Journal of Parallel and Distributed Computing 205 (2025) 105144

Contents lists available at ScienceDirect

Journal of Parallel and Distributed Computing

journal homepage: www.elsevier.com/locate/jpdc

MM-AutoSolver: A multimodal machine learning method for the auto-selection of iterative solvers and preconditioners

Hantao Xiong [a,b], Wangdong Yang [a,b,*], Weiqing He [a,b], Shengle Lin [a,b], Keqin Li [a,b,c], Kenli Li [a,b]

[a] College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, China
[b] The National Supercomputing Center in Changsha, Changsha, Hunan 410082, China
[c] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Keywords:
Auto-selection
HPC
Iterative solver
Multimodal machine learning
Preconditioner
Sparse linear systems

ABSTRACT

The solution of large-scale sparse linear systems of the form $Ax = b$ is an important research problem in the field of High-performance Computing (HPC). With the increasing scale of these systems and the development of both HPC software and hardware, iterative solvers along with appropriate preconditioners have become mainstream methods for efficiently solving these sparse linear systems that arise from real-world HPC applications. Among abundant combinations of iterative solvers and preconditioners, the automatic selection of the optimal one has become a vital problem for accelerating the solution of these sparse linear systems. Previous work has utilized machine learning or deep learning algorithms to tackle this problem, but fails to abstract and exploit sufficient features from sparse linear systems, thus unable to obtain satisfactory results. In this work, we propose to address the automatic selection of the optimal combination of iterative solvers and preconditioners through the powerful multimodal machine learning framework, in which features of different modalities can be fully extracted and utilized to improve the results. Based on the multimodal machine learning framework, we put forward a multimodal machine learning model called MM-AutoSolver for the auto-selection of the optimal combination for a given sparse linear system. The experimental results based on a new large-scale matrix collection showcase that the proposed MM-AutoSolver outperforms state-of-the-art methods in predictive performance and has the capability to significantly accelerate the solution of large-scale sparse linear systems in HPC applications.

1. Introduction

In High-performance Computing (HPC) applications, Partial Differential Equations (PDEs) are commonly used for modeling complex processes. To numerically solve these PDEs on high-performance computers, it is necessary to solve large-scale sparse linear systems of the form $Ax = b$ [1]. The solution time for linear systems largely determines the total execution time of these HPC applications. Therefore, the efficient solution to large-scale sparse linear systems is crucial for enhancing the performance of HPC applications.

As HPC hardware, software, and application scales continue to expand, traditional direct solution methods [2] based on the matrix factorization are gradually being replaced by **iterative solvers** [3] due to their large memory overhead and poor parallelism. Iterative solvers offer better parallelism and require lower memory overhead, thus demonstrating significant advantages in large-scale HPC applications [4]. The convergence rate of iterative solvers is closely related to the numerical properties of the sparse coefficient matrix A. To improve convergence rate and stability, **preconditioning methods** are commonly employed to transform the original sparse linear system into a new one with better numerical properties. These transformation methods are also known as **preconditioners**.

With the continuous advancement of HPC and numerical computation, a wide array of iterative solvers and preconditioners are now available [5]. For a given sparse linear system, different combinations of iterative solvers and preconditioners exhibit significant variations in convergence properties. Table 1 showcases the solution results of several sparse linear systems built from matrices within Suitesparse Matrix Collection [6], and these systems are solved by adopting different combinations of iterative solvers and preconditioners. The results validate significant performance differences caused by different combinations. At present, it is challenging to select the optimal combination of iter-

* Corresponding author at: College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, China.
E-mail address: yangwangdong@hnu.edu.cn (W. Yang).

https://doi.org/10.1016/j.jpdc.2025.105144
Received 19 July 2024; Received in revised form 18 February 2025; Accepted 24 June 2025
Available online 1 July 2025

Table 1
Convergence speed (measured in seconds) of several combinations of iterative solvers and preconditioners. Combinations that do not converge are denoted as *Diverge*, and convergence is detected if the relative residual is less than 10e-8. All solution processes are run on a HUAWEI Taishan Server. Solution results are marked as *Timeout* if they neither converge nor diverge within 300 seconds.

Matrix name	fbcgs+jacobi	cg+ilu	symmlq+jacobi	minres+gamg	cr+jacobi	symmlq+sor	fgmres+gamg	fbcgs+ilu	cgs+gamg	symmlq+icc
2cubes_sphere	**0.476**	0.530	0.684	5.136	0.520	0.715	5.089	0.494	5.183	0.541
af_shell3	11.376	**6.189**	17.469	*Timeout*	9.753	11.103	*Timeout*	7.703	*Diverge*	10.440
af_shell4	13.179	**6.326**	17.556	*Timeout*	9.882	13.070	*Timeout*	7.436	*Timeout*	10.515
airfoil_2d	0.949	0.683	*Diverge*	*Diverge*	*Diverge*	*Diverge*	0.677	**0.291**	0.840	*Diverge*
Andrews	1.332	0.869	1.162	70.330	**0.857**	1.131	47.879	1.184	*Diverge*	1.225
FEM_3D_thermal1	0.385	0.122	0.258	1.136	0.236	0.186	1.086	0.125	1.226	**0.099**
FEM_3D_thermal2	2.115	0.872	1.470	7.362	1.416	1.598	7.164	0.951	7.713	**0.760**
ML_Laplace	225.595	*Timeout*	*Diverge*	*Timeout*	*Diverge*	*Diverge*	203.212	**137.896**	*Diverge*	*Diverge*
rail_20209	3.393	1.737	*Diverge*	*Diverge*	*Diverge*	*Diverge*	**0.474**	2.638	0.492	*Diverge*
poisson3Da	0.665	**0.499**	0.662	*Diverge*	0.634	*Diverge*	1.309	0.533	1.408	*Diverge*

ative solvers and preconditioners, especially for novice users without much proficiency in HPC and numerical analysis. Therefore, the auto-selection of the optimal combination of iterative solvers and preconditioners has become an important research problem for the efficient solution of sparse linear systems [7].

Machine learning [8] was first applied to the automatic selection of solvers and preconditioners by Bhowmick et al. [9,10]. This initial exploration led to the development of more machine learning based methods [11–13], which have increasingly incorporated recent methods focusing on feature selection [14] and specific applications [15]. Furthermore, as deep learning [16] has advanced, there has been a growing trend of employing deep learning based methods to predict the optimal iterative solver and preconditioner [17–20]. However, both two types of methods mentioned above fail to obtain excellent prediction performance due to a common limitation: **the insufficient exploration of features from sparse linear systems**.

Specifically, these methods employ either classical machine learning or deep learning algorithms for the auto-selection modeling. The former depends on **numerical features** derived from expert knowledge, offering strong interpretability, while the latter focuses on learning **structural features** from coefficient matrices, primarily linked to the topological arrangement of non-zero elements. Numerical features, including the diagonal dominance, symmetry, etc, directly impact the selection of iterative solvers and preconditioners [21]. Furthermore, structural features, such as block structures [22], are crucial in the construction and selection of sparse linear solvers and preconditioners. Deep learning has demonstrated the ability to effectively learn structural features, thereby optimizing the choice of iterative solvers and preconditioners [17–20]. However, both machine learning based methods and deep learning based methods utilize either numerical or structural features, without effectively combining the two to create a more comprehensive feature representation for sparse linear systems. Consequently, they fail to achieve satisfactory prediction accuracy in the auto-selection of the optimal iterative solvers and preconditioners.

Inspired by the development of multimodal machine learning [23, 24], a powerful learning paradigm involving integrating and learning features from multiple heterogeneous and interconnected sources of data, we propose to rethink the auto-selection of the optimal combination of iterative solvers and preconditioners as a multimodal machine learning problem that can be tackled in a multimodal machine learning framework. To be specific, we regard numerical features and structural features as two types of features with different modalities, which can be learned separately. The feature representations learned from these two modalities are then fused to create a more comprehensive feature representation for the final auto-selection modeling, aiming at improving prediction performance.

Based on the above insights and the mutimodal learning framework, we further put forward a multimodal machine learning model called MM-AutoSolver for the auto-selection of the optimal combination of iterative solvers and preconditioners for sparse linear systems. This model can learn both numerical features and structural features from coefficient matrices using the Convolutional Neural Network (CNN) [25]. By fusing these two types of features, the model is able to generate a more holistic representation, leading to improved prediction accuracy. Extensive experiments demonstrate that, based on the multimodal framework, the proposed MM-AutoSolver outperforms existing methods in prediction accuracy and is able to accelerate the solution of sparse linear systems originated in various HPC applications.

