



LancBiO: Dynamic Lanczos-aided Bilevel Optimization via Krylov Subspace

Yan Yang

March 30, 2025

State Key Laboratory of Scientific and Engineering Computing
Institute of Computational Mathematics and Scientific/Engineering Computing
Academy of Mathematics and Systems Science
Chinese Academy of Sciences, China

Joint work with **Dr. Bin Gao** and **Prof. Ya-xiang Yuan**

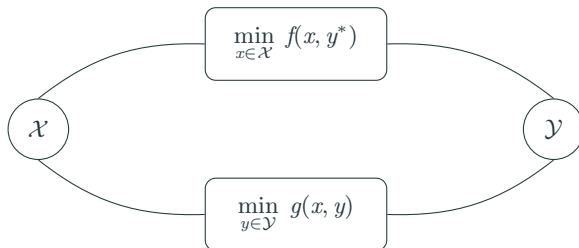
1. Bilevel Optimization (BiO) Problems
2. A Subspace Technique
3. A Dynamic Lanczos-aided Framework
4. Theoretical Analysis
5. Numerical Experiments

Bilevel Optimization (BiO) Problems

Bilevel Optimization (BiO)

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & f(x, y^*) \\ \text{s.t.} \quad & y^* \in \arg \min_{y \in \mathcal{Y}} g(x, y) \end{aligned}$$

The nested structure **couples** the **upper level** and **lower level**



Model selection [Kunapuli et al., 2008; Giovannelli et al., 2021]

Hyper-parameters optimization [Franceschi et al., 2018; Bao et al., 2021]

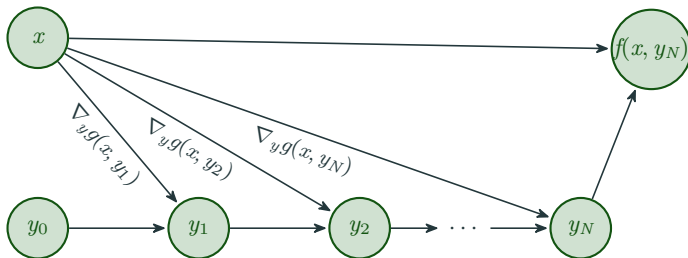
Data poisoning [Liu et al., 2024]

Reinforcement learning [Hong et al., 2023; Chakraborty et al., 2024; Thoma et al., 2024; Yang et al., 2025]

...

I. Automatic differentiation based methods

Main idea



Existing works

- Algorithm Designs: [Domke, 2012; Franceschi et al., 2018; Shaban et al., 2019; Grazi et al., 2020]
- Theoretical Analysis: [Ji, 2021; Gu and Huang, 2022; Zhang et al., 2024]

II. Reformulation based methods

- Value function based reformulation:

$$\min_{x \in \mathcal{X}, y \in \mathcal{Y}} f(x, y), \quad \text{s.t. } g(x, y) \leq g^*(x)$$

- Optimality condition based reformulation:

$$\min_{x \in \mathcal{X}, y \in \mathbb{R}^{d_y}} f(x, y), \quad \text{s.t. } \nabla_y g(x, y) = 0$$

Algorithm	Objective functions		Reference
	Upper-Level	Lower-Level	
Smoothing PG	Smooth	Convex constraint	[Lin-Xu-Ye, 2014]
BOME	Smooth	PL	[Liu-Ye-Wright-Stone-Liu, 2022]
PBGD	Smooth	PL	[Shen-Chen, 2023]
F2SA	Smooth	Strongly Convex	[Kwon-Kwon-Wright-Nowak, 2023]
—	Slater's Constraint Qualification		[Dempe-Dutta, 2012]
CDB	Non-Smooth	Strongly Convex	[Hu-Xiao-Liu-Toh, 2023]
EBSA	Smooth	Non-convex	[Xu-Dai-Liu-Wang, 2024]

