr{D}_i^{tr}) \cdot \frac{d\theta'}{d\theta}$

Experiments

- **Regression (toy)**, Classification, Reinforcement Learning
- tasks: one sine wave (varying amplitude and phase)

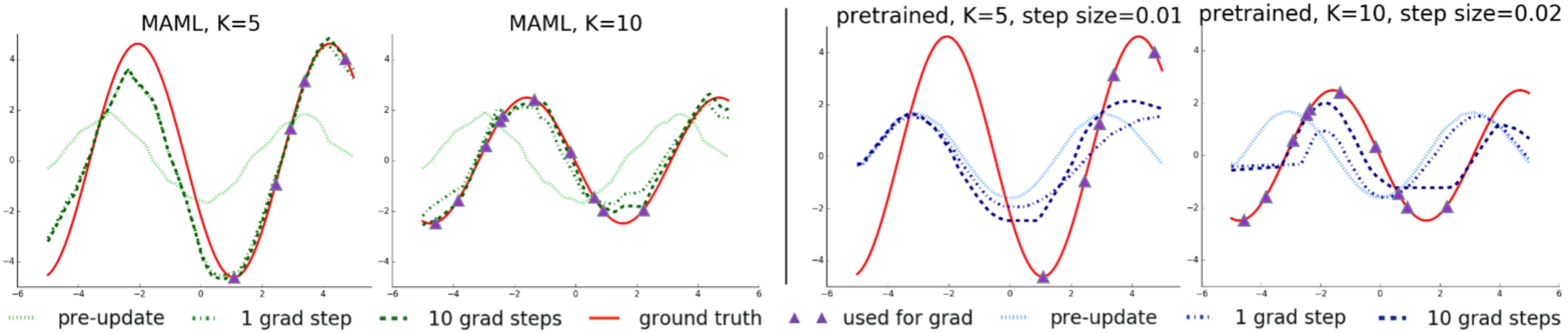
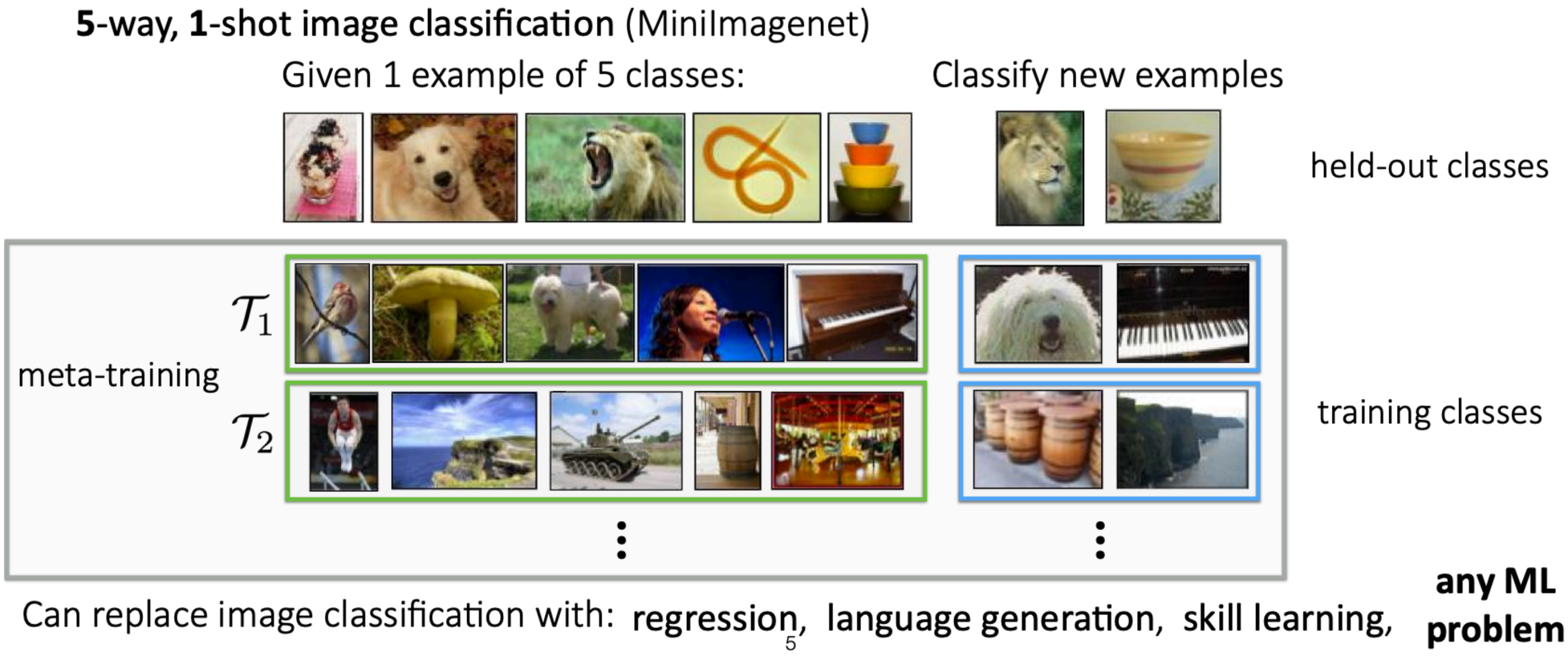


