CARPATHIAN J. MATH.
Volume **42** (2026), No. 2,
Pages 369-392

Online version at https://www.carpathian.cunbm.utcluj.ro/
ISSN 1584-2851 (Print edition); ISSN 1843-4401 (Electronic)
DOI: https://doi.org/10.37193/CJM.2026.02.10

A Novel Fixed-Point Based Two-Step Inertial Algorithm for Convex Bilevel Optimization in Deep Learning Data Classification

SUTHEP SUANTAI [1] AND KOBKOON JANNGAM [2,3,*]

ABSTRACT. In this paper, we present a new two-step inertial algorithm for finding a common fixed-point of a countable family of nonexpansive mappings. We establish a strong convergence theorem for the proposed method under mild conditions. We then apply this algorithm to solve the convex bilevel optimization problem, demonstrating its versatility in handling complex hierarchical optimization tasks. Additionally, we extend the application of the proposed method to the field of data classification using deep learning, specifically utilizing a multihidden-layer extreme learning machine (MELM). Our numerical tests indicate that the proposed method outperforms existing methods in the literature achieving higher classification accuracy. The empirical findings demonstrate our method's efficacy and efficiency and indicate its potential for deep machine learning and optimization applications.

1. INTRODUCTION

Bilevel optimization has received considerable attention recently due to its broad applicability in machine learning, notably in hyperparameter optimization [10, 31], meta-learning [10], and data classification tasks [15]. A general bilevel optimization problem comprises an upper-level (outer) optimization constrained by a lower-level (inner) optimization problem. Specifically, we consider the convex bilevel optimization problem:

$$\min_{x \in S_*} \varphi(x), \tag{1.1}$$

where the feasible set S_* is defined as the solution set of the inner-level optimization:

$$S_* = argmin_{x \in \mathbb{R}^n}\{f(x) + g(x)\}, \tag{1.2}$$

where $\varphi : \mathbb{R}^n \to \mathbb{R}$ is strongly convex and differentiable, $f : \mathbb{R}^n \to \mathbb{R}$ is convex with an L_f-Lipschitz continuous gradient, and $g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is proper, convex, and lower-semicontinuous.

Let Λ be the set of all solutions of (1.1). A well-known first-order condition for solutions $x^* \in \Lambda$ is given by [27]:

$$\langle \nabla\varphi(x^*), x - x^* \rangle \geq 0, \quad \forall x \in S_*.$$

A powerful class of methods for solving problem (1.2) is a class of fixed-point algorithms. These algorithms deal with finding x^* such that $x^* = Tx^*$ where $T = prox_{\lambda g}(I - \lambda\nabla f)$ is a forward–backward (proximal-gradient) operator, f and g are as in equation (1.2), and $\lambda > 0$ is a step size parameter. In general, the proximal operator is defined for a function g as:

Received: 28.04.2025. In revised form: 16.08.2025. Accepted: 18.11.2025
2020 *Mathematics Subject Classification.* 47H09, 65K10, 68T07, 90C25, 90C30.
Key words and phrases. *convex bilevel optimization; Data classification; Deep learning; Fixed-point; Optimization; Two-step inertial algorithm .*
Corresponding author: Kobkoon Janngam; kobkoon.j@cmu.ac.th

$$prox_g(v) = argmin_x \left(g(x) + \frac{1}{2}\|x - v\|^2 \right). \tag{1.3}$$

This operator is used to deal efficiently with functions g that are nonsmooth, a common characteristic of regularization terms in convex optimization. It is well known that if x is a fixed-point of the forward–backward operator,

$$x = prox_{\lambda g}(x - \lambda \nabla f(x)), \tag{1.4}$$

then x is a minimizer of $f + g$. This relationship between fixed-points and minimizers is a key insight in solving problem (1.2). It serves as the theoretical foundation for using fixed-point iterations to solve convex minimization problems [19, 6].

One fundamental method for finding solutions to Equation (1.4), and thus solutions to the inner-level convex optimization problem (1.2), is the iterative proximal forward–backward splitting algorithm (PFBS), defined as:

$$x_{n+1} = prox_{\alpha g}(x_n - \alpha \nabla f(x_n)), \tag{1.5}$$

where convergence is assured if $0 < \alpha < 2/L_f$. Iteration (1.5) can be viewed as a fixed-point iteration $x_{n+1} = T(x_n)$ of the nonexpansive forward–backward operator $T = prox_{\alpha g}(I - \alpha \nabla f)$ with $Fix(T) = S_*$, where $Fix(T) := \{x \in C : Tx = x\}$ denotes the set of fixed-points of T.

This method combines gradient descent for the smooth function f with the proximal step for the nonsmooth function g, making it particularly useful for large-scale optimization problems. It is noted that when L is a Lipschitz constant of ∇f, then $T = prox_{\lambda g}(I - \lambda \nabla f)$ is nonexpansive if $\lambda \in (0, 2/L)$. Moudafi [20] achieved strong convergence of such a proximal algorithm by introducing the viscosity approximation method given by:

$$x_{n+1} = \eta_n S(x_n) + (1 - \eta_n) T(x_n),$$

where T is nonexpansive and S is a contraction mapping satisfying suitable conditions. We shall incorporate a similar technique in our algorithm, as shall be seen in Section 3.

Our algorithm shall also incorporate inertial terms. Inertial methods boost convergence rates by incorporating previous iteration points into the calculations at each step. Numerous researchers have developed iterative fixed-point methods that utilize the properties of the proximal operator and that employ inertial terms alongside other acceleration methods. Important early examples of inertial algorithms include the heavy-ball method introduced by Polyak in 1964 [24], and a similar approach developed by Nesterov in 1983 [22]. In 2009, Beck and Teboulle [4] developed the FISTA algorithm, which extended the aforementioned methods to proximal operators.

Over the past several years, there has been the intense study of single-step methods: that is, algorithms that employ a single inertial parameter. Kaewkhao et al. [17] demonstrated strong convergence of a single-step inertial algorithm using nonexpansive mappings, along with practical applications. Thongsri and Suantai [36] developed accelerated fixed-point algorithms that exhibited high efficiency in regression and classification applications. Sae-jia and Suantai [28] successfully applied inertial forward-backward methods to image restoration tasks. And, Janngam and Suantai [14] developed an inertial framework for solving convex bilevel optimization problems.

Several algorithms have been developed to solve convex bilevel optimization problems. These approaches attempt to manage the hierarchical structure of bilevel optimization using algorithms that can be unconstrainedly applied to both levels. Sabach and Shtern [27] introduced the bilevel gradient sequential averaging method (BiG-SAM), later

improved by Shehu et al. [32] with inertial extrapolation, termed inertial BiG-SAM (iBiG-SAM):

$$\begin{cases} y_n = x_n + \theta_n(x_n - x_{n-1}), \\ p_n = prox_{cg}(y_n - c\nabla f(y_n)), \\ q_n = y_n - \lambda\nabla\varphi(y_n), \\ x_{n+1} = \gamma_n q_n + (1-\gamma_n)p_n. \end{cases}$$

In 2022, Duan and Zhang [7] advanced the iBiG-SAM algorithm by introducing an alternated inertial step, leading to the development of the alternated inertial bilevel gradient sequential averaging method (aiBiG-SAM). This enhanced method achieved accelerated convergence and improved computational efficiency. The aiBiG-SAM algorithm is defined as follows:

$$y_n = \begin{cases} x_n, & \text{if } n \text{ is even}, \\ x_n + \theta_n(x_n - x_{n-1}), & \text{if } n \text{ is odd}, \end{cases}$$

and

$$\begin{cases} p_n = \mathrm{prox}_{cg}(I - c\nabla f)(y_n), \\ q_n = y_n - \lambda\nabla\varphi(y_n), \\ x_{n+1} = \gamma_n q_n + (1-\gamma_n)p_n, \quad n \geq 1. \end{cases}$$

Duan and Zhang established that the aiBiG-SAM algorithm converges strongly to a solution of the bilevel optimization problem. Moreover, they extended their approach by developing a more general method that incorporates an alternating inertial acceleration scheme, thereby enhancing its convergence properties and applicability.

The iBiG-SAM and aiBiG-SAM algorithms both employ a single inertial parameter, θ. Such single-step algorithms can be unstable, slow to converge, and are most effective when the problems are high-dimensional. Two-step inertial algorithms incorporate two inertial parameters, resulting in improved convergence and stability. The basic idea is to update iterates using a combination of current and previous step information, formalized as:

$$x_{n+1} = x_n + \delta(x_n - x_{n-1}) + \theta(x_{n-1} - x_{n-2}),$$

where $\delta > 0$ and $\theta < 0$ are inertial parameters that incorporate momentum from previous iterates. This approach enhances motion modeling, increases redundancy, and improves stability, making it suitable for a broader range of applications. The incorporation of two inertial parameters also provides flexibility and adaptability in algorithm design, allowing fine-tuning for specific optimization problems. Izuchukwu et al. (2023) [13] introduced a two-step inertial forward–reflected–anchored–backward splitting algorithm for monotone inclusion problems, demonstrating strong convergence with fewer computational steps compared to one-step methods. Iyiola and Shehu (2022) [12] proposed a two-step inertial proximal point algorithm for convex minimization problems, establishing a non-asymptotic $O(1/n)$ convergence rate and showcasing the improved stability of two-step inertial algorithms over their one-step counterparts. Thong et al. [35] further developed a double-inertial-step algorithm for split common fixed-point problems with strong convergence results and practical signal processing applications.

Motivated by these insights, this paper introduces a novel two-step inertial viscosity algorithm for convex bilevel optimization. Our algorithm combines two-step inertial and

viscosity approximation methods, achieving strong convergence. We employ our algorithm to solve convex bilevel optimization problems applicable to data classification tasks using deep learning.

The paper is structured as follows. Section 2 provides preliminary concepts. In Section 3, we present our algorithm and the relevant convergence analysis, and we discuss the algorithm's application to solving convex bilevel optimization problems. Section 4 delves into the underlying principles of the machine learning models used for data classification and demonstrates the application of our algorithm by reformulating these models as convex bilevel optimization problems. Section 5 presents numerical experiments that demonstrate the performance of the proposed method. Finally, Section 6 concludes the paper with a summary of findings and potential future research directions.

2. Preliminaries

In Section 2, we establish notation and provide several definitions and lemmas that will be needed for the main results presented in Section 3. Throughout, H denotes a real Hilbert space with inner product $\langle \cdot, \cdot \rangle$ and induced norm $|| \cdot ||$. C denotes a nonempty closed convex subset of H. The symbols $\mathbb{N}, \mathbb{R}, \mathbb{R}_+, \mathbb{R}_{>0}$ denote, respectively, the natural numbers, the real numbers, the non-negative real numbers and the positive real numbers.