II. Reformulation based methods

- Value function based reformulation:

$$\min_{x \in \mathcal{X}, y \in \mathcal{Y}} f(x, y), \quad \text{s.t. } g(x, y) \leq g^*(x)$$

- Optimality condition based reformulation:

$$\min_{x \in \mathcal{X}, y \in \mathbb{R}^{d_y}} f(x, y), \quad \text{s.t. } \nabla_y g(x, y) = 0$$

Algorithm	Objective functions		Reference
	Upper-Level	Lower-Level	
Smoothing PG	Smooth	Convex constraint	[Lin-Xu-Ye, 2014]
BOME	Smooth	PL	[Liu-Ye-Wright-Stone-Liu, 2022]
PBGD	Smooth	PL	[Shen-Chen, 2023]
FZSA	Smooth	Strongly Convex	[Kwon-Kwon-Wright-Nowak, 2023]
---	Slater's Constraint Qualification		[Dempe-Dutta, 2012]
CDB	Non-Smooth	Strongly Convex	[Hu-Xiao-Liu-Toh, 2023]
EBSA	Smooth	Non-convex	[Xu-Dai-Liu-Wang, 2024]

II. Reformulation based methods

- Value function based reformulation

$$\min_{x \in \mathcal{X}, y \in \mathcal{Y}} f(x, y), \quad \text{s.t. } g(x, y) \leq g^*(x)$$

- Optimality condition based reformulation

$$\min_{x \in \mathcal{X}, y \in \mathbb{R}^{d_y}} f(x, y), \quad \text{s.t. } \nabla_y g(x, y) = 0$$

Algorithm	Objective functions		Reference
	Upper-Level	Lower-Level	
Smoothing PG	Smooth	Convex constraint	[Lin-Xu-Ye, 2014]
BOME	Smooth	PL	[Liu-Ye-Wright-Stone-Liu, 2022]
PBGD	Smooth	PL	[Shen-Chen, 2023]
F2SA	Smooth	Strongly Convex	[Kwon-Kwon-Wright-Nowak, 2023]
---	Slater's Constraint Qualification		[Dempe-Dutta, 2012]
CDB	Non-Smooth	Strongly Convex	[Hu-Xiao-Liu-Toh, 2023]
EBSA	Smooth	Non-convex	[Xu-Dai-Liu-Wang, 2024]

III. Approximate implicit differentiation (AID) based methods

Scenario of interest: $g(x, \cdot)$ is **strongly convex**

$$y^*(x) = \arg \min_{y \in \mathbb{R}^{d_y}} g(x, y)$$

By optimality condition & implicit function theorem

$$0 \equiv \nabla_y g(x, y^*(x)) \Rightarrow \nabla_{xy}^2 g(x, y^*(x)) + \nabla_x y^*(x)^\top \nabla_{yy}^2 g(x, y^*(x)) = 0$$

Hyper-gradient computation

$$\begin{aligned} \nabla \phi(x) &:= \nabla f(x, y^*(x)) \\ &= \nabla_x f(x, y^*(x)) + \nabla_x y^*(x)^\top \nabla_y f(x, y^*(x)) \\ &= \nabla_x f(x, y^*) - \nabla_{xy}^2 g(x, y^*) \left[\nabla_{yy}^2 g(x, y^*) \right]^{-1} \nabla_y f(x, y^*) \end{aligned}$$

Hyper-gradient

$$\nabla \phi(x) = \nabla_x f(x, y^*) - \nabla_{xy}^2 g(x, y^*) \left[\nabla_{yy}^2 g(x, y^*) \right]^{-1} \nabla_y f(x, y^*)$$

Main difficulties

- Solving the lower-level problem to obtain $y^*(x)$
- ★ Approximating the **Hessian inverse vector product**

