How To Train Your MAML

Dmitry Protasov

MIPT

March 5, 2024

Introduction to Meta-Learning

Few-shot Learning and Meta-Learning

- *Few-shot learning* is challenging without prior knowledge; *meta-learning* automates the acquisition of *across-task knowledge*.
- *Meta-learning*: Models learn to rapidly assimilate *task-specific knowledge* from limited data, enhancing learning proficiency with experience.
- *MAML's* simplicity and adaptability make it a meta-learning framework that achieves state-of-the-art results.

Related Work

Key Developments in Few-Shot Learning

- *Matching Networks* use cosine distance and a differentiable embedding function to match items between support and target sets, converting distances into probability distributions over classes.
- *Gradient-conditional meta-learner LSTM* – learns how to update a base-learner model
- *MAML*: increasing the gradient update steps and using Batch Stochastic Gradient Descent to speed up learning and enhance generalization. SOTA in Omniglot and Mini-Imagenet.
- *Meta-SGD*: learns static learning rate and update direction for each parameter + initialization parameters.

Formal Problem Setting

Base Model and Meta-Parameters

We define a *base model* f_θ with meta-parameters θ. The goal is to learn initial parameters θ_0 so that after N gradient updates with data from a support set S_b, the model performs well on the target set T_b.

Inner-loop update process

Updated base-network parameters after i steps on data from S_b are given by:

$$\theta_i^b = \theta_{i-1}^b - \alpha \nabla_\theta \mathcal{L}_{S_b}(f_{\theta_{i-1}^b}),$$

where α is the learning rate and $\mathcal{L}_{S_b}$ is the loss on the support set after $i - 1$ update steps.

Formal Problem Setting

Meta-objective and Outer-loop update process

The *meta-objective*, reflecting the quality of θ_0 across all tasks, is minimized to optimize θ_0:

$$\mathcal{L}_{meta}(\theta_0) = \sum_{b=1}^{B} \mathcal{L}_{T_b}(f_{\theta_N^b}(\theta_0)),$$

leading to the meta-parameter update:

$$\theta_0 = \theta_0 - \beta \mathcal{L}_{meta}(\theta_0) = \theta_0 - \beta \nabla_\theta \sum_{b=1}^{B} \mathcal{L}_{T_b}(f_{\theta_N^b}(\theta_0)),$$

with β as the meta-learning rate and $\mathcal{L}_{T_b}$ denotes the loss on the target set for task b.

MAML Issues Summary

- Training Instability: MAML may be unstable during training due to gradient issues exacerbated by networks without skip-connections.
- Second Order Derivative Cost: Computationally expensive + first-order approximations negatively impacting generalization.
- BatchNorm Issues: Using current batch statistics rather than accumulated statistics affects generalization performance and model stability.
- Fixed Biases in Batch Normalization: Inner-loop updates use the same batch normalization biases, assuming incorrectly that feature distribution remains constant.
- Shared Learning Rates: complicates hyperparameter tuning
- Fixed Outer Loop Learning Rate: A static learning rate in the outer loop may hinder generalization and optimization, compared to annealing strategies.

Stabilizing MAML

Multi-Step Loss Optimization (MSL)

To counteract gradient instability, we optimiz a weighted sum of losses after each update, enhancing stability and convergence.

$$\theta = \theta - \beta \nabla_\theta \sum_{b=1}^{B} \sum_{i=0}^{N} v_i L_{T_b}(f_{\theta_i^b}) \tag{1}$$

Derivative-Order Annealing (DA)

- use first-order gradients for the first 50 epochs
- then switch to second-order gradients (achieving the strong generalization)

Batch Normalization Running Statistics (BNRS)

Using per-step running statistics for batch normalization to improve optimization and generalization.

Stabilizing MAML

Per-Step Batch Normalization Weights and Biases (BNWB)

To fix this problem of Shared BatchNorm bias we propose learning a set of biases per-step within the inner-loop update process.

Learning Per-Layer Per-Step Learning Rates and Gradient Directions (LSLR)

Learning rates and gradient directions for each layer and each step, reducing memory and computational load.

Cosine Annealing of Meta-Optimizer Learning Rate (CA)

Implementing cosine annealing for the meta-optimizer's learning rate to fit training data better and enhance generalization.

Results

Omniglot 20-way Few-Shot Classification		
	Accuracy	
Approach	**1-shot**	**5-shot**
Siamese Nets	88.2%	97.0%
Matching Nets	93.8%	98.5%
Neural Statistician	93.2%	98.1%
Memory Mod.	95.0%	98.6%
Meta-SGD	95.93±0.38%	98.97±0.19%
Meta-Networks	97.00%	–
MAML (original)	95.8±0.3%	98.9±0.2%
MAML (local replication)	91.27±1.07%	98.78%
MAML++	**97.65±0.05%**	**99.33±0.03%**
MAML + MSL	91.53±0.69%	-
MAML + LSLR	95.77±0.38%	-
MAML + BNWB + BNRS	95.35±0.23%	-
MAML + CA	93.03±0.44%	-
MAML + DA	92.3±0.55%	-

Figure: MAML++ Omniglot 20-way Few-Shot Results

Results

Mini-Imagenet 5-way Few-Shot Classification			
	Inner Steps	**Accuracy**	
Mini-Imagenet		**1-shot**	**5-shot**
Matching Nets	-	43.56%	55.31%
Meta-SGD	1	50.47±1.87%	64.03±0.94%
Meta-Networks	-	49.21%	-
MAML (original paper)	5	48.70±1.84%	63.11±0.92%
MAML (local reproduction)	5	48.25±0.62%	64.39±0.31%
MAML++	1	51.05±0.31%	-
MAML++	2	51.49±0.25%	-
MAML++	3	51.11±0.11%	-
MAML++	4	51.65±0.34%	-
MAML++	5	**52.15±0.26%**	**68.32±0.44%**

Figure: MAML++ Mini-Imagenet Results.

Results

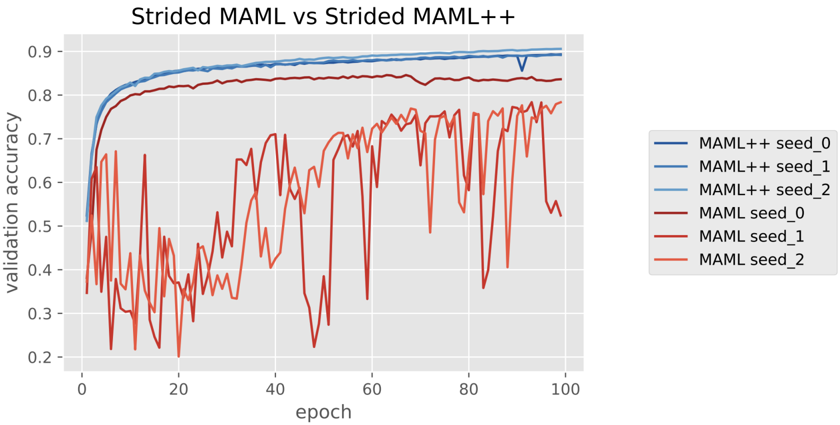

Figure: Stabilizing MAML: This figure illustrates 3 seeds of the original strided MAML vs strided MAML++. One can see that 2 out of 3 seeds with the original strided MAML seem to become unstable and erratic, whereas all 3 of the strided MAML++ models seem to consistently converge very fast, to much higher generalization accuracy without any stability issues

Literature

1. **Main article** How To Train Your MAML