The contributions of this work can be summarized as follows:

- We propose to address the auto-selection of the optimal combination of iterative solvers and preconditioners within a multimodal machine learning framework, where numerical features and structural features learned are seen as the data of different modalities that can be learned and fused to boost the prediction performance.
- Based on the above multimodal machine learning framework, we put forward a multimodal machine learning model named MM-AutoSolver to predict (or select) the optimal combination of iterative solvers and preconditioners for sparse linear systems.
- Extensive experimental results demonstrate that the proposed MM-AutoSolver outperforms existing methods in prediction accuracy and is capable of accelerating the solution of large-scale sparse linear systems.

The rest of this work is organized as follows: In Section 2, we introduce the background on various iterative solvers, preconditioners, and multimodal machine learning techniques. The motivation for employing multimodal machine learning is also discussed in this section. Section 3 describes the methodology that we propose for predicting the optimal combinations of iterative solvers and preconditioners. In section 4, we report experimental results. Section 5 describes related research work on the auto-selection of iterative solvers and preconditioners. Finally, the conclusion of the work is presented in Section 6.

2. Background and motivation

This section covers the fundamental concepts of iterative solvers and preconditioners for addressing sparse linear systems, typically expressed as:

$$Ax = b, \tag{1}$$

and in this context, $A = [a_{ij}]$ denotes an $n \times n$ non-singular sparse coefficient matrix, while b is the specified right-hand side vector and x is the solution vector to be determined.

2.1. Iterative solvers

Due to the expansion of the application size and advancements in HPC, iterative solvers are increasingly favored for efficiently solving large-scale sparse linear systems. These methods are typically divided into two primary categories: stationary iterative methods and non-stationary iterative methods. Methods that can be represented in the following straightforward form

$$x^{(k+1)} = Bx^{(k)} + c, k = 0, 1..., \tag{2}$$

are known as stationary iterative methods, where B signifies a consistently invariable iteration matrix, and c represents a constant vector. Among the most commonly used stationary iterative methods are the Jacobi method, the Gauss-Seidel method, and the Successive Overrelaxation method [3].

The iterative mechanisms utilized in non-stationary iterative methods are notably more complex, posing difficulties in their direct representation via matrix-based formulations. Non-stationary iterative solvers are commonly associated with Krylov subspace methods [26], including the Conjugate Gradient (CG) method [27] and the Generalized Minimal Residual (GMRES) method [28]. Furthermore, non-stationary iterative methods contain sophisticated approaches like the Algebraic Multigrid method (AMG) [29], often utilized as preconditioners for Krylov subspace methods. Amidst the array of iterative methodologies, Krylov subspace methods emerge as the most efficient and widely adopted iterative solvers. Hence, in this work, our focus when discussing iterative solvers primarily revolves around Krylov subspace methods.

Krylov subspace methods belong to the framework of the projection based method. Considering A in Eq. (1) as an $n \times n$ real matrix, and K_m and L_m as m-dimensional subspaces of $\mathbb{R}_n$ ($m < n$), the projection method is designed to discover an approximate solution $\widetilde{x}$ within the affine space $x_0 + K_m$, with an initial guess x_0 for the solution vector x. This method imposes the condition that the new residual vector remains orthogonal to L_m. Specifically, the approximate solution can be defined as

$$r_0 = b - Ax_0, \tag{3}$$

$$\widetilde{x} = x_0 + \alpha, \alpha \in K_m, \tag{4}$$

$$(r_0 - A\alpha, \beta) = 0, \forall \beta \in L_m. \tag{5}$$

Two types of projection methods are commonly recognized: orthogonal and oblique. In an orthogonal projection method, L_m is identical to K_m, while in an oblique projection method, L_m differs from K_m and may not be related to it in any way.

Krylov subspace methods are renowned as the predominant and effective projection-based methodologies, featuring the subspace denoted as:

$$K_m = span\{r_0, Ar_0, A^2r_0, ..., A^{m-1}r_0\}, \tag{6}$$

where $r_0 = b - Ax_0$. Referred to as the Krylov subspace, this mathematical construct serves as a fundamental component in iterative solving techniques.

The CG is one of the most representative iterative solver, whose search space is formed by a set of conjugate vectors and is widely used for solving symmetric positive definite systems. In CG, the search space is the same as the constraint space, i.e. $L_m = K_m$. The GMRES is an iterative solver commonly used to solve nonsymmetric linear systems. It aims to minimize the residual error iteratively by generating a sequence of approximate solutions that converge to the true solution. GMRES iteratively constructs an orthogonal basis for the Krylov subspace, which allows for the efficient solution of the linear system. In GMRES, $L_m = AK_m$. Another class of Krylov subspace methods is based on the equation $L_m = A^T K_m$. Representative methods in this category include the Biconjugate Gradient method (BCG) [30], Biconjugate Gradient Stabilized (BCGS) [31] and others. These methods are primarily used for solving nonsymmetric linear systems. In comparison to GMRES, they do not require storing the basis vectors of the subspace, resulting in lower memory overhead.

2.2. Preconditioning methods

The efficiency of iterative solvers is greatly determined by the eigenvalue distribution of the coefficient matrix. Typically, solvers exhibit better convergence when the condition numbers are smaller or when the eigenvalues are more tightly clustered. To improve the convergence rate, preconditioning methods are adopted to adjust the eigenvalue distribution [4]. There are several steps in preconditioning a given sparse linear system. Initially, a preconditioning matrix M (preconditioner) is specified. Matrix M is then used to convert the original linear system $Ax = b$ into a new form $M^{-1}Ax = M^{-1}b$. Alternatively, the preconditioning matrix can be applied on the right side, leading to $AM^{-1}u = b$, with the solution $x = M^{-1}u$. Provided that the preconditioner M can be decomposed into the product of two matrices, as $M = M_L M_R$, the system can be further transformed into the form $M_L^{-1}AM_R^{-1}u = M_L^{-1}b$, with $x = M_R^{-1}u$. These preconditioned sparse linear systems are then solved using different Krylov subspace methods such as CG or GMRES. By incorporating preconditioner, these methods become their corresponding Preconditioned Krylov subspace solvers, which typically exhibit faster convergence compared to their non-preconditioned counterparts.

For some preconditioning methods, a preconditioning matrix M is constructed from the coefficient matrix A. Common approaches include extracting the diagonal elements of A, the upper triangular part, or performing incomplete LU and Cholesky decomposition of the coefficient matrix A [3]. In some more complex preconditioners, it is often not feasible to explicitly construct the preconditioning matrix M. The preconditioning process typically involves solving a linear system, represented by methods such as AMG [29].

2.3. Motivation

Multimodal machine learning is a method that aims to build models capable of processing data of multiple modalities [32]. In real-world applications, multimodal data is prevalent. For instance, a multimedia application might include a combination of images, text, audio, and video contents [33]. Besides, the generalized multimodal data can be seen as multiple groups of features from raw data using different feature extraction methods [34,35]. Viewed from this perspective, numerical features from expert knowledge and structural features learned by deep learning models can be regarded as two different modalities of data extracted from sparse linear systems.

According to numerical theory, numerical features such as values of non-zero elements, symmetry ratio, diagonal dominance, and others, play a significant role in choosing appropriate solvers and preconditioners [21]. At the same time, the selection of solvers and preconditioners depends on structural features, which deep learning models have proven effective at extracting directly from coefficient matrices. In recent years, CNN [25] has been employed to learn structural features from coefficient matrices and applied in both preconditioner generation [22] and selection [18]. Furthermore, GNN [36] has been utilized for selecting preconditioners and Krylov solvers [20], and other deep learning models, such as FCNN [37], have also been investigated by Funk et al. [19] for the automatic selection of iterative solvers and preconditioners. These studies have all demonstrated the importance of structural features in coefficient matrices for selecting iterative solvers and preconditioners.

Although both numerical and structural features provide important insights for selecting the optimal iterative solver and preconditioner, existing research has not yet combined these two types of features, which has the potential to boost prediction performance. As a result, the prediction accuracy remains suboptimal.

This motivates us to investigate multimodal machine learning approaches that fuse these two types of features, with the goal of enhancing the prediction performance for the automatic selection of the optimal combination of iterative solvers and preconditioners for sparse linear systems.