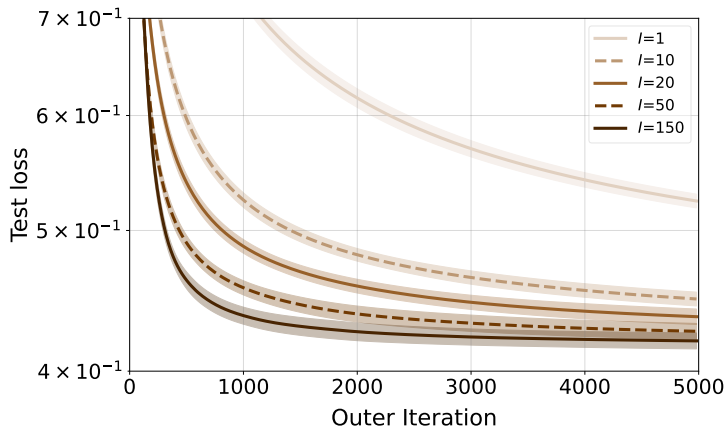
$$v^*(x) := \left[\nabla_{yy}^2 g(x, y^*(x)) \right]^{-1} \nabla_y f(x, y^*(x))$$

Vanilla update rule

$$\mathbf{1} \times : x^+ = x - \beta \left(\nabla_x f(x, y) - \nabla_{xy}^2 g(x, y) \left[\nabla_{yy}^2 g(x, y) \right]^{-1} \nabla_y f(x, y) \right)$$

$$\mathbf{N} \times : y^+ = y - \alpha \nabla_y g(x, y)$$

An observation



How to tackle v^* : approximation principle

- Neumann Series approximation when $\rho(A) < 1$

$$A^{-1} = \sum_{i=0}^{\infty} (I - A)^i$$

[Ghadimi and Wang, 2018; Chen et al., 2021; Ji, 2021]

- At each iteration, solve a linear system approximatively:

$$\mathbf{1} \times : \nabla_{yy}^2 g(x, y) \mathbf{v} \approx \nabla_y f(x, y)$$

$$\mathbf{1} \times : x^+ = x - \beta \left(\nabla_x f(x, y) - \nabla_{xy}^2 g(x, y) \mathbf{v} \right)$$

$$\mathbf{N} \times : y^+ = y - \alpha \nabla_y g(x, y)$$

where $\mathbf{v}$ can be obtained by

- GD [Arbel and Mairal, 2022]
- CG [Yang et al., 2023]
- LS [Xiao et al., 2023]

How to tackle v^* : amortization principle

- Two time-scale **single** loop algorithm [Hong et al., 2020]

$$\mathbf{1} \times : x^+ = x - \beta_k \left(\nabla_x f(x, y) - \nabla_{xy}^2 g(x, y) \left[\nabla_{yy}^2 g(x, y) \right]^{-1} \nabla_y f(x, y) \right)$$

$$\mathbf{1} \times : y^+ = y - \alpha_k \nabla_y g(x, y)$$

- Dynamically approximate (**auxiliary**) variables [Dagr  ou et al., 2022]

$$\mathbf{1} \times : v^+ = v - \gamma \left(\nabla_{yy}^2 g(x, y) v - \nabla_y f(x, y) \right)$$

$$\mathbf{1} \times : x^+ = x - \beta \left(\nabla_x f(x, y) - \nabla_{xy}^2 g(x, y) v^+ \right)$$

$$\mathbf{1} \times : y^+ = y - \alpha \nabla_y g(x, y)$$

How to tackle the Hessian inverse vector product v^* in BiO

- **Approximation** principle
- **Amortization** principle

How to adhere to these two principles

- Subspace methods for **efficient Approximation**: [Yuan, 2014; Carmon and Duchi, 2018]
- Subspace iteration for **reasonable Amortization**: [Yuan, 1995; Saad, 2011]

