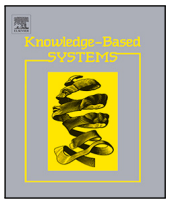
Knowledge-Based Systems 325 (2025) 113946

Contents lists available at ScienceDirect

Knowledge-Based Systems

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



SDRM: A truncated SVD-based dimensionality reduction approach to efficient edge inference in Multi-head Attention Networks

Yuze Du [a], Xiaogang Wang [a,*], Haokun Chen [a], Chenfeng Zhang [a], Jian Cao [b], Rajkumar Buyya [c]

[a] School of Electronic and Information, Shanghai Dianji University, Shanghai, 201306, Pudong New Area District, China
[b] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, Minghang District, China
[c] CLOUDS Laboratory, School of Computing and Information Systems, The University of Melbourne, Parkville, VIC 3010, Melbourne, Australia

ARTICLE INFO

Keywords:
Edge inference
Multi-head attention networks
Truncated SVD
Low-rank approximation

ABSTRACT

In large-scale industrial product detection, the optimization of computing and storage resources becomes a key issue. Most of the existing methods focus on the feature compression of a single dimension or a specific scene, but less consideration is given to how to perform feature selection and compression in high-dimensional data and multi-task scenarios during the process of edge inference. To this end, a Smart Dimensionality Reduction Model (SDRM) with a novel *AdaMatrix* optimization is proposed to enhance the processing efficiency through dynamic compression and dimensionality reduction of 3D tensor. This model adopts a low-rank decomposition strategy based on truncated singular value decomposition (t-SVD), and combines regularization and gradient descent optimization to achieve progressive data compression. A collaborative optimization framework of t-SVD and gradient descent is constructed to optimize the homogeneous distributed multi-head attention mechanism. It reduces storage requirements while ensuring data accuracy, and utilizes the multi-head attention mechanism to enhance parallel processing capability. Theoretical proofs and various experiments with multifaceted metrics show that the proposed SDRM model outperforms existing related methods in terms of compression ratio, computational complexity and data processing efficiency in electronic component defect detection scenarios.

1. Introduction

Industrial product detection plays a vital role in the field of automation and quality control, especially in manufacturing, where it is critical to ensure that products meet specifications and performance requirements. In the process of detection, traditional methods such as machine vision technology based on image processing are often limited by large data volumes and low detection efficiency. At the same time, traditional methods lack flexibility and accuracy in processing data, which cannot meet industrial requirements. These problems are manifested in two main aspects: the limited ability to process large-scale and high-dimensional data, and the insufficient ability to quickly and accurately identify defects in real-time or near-real-time environments.

In recent years, the Transformer model technology has shown better potential in vision tasks due to its self-attention mechanism, which can globally process image data. The structure of Transformer allows features to be captured at different scales [1], adapted to pixel-level prediction tasks, and provides novel solutions in defect detection [2]. Although the Transformer model has demonstrated significant performance in industrial product inspection, there are still some performance gaps in current implementations due to a lack of local information about retained spatial structure, and challenges in model training and convergence.

The computational complexity of multi-head attention mechanisms is very high when dealing with large data sets or high-dimensional inputs. Its temporal and spatial complexity has led to a surge in resource requirements [3–5]. When handling long sequences or large matrices of inputs, distributed sequence parallelism methods have been proposed [6,7] to distribute computational loads across multiple GPUs to process longer sequence data. However, the amount of computation required to compute the attention score matrix and perform matrix multiplication rises exponentially [6]. Meanwhile, there is also a lot of feature redundancy in high-dimensional data. These redundant features not only increase the computational complexity but also may lead to a decline in the model's performance [7–9].

