Efficient Meta-Learning for Continual Learning with Taylor Expansion Approximation

Xiaohan Zou

Boston University

Tong Lin

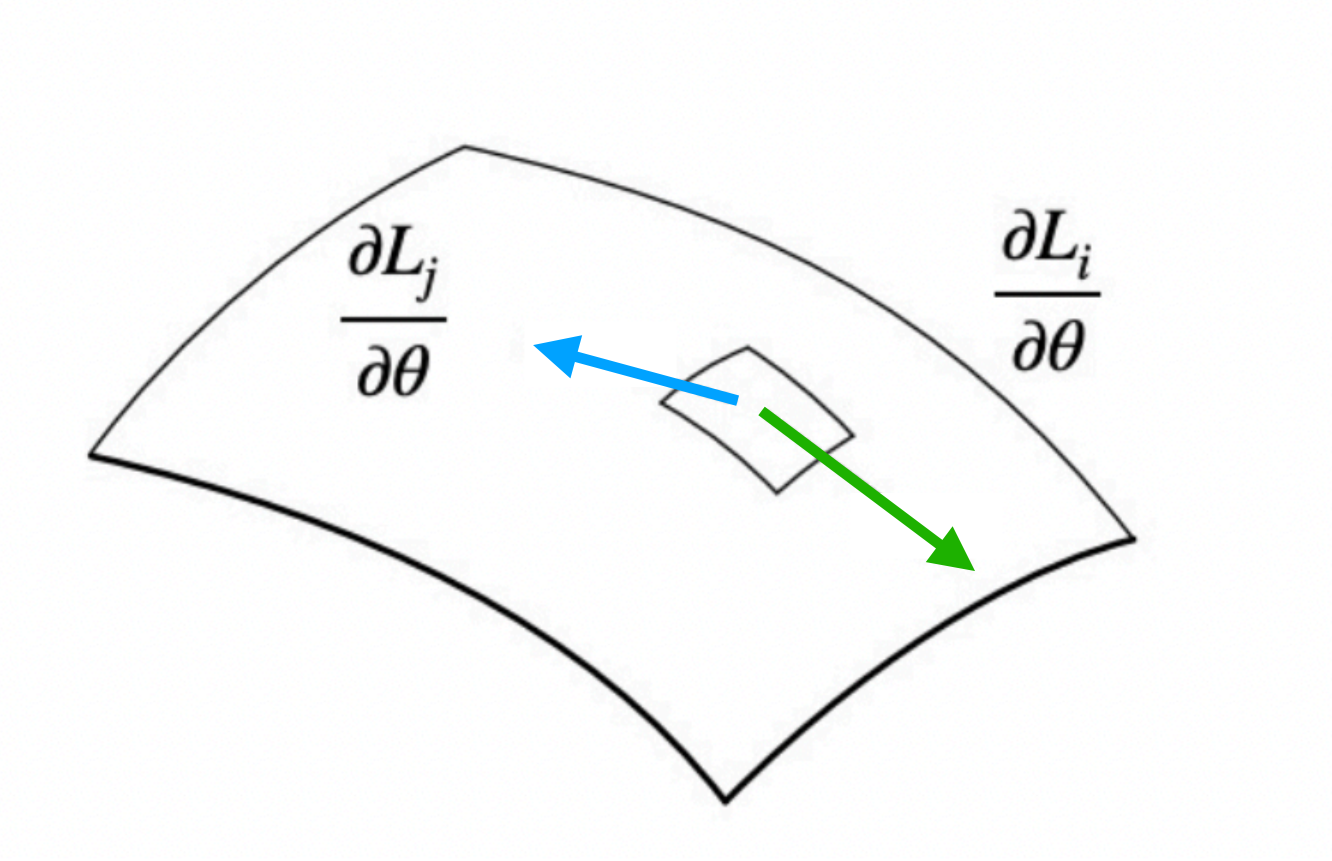
Key Lab. of Machine Perception (MoE),
School of AI, Center for Data Science,
Peking University

IJCNN 2022 Oral

Continual Learning - An Optimization View

Sequential task learning

Interference[1]