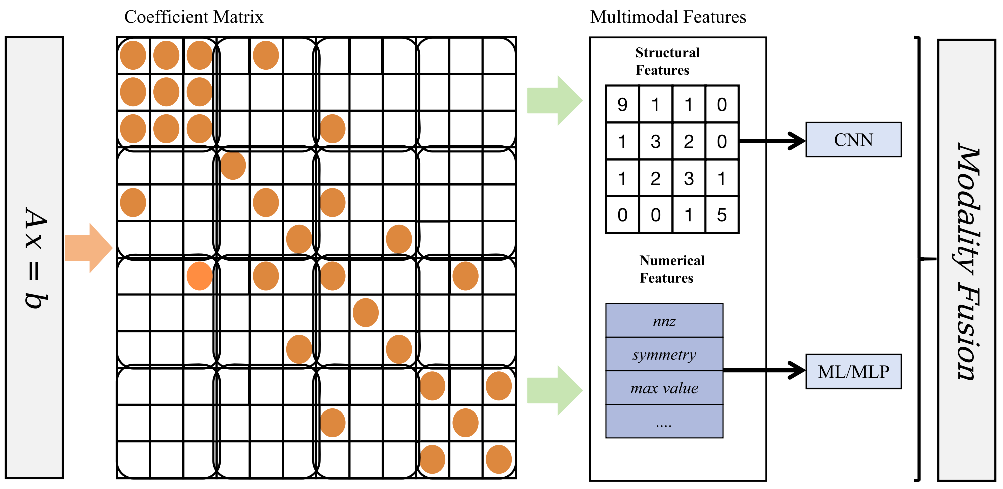


Fig. 1. The multimodal framework for the auto-selection of the optimal combination of iterative solvers and preconditioners, which can be used to learn and fuse features of different modalities to boost the predictive accuracy.

3. Methodology

In this section, we begin by formulating the automatic selection of the optimal combination of iterative solvers and preconditioners within a multimodal multi-class machine learning framework. Following this, we provide a detailed description of the proposed model MM-AutoSolver.

3.1. Multimodal learning framework

For a given sparse linear system $Ax = b$, different combinations of iterative solvers and preconditioners present different convergence properties. The auto-selection of the optimal combination means that given multiple candidate combinations, a model needs to be built to predict the optimal combination in terms of convergence properties. In this work, the optimal combination refers to the one with the shortest solution time, meanwhile, meeting the convergence requirements. From a machine learning perspective, this problem can be viewed as a supervised multi-class classification task, with labels representing all possible candidate combinations. Each sparse linear system with its corresponding optimal solution combination constitutes a sample.

To improve prediction accuracy, extracting informative features from sparse linear systems has become a key challenge. From a numerical analysis standpoint, the features from coefficient matrices play a significant role in determining the iterative convergence rate of the sparse linear system $Ax = b$. These features include both numerical features and structural features.

Numerical features from the coefficient matrix A are important factors influencing the convergence properties of iterative solvers and serve as crucial references for selecting iterative solvers and preconditioners. For instance, the number and distribution of non-zero elements directly influence the computational performance of Sparse Matrix-Vector Multiplication (SpMV) in iterative solvers, which is generally the most time-consuming kernel. Another example is that diagonal dominance has a direct impact on the number of iterations. Overall, numerical features reflect both the computational cost per iteration, in terms of floating-point operations, and the difficulty of achieving iterative convergence (i.e., the number of iterations). These features offer valuable insights for selecting the optimal iterative solvers and preconditioners.

Structural features of the coefficient matrix also have a significant impact on the selection of iterative solvers and preconditioners. For instance, Götz et al. [22] suggested that sparse matrices with dense natural blocks along their diagonal are particularly suitable for the Block Jacobi preconditioner [3], and that CNN can be employed to detect these blocks. Yamada et al. [18] proposed the use of CNN to predict the optimal preconditioner for solving sparse linear systems. In their approach, the input to the CNN is a down-sampled, fixed-size image derived from the original coefficient matrix. These examples demonstrate the importance of structural features in selecting the combination of iterative solvers and preconditioners.

Both numerical features and structural features are extracted from sparse linear systems, but their forms are completely different, and methods of extracting and learning these features are distinct. Therefore, they can be attributed into the spectrum of **generalized multimodal data**. Taking these factors into consideration, we propose to address the automatic selection of the optimal combination of iterative solvers and preconditioners through a powerful multimodal machine learning framework, as shown in Fig. 1, and there are features of two different modalities in this multimodal framework:

- **Numerical Features**. These are features derived from coefficient matrices according to expertise in numerical analysis and HPC, which mainly contain features that are used to distinguish different iterative solvers and preconditioners and have good interpretability. All the numerical features used in this work are shown in Table 2. The main distinction of these features compared to existing methods lies in their lower computational complexity. The maximum time complexity for their computation is $O(n)$ or $O(nnz)$, where n and nnz represent the number of rows and the number of non-zero elements in the matrix, respectively. In order to learn the mapping between these features and target combinations, traditional machine learning algorithms and MLP are adopted for modeling, since these algorithms can directly adopt numerical features as the input and learn from these features directly and effectively.
- **Structural Features**. Features of this type are mainly determined by the sparsity pattern, namely the distribution of non-zero elements. The CNN has been shown to effectively extract structural features from sparse matrices, which can be used to predict the best performance for various numerical computation tasks. For example, the CNN can be employed to predict optimal preconditioners [18], learn structural features from coefficient matrices for the construction of Block Jacobi preconditioners [22], and predict the best format for tasks like SpGEMM [38,39]. Inspired by the work in [18,38,39], we use the CNN for feature extraction and learning to capture the structural features in this study. Since the CNN require fixed-size input for model training and inference, we perform a downsampling-like operation on the original sparse matrices of varying sizes to obtain a density representation matrix of fixed size. Specifically, to compute the density representation, we divide the sparse matrix into n equal partitions along both the rows

Table 2
Numerical features from coefficient matrices.

Feature	Definition
row_num	Number of rows (or columns)
nnz	Number of non-zero elements
nnz_ratio	Ratio of non-zero elements
nnz_lower	Nnz in the lower triangular part
nnz_upper	Nnz in the upper triangular part
nnz_diagonal	Nnz in the diagonal
ave_nnz_row	Average nnz of all rows
max_nnz_row	Max nnz of all rows
arr_nnz_rows	Variance of nnz of all rows
max_value	Max absolute value in all off-diagonal elements
max_value diagonal	Max absolute value in all diagonal elements
diagonal_dominance_ratio	Ratio of diagonal dominant rows
is_symmetry	Symmetry of coefficient matrices
pattern_symmetry	Symmetry ratio of non-zero distribution
value_symmetry	Symmetry ratio of non-zero elements
row_variability	Logarithm of the maximum row-wise max-to-min absolute ratio
col_variability	Logarithm of the maximum column-wise max-to-min absolute ratio

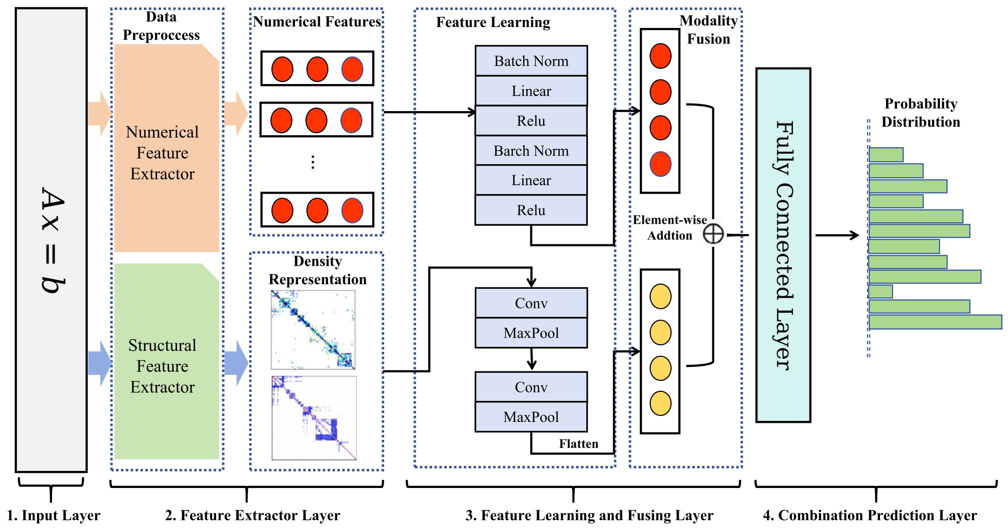


Fig. 2. The model architecture of the proposed MM-AutoSolver.

and columns, forming $n \times n$ blocks. Then, we count the number of non-zero elements in each block and assign this count as the corresponding value in the density representation matrix. Although the density representation loses some information about the distribution of non-zero elements during downsampling, it can still reflect structural features with lower computational and storage costs, while also facilitating subsequent feature learning. Fig. 1 illustrates how the density representation is derived from the input sparse matrix, with $n = 4$.