A Subspace Technique

Revisit hyper-gradient

Recall the hyper-gradient

$$\nabla \phi(x_k) = \nabla_x f(x_k, y_k^*) - \nabla_{xy}^2 g(x_k, y_k^*) \left[\nabla_{yy}^2 g(x_k, y_k^*) \right]^{-1} \nabla_y f(x_k, y_k^*)$$

and its estimator

$$\tilde{\nabla} \varphi(x_k, y_k, v_k) := \nabla_x f(x_k, y_k) - \nabla_{xy}^2 g(x_k, y_k) v_k$$

Geometric interpretation

$$\min_{v \in \mathcal{S}_k} m_k(v) := \frac{1}{2} v^\top \nabla_{yy}^2 g(x_k, y_k) v - \nabla_y f(x_k, y_k)^\top v$$

$$\mathcal{S}_k = \mathbb{R}^{d_y} \implies v_k = \nabla_{yy}^2 g(x_k, y_k)^{-1} \nabla_y f(x_k, y_k) := A_k^{-1} b_k$$

Low-dimensional subspace?

Krylov subspace [Krylov, 1931]:

$$\mathcal{K}_N(A, b) := \text{span} \left\{ b, Ab, A^2b, \dots, A^{N-1}b \right\}$$

- $\mathcal{K}_N(A, b)$ provides a good estimate for $A^{-1}b$ [Carmon and Duchi, 2018]
- An observation

$$\mathcal{K}_N(A, b) = \mathcal{K}_N(I - \eta A, b)$$

Neumann series approximation

- When $\rho(\eta A) < 1$, for some $N \in \mathbb{N}$

$$A^{-1}b = \eta \sum_{i=0}^{\infty} (I - \eta A)^i b \in \mathcal{K}_N(A, b)$$

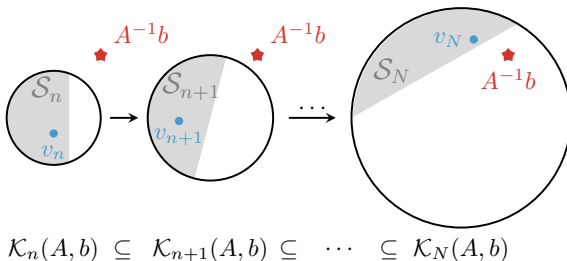
Constructing 2-D subspace in Krylov

Given $v_n \in \mathcal{K}_n(A, b)$, an initial approximation of $A^{-1}b$

$$v_n = \sum_{i=0}^{n-1} c_i (I - \eta A)^i b \approx A^{-1}b$$

Recursively, we can choose

$$v_{n+1} \in \mathcal{S}_{n+1} := \text{span} \{b, (I - \eta A)v_n\} \subseteq \mathcal{K}_{n+1}(A, b)$$



Extend the thoughts in BiO context

- $A_k \approx A_{k-1}, b_k \approx b_{k-1}$ for consecutive iterations
- $\mathcal{S}_k = \text{span} \{b_k, (I - \eta A_k)v_{k-1}\}$

SubBiO

$$\textcolor{red}{1} \times : \mathcal{S}_k := \text{span} \{b_k, (I - \eta A_k)v_{k-1}\}$$

$$\textcolor{red}{1} \times : v_k := \arg \min_{v \in \mathcal{S}_k} m_k(v) := \frac{1}{2} v^T A_k v - b_k v$$

$$\textcolor{blue}{1} \times : x_{k+1} = x_k - \beta \left(\nabla_x f(x_k, y_k) - \nabla_{xy}^2 g(x_k, y_k) v_k \right)$$

$$\textcolor{blue}{1} \times : y_{k+1} = y - \alpha \nabla_y g(x_k, y_k)$$

Two-dimensional subproblem

- Subproblem in SubBiO

$$\min_{z \in \mathbb{R}^2} \frac{1}{2} z^\top (S_k^\top A_k S_k) z - b_k^\top S_k z$$

☹ Projecting Hessian $S_k^\top A_k S_k$ costs $O(\mathbf{2}n^2)$

- $A_k v_{k-1}$: $O(n^2)$
- $\nabla_{xy}^2 g(x_k, y_k) v_k$: $O(n^2)$

SubBiO costs $O(\mathbf{4}n^2)$ per iteration!