Catastrophic forgetting

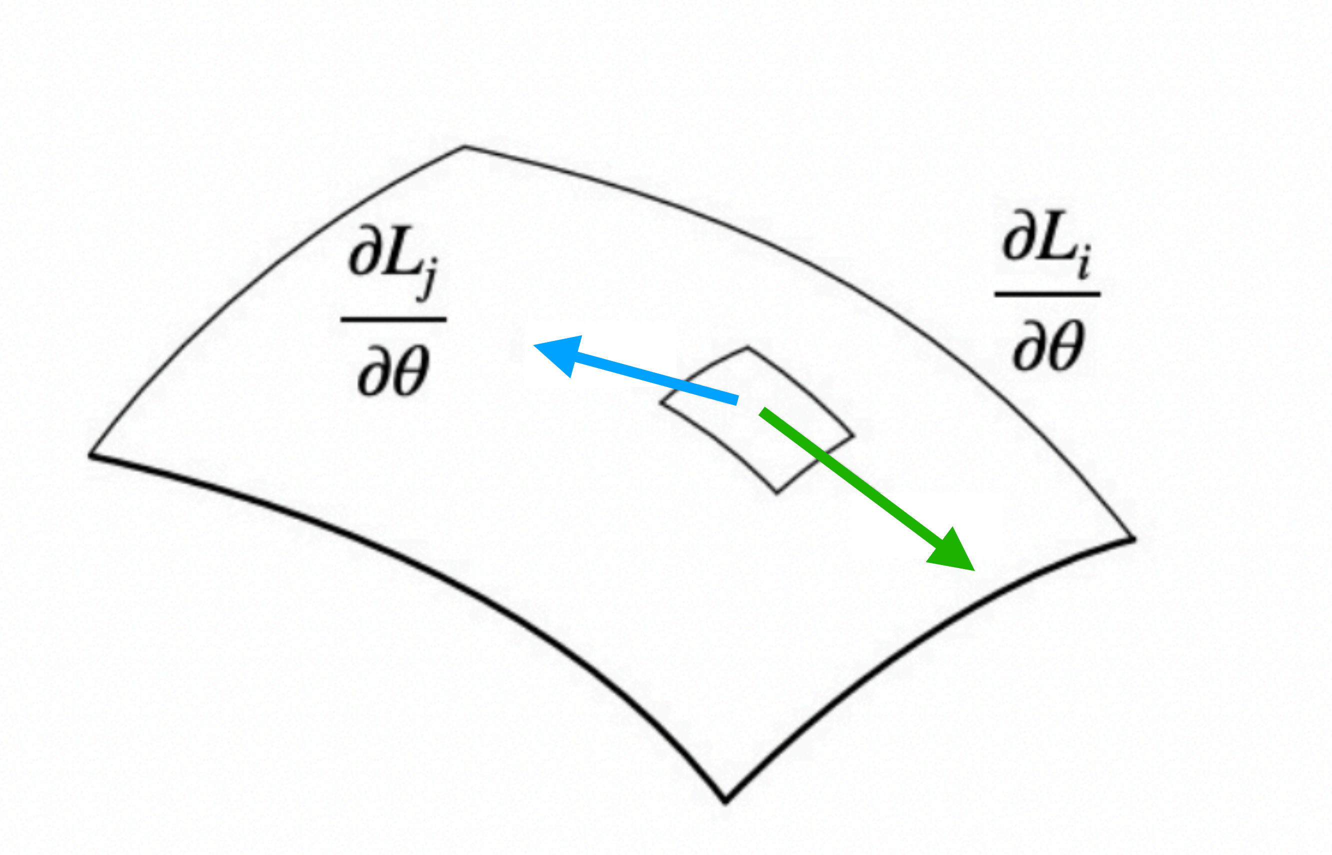
1. "Learning to learn without forgetting by maximizing transfer and minimizing interference" (Riemer et al., ICLR 2019)

Continual Learning - An Optimization View

Sequential task learning

↓

Interference[1]



↓

Catastrophic forgetting

Existing solutions: gradient alignment[2]

$$\theta^j = \arg\min_{\theta^j} \left(\sum_{i=1}^{t} \ell_i(\theta^j) - \alpha \sum_{p,q \leq t} \left(\frac{\partial \ell_p(\theta^j)}{\partial \theta^j} \cdot \frac{\partial \ell_q(\theta^j)}{\partial \theta^j} \right) \right)$$