* Correspondence to: School of Electronic and Information, Shanghai Dianji University, Pudong New Area District, Shanghai, China.
E-mail addresses: duyz@st.sdju.edu.cn (Y. Du), wangxg@sdju.edu.cn (X. Wang), chenhk@st.sdju.edu.cn (H. Chen), zhangcf@st.sdju.edu.cn (C. Zhang), cao-jian@cs.sjtu.edu.cn (J. Cao), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.knosys.2025.113946
Received 31 March 2025; Received in revised form 22 May 2025; Accepted 6 June 2025
Available online 21 June 2025

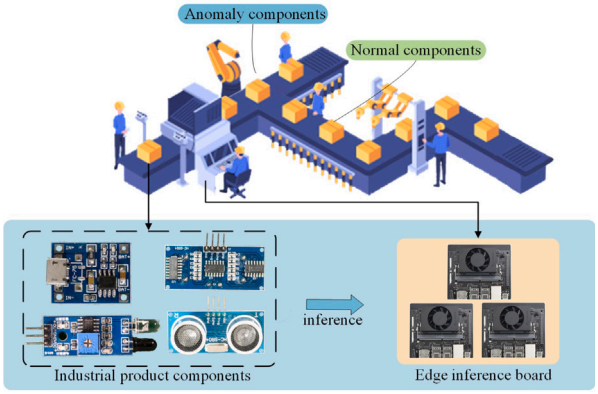


Fig. 1. Utilizing distributed parallel processing for images of industrial components in an assembly Line.

For tensor decomposition and parallel computing, innovative solutions have been proposed. Online tensor decomposition improves accuracy in dynamic data flow scenarios [10], but it is computationally complex and sensitive to hyperparameters. Through the high-dimensional tensor parallel design, the communication cost is effectively reduced and the performance is improved [11,12], but the implementation is complicated and depends on high-speed hardware, and the training cost is high and the interpretability is poor. Despite the improvement of computing efficiency, problems such as computing cost, hardware adaptation and theoretical depth still need to be solved.

In a nutshell, industrial product detection faces the triple challenge of real-time processing of large-scale data, high-precision detection and energy efficiency optimization. Toward this end, we propose a Smart Dimensionality Reduction Model (SDRM) with a fresh *AdaMatrix* optimization in industrial assembly lines, which effectively copes with above problems through intelligent downscaling techniques. The detection scenario for the model is shown in Fig. 1. SDRM adopts the key truncated singular value decomposition (t-SVD) for low-rank approximations and distributed multi-head attention mechanism for efficient processing of industrial detection image data, thus significantly improves the speed and accuracy of data processing by decreasing the number of dimensions in the data processing process, while reducing the demand for computational resources and making industrial detection systems more economical.

The main contributions of this paper are summarized as follows:

1. **Applying low-rank approximations to multi-head attention mechanism.** By establishing the SDRM with a new *AdaMatrix* optimization, the dimensionality of each head in the multi-head attention mechanism is effectively reduced, which not only decreases the number of parameters in the model but also falls memory usage and computational complexity.
2. **Designing the distributed multi-head attention mechanism for data processing.** The proposed model improves the multi-head attention mechanism by using a tensor-based approach, and replacing the traditional dot product attention with a multilinear form, which further reduces the model size, and obtains higher computational efficiency through parameter sharing.
3. **Enhancing the generalization ability of the proposed model.** By adopting t-SVD truncation for low-rank approximation, SDRM can not only reduce the dimension and computational complexity of the model, but also reduce the overfitting of the model. The truncation process only retains the core features in the data, and SDRM can remove noise and redundant information in the data.
4. **Conducting effective experimental validation on real hardware inference boards.** Based on experimental results on open source datasets and our real hardware inference boards, SDRM is not only able to approach real images in the product detection, but also shows superior performance in quantitative and qualitative evaluation. It proves effectiveness in multi-head attention distributed parallel processing.

The rest of this paper is organized as follows. Section 2 outlines the related work. In Section 3, the basic structure and formula of SDRM proposed in this paper are expounded. describes the process of scheduling tasks to edge devices and calculating multi-head attention. In Section 5, some comparative experiments are illustrated. Finally, Section 6 draws the conclusion and future work.

2. Related work

The multi-head attention mechanism is key to the Transformer architecture, enabling models to focus on different parts of the input sequence at the same time. The complexity of traditional perfect attention mechanisms increases with the quadratic length of the sequence, which brings computational challenges. This section explores various tensor parallelism and low-rank decomposition methods aimed at optimizing memory and computational efficiency.