Definition 2.1. *Let $T : C \to C$ be a mapping on a subset C of a real Hilbert space H. We say that T is L-Lipschitz for some constant $L \geq 0$ if*

$$\|Tu - Tv\| \leq L\|u - v\|, \quad \text{for all } u, v \in C.$$

*(i) If $L \in (0, 1)$, then T is called a contraction mapping. In this case, T is also referred to as an L-contraction or a k-**contraction**, where $k := L \in (0, 1)$.*

*(ii) If $L = 1$, then T is said to be **nonexpansive**.*

Definition 2.2 ([21]). *Let $T : C \to C$ be a nonexpansive mapping and $T_n : C \to C$ a family of nonexpansive mappings with $\emptyset \neq Fix(T) \subset \Gamma := \bigcap_{n=1}^{\infty} Fix(T_n)$. The family $\{T_n\}$ is said to satisfy the NST-condition (I) with respect to the mapping T if, when the condition $\lim_{k\to\infty} \|x_k - T_n x_k\| = 0$ holds for all n and for some bounded sequence $\{x_k\} \subset C$, then the condition $\lim_{k\to\infty} \|x_k - Tx_k\| = 0$ also holds.*

Definition 2.3. *Let f, g be as in Section 1, and let $\lambda > 0$ be a real number. The proximity operator, $prox_{\lambda g}$, with respect to g and λ, is defined as*

$$prox_{\lambda g} x = argmin_{y \in H} \left\{ g(y) + \tfrac{1}{2\lambda}\|y - x\|^2 \right\};$$

see [19, 3, 6]. *The forward–backward operator T of f and g with parameter λ is then defined by*

$$T := prox_{\lambda g}(I - \lambda \nabla f).$$

Remark 2.1. *It is well known that if $\lambda \in (0, 2/L)$, where L is a Lipschitz constant of ∇f, then the forward–backward operator T is nonexpansive.*

Definition 2.4. [2, 1] *Let H be a real Hilbert space and $C \subset H$ a non-empty closed convex set. Suppose $\{T_n\}_{n=1}^{\infty}$ is a sequence of nonexpansive self-mappings $T_n : C \to C$ whose common fixed-point set*

$$\Gamma := \bigcap_{n=1}^{\infty} Fix(T_n) \neq \varnothing.$$

The family $\{T_n\}$ is said to satisfy condition (Z) if, for every bounded sequence $\{u_n\} \subset C$ with

$$\lim_{n\to\infty} \|u_n - T_n u_n\| = 0,$$

every weak cluster point of $\{u_n\}$ belongs to F.

Employing the demi-closedness of the operator $I - T$ for a nonexpansive mapping $T : C \to C$, we derive the following remark.

Remark 2.2. *Let $T : C \to C$ be nonexpansive. If a family of nonexpansive mappings $\{T_n\}$ satisfies the NST-condition (I) with respect to T, then $\{T_n\}$ also satisfies Condition (Z).*

The Lemma stated below plays a significant role in proving our main result.

Lemma 2.1 ([34])**.** *Let $\eta, \xi \in H$ and $\kappa \in [0, 1]$. Then the following identities hold in H:*

(i) $\|\eta + \xi\|^2 \leq \|\eta\|^2 + 2\langle \xi, \eta + \xi \rangle$;
(ii) $\|\eta \pm \xi\|^2 = \|\eta\|^2 \pm 2\langle \eta, \xi \rangle + \|\xi\|^2$;
(iii) $\|\kappa\eta + (1-\kappa)\xi\|^2 = \kappa\|\eta\|^2 + (1-\kappa)\|\xi\|^2 - \kappa(1-\kappa)\|\eta - \xi\|^2$.

Lemma 2.2 ([29])**.** *Let $\{d_n\} \subset \mathbb{R}_+$, $\{b_n\} \subset \mathbb{R}$ and $\{h_n\} \subset (0, 1)$ such that $\sum_{n=1}^{\infty} h_n = \infty$. Suppose that*

$$d_{n+1} \leq (1 - h_n)d_n + h_n b_n$$

for all $n \in \mathbb{N}$. If $\limsup_{i\to\infty} b_{n_i} \leq 0$ and for any subsequence $\{d_{n_i}\}$ of $\{d_n\}$ satisfying

$$\liminf_{i\to\infty} (d_{n_i+1} - d_{n_i}) \geq 0,$$

then $\lim_{n\to\infty} d_n = 0$.

Now, the last required result of the literature is presented, which was provided by Bussaban et al.; see [5].

Lemma 2.3. *Let H be a real Hilbert space, and consider the forward–backward operator T associated with the functions f and g for a parameter λ, where $g : H \to \mathbb{R} \cup \{\infty\}$ is a proper, lower semi-continuous convex function, and $f : H \to \mathbb{R}$ is a convex and differentiable function whose gradient ∇f is L-Lipschitz continuous for some constant $L > 0$. Let $\{T_n\}$ be a sequence of forward–backward operators corresponding to the same functions f and g, but with parameters λ_n, where $\lambda_n \to \lambda$ and $\lambda_n, \lambda \in (0, 2/L)$. Then, the sequence $\{T_n\}$ satisfies the NST-condition (I) with respect to T.*

3. Main Results

We define a new modified S-algorithm (Algorithm 1) with two inertial terms. We prove that the sequence, $\{x_n\}$, generated by Algorithm 1 converges strongly to a common fixed-point of a family of nonexpansive mappings in a real Hilbert space.

Algorithm 1 Two-Step Inertial Modified S-Algorithm (TIMSA)

Initialization: Let $\{\alpha_n\}$, $\{\beta_n\}$, $\{\gamma_n\} \subset [0,1]$, $\{\tau_n\} \subset \mathbb{R}_+$ and let $\{\mu_n\}$, $\{\rho_n\} \subset \mathbb{R}_{>0}$ be bounded sequences. Take $x_{-1}, x_0, x_1 \in H$ arbitrarily.
For $n \in \mathbb{N}$, do the following steps:

Step 1. Compute the inertial step:

$$\theta_n = \begin{cases} \min\left\{\mu_n, \dfrac{\alpha_n\tau_n}{\|x_n - x_{n-1}\|}\right\}, & \text{if } x_n \neq x_{n-1}, \\ \mu_n, & \text{otherwise,} \end{cases}$$

$$\delta_n = \begin{cases} \max\left\{-\rho_n, \dfrac{-\alpha_n\tau_n}{\|x_{n-1} - x_{n-2}\|}\right\}, & \text{if } x_{n-1} \neq x_{n-2}, \\ -\rho_n, & \text{otherwise.} \end{cases}$$

Compute:

$$y_n = x_n + \theta_n(x_n - x_{n-1}) + \delta_n(x_{n-1} - x_{n-2}).$$

Step 2. Compute w_n, z_n and x_{n+1}:

$$\begin{aligned} w_n &= (1-\alpha_n)y_n + \alpha_n F(y_n), \\ z_n &= (1-\beta_n)w_n + \beta_n T_n w_n, \\ x_{n+1} &= (1-\gamma_n)T_n w_n + \gamma_n T_n z_n. \end{aligned}$$

Theorem 3.1 establishes the strong convergence of Algorithm 1 to a common fixed-point of the mappings $\{T_n\}$. P_Γ denotes the projection onto the set Γ of common fixed-points.

Theorem 3.1. *Let $\{T_n\}$ be a family of nonexpansive self-mappings on a real Hilbert space H satisfying Condition (Z), and assume that $\Gamma := \bigcap_{n=1}^{\infty} Fix(T_n) \neq \emptyset$. Let $F : H \to H$ be a k-contraction mapping with $k \in (0,1)$. Let $\{x_n\}$ be a sequence generated by Algorithm 1. Suppose that the control sequences $\{\tau_n\}$, $\{\alpha_n\}$, $\{\beta_n\}$, and $\{\gamma_n\}$ satisfy the following conditions:*

(C1) $\lim_{n\to\infty} \tau_n = 0$;
(C2) $\lim_{n\to\infty} \alpha_n = \lim_{n\to\infty} \beta_n = 0$ *and* $\sum_{n=1}^{\infty} \alpha_n = \infty$;
(C3) $0 < a_1 \leq \gamma_n \leq a_2 < 1$ *for some* $a_1, a_2 \in \mathbb{R}$.

Then the sequence $\{x_n\}$ strongly converges to an element $x^* \in \Gamma$, where $x^* = P_\Gamma F(x^*)$.

Proof. Let $x^* = P_\Gamma F(x^*)$. By the definition of y_n, we obtain

$$\begin{aligned} \|y_n - x^*\| &= \|x_n + \theta_n(x_n - x_{n-1}) + \delta_n(x_{n-1} - x_{n-2}) - x^*\| \\ &\leq \|x_n - x^*\| + \theta_n\|x_n - x_{n-1}\| + |\delta_n|\|x_{n-1} - x_{n-2}\|. \end{aligned} \tag{3.6}$$

By the definition of w_n, we obtain

$$\begin{aligned} \|w_n - x^*\| &= \|(1-\alpha_n)y_n + \alpha_n F(x_n) - x^*\| \\ &\leq (1-\alpha_n)\|y_n - x^*\| + \alpha_n\|F(y_n) - F(x^*)\| + \alpha_n\|F(x^*) - x^*\| \\ &\leq \Big(1 - \alpha_n(1-k)\Big)\|y_n - x^*\| + \alpha_n\|F(x^*) - x^*\|. \end{aligned} \tag{3.7}$$

By the definition of z_n, we obtain

$$\begin{aligned} \|z_n - x^*\| &= \|(1-\beta_n)w_n + \beta_n T_n w_n - x^*\| \\ &\leq (1-\beta_n)\|w_n - x^*\| + \beta_n\|T_n w_n - x^*\| \\ &\leq \|w_n - x^*\|. \end{aligned} \tag{3.8}$$

By the definition of x_{n+1} and (3.8), we obtain

$$\begin{aligned}\|x_{n+1}-x^*\| &= \|(1-\gamma_n)T_n w_n+\gamma_n T_n z_n - x^*\|\\ &\le (1-\gamma_n)\|T_n w_n - x^*\| + \gamma_n\|T_n z_n - x^*\|\\ &\le (1-\gamma_n)\|w_n - x^*\| + \gamma_n\|z_n - x^*\|\\ &\le \|w_n - x^*\|. \end{aligned} \tag{3.9}$$