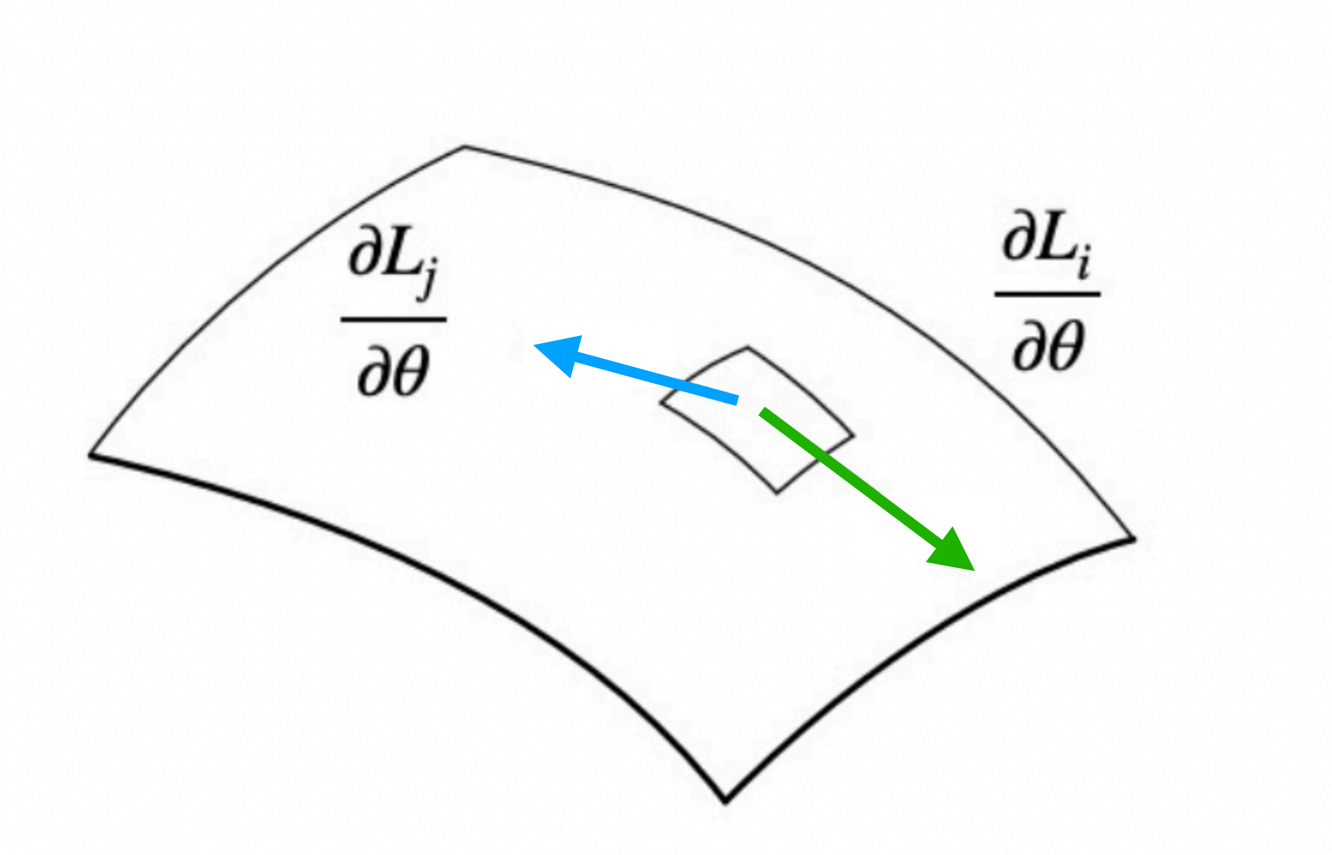
1. "Learning to learn without forgetting by maximizing transfer and minimizing interference" (Riemer et al., ICLR 2019)
2. "Gradient episodic memory for continual learning" (Lopez-Paz et al., NIPS 2017)

Continual Learning - An Optimization View

Sequential task learning

↓

Interference[1]



↓

Catastrophic forgetting

Existing solutions: gradient alignment[2]

$$\theta^j = \arg\min_{\theta^j} \left(\sum_{i=1}^{t} \ell_i(\theta^j) - \alpha \sum_{p,q \le t} \left(\frac{\partial \ell_p(\theta^j)}{\partial \theta^j} \cdot \frac{\partial \ell_q(\theta^j)}{\partial \theta^j} \right) \right)$$

↕ Optimize for the same objective[1]

Model-Agnostic Meta-Learning (MAML)[3]

$$\underbrace{\min_{\theta_0^j} \mathbb{E}_{\tau_{1:t}} \left[L_{\text{meta}} \left(\overbrace{U_k(\theta_0^j)}^{\text{inner-loop}} \right) \right]}_{\text{outer-loop}} = \min_{\theta_0^j} \mathbb{E}_{\tau_{1:t}} \left[L_{\text{meta}} \left(\theta_k^j \right) \right]$$

1. "Learning to learn without forgetting by maximizing transfer and minimizing interference" (Riemer et al., ICLR 2019)
2. "Gradient episodic memory for continual learning" (Lopez-Paz et al., NIPS 2017)
3. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks" (Finn et al., ICML 2017)

MAML for Continual Learning

Problems:

- Interference between old and new task still exists
  (because old data is no longer available)
- Need to compute second-order derivative (Hessian matrix)

$$\underbrace{\min_{\theta_0^j} \mathbb{E}_{\tau_{1:t}} \left[L_{\text{meta}} \left(\overbrace{U_k(\theta_0^j)}^{\text{inner-loop}} \right) \right]}_{\text{outer-loop}} = \min_{\theta_0^j} \mathbb{E}_{\tau_{1:t}} \left[L_{\text{meta}} \left(\theta_k^j \right) \right]$$

MAML for Continual Learning

Problems:

- Interference between old and new task still exists (since old data is no longer available)
- Need to compute second-order derivative (Hessian matrix)

Existing solutions:

- Store old data (privacy & memory)
- Still exists

MAML for Continual Learning

Problems:

- Interference between old and new task still exists (since old data is no longer available)
- Need to compute second-order derivative (Hessian matrix)

Our approach:

according to meta-parameter importance

- Inner update: adds regularization terms to loss function

  Outer (Meta) update: adapts learning rate
- First-order approximation

Inner-Update: Explicit Regularization

Inner-update objective:

$$
\begin{aligned}
\theta_k^j &= \arg\min_{\theta_k^j} \ell_i(\theta_k^j) \\
&= \arg\min_{\theta_k^j} \left\{ \mathcal{L}(\theta_k^j) + \underbrace{\boxed{\frac{\lambda}{2} \sum_m h_m^j \left\| \theta_{k,m}^j - \theta_{0,m}^j \right\|_2^2}}_{} \right\} \\
&= \arg\min_{\theta_k^j} \left\{ \mathcal{L}(\theta_k^j) + \frac{\lambda}{2} \left\| \mathbf{H}^j (\theta_k^j - \theta_0^j) \right\|_2^2 \right\} .
\end{aligned}
$$

regularization term

Motivation:

- Alleviate vanishing gradients

Inner-Update: Explicit Regularization

Inner-update objective:

moving average of importance of m-th meta-parameter Ω^j_m

$$\begin{aligned}\theta^j_k &= \arg\min_{\theta^j_k} \ell_i(\theta^j_k) \\ &= \arg\min_{\theta^j_k} \left\{ \mathcal{L}(\theta^j_k) + \frac{\lambda}{2} \sum_m \boxed{h^j_m} \|\theta^j_{k,m} - \theta^j_{0,m}\|^2_2 \right\} \\ &= \arg\min_{\theta^j_k} \left\{ \mathcal{L}(\theta^j_k) + \frac{\lambda}{2} \left\| \boxed{\mathbf{H}^j} (\theta^j_k - \theta^j_0) \right\|^2_2 \right\}.\end{aligned}$$

diagonal matrix, where $H^j_{m,m} = \sqrt{h^j_m}$

Motivation:

- Alleviate vanishing gradients
- Alleviate catastrophic forgetting

Inner-Update: Explicit Regularization

Inner-update objective:

$$
\begin{aligned}
\theta_k^j &= \arg\min_{\theta_k^j} \ell_i(\theta_k^j) \\
&= \arg\min_{\theta_k^j} \left\{ \mathcal{L}(\theta_k^j) + \frac{\lambda}{2} \sum_m h_m^j \left\| \theta_{k,m}^j - \theta_{0,m}^j \right\|_2^2 \right\} \\
&= \arg\min_{\theta_k^j} \left\{ \mathcal{L}(\theta_k^j) + \frac{\lambda}{2} \left\| \mathbf{H}^j \left(\boxed{\theta_k^j} - \theta_0^j \right) \right\|_2^2 \right\} .
\end{aligned}
$$

minimizer

Motivation:

- Alleviate vanishing gradients
- Alleviate catastrophic forgetting
- Avoid computing Hessian matrix during **meta-update**

Meta-Update: Closed-form First-Order Approximation

The gradient of the inner-loss should be zero:

$$\nabla L(\theta_k) + \lambda \mathbf{H}^2(\theta_k - \theta_0) = 0.$$

Meta-Update: Closed-form First-Order Approximation

The gradient of the inner-loss should be zero:

$$\boxed{\nabla L(\theta_k) + \lambda \mathbf{H}^2(\theta_k - \theta_0)} = 0.$$

Therefore:

$$\begin{aligned}
\nabla_{\theta_0} L_t(\theta_k) &= \left(\frac{\partial \theta_k}{\partial \theta_0}\right)^\top \nabla L(\theta_k) \\
&\quad + \lambda \left(\left(\frac{\partial \theta_k}{\partial \theta_0}\right)^\top - I \right) \mathbf{H}^2(\theta_k - \theta_0) \\
&= \frac{\partial \theta_k}{\partial \theta_0} \boxed{\left(\nabla L(\theta_k) + \lambda \mathbf{H}^2(\theta_k - \theta_0)\right)} - \lambda \mathbf{H}^2(\theta_k - \theta_0) \\
&= \lambda \mathbf{H}^2(\theta_0 - \theta_k).
\end{aligned}$$