Based on above framework, a multimodal machine learning model called MM-AutoSolver is proposed to predict the optimal combination of iterative solvers and preconditioners for sparse linear systems, which is covered in detail in the next subsection.

3.2. Model

In this section, we provide detailed descriptions of the proposed MM-AutoSolver. It exploits multimodal machine learning framework to learn and fuse features of different modalities extracted from sparse linear systems, thereby able to improve the predictive accuracy of combinations of iterative solvers and preconditioners. The architecture of MM-AutoSolver is demonstrated in Fig. 2, which mainly consists of four parts: the input layer, the feature extractor layer, the feature learning and fusing layer, and the combination prediction layer.

3.2.1. Input layer

The input layer is used for reading sparse linear equations and converting them into the Compressed Sparse Row (CSR) [40] format to facilitate subsequent feature extraction.

3.2.2. Feature extractor layer

The feature extractor layer is mainly used to preprocess the original data, namely extracting features of different modalities from original sparse linear systems. In this work, we mainly use features of two modalities, including numerical features and structural features. Therefore, an AnaMod [41] based numerical feature extractor is developed to extract numerical features shown in Table 2, and these features will be organized into a normalized embedding vector. To extract structural features from sparse linear systems, we transform coefficient matrices of different sizes into fixed-size $n \times n$ ($n = 128$ in this study) matrices called density representations. These fixed-size density representations can be seen as images that are well-suited for processing by CNN.

3.2.3. Feature learning and fusing layer

The feature learning and fusing layer is a module that learns features of different modalities and fuse them to obtain a more comprehensive feature representation. Numerical features usually have a more direct relationship with the selection of iterative solvers and preconditioners. We argue that complicated deep learning models with too many layers will fail to learn this direct relationship. Therefore, the MLP based module is designed to learn these numerical features. In this module, we adopt two fully-connected layers to learn numerical features. Additionally, to accelerate the convergence of model training, we apply batch normalization before each fully-connected layer. The output of this module is a numerical feature vector y_n. In this work, structural features are learned from density representations of coefficient matrices by executing CNN on them. As can be seen from Fig. 2, these density representations pass through two convolutional neural network layers with each followed by a maxpooling layer. The output of CNN is finally flattened to obtain a feature vector y_s.

Inspired by insights from the field of multimodal machine learning [32,33], we argue that y_s and y_d are two feature representations that learned from two kinds of features of different modalities, namely representations learned from numerical features and structural features. Given the numerical feature vector $y_n = \{y_1, y_2, ..., y_M\}$ and the structural feature vector $y_s = \{d_1, d_2, ..., d_N\}$, where M and N are dimensions of these two feature vectors respectively, we define the fusion layer as a function:

$$f_{output} = Fusion(y_n, y_s), \tag{7}$$

and in this work, function $Fusion$ is defined as the element-wise addition operation, and therefore, the vector size M is equal to the vector size N.

3.2.4. Combination prediction layer

The output of the feature learning and fusing layer f_{output} is the fused representation of numerical features and structural features, which is denoted as $f_{output} = y_s \oplus y_d$. A fully-connected layer in this module uses this fused representation f_{output} as the input to calculate the final output of MM-AutoSolver. A softmax function is adopted to convert the final output into the probabilities of all candidate combinations of solvers and preconditioners. The loss function of the work is defined as follows:

$$loss = \sum_{i \in G} CrossEntropy(p_i, \bar{p}_i), \tag{8}$$

where $\bar{p}_i$ is the predictive label of the combination, and p_i is the ground-truth label. The letter G denotes the number of the training samples and $CrossEntropy(.,.)$ is the cross entropy function.

4. Experiment and evaluation

In this part, we first give a description of the experimental setting. Then we compare our proposed model MM-AutoSolver with several baseline methods. We also conduct an ablation study to investigate the effectiveness of the MM-AutoSolver. Finally, we evaluate the performance improvements brought by the MM-AutoSolver for solving the sparse linear system from the perspectives of speedup and overhead.

4.1. Experiment setting

4.1.1. Datasets

In this part, we describe the process of building the dataset for training the proposed MM-AutoSolver, which is a multi-class multimodal machine learning model. The well-known SuiteSparse Matrix Collection [6] has a limited number of matrices suitable for building sparse linear systems, with many being non-sparse, non-square, or pattern-type matrices. After filtering these out, fewer than 1,700 matrices remain. Additionally, many matrices are small, with over 50% having row sizes under 5,000, and 5.21% exceeding 1 million. Moreover, many large-scale matrices, such as those from graph applications, do not originate from sparse linear systems, making their sparsity patterns unsuitable for this purpose. Therefore, the SuiteSparse Matrix Collection is not ideal as the dataset for automatic selection modeling. In this work, the finite element based software FreeFEM [42] and the computational fluid dynamic software OpenFoam [43] are used to produce a large-scale sparse matrix collection for constructing linear systems. We use well-defined sample programs from packages FreeFEM and OpenFOAM, including those governed by well-known equations such as the Stokes and Poisson equations. Solving these sample programs using finite element and finite volume methods requires solving sparse linear systems, which typically results in sparse matrices that can be used to construct linear systems. Furthermore, by varying the parameters in these samples, a large number of sparse matrices can be generated. We have assembled a new collection of 12,914 sparse matrices. Among these, 12,651 matrices are generated from sample programs in FreeFEM and OpenFOAM, with more than 4,000 matrices having sizes exceeding 1 million, better reflecting the matrix sizes encountered in real-world applications. The remaining 263 matrices come from the SuiteSparse Matrix Collection [6]. Subsequently, for each sparse matrix, we generate the right-hand side vector b by randomly assigning a value within the interval $[0, 1)$ to each element of the x vector, and obtaining the b vector by multiplying A with x. For all sparse linear systems, the combinations of iterative solvers and preconditioners from PETSc [44] v3.17.3 are utilized to solve them on a HUAWEI Taishan server (with 2 Kunpeng 920 processors [45]) with unified parameter settings. By analyzing the iterative solution results of these linear systems, we screen out 19 optimal combinations with high frequency as labels of the multi-class problem. Note that for a given sparse linear system, the combination that meets the convergence accuracy and takes the shortest time to solve is the optimal combination, namely the label. Each sparse linear system and its corresponding label constitute a data sample. Among 12,914 data samples, one-tenth of them are used as the test set and the rest as the training set. For the training set and test set, the number of sparse linear systems corresponding to each label is shown in Table 3.

Table 3
The statistics of labels in the training and test sets.

Labels	Training set	Test set
fbcgs+jacobi	2,173	231
bcgsl+none	2,054	223
symmlq+icc	1,201	125
symmlq+jacobi	923	99
dgmres+none	650	85
gmres+gamg	640	70
cr+eisenstat	598	66
symmlq+sor	582	57
fbcgs+ilu	562	56
minres+gamg	524	61
fcg+gamg	342	41
cr+jacobi	310	40
cg+ilu	275	38
fgmres+gamg	226	34
cg+eisenstat	224	23
cg+bjacobi	193	29
cr+ilu	68	5
cgs+gamg	49	4
bcgsl+asm	29	4
Total	11,623	1,291

4.1.2. Evaluation metrics

Given that the task of predicting the optimal combination is framed as a multi-class classification problem, we utilize four commonly used metrics: Accuracy (Acc), Macro Precision (MP), Macro Recall (MR), and Macro F1 score (F1). MP, MR, and F1 are computed as macro averages, treating all classes with equal importance. While Acc can indicate the overall predictive accuracy, it may be skewed by classes with more training samples. Consequently, MP, MR, and F1 are adopted to provide a more balanced and comprehensive assessment of the predictive performance.