Using (3.6), (3.7) and (3.9), we have

$$\begin{aligned}\|x_{n+1}-x^*\| &\le \Big(1-\alpha_n(1-k)\Big)\|y_n-x^*\| + \alpha_n\|F(x^*)-x^*\|\\ &\le \Big(1-\alpha_n(1-k)\Big)\Big(\|x_n-x^*\| + \theta_n\|x_n-x_{n-1}\| + |\delta_n|\|x_{n-1}-x_{n-2}\|\Big)\\ &\quad + \alpha_n\|F(x^*)-x^*\|\\ &\le \Big(1-\alpha_n(1-k)\Big)\|x_n-x^*\| + \alpha_n\left(\frac{\theta_n}{\alpha_n}\|x_n-x_{n-1}\| + \frac{|\delta_n|}{\alpha_n}\|x_{n-1}-x_{n-2}\|\right.\\ &\quad \left. + \|F(x^*)-x^*\|\right).\end{aligned}$$

By the definition of θ_n, δ_n and (C1), we obtain

$$\frac{\theta_n}{\alpha_n}\|x_n-x_{n-1}\| \to 0 \text{ as } n\to\infty \quad \text{and} \quad \frac{|\delta_n|}{\alpha_n}\|x_{n-1}-x_{n-2}\| \to 0 \text{ as } n\to\infty. \tag{3.10}$$

From (3.10), we know that there exists $M>0$ such that

$$\frac{\theta_n}{\alpha_n}\|x_n-x_{n-1}\| \le M \qquad \text{and} \qquad \frac{|\delta_n|}{\alpha_n}\|x_{n-1}-x_{n-2}\| \le M$$

for all $n\ge 1$. Then,

$$\begin{aligned}\|x_{n+1}-x^*\| &\le \Big(1-\alpha_n(1-k)\Big)\|x_n-x^*\| + \alpha_n(1-k)\left(\frac{M+M+\|F(x^*)-x^*\|}{1-k}\right)\\ &\le \max\left\{\|x_n-x^*\|,\ \frac{2M+\|F(x^*)-x^*\|}{1-k}\right\}.\end{aligned}$$

Thus, by mathematical induction, we deduce that

$$\|x_n-x^*\| \le \max\left\{\|x_1-x^*\|,\ \frac{2M+\|F(x^*)-x^*\|}{1-k}\right\}.$$

for all $n\ge 1$. Therefore, the sequence$\{x_n\}$ is bounded, also the sequences $\{w_n\}$, $\{T_n w_n\}$, $\{y_n\}$ are bounded. Now, by Lemma 2.1 (i) and (ii), we obtain

$$\begin{aligned}\|w_n-x^*\|^2 &= \|(1-\alpha_n)(y_n-x^*) + \alpha_n(F(y_n)-F(x^*)) + \alpha_n(F(x^*)-x^*)\|^2\\ &\le \|(1-\alpha_n)(y_n-x^*) + \alpha_n(F(y_n)-F(x^*))\|^2 + 2\alpha_n\langle F(x^*)-x^*, w_n-x^*\rangle\\ &\le (1-\alpha_n)\|y_n-x^*\|^2 + \alpha_n\|F(y_n)-F(x^*)\|^2 + 2\alpha_n\langle F(x^*)-x^*, w_n-x^*\rangle\\ &\le \Big(1-\alpha_n(1-k)\Big)\|y_n-x^*\|^2 + 2\alpha_n\langle F(x^*)-x^*, w_n-x^*\rangle\end{aligned} \tag{3.11}$$

and

$$\begin{aligned}\|y_n - x^*\|^2 &= \|x_n - x^*\|^2 + 2\langle x_n - x^*, \theta_n(x_n - x_{n-1}) + \delta_n(x_{n-1} - x_{n-2})\rangle \\ &\quad + \|\theta_n(x_n - x_{n-1}) + \delta_n(x_{n-1} - x_{n-2})\|^2 \\ &\leq \|x_n - x^*\|^2 + 2\theta_n\|x_n - x^*\|\|x_n - x_{n-1}\| + 2|\delta_n|\|x_n - x^*\|\|x_{n-1} - x_{n-2}\| \\ &\quad + \theta_n^2\|x_n - x_{n-1}\|^2 + 2\theta_n|\delta_n|\|x_n - x_{n-1}\|\|x_{n-1} - x_{n-2}\| \\ &\quad + \delta_n^2\|x_{n-1} - x_{n-2}\|^2. \qquad (3.12)\end{aligned}$$

Also, from Lemma 2.1 (iii), (3.8), (3.11) and (3.12), we obtain

$$\begin{aligned}\|x_{n+1} - x^*\|^2 &\leq (1-\gamma_n)\|T_n w_n - x^*\|^2 + \gamma_n\|T_n z_n - x^*\|^2 - \gamma_n(1-\gamma_n)\|T_n w_n - T_n z_n\|^2 \\ &\leq (1-\gamma_n)\|w_n - x^*\|^2 + \gamma_n\|z_n - x^*\|^2 - \gamma_n(1-\gamma_n)\|T_n w_n - T_n z_n\|^2 \\ &\leq \|w_n - x^*\|^2 - \gamma_n(1-\gamma_n)\|T_n w_n - T_n z_n\|^2 \\ &\leq \Big(1 - \alpha_n(1-k)\Big)\|y_n - x^*\|^2 + 2\alpha_n\langle F(x^*) - x^*, w_n - x^*\rangle \\ &\quad - \gamma_n(1-\gamma_n)\|T_n w_n - T_n z_n\|^2 \\ &\leq \Big(1 - \alpha_n(1-k)\Big)\|x_n - x^*\|^2 + 2\theta_n\|x_n - x^*\|\|x_n - x_{n-1}\| \\ &\quad + 2|\delta_n|\|x_n - x^*\|\|x_{n-1} - x_{n-2}\| + \theta_n^2\|x_n - x_{n-1}\| \\ &\quad + 2\theta_n|\delta_n|\|x_n - x_{n-1}\|\|x_{n-1} - x_{n-2}\| + \delta_n^2\|x_{n-1} - x_{n-2}\| \\ &\quad + 2\alpha_n\langle F(x^*) - x^*, w_n - x^*\rangle - \gamma_n(1-\gamma_n)\|T_n w_n - T_n z_n\|^2 \\ &= \Big(1 - \alpha_n(1-k)\Big)\|x_n - x^*\|^2 - \gamma_n(1-\gamma_n)\|T_n w_n - T_n z_n\|^2 + \alpha_n(1-k)b_n, \qquad (3.13)\end{aligned}$$

where

$$\begin{aligned}b_n = \frac{1}{1-k}\Bigg(&\frac{2\theta_n}{\alpha_n}\|x_n - x^*\|\|x_n - x_{n-1}\| + \frac{2|\delta_n|}{\alpha_n}\|x_n - x^*\|\|x_{n-1} - x_{n-2}\| + \frac{\theta_n^2}{\alpha_n}\|x_n - x_{n-1}\|^2 \\ &+ \frac{\delta_n^2}{\alpha_n}\|x_{n-1} - x_{n-2}\|^2 + \frac{2\theta_n|\delta_n|}{\alpha_n}\|x_n - x_{n-1}\|\|x_{n-1} - x_{n-2}\| + 2\langle F(x^*) - x^*, w_n - x^*\rangle\Bigg).\end{aligned}$$

It follows that

$$\gamma_n(1-\gamma_n)\|T_n w_n - T_n z_n\|^2 \leq \|x_n - x^*\|^2 - \|x_{n+1} - x^*\|^2 + \alpha_n(1-k)S', \qquad (3.14)$$

where $S' = \sup\{b_n : n \in \mathbb{N}\}$. We proceed to show that $x_n \to x^*$. To apply Lemma 2.2, define $d_n := \|x_n - x^*\|^2$ and $h_n := \alpha_n(1-k)$. Then, by (3.13), we obtain

$$d_{n+1} \leq (1-h_n)a_n + h_n b_n.$$

Suppose that $\{d_{n_i}\}$ is a subsequence of $\{d_n\}$ such that $\liminf_{i\to\infty}(d_{n_i+1} - d_{n_i}) \geq 0$. Using (3.14) and (C2), we obtain

$$\begin{aligned}\limsup_{i\to\infty}\gamma_{n_i}(1-\gamma_{n_i})\|T_{n_i}w_{n_i} - T_{n_i}z_{n_i}\|^2 &\leq \limsup_{i\to\infty}(d_{n_i} - d_{n_i+1} + \alpha_n(1-k)S') \\ &= -\liminf_{i\to\infty}(d_{n_i+1} - d_{n_i}) \\ &\leq 0. \qquad (3.15)\end{aligned}$$

From (3.15) and (C3), we obtain

$$\lim_{i\to\infty}\|T_{n_i}w_{n_i} - T_{n_i}z_{n_i}\| = 0. \qquad (3.16)$$

It follows that

$$\|T_{n_i}w_{n_i} - x_{n_i+1}\| \leq \gamma_{n_i}\|T_{n_i}w_{n_i} - T_{n_i}z_{n_i}\| \to 0 \text{ as } i \to \infty. \tag{3.17}$$

Using (C2), we obtain

$$\|z_n - w_n\| = \beta_n\|w_n - T_nw_n\| \to 0 \text{ as } n \to \infty. \tag{3.18}$$

By the definition of θ_n and δ_n, we obtain

$$\|y_n - x_n\| = \theta_n\|x_n - x_{n-1}\| + |\delta_n|\|x_{n-1} - x_{n-2}\| \to 0 \text{ as } n \to \infty. \tag{3.19}$$

From (3.19) and (C2), we obtain

$$\begin{aligned}\|w_n - x_n\| &= \|y_n - \alpha_n y_n + \alpha_n F(y_n) - x_n\| \\ &\leq \|y_n - x_n\| + \alpha_n\|F(y_n) - y_n\| \to 0 \text{ as } n \to \infty.\end{aligned} \tag{3.20}$$

From (3.17) and (3.18), we obtain

$$\begin{aligned}\|T_{n_i}w_{n_i} - w_{n_i}\| &\leq \|T_{n_i}w_{n_i} - T_{n_i}z_{n_i}\| + \|T_{n_i}z_{n_i} - x_{n_i}\| + \|x_{n_i} - w_{n_i}\| \\ &\to 0 \text{ as } i \to \infty.\end{aligned} \tag{3.21}$$

Next, we show that $\limsup_{i\to\infty} b_{n_i} \leq 0$. Obviously, it suffices to show that

$$\limsup_{i\to\infty}\langle F(x^*) - x^*, w_{n_i} - x^*\rangle \leq 0. \tag{3.22}$$

Since $\{w_{n_i}\}$ is bounded, there exists a subsequence $\{w_{n_{i_j}}\}$ of $\{w_{n_i}\}$ and $y \in H$ such that $w_{n_{i_j}} \rightharpoonup y$ as $j \to \infty$ and

$$\limsup_{i\to\infty}\langle F(x^*) - x^*, w_{n_i} - x^*\rangle = \lim_{j\to\infty}\langle F(x^*) - x^*, w_{n_{i_j}} - x^*\rangle = \langle F(x^*) - x^*, y - x^*\rangle.$$

Using (3.21) and $\{T_n\}$ satisfies the condition (Z), we obtain $y \in \Gamma$. From $x^* = P_\Gamma F(x^*)$, we obtain

$$\langle F(x^*) - x^*, z - x^*\rangle \leq 0$$

Since the inequality holds for every $z \in \Gamma$, choosing $z = y$ gives

$$\langle F(x^*) - x^*, y - x^*\rangle \leq 0.$$

Hence, the variational inequality (3.22) holds. By Lemma 2.2 the sequence $\{x_n\}$ converges to x^* as required. □

We now turn to a special case of Algorithm 1 in which the nonexpansive mappings $\{T_n\}$ are forward–backward operators. The resulting algorithm (Algorithm 2) will allow us to solve the convex bilevel optimization problems (1.1) and (1.2) that were posed in Section 1. The mappings f, g, and φ referenced in Section 1 are assumed to satisfy the following conditions:

(A1) $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable and convex function such that ∇f is Lipschitz continuous with constant $L_f > 0$ and $g : \mathbb{R}^n \to (-\infty, \infty]$ is lower semicontinuous, proper and convex.