A Dynamic Lanczos-aided Framework

- How to reduce $O(4n^2)$ while preserving the advantages of Krylov subspace?

- How to reduce $O(4n^2)$ while preserving the advantages of Krylov subspace?
- Main computation comes from the **projection of Hessian** A_k

- How to reduce $O(4n^2)$ while preserving the advantages of Krylov subspace?
- Main computation comes from the **projection of Hessian** A_k
- It's reasonable to consider the **Lanczos Process**

Lanczos process [Lanczos, 1950] computes the eigenvalues and corresponding eigenvectors of symmetric matrix.

- Construct Krylov subspace

$$\mathcal{K}_k(A, b) = \text{span} \{ b, Ab, \dots, A^{k-1}b \}$$

and an orthogonal basis Q_k

- Project A to the Krylov subspace, $T_k = Q_k^T A Q_k$
- Compute eigenvalues and eigenvectors of T_k

Implementation details

Goal

- Maintain an **orthogonal basis** $Q_j = [q_1, \dots, q_j]$ of $\mathcal{K}_j(A, b)$
- Keep the (approximate) projection matrix T_j **tridiagonal**

Iterates

$$\begin{aligned} u_j &= Aq_j - \beta_j q_{j-1} & \alpha_j &= q_j^\top u_j & \omega_j &= u_j - \alpha_j q_j \\ \beta_{j+1} &= \|\omega_j\| & q_{j+1} &= \omega_j / \beta_{j+1} \end{aligned}$$

Tridiagonalization

$$A Q_j = Q_j T_j + \beta_{j+1} q_{j+1} e_j^\top$$

Core principles

- Incrementally construct subspaces $\mathcal{S}_k$
- Dynamically solve quadratic subproblems:

$$v_k := \arg \min_{v \in \mathcal{S}_k} m_k(v) := \frac{1}{2} v^T A_k v - b_k^T v$$

Adapt standard Lanczos process in BiO

$$\begin{aligned} u_j &= A_j q_j - \beta_j q_{j-1} \\ \alpha_j &= q_j^\top u_j \\ \omega_j &= u_j - \alpha_j q_j \\ \beta_{j+1} &= \|\omega_j\| \\ q_{j+1} &= \omega_j / \beta_{j+1} \end{aligned} \quad T_j = \left(\begin{array}{ccc|c} & & & \mathbf{0} \\ & T_{j-1} & & \\ & & & \beta_j \\ \hline \mathbf{0} & & \beta_j & \alpha_j \end{array} \right).$$

Lanczos process is inherently unstable!

[Paige, 1980; Meurant and Strakoš, 2006]

Two "Res" strategies

Restart mechanism

- Restart subspaces each m steps
- Mitigate the accumulation of difference among $\{A_1, \dots, A_k\}$

Restart mechanism

- Restart subspaces each m steps
- Mitigate the accumulation of difference among $\{A_1, \dots, A_k\}$

Residual minimization

- Minimal residual subproblems

$$\min_{\Delta v \in \mathcal{S}_k} \|(b_k - A_k \bar{v}) - A_k \Delta v\|^2$$

- Correct current $\bar{v}$, $v_k = \bar{v} + \Delta v_k$
- Collect historical information