4.1.3. Baselines

In order to assess the performance of our proposed MM-AutoSolver, we compare it against two notable baseline methods:

Table 4
Hyper parameter setting.

Hyper parameters	Values
Epoch	256
Batch size	512
Optimizer	Adam
Cost function	Cross Entropy
Learning rate	1e-3
Number of numerical features	17
Neuron number of hidden layers in MLP	1,024 & 128
Output vector size of MLP (y_n)	19
Activation function of MLP	Relu
Size of the density representation	128×128
Size of the first conv kernel	3×3
Size of the second conv kernel	5×5
Size of the maxpooling	2×2
Activation function of CNN	Tanh
Output vector size of CNN (y_s)	19

- **Machine Learning Based Methods**: For multi-class classification tasks, we employ eXtreme Gradient Boosting (XGBoost) [46], a powerful tree-based ensemble learning algorithm. XGBoost, an enhanced version of the ADT [47], is widely recognized for its robustness and efficiency in handling classification problems. Thus, it serves as a strong benchmark representing traditional machine learning approaches.
- **Deep Learning Based Methods**: Inspired by the work of Funk et al. [19], who introduced a deep learning model (denoted as DLSolver in this work) for predicting the optimal solver for a given sparse linear system, we use DLSolver as the deep learning baseline. This model, designed for multi-class classification, is considered state-of-the-art methodology in predicting iterative solvers, making it an ideal candidate for comparison with the proposed model MM-AutoSolver.

4.1.4. Implementation details

In this work, TensorFlow [48] is leveraged to implement the proposed MM-AutoSolver model, which is trained by adopting the aforementioned training set. MM-AutoSolver is trained on a heterogeneous computing system with an Nvidia A100 GPU. The detailed settings for hyper parameters in MM-AutoSolver are shown in Table 4. The density representation size is chosen to be 128×128 from the evaluated options of 64×64, 128×128, and 256×256, as it yields the best prediction performance.

4.2. Overall prediction performance

Table 5 shows the overall prediction performance of this work, including two parts. The first part showcases the performance evaluation of the model MM-AutoSolver and baseline models that we employ: XGBoost and DLSolver. In the second part, we demonstrate the evaluation results of the ablation study.

The results indicate that MM-AutoSolver consistently outperforms both XGBoost and DLSolver across all four evaluation metrics, reaching 78.54%, 63.41%, 62,81%, and 62.53%, respectively. Specifically, it achieves improvements of 3.56% in accuracy (Acc), 2.95% in mean precision (MP), 5.92% in mean recall (MR), and 4.84% in F1 score compared to DLSolver, which can be seen as state-of-the-art methodology. Additionally, MM-AutoSolver surpasses XGBoost by a significant margin in all metrics, as illustrated in Table 5. XGBoost, known for its robust ensemble learning capabilities in multi-class classification and regression, is a tree-based model that enhances prediction accuracy through the integration of multiple decision trees. However, the lower accuracy of XGBoost in Table 5 suggests that machine learning models struggle to accurately predict multiple combinations of iterative solvers and preconditioners, due to their inability in capturing complex relationships within matrix features. MM-AutoSolver achieves improvements over DLSolver in all metrics. The primary advantage of MM-AutoSolver lies in its ability to incorporate structural features of matrices, which is further combined with numerical features using multimodal machine learning framework. We hypothesize that these features are crucial in enhancing prediction performance and will test this hypothesis in an ablation study. Overall, our results demonstrate the superior efficacy of MM-AutoSolver in predicting the optimal combinations of iterative solvers and preconditioners.

Table 5
The overall performance of different methods for predicting the optimal combination of iterative solvers and preconditioners.

Metrics	Acc	MP	MR	F1
MM-AutoSolver	**78.54%**	**63.41%**	**62.81%**	**62.53%**
XGBoost	33.85%	17.33%	14.96%	12.56%
DLSolver	74.98%	60.46%	56.89%	57.69%
MM-AutoSolver w/o MLP	65.22%	43.80%	42.52%	41.92%
MM-AutoSolver w/o CNN	75.60%	59.87%	58.21%	57.67%

4.3. Case analysis

This section presents a case analysis to demonstrate how MM-AutoSolver interacts with several types of combinations and their corresponding matrices. For example, the fbcgs+jacobi combination, with 2,173 training and 231 test samples, achieves 98.19% prediction accuracy. This combination performs best in linear systems from the FreeFEM Elasticity routine, where homogeneous Dirichlet boundary conditions are applied to two edges in 3D linear elasticity PDEs, resulting in a lower triangular sparse matrix. Similarly, the bcgsl+none combination, with 2,054 training and 223 test samples, reaches 99.10% prediction accuracy, corresponding to nearly symmetric matrices from FreeFEM solving Laplace and Stokes equations. The symmlq+icc combination, with 1,201 training and 125 test samples, achieves 83.85% prediction accuracy, corresponding to symmetric matrices with strong diagonal dominance from the icoFoam sample program in OpenFOAM.

4.4. Ablation study

The second part of Table 5 demonstrates the results of the ablation study. The key innovation of MM-AutoSolver lies in its utilization of multimodal machine learning techniques to extract matrix features of different modalities from sparse linear systems, namely numerical features and structural features. Subsequently, these extracted features of two modalities are learned using different machine learning or deep learning modules. Finally, the learned information from both modalities are fused to enhance the MM-AutoSolver's prediction performance. Specifically, as can be seen from Fig. 2, features of two different modalities from sparse linear systems pass through an MLP based sub-model and a CNN based sub-model that learn numerical features and structural features, respectively. The main purpose of the ablation study is to validate the contribution of features of different modalities and their corresponding learning components to the improvement of the model's effectiveness. Therefore, we conduct experiments by removing features of different modalities separately to observe their impact on the model's performance. By this means, we construct two models, MM-AutoSolver w/o MLP and MM-AutoSolver w/o CNN, to investigate the impact of numerical features and structural features, along with their corresponding learning components, on the model's performance. Among these two models, the former one removes numerical features and their learning components (MLP based sub-model), while the latter one eliminates structural features and its corresponding machine learning components (CNN-based sub-model).

As can be seen from Table 5, both MM-AutoSolver w/o MLP and MM-AutoSolver w/o CNN have declined in all four metrics compared to MM-AutoSolver. MM-AutoSolver w/o MLP's four metrics decrease by

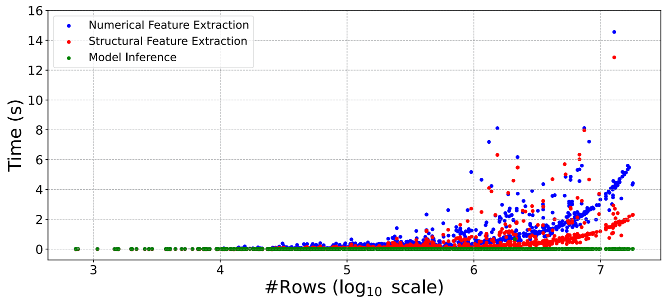


Fig. 3. The overhead of model inference of MM-AutoSolver and extracting numerical features and structural features. The horizontal axis represents the matrix scale (number of rows) of all test sparse linear systems sorted in increasing order of the base-10 logarithm.

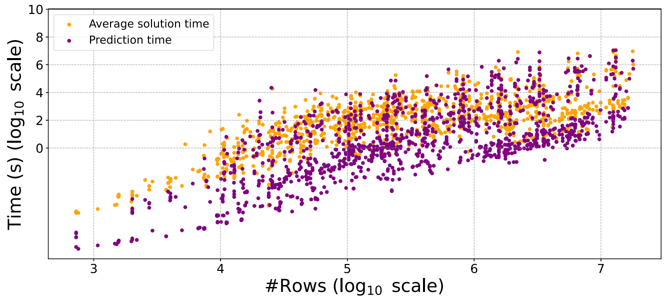


Fig. 4. The average solution time and the time of the predicted combination. The horizontal axis represents the matrix scale (number of rows) of all test sparse linear systems sorted in increasing order of the base-10 logarithm. The vertical axis represents the base-10 logarithm of the time.