(A2) $\varphi : \mathbb{R}^n \to \mathbb{R}$ is strongly convex with parameter σ such that $\nabla\varphi$ is L_φ-Lipschitz continuous. As can be seen below, φ plays the role of the contraction map F of Algorithm 1. An additional parameter s is needed that will satisfy the following condition: $s \in \left(0, \frac{2}{L_\varphi+\sigma}\right)$. The parameter s is addressed further in Proposition 3.1

Algorithm 2 Two-Step Inertial Bilevel Gradient Modified S-Algorithm (TIBiG-MSA)

Initialization: Let $\{\alpha_n\}, \{\beta_n\}, \{\gamma_n\} \subset [0,1], \{\tau_n\} \subset \mathbb{R}_+$, and let $\{\mu_n\}, \{\rho_n\} \subset \mathbb{R}_{>0}$ be bounded sequences. Take $x_{-1}, x_0, x_1 \in \mathbb{R}^n$ arbitrarily. Let $\{c_n\} \subset (0, \frac{2}{L_f})$ with $c_n \to c$ as $n \to \infty$, where $c \in (0, \frac{2}{L_f})$. For $n \in \mathbb{N}$, do the following steps:

Step 1. Compute the inertial step:

$$\theta_n = \begin{cases} \min\left\{\mu_n, \frac{\alpha_n \tau_n}{\|x_n - x_{n-1}\|}\right\} & \text{if } x_n \neq x_{n-1}, \\ \mu_n & \text{otherwise,} \end{cases}$$

and

$$\delta_n = \begin{cases} \max\left\{-\rho_n, \frac{-\alpha_n \tau_n}{\|x_{n-1} - x_{n-2}\|}\right\} & \text{if } x_{n-1} \neq x_{n-2}, \\ -\rho_n & \text{otherwise.} \end{cases}$$

$$y_n = x_n + \theta_n(x_n - x_{n-1}) + \delta_n(x_{n-1} - x_{n-2}).$$

Step 2. Compute w_n, z_n and x_{n+1}:

$$\begin{aligned} w_n &= (1-\alpha_n)y_n + \alpha_n(I - s\nabla\varphi)(y_n), \\ z_n &= (1-\beta_n)w_n + \beta_n prox_{c_n g}(I - c_n\nabla f)(w_n), \\ x_{n+1} &= (1-\gamma_n)prox_{c_n g}(I - c_n\nabla f)(w_n) + \beta_n prox_{c_n g}(I - c_n\nabla f)(z_n). \end{aligned}$$

The following proposition is established by Sabach and Shtern [27] and plays a key role in our next theorem.

Proposition 3.1. *Suppose that $\varphi : \mathbb{R}^n \to \mathbb{R}$ is strongly convex with parameter $\sigma > 0$ and that its gradient $\nabla\varphi$ is L_φ-Lipschitz continuous. The mapping $F_s = I - s\nabla\varphi$ is a contraction such that*

$$\|(a - s\nabla\varphi(a)) - (b - s\nabla\varphi(b))\| \leq \sqrt{1 - \frac{2s\sigma L_\varphi}{\sigma + L_\varphi}}\|a - b\|$$

for all $s \in \left(0, \frac{2}{\sigma + L_\varphi}\right)$ and all $a, b \in \mathbb{R}^n$.

Theorem 3.2 (Strong convergence of TIBiG-MSA)**.** *Let the sequence $\{x_n\}$ be generated by Algorithm 2 with $c_n \in (0, 2/L_f)$, and assume that all control parameters satisfy conditions* (C1)–(C3) *in Theorem 3.1. Suppose further that the functions f, g, and φ satisfy assumptions* (A1)–(A2) *stated before Algorithm 2. Then $\{x_n\}$ converges strongly to a point $x^* \in \Lambda$, where Λ denotes the solution set of* (1.1)*, and $x^* = P_{S_*}(I - s\nabla\varphi)(x^*)$.*

Proof. Let $F := I - s\nabla\varphi$ and define $T_n := \operatorname{prox}_{c_n g}(I - c_n\nabla f)$, where $c_n \in (0, \frac{2}{L_f})$. According to Proposition 3.1, F is a contraction. Since T_n is nonexpansive, Lemma 2.3 implies that the sequence $\{T_n\}$ fulfills the NST-condition (I) with respect to its limit operator $T := \operatorname{prox}_{cg}(I - c\nabla f)$. Consequently, all hypotheses of Theorem 3.1 are met, so Algorithm 2 yields

$$x_n \longrightarrow x^* \in \Gamma := \bigcap_{n=1}^{\infty} Fix(T_n) = S_*,$$

with $x^* = P_\Gamma F(x^*)$. Then, for all $x \in S_*$, we obtain

$$0 \;\geq\; \langle F(x^*) - x^*,\, x - x^* \rangle = \langle -s\nabla\varphi(x^*),\, x - x^* \rangle .$$

Dividing by $-s > 0$ gives the variational inequality $\langle \nabla\varphi(x^*), x - x^* \rangle \geq 0$ for all $x \in S_*$, which is exactly the first-order optimality condition for problem (1.1). Hence $x^* \in \Lambda$ and $x_n \to x^* \in \Lambda$ as required. □

Remark 3.3. *We note that Theorem 3.1 proves strong convergence of our two-step inertial viscosity algorithm for a countable families of nonexpansive mappings, while Theorem 3.2 is a consequence result of Theorem 1 by putting* $T_n = prox_{c_n g}(I - c_n \nabla f)$ *and the main result of this theorem aim to solve convex bilevel optimization problem (1.1) and (1.2). The novelty of our algorithm is employing two-step inertial technique and viscosity approximation method for accelerating convergence behavior of the algorithm while BiG-SAM, iBiG-SAM, and aiBiG-SAM contain only a single inertial step and having no viscosity approximation technique. We will see in the next section that our algorithm has a better performance than the others.*

4. APPLICATIONS

In this section, we apply the proposed algorithm (TIBiG-MSA) to convex bilevel problems arising in the training of ELM, TELM, and MELM models. The convergence results in Section 3 are established for the general bilevel framework (1.1) and (1.2). In this way, the algorithms of Section 3 are not only proved to converge in theory but are also tested in practice through bilevel reformulations of standard machine learning models, thereby linking the theoretical analysis with practical applications.

The integration of artificial intelligence (AI) technology with machine learning (ML) and deep learning (DL) in data-driven image analysis and classification operations has had significant impact across a number of fields. These technologies allow researchers to solve problems that were previously deemed very challenging or even insurmountable. Deep learning approaches have enhanced the accuracy of image identification procedures as well as object search capabilities and this innovation now benefits medical imaging analysis and autonomous vehicle systems [18, 30].

One of the major building blocks for the development of AI is a type of ML algorithm called a neural network, which is a model inspired by the structure of the human brain, consisting of interconnected layers of nodes (or "neurons") that process data. In the very early days of neural networks, it was found that multiple-layered neural networks, called deep neural networks, could be used to model complex nonlinear relationships in data. However, traditional training algorithms, such as backpropagation, were computationally intensive and prone to overfitting and local minima [26].

To address these challenges, Huang et al. (2006) [11] created the extreme learning machine (ELM), which serves as an algorithm to speed up the training process of single hidden-layer feedforward neural networks (SLFNs). The process of ELM operates differently from traditional approaches by randomly selecting the initial weight values and bias levels linking input and hidden layers before executing a basic least squares optimization for output weight calculation. The training process becomes faster while the generalization ability increases when using this method. Subsequent research of the ELM model suggested more complex learning machines with an increased number of hidden layers. A two-hidden-layer extreme learning machine (TELM) was developed in 2016 by Qu et al. [25] which delivered enhanced modeling results for complicated, high-dimensional datasets.

Xiao et al. (2017) further generalized the ELM concept into a multihidden-layer extreme learning machine (MELM), incorporating more than two hidden layers. MELM maintains the foundational random weight initialization characteristic of ELM for the initial layer, while weights for subsequent layers are refined iteratively. This deeper architecture enhances the model's capability to learn sophisticated representations from data, making it particularly well-suited for challenging classification and regression tasks [39]. In the following paragraphs, we will describe the ELM, TELM, and MELM models in detail.

Consider a training set defined by $\{(\mathbf{X}, \mathbf{T}) : \mathbf{X} \in \mathbb{R}^{N\times n}, \mathbf{T} \in \mathbb{R}^{N\times m}\}$, where N represents the number of training samples, n denotes the input dimension (features), and m corresponds to the output dimension (number of classes).

