Evaluating Data Influence in Meta Learning

Chenyang Ren, Huanyi Xie, Shu Yang, Meng Ding, Dongrui Liu, Lijie Hu, Di Wang
KAUST MBZUAI SJTU SUNY Buffalo Shanghai AI Laboratory

Motivation

Meta-learning is naturally a **bilevel optimization** problem: the inner loop adapts task-specific parameters, while the outer loop updates shared meta-parameters. In practice, however, not all tasks or instances are equally useful:

- some tasks are redundant,
- some samples are mislabeled or harmful,
- some data dominate meta-updates disproportionately.

This paper studies a central question:

How can we estimate the influence of tasks and samples in meta learning without full retraining?

Bilevel Formulation

For each task i, let $D_i = D_i^{\mathrm{tr}} \cup D_i^{\mathrm{val}}$. The meta-learning objective is:

$$\theta_i(\lambda) = \arg\min_{\theta_i} L_I(\lambda, \theta_i; D_i^{\mathrm{tr}}), \qquad \lambda^* = \arg\min_{\lambda} \sum_{i \in \mathcal{I}} L_O(\lambda, \theta_i(\lambda); D_i^{\mathrm{val}}) + \frac{\delta}{2}\|\lambda\|^2.$$

Key challenge: data affect the meta-parameter λ both directly and indirectly through the adapted parameter $\theta_i(\lambda)$.

Why Classical IF Fails in Meta Learning

Classical influence functions assume a **single-level objective**. That is not enough here, because the outer loss depends on the inner adaptation path.

$$D_\lambda L_O(\lambda, \theta_i(\lambda); D_i^{\mathrm{val}}) = \partial_\lambda L_O + \frac{d\theta_i(\lambda)}{d\lambda} \partial_{\theta_i} L_O$$

Therefore, accurate attribution in meta learning must use **total gradients** and **total Hessians**, not only partial derivatives.

Core Idea

The sensitivity of the adapted parameter is obtained by implicit differentiation:

$$\frac{d\theta_i(\lambda)}{d\lambda} = -\partial_{\lambda\theta_i} L_I(\lambda, \theta_i(\lambda); D_i^{\mathrm{tr}})\, H_{i,\mathrm{in}}^{-1}, \qquad H_{i,\mathrm{in}} = \partial_{\theta\theta_i} L_I$$

This gives the **total Hessian** for the meta objective:

$$H_{\lambda,\mathrm{Total}} = \sum_{i \in \mathcal{I}} D_\lambda D_\lambda L_O(\lambda, \theta_i(\lambda); D_i^{\mathrm{val}})$$

Task Influence Function (task-IF)

Removing task k changes both the inner adaptation and the outer objective. The proposed task-level influence is:

$$\text{task-IF}(D_k) = -(\delta I + H_{\lambda,\mathrm{Total}})^{-1} D_\lambda L_O(\lambda^*, \theta_k(\lambda^*); D_k^{\mathrm{val}})$$

The retrained meta-parameter after deleting task k is approximated by:

$$\lambda^*_{-k} \approx \lambda^* - \text{task-IF}(D_k)$$

Key intuition

A task is influential if it induces a large bilevel-aware gradient signal and this signal remains important after curvature correction.

Instance Influence Function (instance-IF)

Validation sample. If $\tilde{z} \in D_k^{\mathrm{val}}$, the effect is measured directly at the outer level:

$$\text{instance-IF}_{\mathrm{val}}(\tilde{z}) = -(\delta I + H_{\lambda,\mathrm{Total}})^{-1} \nabla \ell(\tilde{z}; \theta_k(\lambda^*))$$

Training sample. If $\tilde{z} \in D_k^{\mathrm{tr}}$, it only influences the meta-parameter through the adapted task parameter.
Stage 1: influence on the adapted parameter

$$\text{inner-IF}(\tilde{z}) = -H_{k,\mathrm{in}}^{-1} \nabla_{\theta_k} \ell(\tilde{z}; \theta_k(\lambda^*))$$

Stage 2: propagate to the meta level

$$\text{instance-IF}_{\mathrm{tr}}(\tilde{z}) = -H_{\lambda,\mathrm{Total}}^{-1} D_\lambda P(\lambda^*, \theta_k(\lambda^*); \tilde{z})$$

This two-stage construction is the key novelty for training-sample attribution.

Efficient Approximation

The exact formulas require repeated inverse-Hessian vector products, which are expensive for neural networks. The paper uses:

- **EK-FAC** for efficient approximation of the inner Hessian inverse;
- **Neumann series** for approximating the inverse total Hessian.

$$\widetilde{H}_{\lambda,\mathrm{Total}}^{-1} v \approx v + (I - \widetilde{H}_{\lambda,\mathrm{Total}}) v + \cdots + (I - \widetilde{H}_{\lambda,\mathrm{Total}})^J v$$

Task / sample → Gradient → Curvature → Updated meta-model

Experimental Setup

Datasets: Omniglot, MNIST, Mini-ImageNet, FC100.
Baselines: Retraining, Direct-IF, EKFAC IF, TRAK, TracIn.
Metrics:

- test accuracy after removing harmful tasks / instances,
- runtime for updating the model.

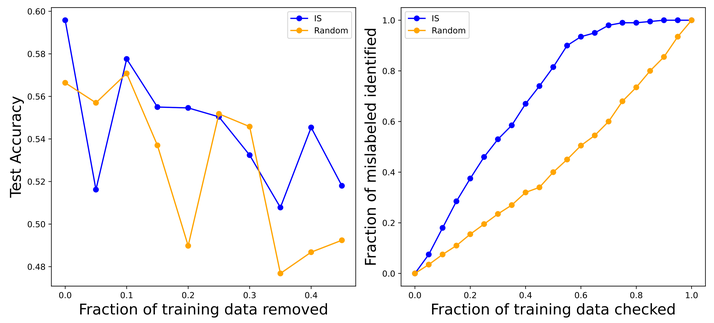
Main Quantitative Result

Omniglot task-removal benchmark

Method	Omniglot Acc.	Runtime (s)
Retrain	0.7908	316.74
Task-IF	**0.7804**	65.69
Direct-IF	0.5422	**7.46**
TracIn	0.7690	375.42

✓ **Task-IF gives the best trade-off:** it is much closer to retraining than Direct-IF, while being substantially faster than full retraining.

Visual Evidence



Harmful-task removal: Task-IF finds mislabeled tasks much earlier than random checking and improves downstream accuracy as harmful tasks are removed.