Two "Res" strategies

Restart mechanism

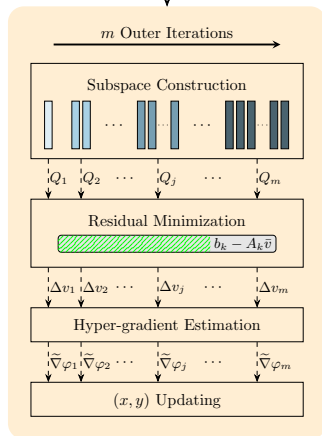
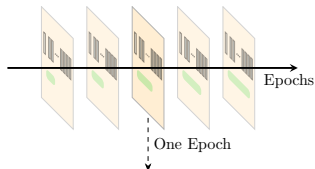
- Restart subspaces each m steps
- Mitigate the accumulation of difference among $\{A_1, \dots, A_k\}$

Residual minimization

- Minimal residual subproblems

$$\min_{\Delta v \in \mathcal{S}_k} \|(b_k - A_k \bar{v}) - A_k \Delta v\|^2$$

- Correct current $\bar{v}$, $v_k = \bar{v} + \Delta v_k$
- Collect historical information



- 😊 Low-dimensional subproblems
- 😊 No cost of Hessian projection
- LancBiO needs $O(2n^2)$ instead of $O(4n^2)$ of SubBiO

Existing work [Ghadimi and Wang, 2018; Hong et al., 2020; Ji, 2021; Dagr  ou et al., 2022]

$$\text{stocBiO, TTSA, BSA} : v_k = \eta \sum_{i=0}^N (I - \eta A_k)^i b_k$$

$$\text{SOBA} : v_k = v_{k-1} - \eta (A_k v_{k-1} - b_k)$$

Ours

$$\text{SubBiO} : v_k = \arg \min_{v \in \mathcal{S}_k^{\text{Sub}}} m_k(v) := \frac{1}{2} v^T A_k v - b_k v$$

$$\text{LancBiO} : v_k \approx \arg \min_{v \in \mathcal{S}_k^{\text{Lanc}}} m_k(v) := \frac{1}{2} v^T A_k v - b_k v$$

Theoretical Analysis

Divide iterates into epochs of m -step dynamic Lanczos process

$$\varepsilon_{st}^h := \left(1 + \frac{L_{gx}}{\mu_g}\right) \|x_{mh+s} - x_{mh+t}\| + \|y_{mh+s} - y_{mh+s}^*\|$$

$$\varepsilon_j^h := \max_{1 \leq s, t \leq j} \varepsilon_{st}^h$$

Proposition

The dynamic Lanczos process in LancBiO with normalized q_1 and α_j, β_j, q_j satisfies

$$A_j^* Q_j = Q_j T_j + \beta_{j+1} q_{j+1} e_j^\top + \delta Q_j, \quad \text{for } j = 1, 2, \dots, m,$$

with $\|\delta q_j\| \leq L_{gyy} \varepsilon_j$

Theorem

Within each epoch, set the step size $\theta \sim \mathcal{O}(1/m)$ for y and the step size for x as zero in the first m_0 steps, and the others as $\lambda \sim \mathcal{O}(1/m^4)$, then the iterates $\{x_k\}$ satisfy

$$\frac{m}{K(m - m_0)} \sum_{\substack{k=0, \\ (k \bmod m) > m_0}}^K \|\nabla \varphi(x_k)\|^2 = \mathcal{O}\left(\frac{m\lambda^{-1}}{K(m - m_0)}\right)$$

where m is the subspace dimension and $m_0 \sim \Omega(\log m)$

Numerical Experiments

Compared methods

- TTSA: first single-loop algorithm [Hong et al., 2020]
- stocBiO: double-loop, truncated Neumann Series [Ji, 2021]
- AmIGO: double-loop, GD or CG [Arbel et al., 2022]
- SOBA: single-loop, auxiliary variable v [Dagr  ou et al., 2022]
- F2SA: penalty-based method [Kwon et al., 2023]
- HJFBiO: Hessian/Jacobian-free method [Huang, 2024]