4.1. **Extreme Learning Machine (ELM).** In the context of ELM (see Figure 1 for the ELM structure), the training set consists of N samples. The training data matrix $\mathbf{X}$, the target matrix $\mathbf{T}$, the input weight matrix $\mathbf{W}$, and the bias matrix $\mathbf{B}$ are defined as follows:

$$\mathbf{X} = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{Nn} \end{bmatrix} \in \mathbb{R}^{N\times n}, \qquad \mathbf{T} = \begin{bmatrix} t_{11} & \cdots & t_{1m} \\ \vdots & \ddots & \vdots \\ t_{N1} & \cdots & t_{Nm} \end{bmatrix} \in \mathbb{R}^{N\times m},$$

$$\mathbf{W} = \begin{bmatrix} w_{11} & \cdots & w_{1L} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nL} \end{bmatrix} \in \mathbb{R}^{n\times L}, \qquad \mathbf{B} = \begin{bmatrix} b_{11} & \cdots & b_{1L} \\ \vdots & \ddots & \vdots \\ b_{N1} & \cdots & b_{NL} \end{bmatrix} \in \mathbb{R}^{N\times L}.$$

First, the input data matrix $\mathbf{X}$ passes through the model, where it is multiplied by the weight matrix $\mathbf{W}$ that connects the input layer to the hidden layer. After multiplying by $\mathbf{W}$, the bias matrix $\mathbf{B}$ is added, where L is the number of hidden nodes. The core principle of ELM is that with a sufficiently large number of hidden neurons, the random initialization of the weight matrix $\mathbf{W}$ and bias vector $\mathbf{B}$ has minimal impact on the model's accuracy. As such, in ELM, $\mathbf{W}$ and $\mathbf{B}$ are typically initialized randomly without significant consequences on performance [11].

The output of this operation is passed through an activation function $g(\cdot)$ at the hidden layer. The purpose of the activation function is to introduce nonlinearity into the model, enabling it to learn complex patterns in the data. Common activation functions include the sigmoid, ReLU, and hyperbolic tangent functions. The choice of activation function depends on the specific application and the nature of the data.

The hidden layer output matrix $\mathbf{H} \in \mathbb{R}^{N\times L}$ is computed as:

$$\mathbf{H} = g(\mathbf{X}\mathbf{W} + \mathbf{B}) = \begin{bmatrix} g\left(\sum_{j=1}^{n} x_{1j}w_{j1} + b_{11}\right) & \cdots & g\left(\sum_{j=1}^{n} x_{1j}w_{jL} + b_{1L}\right) \\ g\left(\sum_{j=1}^{n} x_{2j}w_{j1} + b_{21}\right) & \cdots & g\left(\sum_{j=1}^{n} x_{2j}w_{jL} + b_{2L}\right) \\ \vdots & \ddots & \vdots \\ g\left(\sum_{j=1}^{n} x_{Nj}w_{j1} + b_{N1}\right) & \cdots & g\left(\sum_{j=1}^{n} x_{Nj}w_{jL} + b_{NL}\right) \end{bmatrix},$$

The ELM model uses the equation $\mathbf{H}\beta = \mathbf{T}$, where the weight matrix between the hidden and output layers (output weights) $\beta \in \mathbb{R}^{L\times m}$ is defined by

$$\beta = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1m} \\ \vdots & \ddots & \vdots \\ \beta_{L1} & \cdots & \beta_{Lm} \end{bmatrix}$$

To determine the output weights β, the ELM model minimizes the least-squares error between the predicted output $\mathbf{Y}$ and the target output $\mathbf{T}$. This is achieved using the Moore-Penrose pseudoinverse:

$$\beta = \mathbf{H}^{\dagger}\mathbf{T},$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose pseudoinverse of $\mathbf{H}$. The pseudoinverse $\mathbf{H}^{\dagger}$ is computed as:

$$\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T,$$

assuming $\mathbf{H}^T\mathbf{H}$ is invertible. The output of the network is then given by:

$$\mathbf{Y} = \mathbf{H}\beta,$$

where $\mathbf{Y} \in \mathbb{R}^{N\times m}$ is the predicted output matrix.

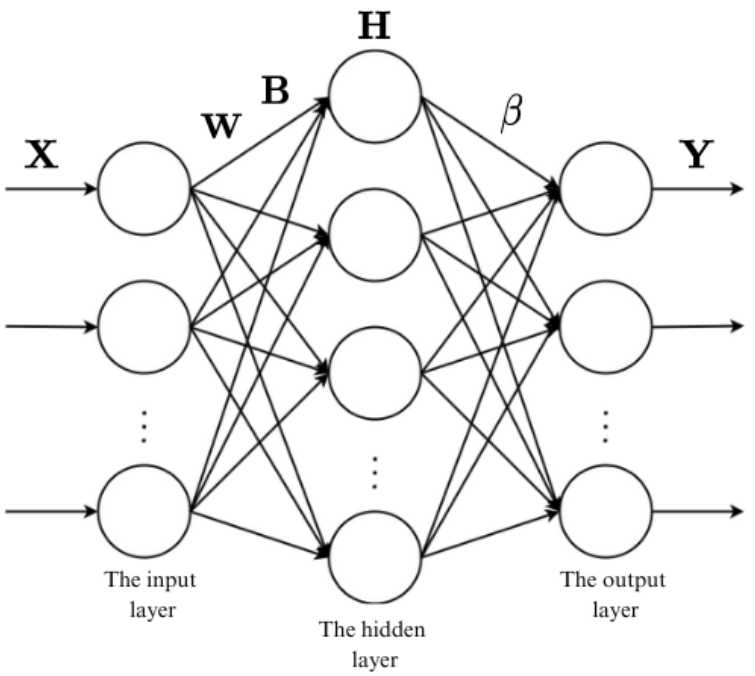


FIGURE 1. The structure of the ELM [16]

4.2. **Two-Hidden-Layer Extreme Learning Machine (TELM).** The TELM algorithm tries to calculate and update the weights and biases between the first and second hidden layers, as well as between the second hidden layer and the output layer (see Figure 2 for the TELM structure). Initially, the computation is simplified by merging the two hidden layers into one equivalent hidden layer. Output weights are calculated for the combined hidden layer, and then the combined layer is separated, and the model is refined further by updating the weights and biases via additional steps.

The TELM algorithm starts by treating the two hidden layers as one hidden layer to simplify the initialization. The output of this combined hidden layer, denoted by $\mathbf{H} \in \mathbb{R}^{N\times L}$, is expressed as:

$$\mathbf{H} = g(\mathbf{X}\mathbf{W} + \mathbf{B}),$$

where $\mathbf{X} \in \mathbb{R}^{N\times n}$ is the input matrix, $\mathbf{W} \in \mathbb{R}^{n\times L}$ is the weight matrix of the first hidden layer, $\mathbf{B} \in \mathbb{R}^{N\times L}$ is the bias matrix of the first hidden layer (broadcasted to match the dimensions), L is the number of hidden nodes, and $g(\cdot)$ is the activation function applied element-wise. The weight matrix $\mathbf{W}$ and bias matrix $\mathbf{B}$ are initialized randomly.

Once $\mathbf{H}$ is obtained, the output weight matrix β between the second hidden layer and the output layer is determined using:

$$\beta = \mathbf{H}^{\dagger}\mathbf{T}, \tag{4.23}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose pseudoinverse of $\mathbf{H}$, and $\mathbf{T} \in \mathbb{R}^{N\times m}$ is the target matrix.

After the initial output weight matrix β is determined, the TELM algorithm separates the two hidden layers, which were previously merged, so that the network now contains two distinct hidden layers. To refine the model, the expected output, $\mathbf{H}_1$, of the second hidden layer is computed in a way that minimizes error, with the previously calculated output weight matrix beta acting as a constraint. This is done using β and the target matrix $\mathbf{T}$:

$$\mathbf{H}_1 = \mathbf{T}\beta^{\dagger}, \tag{4.24}$$

where $\beta^{\dagger}$ is the Moore-Penrose pseudoinverse of the matrix β.

Next, the algorithm updates the weights between the first and second hidden layers in a way that minimizes error, with $\mathbf{H}_1$ and $\mathbf{H}$ acting as constraints. Observe that, according to the typical algorithm, the expected output of the second hidden layer would be calculated using

$$\mathbf{H}_1 = g(\mathbf{H}\mathbf{W}_1 + \mathbf{B}_1),$$

where $\mathbf{W}_1 \in \mathbb{R}^{L\times L}$ is the weight matrix between the first and second hidden layers, and $\mathbf{B}_1 \in \mathbb{R}^{N\times L}$ is the bias matrix for the second hidden layer. Using the expected output $\mathbf{H}_1$ and the inverse function of the activation function g(·), the updated weight matrix $\mathbf{W}_{HE}$ is calculated as:

$$\mathbf{W}_{HE} = \mathbf{H}_E^{\dagger} g^{-1}(\mathbf{H}_1), \tag{4.25}$$

where $\mathbf{W}_{HE} = [\mathbf{B}_1 \quad \mathbf{W}_1]^T \in \mathbb{R}^{(L+1)\times L}$, $\mathbf{H}_E^{\dagger}$ is the Moore-Penrose pseudoinverse of $\mathbf{H}_E = [1 \quad \mathbf{H}] \in \mathbb{R}^{N\times(L+1)}$, and $g^{-1}(\cdot)$ is the inverse of the activation function $g(\cdot)$.

With the updated weight matrix $\mathbf{W}_{HE}$, the actual output of the second hidden layer $\mathbf{H}_2 \in \mathbb{R}^{N\times L}$ is updated as follows:

$$\mathbf{H}_2 = g(\mathbf{H}_E\mathbf{W}_{HE}),$$

where $\mathbf{H}_2$ represents the final output of the second hidden layer after weight and bias adjustments.

Finally, the output weight matrix $\beta_{new} \in \mathbb{R}^{L\times m}$ between the second hidden layer and the output layer is updated as follows:

$$\beta_{new} = \mathbf{H}_2^{\dagger}\mathbf{T}, \tag{4.26}$$

where $\mathbf{H}_2^{\dagger}$ is the Moore-Penrose pseudoinverse of $\mathbf{H}_2$. The final output (predicted output matrix) $\mathbf{Y} \in \mathbb{R}^{N\times m}$ of the TELM network is then computed by:

$$Y = \mathbf{H}_2\beta_{new}.$$

Briefly summarized, the TELM first simplifies the network by merging the hidden layers, calculates the output weight matrix as in the one-hidden layer case, then separates the layers, iteratively refining the weight and bias by a series of updates. This process guarantees that the network can replicate complex relationships in the data by using multiple hidden layers and fine-tuning its parameters.