13.32%, 19.61%, 20.29%, and 20.61%, respectively. Meanwhile, MM-AutoSolver w/o CNN's four metrics drop by 2.94%, 3.54%, 4.60%, and 4.86%, respectively. Clearly, the performance decline of MM-AutoSolver w/o MLP is much greater than that of MM-AutoSolver w/o CNN. Therefore, we can conclude that both numerical features and structural features are complementary to each other and play a positive role in boosting the predictive accuracy. Excluding numerical features along with their learning components leads to a notable drop in performance. Therefore, we can conclude that numerical features are more significant in predicting iterative solvers and preconditions. After removing structural features along with their learning components, there is also a decline in all four metrics. Hence, it can be concluded that the learned structural features play a positive role in enhancing the model's performance. In summary, the ablation study confirms the validity of the multimodal machine learning framework and the effectiveness of the proposed MM-AutoSolver.

4.5. Speedup evaluation

In this part, we mainly evaluate the performance improvement brought by the proposed MM-AutoSolver for large-scale sparse linear systems from two aspects. The first aspect is the overhead incurred by MM-AutoSolver, which primarily includes feature extraction overhead and model inference overhead. The second aspect is the speedup effect on solving sparse linear systems introduced by the automatic selection model.

4.5.1. Overhead

MM-AutoSolver's overhead for the automatic selection of the optimal combination of iterative solvers and preconditioners mainly includes two aspects: feature extraction overhead and model inference overhead.

In this work, we employ features of two different modalities, including numerical features and structural features. The numerical features that we adopt in this work are shown in Table 2. These features can be obtained by traversing all *nnz* of the coefficient matrix, meaning that the time complexity for calculating numerical features does not exceed $O(nnz)$. Therefore, theoretically, the feature extraction time should be relatively low. The primary overhead for extracting structural features is converting original coefficient matrices into fixed-size density representations. This process still requires traversing all *nnz*, so the time complexity remains $O(nnz)$, resulting in relatively low theoretical overhead. As shown in Fig. 2, we implement a data preprocess module in feature extractor layer for extracting two different types of features.

Model inference overhead refers to the time cost of inputting numerical and structural features into a trained automatic selection model and performing forward inference computations to obtain the probabilities corresponding to different candidate combinations. The inference overhead of the model is mainly related to the model structure. Therefore, in this work, the inference overhead of the model is essentially consistent for all sparse linear systems.

Therefore, in this work, we define the overhead of employing MM-AutoSolver as

$$T_{\text{overhead}} = T_{\text{numerical}} + T_{\text{structural}} + T_{\text{inference}}, \tag{9}$$

where $T_{\text{numerical}}$, $T_{\text{structural}}$, and $T_{\text{inference}}$ respectively represent the numerical feature extraction overhead, structural feature extraction overhead, and model overhead. Fig. 3 shows the overhead of data preprocess module for extracting numerical and structural features, as well as the model inference overhead. The model inference cost remains essentially at the millisecond level and largely unchanged, independent of the matrix scale. The extraction overhead of both numerical and structural features is positively correlated with the matrix scale. Because larger matrices generally have more non-zero elements. Additionally, the extraction overhead of numerical features is generally greater than that of structural features.

4.5.2. Speedup

The automatic selection of the optimal combination of iterative solvers and preconditioners aims to build an model that can automatically specify the optimal combination for a given sparse linear system to minimize the solution time. In this work, we define the average solution time as

$$T_{\text{aver}} = \frac{1}{N}\sum_{i=1}^{N} T_i, \tag{10}$$

where T_i represents the solution time of the *i-th* candidate combination, and N is the number of total candidate combinations. Specifically, T_{aver} represents the expectation of solution time without using any automatic selection models. $T_{\text{prediction}}$ is used to denote the solution time of the predictive combination. Fig. 4 describes the general trend of solution time of the predictive combination and average solution time of all combinations as the matrix size increases. It can be observed that, overall, the prediction time is less than the average solution time, and both tend to increase as the matrix size grows.

We define the speedup as

$$P_{\text{speedup}} = T_{\text{aver}}/T_{\text{prediction}}, \tag{11}$$

in which P_{speedup} represents the ratio of T_{aver} to $T_{\text{prediction}}$, thus can be seen as the performance improvement factor brought by the automatic selection model. Considering the overhead of feature extraction and model inference, the speedup can be further defined as

$$P_{\text{speedup\_over}} = T_{\text{aver}}/(T_{\text{prediction}} + T_{\text{overhead}}), \tag{12}$$

which can be used to evaluate the performance improvement that MM-AutoSolver brings to the solution of sparse linear systems.

Fig. 5 illustrates the performance speedup brought by MM-AutoSolver in solving 1,291 sparse linear systems from the test set. It is evident that for the majority of sparse linear systems in the test set, both

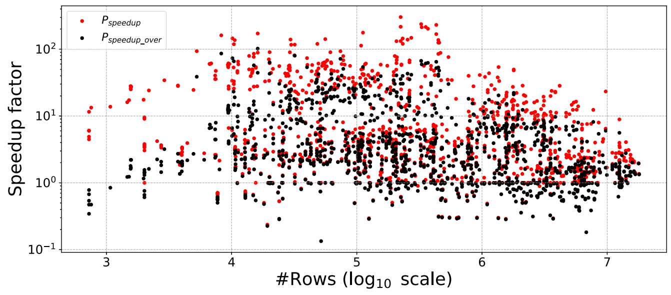


Fig. 5. The speedup that MM-AutoSolver brings to the solution of sparse linear systems in the test set. The horizontal axis represents the matrix scale (number of rows) of all test sparse linear systems sorted in increasing order of the base-10 logarithm.

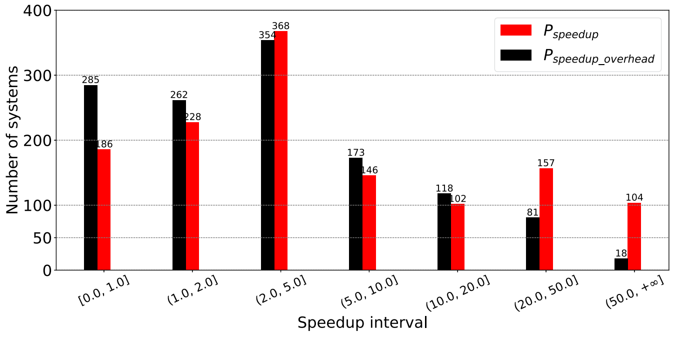


Fig. 6. The overview of the speedup ratios for sparse linear systems in the test set under two evaluation metrics.

metrics P_{speedup} and $P_{\text{speedup\_over}}$ show a speedup ratio greater than 1.0 (10^0), indicating a positive acceleration effect. In fact, among the 1,291 test cases, 1,105 (85.6%) of them have P_{speedup} greater than 1.0 and 1,006 (77.9%) of them have $P_{\text{speedup\_over}}$ greater than 1.0. Specifically, a significant portion of the test cases achieve speedup ratios of several tens, and a few sparse linear systems achieve nearly 100x speedup. Fig. 6 further illustrates the speedup achieved by MM-AutoSolver for sparse linear systems in the test set under two speedup metrics. These results clearly demonstrate that MM-AutoSolver can significantly enhance the solution of large-scale sparse linear systems.

5. Related work

Recent advancements in machine learning [8], with a particular focus on deep learning [16], have created new opportunities to automate the selection process of iterative solvers and preconditioners for sparse linear systems. This section delves into a thorough review of contemporary research that leverages machine learning and deep learning methodologies to improve the automatic selection or prediction of suitable iterative solvers and preconditioners.

Bhowmick et al. [9] pioneered the use of machine learning techniques for the auto-selection of solvers in sparse linear systems. They provided a comprehensive methodology for the organization of datasets, the identification of crucial features, the formulation of selection criteria for various iterative solvers, and the training of machine learning models (e.g., SVM [49] and Adaboost [50]). This work clearly exemplifies the use of machine learning for solver selection in a structured and systematic manner. In their later work, Bhowmick et al. [10] introduced a low-cost strategy for feature computation and selection. The strategy enables the streamlined development of conventional machine learning models, including Nearest Neighbor [51], Naive Bayes [52], and SVM, which are leveraged to forecast the most suitable iterative solver and preconditioner. Eijkhout et al. [53] proposed a comprehensive framework for advising on numerical methods and created supporting libraries to apply the principles set forth in this framework. Eller et al. [11] applied machine learning algorithms to determine the best combination of iterative solvers and preconditioners for transient simulations. These simulations often involve larger-scale linear systems than those from the SuiteSparse Matrix Collection [6]. Sakurai et al. [12] developed a method using historical residual data to anticipate the best pairing of iterative solvers and preconditioning methods. Jessup et al. [13,54] launched the Lighthouse project [55], which utilizes machine learning models to predict the effectiveness of various combinations of iterative solvers and preconditioners for sparse linear systems. Moreover, they developed a performance model to analyze the communication overhead of different iterative methods, with the goal of identifying the most suitable solver for distributed computing environments [56]. Recently, Sun et al. [14] introduced a feature selection strategy for machine learning models predicting Krylov subspace solvers. Zabegaev et al. [15] proposed to automate the selection of linear solvers for time-dependent problems. The aforementioned research work primarily explores the use of traditional machine learning methods to predict appropriate iterative solvers and preconditioners. These approaches are notable for their interpretability, enabling them to elucidate the relationship between sparse linear systems and the suitable iterative solvers or preconditioners.