Running platform

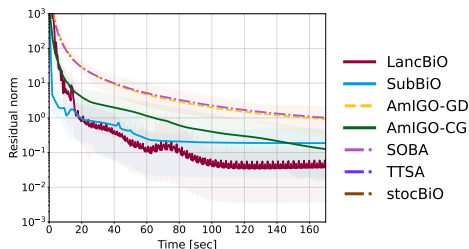
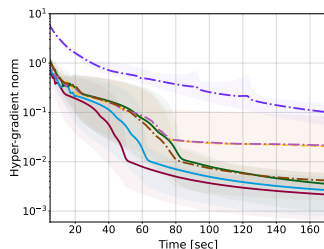
- Intel® Xeon® Gold 6330 CPUs & NVIDIA A800 GPU
- Python 3.8.0 + Pytorch 1.13.1

Synthetic problem

$$\min_{x \in \mathbb{R}^d} f(x, y) := c_1 \cos(x^\top D_1 y) + \frac{1}{2} \|D_2 x - y\|^2,$$

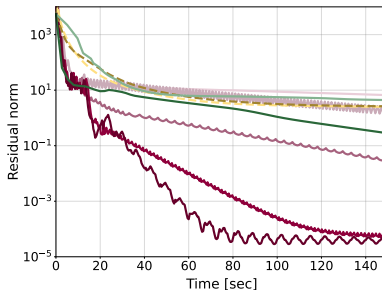
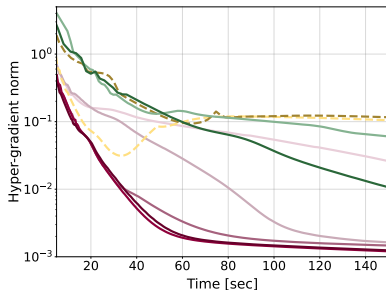
$$\text{s. t. } y^* = \arg \min_{y \in \mathbb{R}^d} g(x, y)$$

$$:= c_2 \sum_{i=1}^d \sin(x_i + y_i) + \log \left(\sum_{i=1}^d e^{x_i y_i} \right) + \frac{1}{2} y^\top (D_3 + G) y,$$



Influence of the subspace dimension m

LancBiO_m10 LancBiO_m50 LancBiO_m150 AmlGO-GD_I10 AmlGO-CG_I10
LancBiO_m20 LancBiO_m80 AmlGO-GD_I3 AmlGO-CG_I3

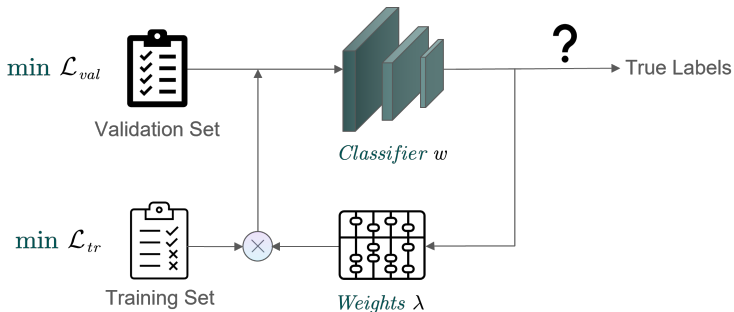


- 😊 Increases in m enhance the convergence of the residual norm
- 😊 When $m = 50$, the estimate of v^* is sufficiently accurate

Data hyper-cleaning

Goal: train a classifier in a **corrupted setting** where some labels of training data are replaced by random class numbers

- **Upper level:** data weights λ
- **Lower level:** classifier defined by deep nets w