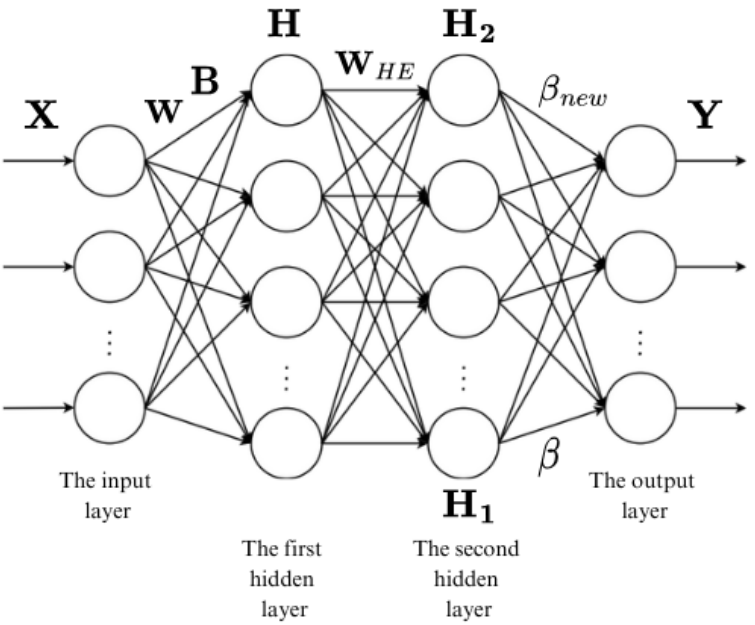


FIGURE 2. The structure of the TELM [16]

4.3. **Multihidden-Layer Extreme Learning Machine (MELM).** The multihidden-layer extreme learning machine (MELM) architecture extends ELM and TELM by adding additional hidden layers, improving the learning capabilities of the model. The incorporation of the additional layers enables the network to leverage the data in finer detail, making the model especially effective for complicated tasks. An examination of a model with three hidden layers, as shown in Figure 3, serves to illustrate how MELM works. The strategy is to reduce the three-layer network by replacing the second and third hidden layers with a single combined equivalent hidden layer, thus reducing it to a TELM framework. In this way, the TELM framework is extended for use on more complex network structures.

A 3-layer MELM begins by combining the second and third hidden layers into one. The TELM methodology is then used to derive the output weight matrix, β_{new}, between the combined second hidden layer and the output layer. After obtaining the initial output weight matrix, β_{new}, the MELM algorithm then separates the merged hidden layers, resulting in three distinct hidden layers. To further refine the model, the expected output of the third hidden layer, $\mathbf{H}_3 \in \mathbb{R}^{N\times L}$, is computed using the updated weight matrix β_{new} as a constraint:

$$\mathbf{H}_3 = \mathbf{T}\beta_{new}^{\dagger},$$

where $\beta_{new}^{\dagger} \in \mathbb{R}^{L\times m}$ is the Moore-Penrose pseudoinverse of the weight matrix β_{new}.

Observe that, according to the typical algorithm, the expected output of the third hidden layer, $\mathbf{H}_3$, would be calculated using

$$\mathbf{H}_3 = g(\mathbf{H}_2\mathbf{W}_2 + \mathbf{B}_2),$$

where $\mathbf{W}_2 \in \mathbb{R}^{L\times L}$ is the weight matrix between the second and third hidden layers, $\mathbf{H}_2$ is the output from the second hidden layer, and $\mathbf{B}_2 \in \mathbb{R}^{N\times L}$ is the bias term for the third hidden layer. The algorithm updates the weights between the second and third hidden

layers using the inverse function of the activation function g(·) and the expected output $\mathbf{H}_3$. The weight matrix $\mathbf{W}_{HE1}$ is calculated as:

$$\mathbf{W}_{HE1} = \mathbf{H}_{E1}^{\dagger} g^{-1}(\mathbf{H}_3),$$

where $\mathbf{W}_{HE1} = [\mathbf{B}_2 \quad \mathbf{W}_2]^T \in \mathbb{R}^{(L+1)\times L}$, $\mathbf{H}_{E1}^{\dagger}$ is the Moore-Penrose pseudoinverse of $\mathbf{H}_{E1} = [1 \quad \mathbf{H}_2] \in \mathbb{R}^{N\times(L+1)}$, and $g^{-1}(\cdot)$ is the inverse of the activation function $g(\cdot)$.

With the updated weight matrix $\mathbf{W}_{HE1}$, the actual output of the third hidden layer, $\mathbf{H}_4 \in \mathbb{R}^{N\times L}$, is calculated as follows:

$$\mathbf{H}_4 = g(\mathbf{H}_{E1}\mathbf{W}_{HE1}),$$

where $\mathbf{H}_4$ represents the final output of the third hidden layer before reaching the output layer.

Finally, the output weight matrix $\beta_{final} \in \mathbb{R}^{L\times m}$ between the second hidden layer and the output layer is updated as follows:

$$\beta_{final} = \mathbf{H}_4^{\dagger}\mathbf{T}.$$

The final output (predicted output matrix) $\mathbf{Y} \in \mathbb{R}^{N\times m}$ of the three-hidden-layer ELM network is given by:

$$\mathbf{Y} = \mathbf{H}_4\beta_{final}.$$

This process can be generalized to networks with more than three hidden layers in a natural inductive way. For a network with n hidden layers, the network structure is simplified by merging the last two layers, applying the algorithm for $n-1$ layers to obtain an output weight matrix, then separating the merged layers, updating the weights between layers $n-1$ and n, and completing the calculation for the last layer, as shown above in the discussions of TELM and MELM. This approach guarantees that the network can handle even more complicated data patterns with little additional computational cost. Specifically, the computational complexity of this process is $\mathcal{O}(n)$, as each additional layer contributes a fixed computational cost, ensuring scalability for deeper networks.

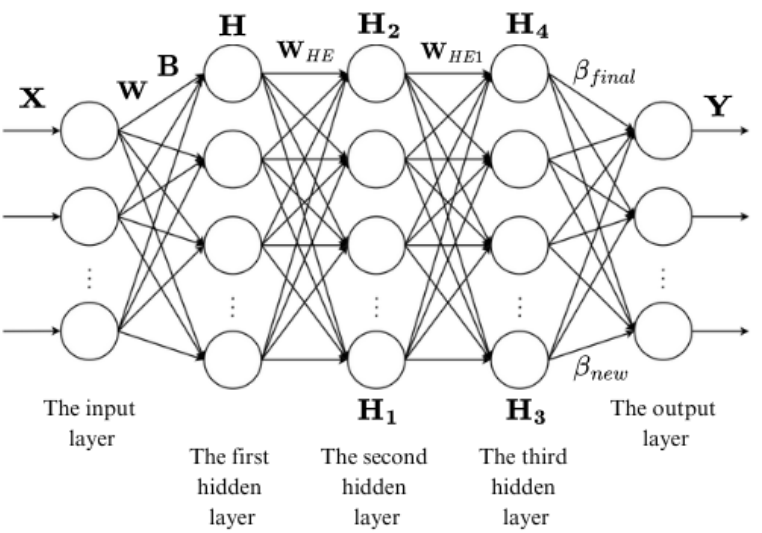


FIGURE 3. The structure of the three-hidden-layer ELM [16]

4.4. **Reformulation as a convex bilevel optimization problem and application of TIBiG-MSA.** The straightforward calculation of Moore–Penrose pseudoinverses occurs in equations (4.23)–(4.25) and (4.26) when such pseudoinverses actually exist. The direct application of these methods proves difficult whenever a pseudoinverse either does not exist or exhibits numerical instability. We solve this problem by implementing regularization techniques through the least absolute shrinkage and selection operator (LASSO) [37], which adds a stabilization penalty term. The transformation method changes the original equations into the following convex minimization problem.

Consequently, the original equations (4.23)–(4.25) and (4.26) can be reformulated into the following convex optimization problems:

$$\min_{\beta} \|\mathbf{H}\beta - \mathbf{T}\|_2^2 + \lambda\|\beta\|_1, \tag{4.27}$$

$$\min_{\mathbf{H}_1} \|\mathbf{H}_1\beta - \mathbf{T}\|_2^2 + \lambda\|\mathbf{H}_1\|_1, \tag{4.28}$$

$$\min_{\mathbf{W}_{HE}} \|\mathbf{H}_E\mathbf{W}_{HE} - g^{-1}(\mathbf{H}_1)\|_2^2 + \lambda\|\mathbf{W}_{HE}\|_1, \tag{4.29}$$

$$\min_{\beta_{new}} \|\mathbf{H}_2\beta_{new} - \mathbf{T}\|_2^2 + \lambda\|\beta_{new}\|_1, \tag{4.30}$$

where λ is a regularization parameter that balances data fitting accuracy with model simplicity, specifically penalizing the magnitude of parameters (through the L1 norm). This ensures that even under ill-conditioned or non-invertible scenarios.

Each optimization problem described by (4.27)–(4.30) can be viewed as minimizing the sum of two convex functions:

$$\min_{x\in\mathbb{R}^n} \left(f(x) + g(x)\right).$$

Let $S_*^{(i)}$ be the set of all solutions to the inner-level problems:

$$S_*^{(i)} = \left\{ K_i : \begin{array}{ll} K_i = \arg\min_{\beta} \|\mathbf{H}\beta - \mathbf{T}\|_2^2 + \lambda\|\beta\|_1, & \text{if } i = 1, \\ K_i = \arg\min_{\mathbf{H}_1} \|\mathbf{H}_1\beta - \mathbf{T}\|_2^2 + \lambda\|\mathbf{H}_1\|_1, & \text{if } i = 2, \\ K_i = \arg\min_{\mathbf{W}_{HE}} \|\mathbf{H}_E\mathbf{W}_{HE} - g^{-1}(\mathbf{H}_1)\|_2^2 + \lambda\|\mathbf{W}_{HE}\|_1, & \text{if } i = 3, \\ K_i = \arg\min_{\beta_{new}} \|\mathbf{H}_2\beta_{new} - \mathbf{T}\|_2^2 + \lambda\|\beta_{new}\|_1, & \text{if } i = 4, \end{array} \right\}.$$

We desire the solution $K_i \in S_*^{(i)}$ that minimizes the following expression:

$$\min_{K_i\in S_*^{(i)}} \frac{1}{2}\|K_i\|^2. \tag{4.31}$$

Therefore, our main obtained result (Theorem 3.2) can be applied for solving problem (4.31) which is a problem of data classification using the concept of MELM. The numerical experiment of our algorithm (TIBiG-MSA) will be presented and compare with the other algorithms in the next section.

5. Numerical Experiments

In this section, we present a series of numerical experiments designed to demonstrate the effectiveness of TIBiG-MSA in solving convex bilevel optimization problems commonly encountered in classification tasks. Our main objective is to compare TIBiG-MSA's performance with the established iBiG-SAM and aiBiG-SAM methods. We detail the experimental setup, datasets, algorithm parameters, and results to ensure clarity and reproducibility.

The specific inner-level problems (1.2) considered in our experiments appear in Table 1. Each row of the table shows the functions $f(\cdot)$ and $g(\cdot)$ that make up the MELM algorithm (see Section 4.4). We use $\lambda = 10^{-5}$ to balance model complexity against data fitting. By applying TIBiG-MSA to these problems, we highlight its ability to address both the accuracy and complexity demands of real-world classification scenarios.

TABLE 1. Inner-level problem setting for all algorithms.