Meta-Update: Closed-form First-Order Approximation

The gradient of the inner-loss should be zero：

$$\nabla L(\theta_k) + \lambda \mathbf{H}^2(\theta_k - \theta_0) = 0.$$

Therefore:

$$\begin{aligned}
\nabla_{\theta_0} L_t(\theta_k) &= \left(\frac{\partial \theta_k}{\partial \theta_0}\right)^\top \nabla L(\theta_k) \\
&\qquad + \lambda \left(\left(\frac{\partial \theta_k}{\partial \theta_0}\right)^\top - I \right) \mathbf{H}^2(\theta_k - \theta_0) \\
&= \frac{\partial \theta_k}{\partial \theta_0} \left(\nabla L(\theta_k) + \lambda \mathbf{H}^2(\theta_k - \theta_0)\right) - \lambda \mathbf{H}^2(\theta_k - \theta_0) \\
&= \boxed{\lambda \mathbf{H}^2(\theta_0 - \theta_k).}
\end{aligned}$$

first-order approximation

Meta-Update: Adaptive Learning Rate

Scale the learning rate for each parameter inversely proportional to the moving average of its importance:

$$\alpha_m^j \leftarrow \frac{r}{h_m^j} \alpha_m^{j-1},$$

In this way:

- Changes in important parameters can be reduced
- Less important parameters allow having larger step sizes in future tasks

Meta-Parameter Importance Estimation

The importance of the m-th meta-parameter can be quantified by the impact on the total loss after zeroing it out:

$$\Omega_m = \left| L_t(\theta_0) - L_t\left(\theta_0\big|_{\theta_{0,m}=0}\right) \right|.$$

Meta-Parameter Importance Estimation

The importance of the m-th meta-parameter can be quantified by the impact on the total loss after zeroing it out:

$$\Omega_m = \left| L_t(\theta_0) - L_t\left(\theta_0|_{\theta_{0,m}=0}\right) \right|.$$

Approximate $L_t\left(\theta_0|_{\theta_{0,m}=0}\right)$ using first-order Taylor expansion:

$$L_t\left(\theta_0|_{\theta_{0,m}=0}\right) = L_t(\theta_0) + \frac{\partial L_t(\theta_0)}{\partial \theta_{0,m}}(\theta_{0,m} - 0) + o(\theta_{0,m})$$

Meta-Parameter Importance Estimation

The importance of the m-th meta-parameter can be quantified by the impact on the total loss after zeroing it out:

$$\Omega_m = \left| L_t(\theta_0) - L_t\left(\theta_0|_{\theta_{0,m}=0}\right) \right|.$$

Approximate $L_t\left(\theta_0|_{\theta_{0,m}=0}\right)$ using first-order Taylor expansion:

$$L_t\left(\theta_0|_{\theta_{0,m}=0}\right) = L_t(\theta_0) + \frac{\partial L_t(\theta_0)}{\partial \theta_{0,m}}(\theta_{0,m} - 0) + o(\theta_{0,m})$$

Finally:

meta-gradient, which is already available after inner-update

$$\Omega_m = \left| L_t\left(\theta_0|_{\theta_{0,m}=0}\right) - L_t(\theta_0) \right| \approx \left| \boxed{\frac{\partial L_t(\theta_0)}{\partial \theta_{0,m}}} \theta_{0,m} \right|$$

Inner-Update: Proximal Gradient Descent (PGD)

Inner-update objective:

$$\theta_k^j = \arg\min_{\theta_k^j} \left\{ \mathcal{L}(\theta_k^j) + \frac{\lambda}{2} \sum_m h_m^j \left\| \theta_{k,m}^j - \theta_{0,m}^j \right\|_2^2 \right\}$$

Inner-Update: Proximal Gradient Descent (PGD)

Inner-update objective:

$$\theta_k^j = \arg\min_{\theta_k^j} \left\{ \mathcal{L}(\theta_k^j) + \overset{f(\theta_{k,m})}{\boxed{\frac{\lambda}{2} \sum_m h_m^j \|\theta_{k,m}^j - \theta_{0,m}^j\|_2^2}} \right\}$$

$$f(\theta_{k,m}) = \frac{\lambda}{2} h_m \|\theta_{k,m} - \theta_{0,m}\|_2^2$$

Inner-Update: Proximal Gradient Descent (PGD)