2.1. The multi-head attention mechanism and its tensor parallel

Most of the work has recently focused on improving the efficiency of data processing in distributed systems, optimizing resource allocation, and improving the performance of large-scale model training. Khalesi's model [13] and Li's flexible matrix multiplication framework [14] enhance multitasking efficiency through dynamic data allocation strategies. Jin's edge co-training system [15] for balanced optimization of computation and communication in IoT scenarios. These studies achieve an important breakthrough at the computing paradigm level. In terms of model training acceleration, Dash's LLM distributed training scheme [16] and Lai's Merak 3D parallel framework [17] effectively mitigate giant model resource consumption. Liu's Colossal-Auto [18] linearizes computational graphs by automating parallel strategies. Chen and Liang's study optimizes the parallelization system at the method level and improves the computational efficiency [19,20]. Above all, these studies have driven the development of distributed computing techniques, especially in applications that handle large-scale data and model training, significantly improving efficiency and performance.

The multidimensional innovation of the Transformer framework is reflected in the synergistic breakthroughs of the underlying components and cross-domain applications. Multi-head attention enhances feature capture by parallel parsing in subspace [1]. The Transformer framework has shown its potential in cross-domain applications, and the communication optimization strategy in [21] for dynamic tensor block scheduling has led to an increase in distributed data bandwidth utilization. Han [22] breaks through the traditional convolutional limitations by hierarchical spatial modeling paradigm. The advantages of visual Transformer in object detection were found in the subsequent research [17,23]. And the end-to-end model based on semantic segmentation reaches the frontier metrics [24]. In combination with the architecture auto-search algorithm, the parameter efficiency of the Vision Transformer (ViT) is further optimized [25]. The temporal prediction direction reduces the prediction error through frequency enhancement and attention mechanisms [7,26]. The ViT improves the performance of image processing by segmenting an image into a sequence of blocks and applying the pure Transformer architecture directly [27]. Despite the breakthroughs in scalability that these distributed schemes go through, there are still common problems such as underutilization of memory bandwidth and high percentage of time consuming communication synchronization [18,21,26].

In recent years, the edge-optimized attention mechanism has significantly improved the performance of visual tasks in resource-constrained

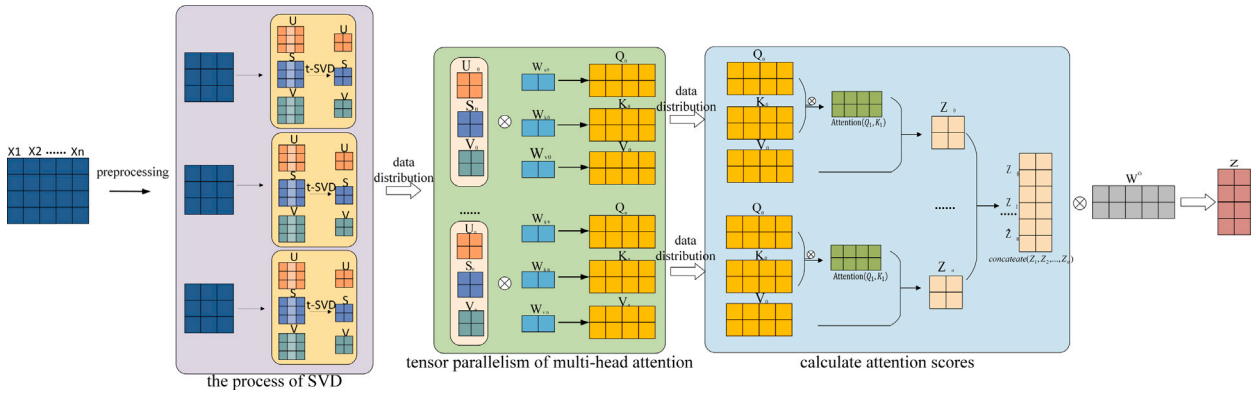


Fig. 2. This is the process of the proposed SDRM which performs low-rank approximation through the t-SVD process. The detailed process of t-SVD is shown in Fig. 3. By performing low-rank approximation of data, SDRM can speed up data processing.

scenarios through structural innovation and the reform of evaluation indicators. Song proposes the C2f-FCA module [28], combined with multi-scale context attention and hardware adaptability design, to improve the real-time accuracy in the detection of small targets in unmanned aerial vehicles. EdgeViTs [29] inherits from attention and convolution through the local-all-local (LGL) bottleneck, significantly reducing the latency of ViT on mobile devices for the first time. BAAN [30] adopts the axial attention decomposition and boundary regularization module to solve the imbalance problem in pavement detection with low complexity.