Figure 2. Few-shot adaptation for the simple regression task. Left: Note that MAML is able to estimate parts of the curve where there are no datapoints, indicating that the model has learned about the periodic structure of sine waves. Right: Fine-tuning of a model pretrained on the same distribution 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.

Experiments

- Regression (toy), **Classification**, Reinforcement Learning
- MiniImagenet: 64 training classes, 12 validation classes, 24 test classes



Experiments

- MAML compares well to SOTA or narrowly outperforming
- omit the second-order derivatives almost the same results: ReLU are locally mostly linear, second-order derivatives close to zero (33% speed-up)
  - $$\frac{d}{d_\theta}\mathscr{L}(\theta_i, \mathscr{D}_i^{val}) = \frac{d}{d_{\theta_i}}\mathscr{L}(\theta_i, \mathscr{D}_i^{val})\Big|_{\theta_i = U_i(\theta)} \cdot \frac{d}{d_\theta}U(\theta, \mathscr{D}_i^{tr})$$
  - $$\frac{d}{d_\theta}U(\theta, \mathscr{D}_i^{tr}) = I - \alpha\nabla^2\mathscr{L}(\theta, \mathscr{D}_i^{tr})$$

MiniImagenet (Ravi & Larochelle, 2017)	5-way Accuracy	
	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)	$\mathbf{48.70 \pm 1.84\%}$	$\mathbf{63.11 \pm 0.92\%}$

Challenges & solutions

- bi-level optimization can exhibit instabilities
- backpropagating through many inner gradient steps is compute- & memory-intensive
- architecture effective for inner gradient step

- https://cs330.stanford.edu/slides/cs330_optbased_metalearning_2020.pdf

Online Meta-learning

- Meta-learning: learn a prior over parameters, tasks are available as a batch
  - no sequential or non-stationary
- Online learning: tasks in sequence; one model w/o task-specific adaptation
  - no use of past experience to accelerate adaptation
- Meta-learning + Online learning = continual lifelong learning
- => Follow the Meta Leader (FTML): extends MAML

Online learning: follow the leader (FTL)

- agent faces a sequence of loss function $\{\mathscr{L}_t\}_{t=1}^{\infty}$
- decide model parameters $\{\theta_t\}_{t=1}^{\infty}$
- loss/regret: $\sum_{t=1}^{T} \mathscr{L}_t(\theta_t) - \min_{\theta} \sum_{t=1}^{T} \mathscr{L}_t(\theta)$
- Goal: regret grows with T as slowly as possible (e.g. sub-linearly)
- FTL: $\theta_{t+1} = \arg\min_{\theta} \sum_{k=1}^{t} \mathscr{L}_k(\theta)$, i.e. joint training on all observed data