Inner-update objective:

$$\theta_k^j = \arg\min_{\theta_k^j} \left\{ \mathcal{L}(\theta_k^j) + \frac{\lambda}{2} \sum_m h_m^j \|\theta_{k,m}^j - \theta_{0,m}^j\|_2^2 \right\}$$

$$f(\theta_{k,m}) = \frac{\lambda}{2} h_m \|\theta_{k,m} - \theta_{0,m}\|_2^2$$

Proximal operator of f:

$$\mathrm{prox}_{\gamma f}(v) = \arg\min_x \left(f(x) + \frac{1}{2\gamma} \|x - v\|_2^2 \right)$$

Inner-Update: Proximal Gradient Descent (PGD)

Inner-update objective:

$$\theta_k^j = \arg\min_{\theta_k^j} \left\{ \mathcal{L}(\theta_k^j) + \frac{\lambda}{2} \sum_m h_m^j \|\theta_{k,m}^j - \theta_{0,m}^j\|_2^2 \right\}$$

$$f(\theta_{k,m}) = \frac{\lambda}{2} h_m \|\theta_{k,m} - \theta_{0,m}\|_2^2$$

Proximal operator of f:

$$\mathrm{prox}_{\gamma f}(v) = \arg\min_x \left(f(x) + \frac{1}{2\gamma} \|x - v\|_2^2 \right)$$

Modified gradient steps ($\kappa = \{1, \ldots, k\}$):

$$\hat{\theta}_\kappa = \theta_{\kappa-1} - \gamma \nabla L(\theta_{\kappa-1}),$$

$$\theta_\kappa = \mathrm{prox}_{\gamma f}(\hat{\theta}_\kappa),$$

Inner-Update: Proximal Gradient Descent (PGD)

Inner-update objective:

$$\theta_k^j = \arg\min_{\theta_k^j} \left\{ \mathcal{L}(\theta_k^j) + \frac{\lambda}{2} \sum_m h_m^j \|\theta_{k,m}^j - \theta_{0,m}^j\|_2^2 \right\}$$

$$f(\theta_{k,m}) = \frac{\lambda}{2} h_m \|\theta_{k,m} - \theta_{0,m}\|_2^2$$

Proximal operator of f:

$$\mathrm{prox}_{\gamma f}(v) = \arg\min_x \left(f(x) + \frac{1}{2\gamma} \|x - v\|_2^2 \right)$$

Modified gradient steps ($\kappa = \{1, \ldots, k\}$):

$$\hat{\theta}_\kappa = \theta_{\kappa-1} - \gamma \nabla L(\theta_{\kappa-1}),$$

$$\theta_\kappa = \mathrm{prox}_{\gamma f}(\hat{\theta}_\kappa),$$

Closed-form proximal gradient update:

$$\theta_{\kappa,m} = \frac{\hat{\theta}_{\kappa,m} + \gamma\lambda h_m \theta_{0,m}}{\gamma\lambda h_m + 1}$$

Experiments: Setup

- Datasets: MNIST Permutations, Many Permutations, Split CIFAR100, Split miniImageNet
- Architecture: MLP (2 layers, 100 ReLU units), ResNet18
- Metric: Average Accuracy (ACC), Backward Transfer (BWT)

Method	MNIST Perm.		Many Perm.		CIFAR100		miniImageNet	
	ACC	BWT	ACC	BWT	ACC	BWT	ACC	BWT
EWC	62.32 ± 1.34 †	-13.32 ± 2.24 †	33.46 ± 0.46 †	-17.84 ± 1.15 †	39.60 ± 1.11	-23.53 ± 1.19	34.34 ± 2.06	-28.17 ± 1.49
GEM	55.42 ± 1.10 †	-24.42 ± 1.10 †	32.14 ± 0.50 †	-23.52 ± 0.87 †	43.41 ± 2.09	-20.76 ± 1.31	37.02 ± 1.91	-25.29 ± 2.10
A-GEM	56.04 ± 2.36	-24.05 ± 2.47	29.98 ± 1.84	-27.23 ± 1.79	43.87 ± 2.61	-23.38 ± 1.52	36.37 ± 1.56	-25.11 ± 2.92
MER	73.46 ± 0.45 †	-9.96 ± 0.45 †	47.40 ± 0.35 †	-17.78 ± 0.39 †	-	-	-	-
La-MAML	**73.92 ± 1.05**	-7.91 ± 0.87	47.69 ± 0.41	-13.24 ± 0.95	61.23 ± 0.94	-19.84 ± 2.20	45.29 ± 1.76	-18.57 ± 2.94
EMCL	73.61 ± 1.12	-10.25 ± 0.73	**48.12 ± 1.48**	-14.09 ± 0.74	**61.95 ± 1.20**	-16.48 ± 1.96	**46.52 ± 0.83**	-17.45 ± 2.38

Experiments: Result Summary for Our Method

- Outperforms commonly used baselines (EWC, GEM and A-GEM) significantly
- Better or on-par performance compared to MER and La-MAML (MAML based)
- Memory-based methods are not very effective when the size of replay buffer is small