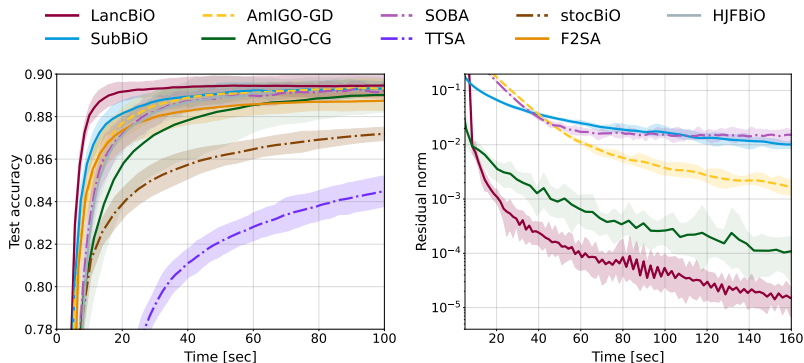


Goal: train a classifier in a **corrupted setting** where some labels of training data are replaced by random class numbers

BiO formulation:

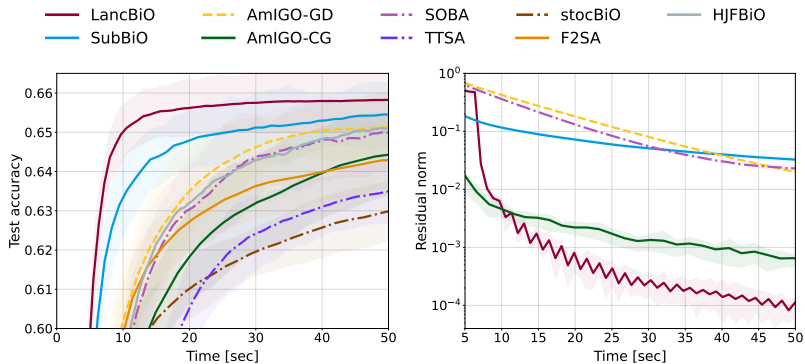
$$\begin{aligned} \min_{\lambda} \quad & \mathcal{L}_{val}(\lambda, w^*) = \frac{1}{|\mathcal{D}_{val}|} \sum_{(x_i, y_i) \in \mathcal{D}_{val}} L(w^* x_i, y_i) \\ \text{s.t.} \quad & w^* = \arg \min_w \mathcal{L}_{tr}(w, \lambda) \\ & := \frac{1}{|\mathcal{D}_{tr}|} \sum_{(x_i, y_i) \in \mathcal{D}_{tr}} \sigma(\lambda_i) L(w x_i, y_i) + C_r \|w\|^2 \end{aligned}$$

Test on MNIST (pollution rate=0.8)

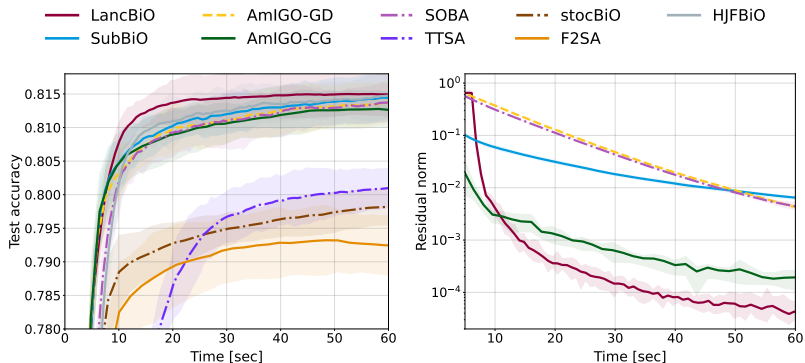


- 😊 LancBiO reaches the plateau fastest
- 😊 LancBiO achieves the lowest residual

Test on Kuzushiji-MNIST (pollution rate=0.6)



Test on Fashion-MNIST (pollution rate=0.5)



Take-home notes

- First subspace technique in BiO
- Improved numerical performance
- Analysis to tackle instability of constructing Krylov subspaces in BiO
 - Extension to stochastic settings

References

- Yan Yang, Bin Gao, Ya-xiang Yuan. *LancBiO: dynamic Lanczos-aided bilevel optimization via Krylov subspace*. ICLR (2025)
- Code is publicly available from <https://github.com/UCAS-YanYang/LancBiO>