Problem Setup

- Protocol
  - At round t, the agent choose a model defined by θ_t
  - The world simultaneously choose task defined by population risk R_t
  - The agent update model $\theta_t' = U_t(\theta_t)$, e.g. $U_t(\theta_t) = \theta_t - \alpha \nabla \hat{R}_t(\theta_t)$
  - The agent incurs loss $R_t(\theta_t')$. Advance to round t + 1
  - Goal: minimize regret over rounds $\sum_{t=1}^{T} R_t(U_t(\theta_t)) - \min_{\theta} \sum_{t=1}^{T} R_t(U_t(\theta))$

Follow The Meta Leader (FTML)

- Meta-update: $\theta_{t+1} = \arg\min_{\theta} \sum_{k=1}^{t} R_k(U_k(\theta))$

- FTL: $\theta_{t+1} = \arg\min_{\theta} \sum_{k=1}^{t} \mathscr{L}_k(\theta)$ as a reference

Convexity of MAML

- Assumptions:
  - 1. C^2- smoothness: R is G-Lipschitz, β-smooth, and has ρ-Lipschitz Hessians
  - 2. Strong convexity: R is μ-strongly convex
- **Theorem 1.** $\hat{R}(\theta')$ is convex given .., and $\tilde{\beta}$-smooth and $\tilde{\mu}$-strongly convex
- **Corollary 1.** MAML optimization is convex given …

Inherited regret bound for FTML

- FTML has the same regret guarantees (up to a constant factors) as FTL with strongly convex losses
- $$\sum_{t=1}^{T} R_t(U_t(\theta_t)) - \min_{\theta} \sum_{t=1}^{T} R_t(U_t(\theta)) = \mathcal{O}(\frac{32G^2}{\mu} \log T)$$
- (FTL: Cesa-Bianchi & Lugosi (2006) Theorem 3.1)

Practical FTML

- Meta-update: $\theta_{t+1} = \arg\min_{\theta} \sum_{k=1}^{t} R_k(U_k(\theta))$
- problem: 1) no closed form 2) no access to population risk R_t
- idea: iterative stochastic optimization
- gradient for meta-update: $g_t(\theta) = \nabla_\theta \mathbb{E}_k(\mathscr{L}(\theta', D_k^{val}))$
- inner gradient update: $\theta' = \theta - \alpha \nabla_\theta \mathscr{L}(\theta, D_k^{tr})$

FTML algorithm

Algorithm 1 Online Meta-Learning with FTML

Input: Performance threshold of proficiency, γ
randomly initialize $\mathbf{w}_1$
initialize the task buffer as empty, $\mathcal{B} \leftarrow [\,]$
for $t = 1, \ldots$ **do**
  initialize $\mathcal{D}_t = \emptyset$
  Add $\mathcal{B} \leftarrow \mathcal{B} + [\,\mathcal{T}_t\,]$
  while $|\mathcal{D}_{\mathcal{T}_t}| < N$ **do**
    Append batch of n new datapoints $\{(\mathbf{x}, \mathbf{y})\}$ to $\mathcal{D}_t$
    $\mathbf{w}_t \leftarrow \texttt{Meta-Update}(\mathbf{w}_t, \mathcal{B}, t)$
    $\tilde{\mathbf{w}}_t \leftarrow \texttt{Update-Procedure}\,(\mathbf{w}_t, \mathcal{D}_t)$
    if $\mathcal{L}\,(\mathcal{D}_t^{\text{test}}, \tilde{\mathbf{w}}_t) < \gamma$ **then**
      Record efficiency for task $\mathcal{T}_t$ as $|\mathcal{D}_t|$ datapoints
    end if
  end while
  Record final performance of $\tilde{\mathbf{w}}_t$ on test set $\mathcal{D}_t^{\text{test}}$ for task t.
  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
end for