Method	MNIST Perm.		Many Perm.		CIFAR100		miniImageNet	
	ACC	BWT	ACC	BWT	ACC	BWT	ACC	BWT
EWC	62.32 ± 1.34 †	-13.32 ± 2.24 †	33.46 ± 0.46 †	-17.84 ± 1.15 †	39.60 ± 1.11	-23.53 ± 1.19	34.34 ± 2.06	-28.17 ± 1.49
GEM	55.42 ± 1.10 †	-24.42 ± 1.10 †	32.14 ± 0.50 †	-23.52 ± 0.87 †	43.41 ± 2.09	-20.76 ± 1.31	37.02 ± 1.91	-25.29 ± 2.10
A-GEM	56.04 ± 2.36	-24.05 ± 2.47	29.98 ± 1.84	-27.23 ± 1.79	43.87 ± 2.61	-23.38 ± 1.52	36.37 ± 1.56	-25.11 ± 2.92
MER	73.46 ± 0.45 †	-9.96 ± 0.45 †	47.40 ± 0.35 †	-17.78 ± 0.39 †	-	-	-	-
La-MAML	**73.92 ± 1.05**	-7.91 ± 0.87	47.69 ± 0.41	-13.24 ± 0.95	61.23 ± 0.94	-19.84 ± 2.20	45.29 ± 1.76	-18.57 ± 2.94
EMCL	73.61 ± 1.12	-10.25 ± 0.73	**48.12 ± 1.48**	-14.09 ± 0.74	**61.95 ± 1.20**	-16.48 ± 1.96	**46.52 ± 0.83**	-17.45 ± 2.38

Experiments: Result Summary for Our Method

- Outperforms commonly used baselines (EWC, GEM and A-GEM) significantly
- Better or on-par performance compared to MER and La-MAML (MAML based)
- Memory-based methods are not very effective when the size of replay buffer is small

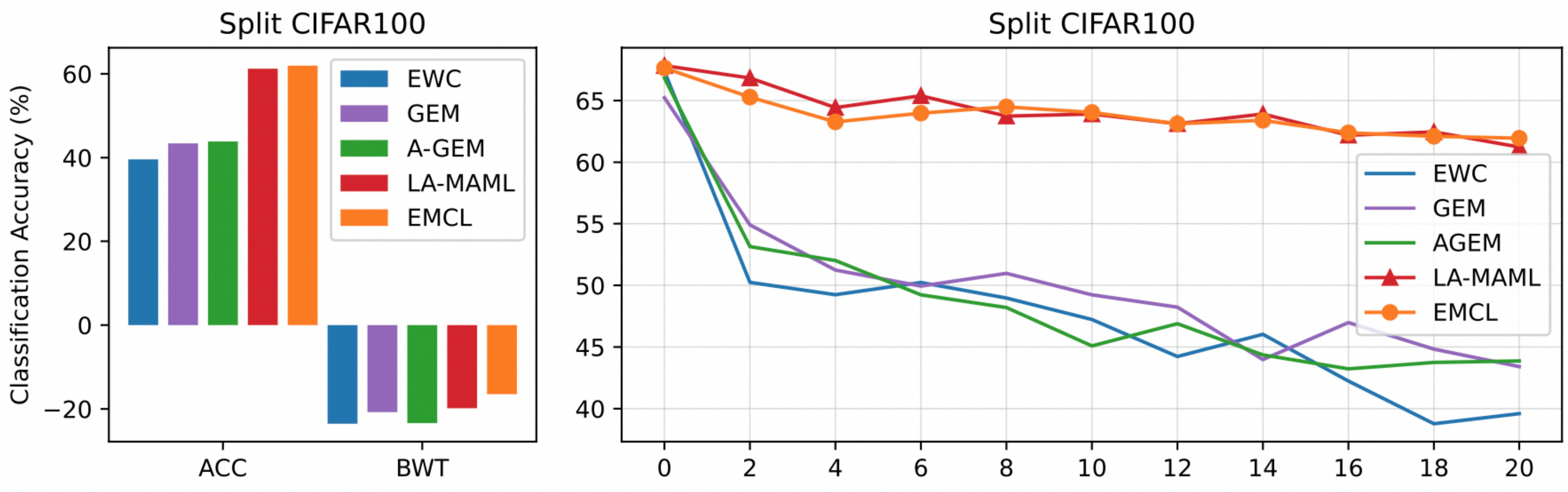


Fig. 1: Left: ACC and BWT for all appraoches on CIFAR100. Right: evolution of the average test accuracy as more tasks are learned.