In recent years, researchers have been leveraging deep learning models to harness the structural features of coefficient matrices, aiming to predict the optimal iterative solver and preconditioner for sparse linear systems. Holloway et al. [57] introduced the application of neural networks [16] for forecasting the performance of preconditioned iterative solvers, specifically predicting their capability to effectively solve given problems. Similarly, Kuefler et al. [58] explored the use of reinforcement learning [59] to address sparse linear systems. They highlighted that reinforcement learning could not only recommend suitable solvers but also directly solve these systems without needing labeled data. Yeom et al. [17] took insights from natural language processing to create a performance vector space [60], mapping different combinations of iterative solvers and preconditioners into the space. For the prediction, they projected the sparse linear system into the vector space and used the nearest neighbor algorithm to identify the most effective combination. Yamada et al. [18] applied CNN [25] to identify patterns directly from the coefficient matrix for predicting the best preconditioner for sparse linear systems. Their method involves converting the coefficient matrix into a fixed-size, three-channel color image. CNN is then used to process this image, extracting feature maps that assist in the preconditioner prediction. Funk et al. [19] explored several deep learning models (e.g., MLP [61], FCNN [37], etc.) to automatically select appropriate iterative solvers for a given sparse linear system. More recently, Tang et al. [20] proposed the use of the GNN [36] for predicting the optimal combination of iterative solvers and preconditioners. Their motivation stems from the observation that non-zero elements of coefficient matrices can be perfectly represented as graphs, and GNN is well-suited for learning informative features from such graph representations. These deep learning based methods excel at directly learning features from the coefficient matrix. However, as the size of the matrix grows, their efficiency in processing data diminishes, and they suffer from a lack of interpretability as well. Thus, it becomes imperative to integrate the strengths of both traditional machine learning models and deep learning algorithms to achieve more efficient auto-selection of iterative solvers and preconditioners.

6. Conclusion

In this paper, we propose to solve the problem of the automatic selection of the optimal combination of iterative solvers and preconditioners for sparse linear systems by employing a multimodal machine learning framework, and then put forward MM-AutoSolver, a multimodal machine learning model to predict the optimal combination among candidate ones for a given sparse linear system. MM-AutoSolver aims to learn from numerical features and structural features and fuse these two kinds of features of different modalities to boost the prediction performance. Experimental results on the real-world sparse linear systems demonstrate the advantage of the proposed MM-AutoSolver over state-of-art baselines and the performance speedup that it can bring to the solution of large-scale sparse linear systems.

CRediT authorship contribution statement

Hantao Xiong: Writing – original draft, Methodology, Investigation. **Wangdong Yang:** Project administration, Conceptualization. **Weiqing He:** Data curation. **Shengle Lin:** Visualization, Validation. **Keqin Li:** Writing – review & editing. **Kenli Li:** Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors sincerely thank the editors and all the anonymous reviewers for their comments, which are valuable and constructive. The research was partially funded by the Major Projects of Xiangjiang Laboratory (22xj01011), the Key Program of National Natural Science Foundation of China (Grant No. U21A20461), the National Natural Science Foundation of China (Grant Nos. 61872127 and 61751204), the Research Innovation Project for Postgraduate Students of Hunan Province (No. CX20220412), and Guangdong Province Core Software Research and Development Project (No. 2024QT006).

Data availability

Data will be made available on request.

References

[1] H. Anzt, E. Boman, R. Falgout, P. Ghysels, M. Heroux, X. Li, L. Curfman McInnes, R. Tran Mills, S. Rajamanickam, K. Rupp, et al., Preparing sparse solvers for exascale computing, Philos. Trans. R. Soc. A 378 (2166) (2020) 20190053.

[2] T.A. Davis, Direct Methods for Sparse Linear Systems, SIAM, 2006.

[3] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM, 2003.

[4] M. Benzi, Preconditioning techniques for large linear systems: a survey, J. Comput. Phys. 182 (2) (2002) 418–477.

[5] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. Van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, 1994.

[6] T.A. Davis, Y. Hu, The University of Florida sparse matrix collection, ACM Trans. Math. Softw. 38 (1) (2011) 1–25.

[7] H. Zou, X. Xu, C.-S. Zhang, A survey on intelligent iterative methods for solving sparse linear algebraic equations, arXiv preprint, arXiv:2310.06630, 2023.

[8] T.M. Mitchell, Machine Learning, 1997.

[9] S. Bhowmick, V. Eijkhout, Y. Freund, E. Fuentes, D. Keyes, Application of machine learning to the selection of sparse linear solvers, Int. J. High Perform. Comput. Appl. (2006).

[10] S. Bhowmick, B. Toth, P. Raghavan, Towards low-cost, high-accuracy classifiers for linear solver selection, in: Computational Science–ICCS 2009: 9th International Conference, Baton Rouge, LA, USA, May 25-27, 2009, Proceedings, Part I, Springer, 2009, pp. 463–472.

[11] P.R. Eller, J.-R.C. Cheng, R.S. Maier, Dynamic linear solver selection for transient simulations using machine learning on distributed systems, in: 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, IEEE, 2012, pp. 1915–1924.

[12] T. Sakurai, T. Katagiri, H. Kuroda, K. Naono, M. Igai, S. Ohshima, A sparse matrix library with automatic selection of iterative solvers and preconditioners, Proc. Comput. Sci. 18 (2013) 1332–1341.

[13] E. Jessup, P. Motter, B. Norris, K. Sood, Performance-based numerical solver selection in the lighthouse framework, SIAM J. Sci. Comput. 38 (5) (2016) S750–S771.

[14] H.-B. Sun, Y.-F. Jing, X.-W. Xu, A new matrix feature selection strategy in machine learning models for certain Krylov solver prediction, J. Classif. (2024) 1–18.

[15] Y. Zabegaev, E. Keilegavlen, E. Iversen, I. Berre, Automated linear solver selection for simulation of multiphysics processes in porous media, Comput. Methods Appl. Mech. Eng. 426 (2024) 117031.

[16] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[17] J.-S. Yeom, J.J. Thiagarajan, A. Bhatele, G. Bronevetsky, T. Kolev, Data-driven performance modeling of linear solvers for sparse matrices, in: 2016 7th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), IEEE, 2016, pp. 32–42.

[18] K. Yamada, T. Katagiri, H. Takizawa, K. Minami, M. Yokokawa, T. Nagai, M. Ogino, Preconditioner auto-tuning using deep learning for sparse iterative algorithms, in: 2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW), IEEE, 2018, pp. 257–262.

[19] Y. Funk, M. Götz, H. Anzt, Prediction of optimal solvers for sparse linear systems using deep learning, in: Proceedings of the 2022 SIAM Conference on Parallel Processing for Scientific Computing, SIAM, 2022, pp. 14–24.

[20] Z. Tang, H. Zhang, J. Chen, Graph neural networks for selection of preconditioners and Krylov solvers, in: NeurIPS 2022 Workshop: New Frontiers in Graph Learning, 2022.

[21] R.L. Burden, Numerical Analysis, Brooks/Cole Cengage Learning, 2011.

[22] M. Götz, H. Anzt, Machine learning-aided numerical linear algebra: convolutional neural networks for the efficient preconditioner generation, in: 2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA), IEEE, 2018, pp. 49–56.

[23] L.-P. Morency, P.P. Liang, A. Zadeh, Tutorial on multimodal machine learning, in: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts, 2022, pp. 33–38.

[24] C. Chen, K. Li, C. Zhongyao, F. Piccialli, S.C. Hoi, Z. Zeng, A hybrid deep learning based framework for component defect detection of moving trains, IEEE Trans. Intell. Transp. Syst. 23 (4) (2020) 3268–3280.