Algorithm 2 FTML Subroutines

Input: Hyperparameters parameters α, η
function $\texttt{Meta-Update}(\mathbf{w}, \mathcal{B}, t)$
  for $n_{\text{m}} = 1, \ldots, N_{\text{meta}}$ steps **do**
    Sample task $\mathcal{T}_k$: $k \sim \nu^t(\cdot)$ *// (or a minibatch of tasks)*
    Sample minibatches $\mathcal{D}_k^{\text{tr}}, \mathcal{D}_k^{\text{val}}$ uniformly from $\mathcal{D}_k$
    Compute gradient $\mathbf{g}_t$ using $\mathcal{D}_k^{\text{tr}}, \mathcal{D}_k^{\text{val}}$, and Eq. 5
    Update parameters $\mathbf{w} \leftarrow \mathbf{w} - \eta\, \mathbf{g}_t$ *// (or use Adam)*
  end for
  Return $\mathbf{w}$
end function
function $\texttt{Update-Procedure}(\mathbf{w}, \mathcal{D})$
  Initialize $\tilde{\mathbf{w}} \leftarrow \mathbf{w}$
  for $n_{\text{g}} = 1, \ldots, N_{\text{grad}}$ steps **do**
    $\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{w}} - \alpha \nabla \mathcal{L}(\mathcal{D}, \tilde{\mathbf{w}})$
  end for
  Return $\tilde{\mathbf{w}}$
end function

Experiments

- Rainbow MNIST, **Five-Way CIFAR-100**, Object Pose Prediction
- Last Layer FTML vs. FTML: whether online meta-leaning is simply learning features and performing training on the last layer vs. adapting the features to each task

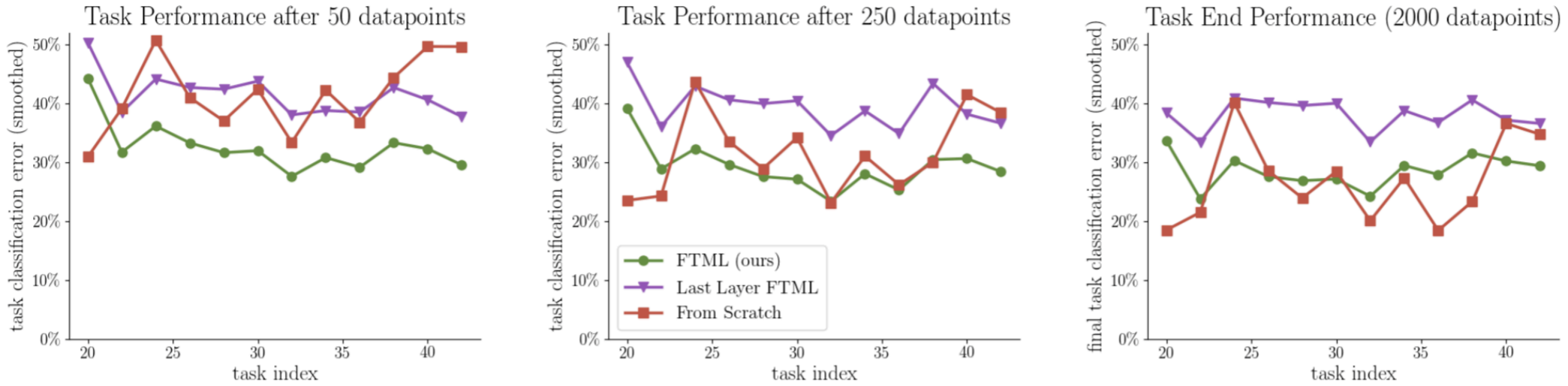


Figure 4. Online CIFAR-100 results, evaluating task performance after seeing 50, 250, and 2000 datapoints for each task. FTML learns each task much more efficiently than models trained from scratch, while both achieve similar asymptotic performance after 2000 datapoints. FTML benefits from adapting all layers rather than learning a shared feature space across tasks while adapting only the last layer.

Future work

- better inner gradient update procedures
- compute- & memory-intensive: store all datapoints -> streaming?