Problems	$f(\cdot)$	$g(\cdot)$
(4.27)	$f(\boldsymbol{\beta}) = \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_2^2$	$g(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|_1$
(4.28)	$f(\mathbf{H}_1) = \|\mathbf{H}_1\boldsymbol{\beta} - \mathbf{T}\|_2^2$	$g(\mathbf{H}_1) = \lambda\|\mathbf{H}_1\|_1$
(4.29)	$f(\mathbf{W}_{HE}) = \left\|\mathbf{H}_E\mathbf{W}_{HE} - g^{-1}(\mathbf{H}_1)\right\|_2^2$	$g(\mathbf{W}_{HE}) = \lambda\|\mathbf{W}_{HE}\|_1$
(4.30)	$f(\boldsymbol{\beta}_{\text{new}}) = \|\mathbf{H}_2\boldsymbol{\beta}_{\text{new}} - \mathbf{T}\|_2^2$	$g(\boldsymbol{\beta}_{\text{new}}) = \lambda\|\boldsymbol{\beta}_{\text{new}}\|_1$

We test the performance of our algorithm by applying it to several datasets that vary in their dimensions (i.e., number of features), number of samples, and class distributions. Applying the algorithm to data sets with diverse characteristics allows for a comprehensive study of the algorithm's capability to handle different kinds and amounts of data. Each dataset is divided into training and testing sets. Table 2 provides an overview.

TABLE 2. Details of each dataset.

Dataset	**Training samples**	**Testing samples**	**Features**	**Classes**
Iris	105	45	4	3
Ionosphere	245	106	34	2
Hypertension	4453	1909	9	2
Weather Type	9240	3960	10	4

The datasets were chosen for their relevance to common classification tasks. Each dataset is briefly described below.

- **Iris** [9]: A well-known dataset in the ML community, this dataset is used for basic classification algorithm testing. It has 4 features and 3 classes.
- **Ionosphere** [33]: This is a binary classification using 34 features on a radar signal classification dataset.
- **Hypertension:** A healthcare related dataset with 9 features and binary outcomes. The data was collected by Sripat Medical Center, Faculty of Medicine, Chiang Mai University, Thailand. The data is real-world relevant and authentic.

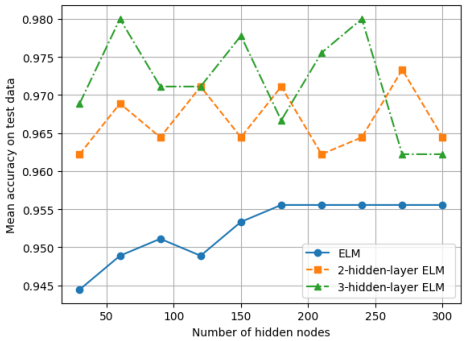
- **Weather Type** [23]: A weather-related dataset with 10 features and 4 classes. It can be used to test an algorithm's performance on multi-class classification problems containing more complex data patterns. Because this dataset contains outliers, it can be used to test a machine learning model's accuracy and resiliency, and provides a good platform to test different types of machine learning models.

In the first experiment, we assess and compare the accuracy of the ELM, TELM and MELM models across different datasets, defining accuracy as follows:

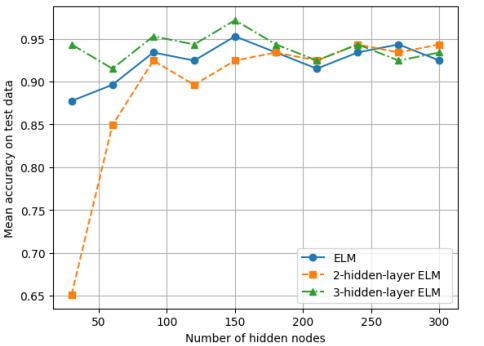
$$\text{Classification accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of samples}} \times 100.$$

To examine how the number of hidden nodes influences accuracy, the models are trained and tested for each dataset across a range of hidden nodes, varying from 50 to 300. Our proposed fixed-point algorithm (TIBiG-MSA) is applied to each experimental configuration, running 500 iterations across 10 executions. The random assignment of the initial weights and biases produces variability across executions in the classification accuracy. The mean accuracy across the ten executions is taken as a measure of the model's performance.

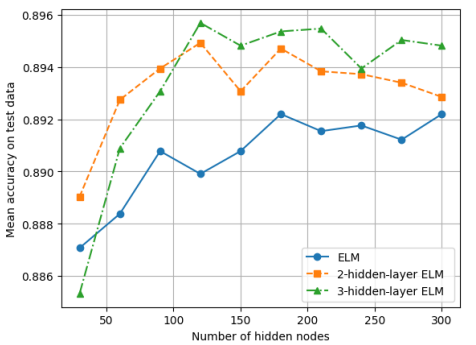
The results of the experiment are summarized in Figure 4. They demonstrate that in most of the tested datasets, MELM performs better than both the traditional ELM and the TELM models. The most transparently observed advantage is in the Weather Type dataset (Figure 4d). Here, the MELM's better performance reflects that model's increased ability to capture complicated data trends. High variability and the presence of outliers in the weather data pose problems for models with fewer hidden layers. The three hidden layers of MELM enable the model to flow through the data and accept more features. This allows modeling of more complicated relationships, increasing the algorithm's accuracy.



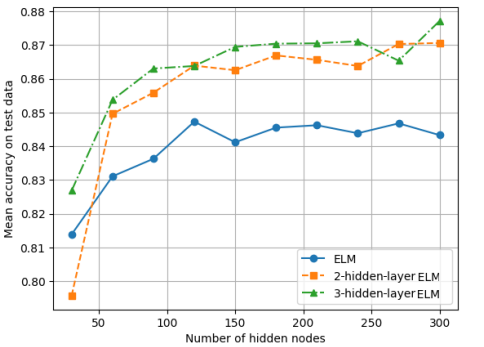
(**a**) Iris Dataset



(**b**) Ionosphere Dataset



(**c**) Hypertension Dataset



(**d**) Weather Type Dataset

FIGURE 4. Mean accuracy on test data with different numbers of hidden nodes for the ELM, 2-hidden-layer ELM, and 3-hidden-layer ELM using (a) Iris, (b) Ionosphere, (c) Hypertension, and (d) Weather Type datasets.

In the second experiment, the MELM model is implemented using the TIBiG-MSA, iBiG-SAM, and aiBiG-SAM algorithms, and the results are compared in terms of accuracy and convergence behavior. Mean classification accuracy is computed over 10 executions for varying iterations to assess convergence. Figure 5 illustrates the mean accuracy of the models as a function of the number of iterations, highlighting the superior convergence performance of TIBiG-MSA. The datasets used, Iris ($L = 50$), Ionosphere ($L = 90$), Hypertension ($L = 100$), and Weather Type ($L = 100$), are configured with the optimal number of hidden nodes (L), based on the results of the first experiment (Figure 5). In particular, the number of hidden nodes for each dataset was selected to balance accuracy, computational efficiency, and the risk of overfitting, as larger numbers of nodes provide diminishing returns in accuracy while increasing complexity.

It is crucial to carefully select the algorithm parameters, as they directly influence convergence behavior and the quality of the solution. Table 3 summarizes the parameters used for TIBiG-MSA, iBiG-SAM, and iBiG-SAM. Careful selection of the parameters ensures that each algorithm performs optimally under the same conditions, allowing for fair comparison. The parameters for TIBiG-MSA are chosen to ensure a balance between convergence speed and solution accuracy. The choices for iBiG-SAM and iBiG-SAM are based on recommendations from the literature.

TABLE 3. Algorithm parameters and control settings

Methods	**Setting**
TIBiG-MSA	$\beta_n = \alpha_n = \dfrac{1}{n+1},\ \gamma_n = \dfrac{0.9n}{n+1},\ c_n = \dfrac{1}{L_f},\ \tau_n = \dfrac{10^{14}}{n^2},\ \mu_n = \dfrac{n}{n+1},\ \rho_n = \dfrac{1}{n+1}$
iBiG-SAM	$\lambda = 0.01,\ c = \dfrac{1}{L_f},\ \alpha = 3,\ \gamma_n = \dfrac{0.2}{1-\frac{2+cL_f}{4}},\ \beta_n = \gamma_n n^{-0.01},$ $\theta_n = \begin{cases} \min\left\{\dfrac{n}{n+\alpha-1}, \dfrac{\beta_n}{\|x_n - x_{n-1}\|}\right\}, & x_n \neq x_{n-1}, \\ \dfrac{n}{n+\alpha-1}, & \text{otherwise.} \end{cases}$
aiBiG-SAM	$\lambda = 0.01,\ c = \dfrac{1}{L_f},\ \alpha = 3,\ \gamma_n = \dfrac{0.2}{1-\frac{2+cL_f}{4}},\ \beta_n = \gamma_n n^{-0.01},$ $\theta_n = \begin{cases} \min\left\{\dfrac{n}{n+\alpha-1}, \dfrac{\beta_n}{\|x_n - x_{n-1}\|}\right\}, & x_n \neq x_{n-1}, \\ \dfrac{n}{n+\alpha-1}, & \text{otherwise.} \end{cases}$

The numerical experiments clearly show the superior convergence speed and accuracy of TIBiG-MSA compared with the existing methods iBiG-SAM and aiBiG-SAM. A diverse set of datasets and carefully tuned parameters were used to ensure a fair evaluation. As shown in Figure 5, the figure provides a direct comparison among the three algorithms, demonstrating that our method consistently achieves faster convergence and higher classification accuracy