Experiments: Training Time

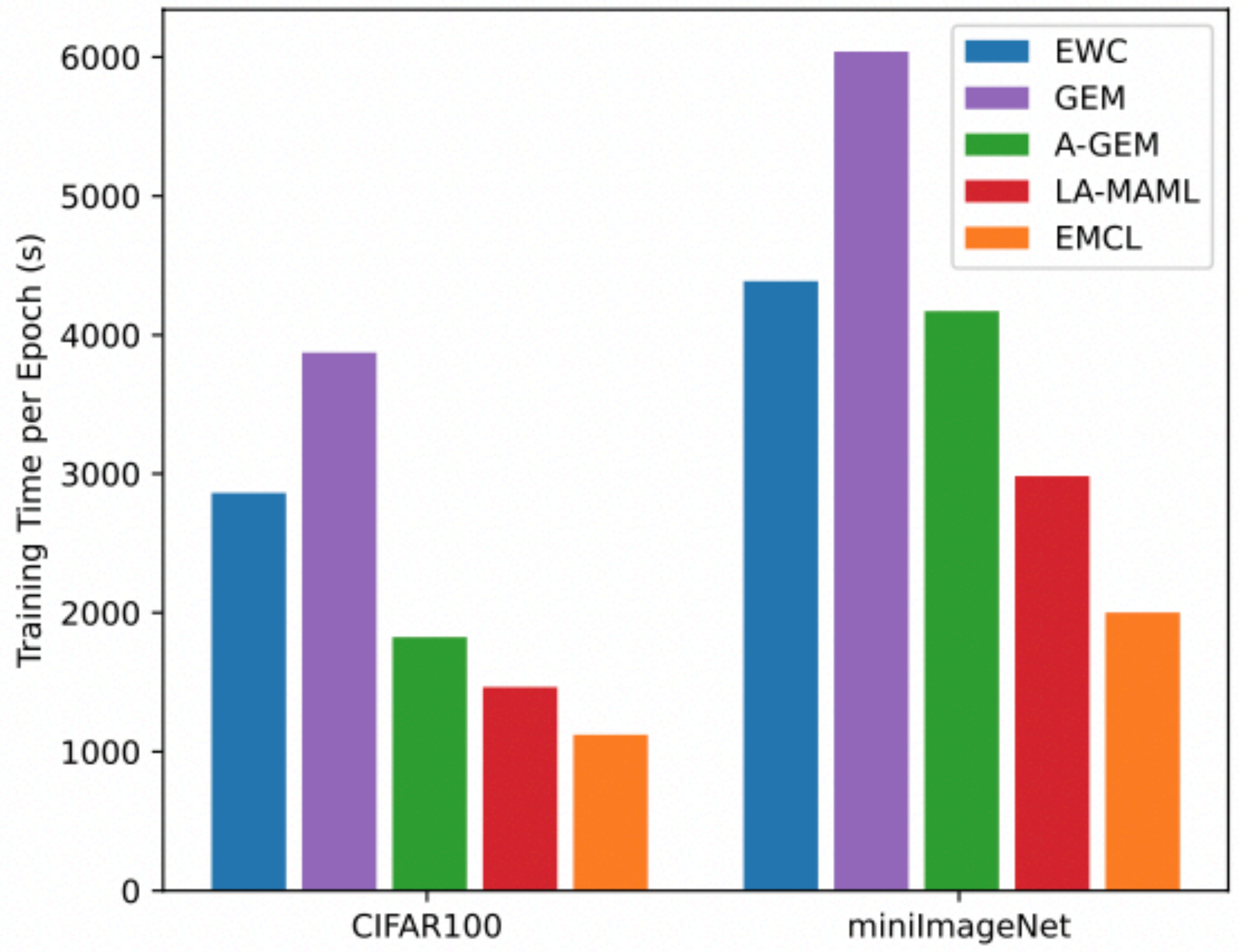


Fig. 2: Training time for all algorithms on Split CIFAR100 and miniImageNet.

Setup:

- Measured on a single GPU
- Include time spent for memory management and weight importance calculation

Results:

- Faster than all baselines (while achieving higher or on-par accuracy)

Analysis:

- No need to compute second-order derivatives
- Efficient weight importance calculation
- Inner-update using proximal gradient descent

Conclusion

- Introduced a novel meta-learning algorithm for continual learning problems
- Modulated the meta-update learning rates and add explicit regularization terms to the inner loss to alleviate catastrophic forgetting
- The proposed method is fast, because it
  - Expresses the gradient of meta-updated in closed-form to avoid accessing the Hessian information
  - Uses proximal gradient descent to solve the inner objective easier and improve the computational efficiency
  - Estimates parameter importance efficiently using the Taylor series

Thanks!