[25] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551.

[26] S.J. Wright, Numerical Optimization, 2006.

[27] M.R. Hestenes, E. Stiefel, et al., Methods of Conjugate Gradients for Solving Linear Systems, vol. 49, NBS, Washington, DC, 1952.

[28] Y. Saad, M.H. Schultz, Gmres: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869.

[29] J.W. Ruge, K. Stüben, Algebraic multigrid, in: Multigrid Methods, SIAM, 1987, pp. 73–130.

[30] R. Fletcher, Conjugate gradient methods for indefinite systems, in: Numerical Analysis: Proceedings of the Dundee Conference on Numerical Analysis, 1975, Springer, 2006, pp. 73–89.

[31] H.A. Van der Vorst, Bi-cgstab: a fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 13 (2) (1992) 631–644.

[32] T. Baltrušaitis, C. Ahuja, L.-P. Morency, Multimodal machine learning: a survey and taxonomy, IEEE Trans. Pattern Anal. Mach. Intell. 41 (2) (2018) 423–443.

[33] X. Zou, L. Zhou, K. Li, A. Ouyang, C. Chen, Multi-task cascade deep convolutional neural networks for large-scale commodity recognition, Neural Comput. Appl. 32 (2020) 5633–5647.

[34] X. Liang, Y. Qian, Q. Guo, H. Cheng, J. Liang, Af: an association-based fusion method for multi-modal classification, IEEE Trans. Pattern Anal. Mach. Intell. 44 (12) (2022) 9236–9254, https://doi.org/10.1109/TPAMI.2021.3125995.

[35] H. Tao, C. Hou, Y. Qian, J. Zhu, D. Yi, Latent complete row space recovery for multi-view subspace clustering, IEEE Trans. Image Process. 29 (2020) 8083–8096, https://doi.org/10.1109/TIP.2020.3010631.

[36] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2020) 4–24.

[37] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

[38] Z. Xie, G. Tan, W. Liu, N. Sun, Ia-spgemm: an input-aware auto-tuning framework for parallel sparse matrix-matrix multiplication, in: Proceedings of the ACM International Conference on Supercomputing, 2019, pp. 94–105.

[39] Z. Xie, G. Tan, W. Liu, N. Sun, A pattern-based spgemm library for multi-core and many-core architectures, IEEE Trans. Parallel Distrib. Syst. 33 (1) (2021) 159–175.

[40] A. Buluç, J.T. Fineman, M. Frigo, J.R. Gilbert, C.E. Leiserson, Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks, in: Proceedings of the Twenty-First Annual Symposium on Parallelism in Algorithms and Architectures, 2009, pp. 233–244.

[41] V. Eijkhout, E. Fuentes, A standard and software for numerical metadata, ACM Trans. Math. Softw. 35 (4) (2009) 1–20.

[42] F. Hecht, New development in FreeFEM++, J. Numer. Math. 20 (3–4) (2012) 251–265, https://freefem.org/.

[43] H. Jasak, A. Jemcov, Z. Tukovic, et al., OpenFOAM: a C++ library for complex physics simulations, in: International Workshop on Coupled Methods in Numerical Dynamics, 2007, pp. 1–20.

[44] S. Balay, S. Abhyankar, M.F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E.M. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, J. Faibussowitsch, W.D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M.G. Knepley, F. Kong, S. Kruger, D.A. May, L.C. McInnes, R.T. Mills, L. Mitchell, T. Munson, J.E. Roman, K. Rupp, P. Sanan, J. Sarich, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, J. Zhang, PETSc web page, https://petsc.org/, 2023.

[45] J. Xia, C. Cheng, X. Zhou, Y. Hu, P. Chun, Kunpeng 920: the first 7-nm chiplet-based 64-core arm soc for cloud services, IEEE MICRO 41 (5) (2021) 67–75.

[46] T. Chen, C. Guestrin, Xgboost: a scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.

[47] Y. Freund, L. Mason, The alternating decision tree learning algorithm, in: ICML, vol. 99, 1999, pp. 124–133.

[48] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., TensorFlow: a system for large-scale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 16, 2016, pp. 265–283.

[49] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (1995) 273–297.

[50] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, in: European Conference on Computational Learning Theory, Springer, 1995, pp. 23–37.

[51] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.

[52] G.I. Webb, E. Keogh, R. Miikkulainen, Naïve Bayes, Encycl. Mach. Learn. 15 (1) (2010) 713–714.

[53] V. Eijkhout, E. Fuentes, Machine learning for multi-stage selection of numerical methods, in: New Advances in Machine Learning, INTECH, 2010, pp. 117–136.

[54] K. Sood, Iterative solver selection techniques for sparse linear systems, Ph.D. thesis, University of Oregon, 2019.

[55] P. Motter, K. Sood, E. Jessup, B. Norris, Lighthouse: an automated solver selection tool, in: Proceedings of the 3rd International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering, 2015, pp. 16–24.

[56] K. Sood, B. Norris, E. Jessup, Comparative performance modeling of parallel preconditioned Krylov methods, in: 2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEEE, 2017, pp. 26–33.

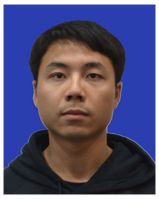
[57] A. Holloway, T.-Y. Chen, Neural networks for predicting the behavior of preconditioned iterative solvers, in: Computational Science–ICCS 2007: 7th International Conference, Beijing, China, May 27-30, 2007, Proceedings, Part I 7, Springer, 2007, pp. 302–309.

[58] E. Kuefler, T.-Y. Chen, On using reinforcement learning to solve sparse linear systems, in: Computational Science–ICCS 2008: 8th International Conference, Kraków, Poland, June 23-25, 2008, Proceedings, Part I 8, Springer, 2008, pp. 955–964.

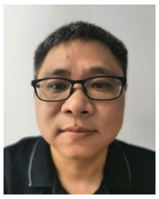
[59] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: a survey, J. Artif. Intell. Res. 4 (1996) 237–285.

[60] K. Erk, S. Padó, A structured vector space model for word meaning in context, in: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, 2008, pp. 897–906.

[61] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.



Hantao Xiong is currently working toward the Ph.D. degree in Computer Science and Technology with the College of Information Science and Electronic Engineering, Hunan University, Changsha, China. His research interests include parallel computing, numerical computation, and artificial intelligence.



Wangdong Yang received the Ph.D. degree in computer science from Hunan University, China, and the M.S. degree in computer science from Central South University, China. He is a professor of computer science and technology at Hunan University, China. His research interests include modeling and programming for heterogeneous computing systems, parallel and distributed computing, and numerical computation. He has published more than 60 papers in International conferences and journals. He is currently served on the editorial boards of IEEE Internet of Things Journal.

Weiqing He received the B.E. degree in 2022 from College of Computer Science and Technology, Hainan University, China. He is currently pursuing the PhD degree with the College of Information Science and Engineering, Hunan University, China. His research interests include parallel computing, artificial intelligence, and numerical computation.

Shengle Lin is currently working toward the Ph.D. degree in Computer Science and Technology with the College of Information Science and Electronic Engineering, Hunan University, Changsha, China. He is now working on a one-year joint Ph.D. program at Agency for Science, Technology and Research (A*STAR), Singapore. His research interests include high-performance computing, parallel computing, numerical computation and artificial intelligence.

Dr. Keqin Li is a SUNY Distinguished Professor of computer science with the State University of New York. He is also a National Distinguished Professor with Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has authored or coauthored more than 780 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He holds over 60 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top 10 most influential scientists in distributed computing based on a composite indicator of Scopus citation database. He has chaired many international conferences. He is currently an associate editor of the ACM Computing Surveys and the CCF Transactions on High Performance Computing. He has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.

Kenli Li received the Ph.D. degree in computer science from Huazhong University of Science and Technology, China, in 2003. and the M.S. degree in mathematics from Central South University, China, in 2000. He was a visiting scholar at University of Illinois at Urbana-Champaign from 2004 to 2005. He is a full professor of computer science and technology at Hunan University. The main research fields are parallel and distributed processing, supercomputing and cloud computing, high-performance computing for big data and artificial intelligence, etc. He has published more than 300 papers in international conferences and journals. He is currently served on the editorial boards of IEEE Transactions on Computers. He is an outstanding member of CCF and a member of the IEEE.