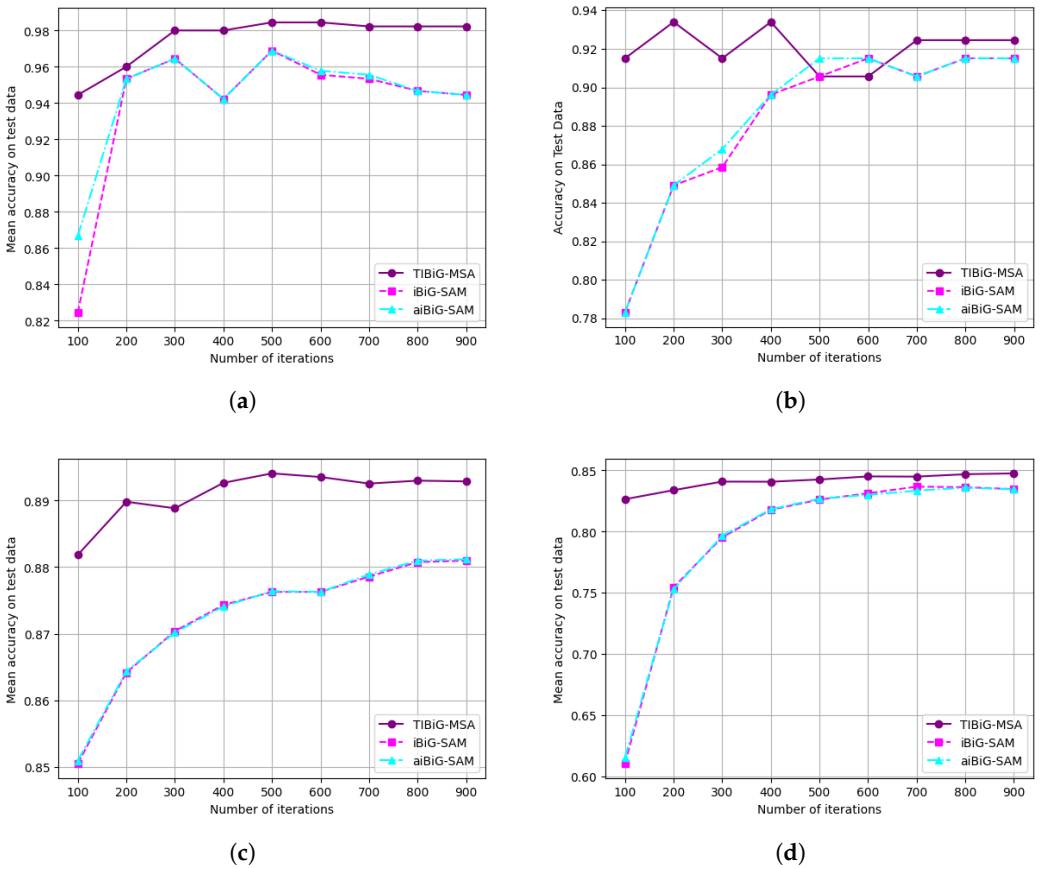


FIGURE 5. Comparison of mean accuracy on test data for TIBiG-MSA, iBiG-SAM, and aiBiG-SAM across different numbers of iterations on (a) Iris ($L = 50$), (b) Ionosphere ($L = 90$), (c) Hypertension ($L = 100$), and (d) Weather Type ($L = 100$).

6. CONCLUSIONS

In this paper, we proposed a novel two-step inertial algorithm (TIBiG-MSA) to tackle convex minimization problems when they arise in hierarchical optimization tasks. We proved a strong convergence theorem that the fixed-point of the algorithm is indeed a solution of the associated convex bilevel optimization problems. This algorithm was then applied to a series of data classification tasks using the ELM, TELM, and MELM models. All models were reformulated in the form of a series of convex bilevel optimization problems, which enabled effective use of the TIBiG-MSA algorithm. Numerical experiments performed on multiple datasets showed that the MELM model plus our proposed algorithm achieved the best classification accuracy and required fewer iterations to converge compared to iBiG-SAM and aiBiG-SAM. These results highlight the superior efficiency and performance of TIBiG-MSA. We demonstrated that the proposed TIBiG-MSA algorithm is both theoretically robust and practically effective, making it a valuable tool for addressing machine learning and optimization challenges. This method can be extended to other optimization problems, and future research may investigate how this method can be integrated with other advanced machine learning approaches.

ACKNOWLEDGMENTS

This work was supported by Chiang Mai University and Fundamental Fund 2025, Chiang Mai University, Chiang Mai, Thailand. K. Janngam was supported by the CMU Proactive Researcher Program, Chiang Mai University, under Contract No. 568/2567.

REFERENCES

[1] Aoyama, K.; Kohsaka, F.; Takahashi, W. Strong convergence theorems by shrinking and hybrid projection methods for relatively nonexpansive mappings in Banach spaces. *Nonlinear Analysis and Convex Analysis*, 7–26, Yokohama Publishers, Yokohama, 2009.

[2] Aoyama, K.; Kimura, Y. Strong convergence theorems for strongly nonexpansive sequences. *Appl. Math. Comput.* **217** (2011), 7537–7545.

[3] Bauschke, H.H.; Combettes, P.L. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. 2nd ed. Springer, Berlin, 2017.

[4] Beck, A.; Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.* **2** (2009), no. 1, 183–202.

[5] Bussaban, L.; Suantai, S.; Kaewkhao, A. A parallel inertial S-iteration forward–backward algorithm for regression and classification problems. *Carpathian J. Math.* **36** (2020), 35–44.

[6] Combettes, P.L.; Wajs, V.R. Signal recovery by proximal forward–backward splitting. *Multiscale Model. Simul.* **4** (2005), 1168–1200.

[7] Duan, P.; Zhang, Y. Alternated and multi-step inertial approximation methods for solving convex bilevel optimization problems. *Optimization* **71** (2022), no. 11, 3867–3896.

[8] Duan, P.; Li, H. A self-adaptive algorithm with multi-step inertia for solving convex bilevel optimization problems. *J. Appl. Numer. Optim.* **6** (2024), no. 3, 447–460.

[9] Fisher, R.A. Iris. *UCI Mach. Learn. Repos.* (1988). (Accessed 10 Sep 2024).

[10] Franceschi, L.; Frasconi, P.; Salzo, S.; Grazzi, R.; Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. *Proc. Int. Conf. Mach. Learn.* (2018), 1568–1577.

[11] Huang, G.-B.; Zhu, Q.-Y.; Siew, C.-K. Extreme learning machine: Theory and applications. *Neurocomputing* **70** (2006), 489–501.

[12] Iyiola, O.S.; Shehu, Y. Convergence results of two-step inertial proximal point algorithm. *Appl. Numer. Math.* **182** (2022), 57–75.

[13] Izuchukwu, C.; Aphane, M.; Aremu, K.O. Two-step inertial forward-reflected-anchored-backward splitting algorithm for solving monotone inclusion problems. *Comput. Appl. Math.* **42** (2023), 132.

[14] Janngam, K.; Suantai, S. An accelerated forward–backward algorithm with applications to image restoration problems. *Thai J. Math.* **19** (2021), 325–339.

[15] Janngam, K.; Suantai, S.; Cho, Y.J.; Kaewkhao, A.; Wattanataweekul, R. A novel inertial viscosity algorithm for bilevel optimization problems applied to classification problems. *Mathematics* **11** (2023), no. 14, 3241.

[16] Janngam, K.; Suantai, S.; Wattanataweekul, R. A novel fixed-point based two-step inertial algorithm for convex minimization in deep learning data classification. *AIMS Math.* **10** (2025), no. 3, 6209–6232.

[17] Kaewkhao, A.; Bussaban, L.; Suantai, S. Convergence theorem of inertial P-iteration method for a family of nonexpansive mappings with applications. *Thai J. Math.* **18** (2020), 1743–1751.

[18] LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **521** (2015), 436–444.

[19] Moreau, J.J. Fonctions convexes duales et points proximaux dans un espace hilbertien. *C. R. Acad. Sci. Paris Ser. A* **255** (1962), 2897–2899.

[20] Moudafi, A. Viscosity approximation methods for fixed-point problems. *J. Math. Anal. Appl.* **241** (2000), 46–55.

[21] Nakajo, K.; Shimoji, K.; Takahashi, W. Strong convergence to a common fixed-point of families of nonexpansive mappings in Banach spaces. *J. Nonlinear Convex Anal.* **8** (2007), 11–34.

[22] Nesterov, Y. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Sov. Math. Dokl.* **27** (1983), no. 2, 372–376.

[23] Nikhil7280. Weather type classification dataset. Kaggle (2021). Available at: `https://www.kaggle.com/datasets/nikhil7280/weather-type-classification/data` (Accessed 10 Sep 2024).

[24] Polyak, B.T. Some methods of speeding up the convergence of iterative methods. *USSR Comput. Math. Math. Phys.* **4** (1964), no. 5, 1–17.

[25] Qu, B.Y.; Lang, B.F.; Liang, J.J.; Qin, A.K.; Crisalle, O.D. Two-hidden-layer extreme learning machine for regression and classification. *Neurocomputing* **175** (2016), 826–834.

[26] Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **323** (1986), 533–536.

[27] Sabach, S.; Shtern, S. A first order method for solving convex bilevel optimization problems. *SIAM J. Optim.* **27** (2017), no. 2, 640–660.
[28] Sae-jia, P.; Suantai, S. A novel algorithm for convex bi-level optimization problems in Hilbert spaces with applications. *Thai J. Math.* **21** (2023), 625–645.
[29] Saejung, S.; Yotkaew, P. Approximation of zeros of inverse strongly monotone operators in Banach spaces. *Nonlinear Anal.* **75** (2012), 724–750.
[30] Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **61** (2015), 85–117.
[31] Shaban, A.; Cheng, C.A.; Hatch, N.; Boots, B. Truncated back-propagation for bilevel optimization. *Proc. Artif. Intell. Statist.* (2019), 1723–1732.
[32] Shehu, Y.; Vuong, P.T.; Zemkoho, A. An inertial extrapolation method for convex simple bilevel optimization. *Optim. Methods Softw.* **36** (2019), no. 1, 1–19.
[33] Sigillito, V.; Wing, S.; Hutton, L.; Baker, K. Ionosphere. *UCI Mach. Learn. Repos.* (1989). (Accessed 10 Sep 2024).
[34] Takahashi, W. *Introduction to Nonlinear and Convex Analysis*. Yokohama Publishers, Yokohama, 2009.
[35] Thong, D.V.; Reich, S.; Li, X.-H.; Tham, P.T.H. An efficient algorithm with double inertial steps for solving split common fixed-point problems and an application to signal processing. *Comput. Appl. Math.* **44** (2025), 102.
[36] Thongsri, P.; Suantai, S. New accelerated fixed-point algorithms with applications to regression and classification problems. *Thai J. Math.* **18** (2020), 2001–2011.
[37] Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B* **58** (1996), 267–288.
[38] Wattanataweekul, R.; Janngam, K.; Suantai, S. A novel two-step inertial viscosity algorithm for bilevel optimization problems applied to image recovery. *Mathematics* **11** (2023), no. 16, 3518.
[39] Xiao, D.; Li, B.; Mao, Y. A multiple hidden layers extreme learning machine method and its application. *Math. Probl. Eng.* **2017** (2017), 1–10.

[1] Research Center in Optimization and Computational Intelligence for Big Data Prediction, Department of Mathematics, Faculty of Science, Chiang Mai University, Chiang Mai 50200, Thailand
Email address: suthep.s@cmu.ac.th

[2] Department of Mathematics, Faculty of Science, Chiang Mai University, Chiang Mai 50200, Thailand

[3] Office of Research Administration, Chiang Mai University, Chiang Mai 50200, Thailand
Email address: kobkoon.j@cmu.ac.th