The existing parallel frameworks of dynamic tensors still have certain limitations. High computational complexity and insufficient dynamic adaptability of static data allocation strategies. The feature collaboration mechanism is weak, and the static tensor transformation and lightweight strategy lead to certain deficiencies in the model's capabilities.

2.2. Low-rank approximation of the tensor

The low-rank approximation technique reduces model complexity through matrix decomposition and tensor representation. It enables parameter scale compression in convolutional neural networks [31] and optimizes unilateral data processing efficiency [32]. Wang's research [33] in the field of multidimensional data complementation demonstrates that low-rank tensor representations based on coupled nonlinear transformations can improve the quality of complex image restoration. At the theoretical level, Sarlos' complexity analysis explores [34] the fact that matrix computation with varying principal elements leads to a decrease in the convergence speed of the algorithms, but the feature retention and computational overhead can be balanced by a dynamic rank-adaptive strategy [35].

Low rank approximation have wide applicability in different fields, and it plays a greater role in dealing with large scale and high dimensional data. These studies have proposed different technical schemes, but the computational complexity is high and the essential characteristics of the data are insufficient. The proposed SDRM in this paper utilizes an efficient dimensionality reduction strategy to reduce computation and storage requirements, while it is able to retain more data features.

2.3. Tensor decomposition and operational optimization

Tensor decomposition technology achieves computational performance breakthroughs in multiple domains through singular value optimization strategies. Where the stochastic fixed-precision algorithm reduces the decomposition elapsed time [36] and the hierarchical parallel architecture greatly improves the efficiency in distributed systems [37]. In the field of machine learning, Amiridi et al. [38] use feature selection to improve the accuracy of classification tasks, and Razin et al. [39] avoided overfitting through implicit regularization. The Tesseract framework [11] greatly improves resource utilization through multidimensional parallelization. Cheng et al. [40] propose the dynamic load balancing via adaptive tensor parallel systems. And the MERIT method [41] makes the visual processing memory footprint reduced by tensor transformation. Tensor decomposition has a wide range of applications in real-time computing. The Nesterov-accelerated CP decomposition algorithm [10] improves the speed of iterative convergence. Incremental decomposition techniques [42] greatly improve the efficiency of processing spatio-temporal data. Fawzi et al. [12] combine reinforcement learning with traditional matrix multiplication algorithms to provide new methods for machine learning to solve traditional tensor operational optimization problems. Existing optimization frameworks currently studied are deficient in dynamic adaptation, real-time efficiency and feature synergy.

As for the existing research on mature tensor decomposition methods. Incremental singular value decomposition gradually updates the decomposition to adapt to dynamic data streams. However, its block update mechanism brings relatively high computational complexity. Tucker decomposition utilizes high-order singular value decomposition to decompose tensors into core tensors and factor matrices. But it requires a large amount of storage and computing costs. CP decomposition represents tensors as the sum of the ranks and tensors, and is suitable for data with simple patterns. However, when dealing with complex structures, its convergence speed is slow and the accuracy decreases.

Distributed tensor decomposition is widely applied in scenarios such as intelligent perception in the Internet of Things and cross-device data fusion. Currently, it is facing the communication and computing challenges of the surging multi-source data of edge devices. The FlyCom^2 framework [43] realizes stream tensor decomposition through random sketch compression and MIMO aerial computing, and verifies its advantages of low complexity and high precision in scenarios with large feature value gaps. DiaMASTD [44] proposes a distributed framework for the dynamic multi-dimensional streaming tensor decomposition problem. This model designs two heuristic load balancing methods. However, there are still bottlenecks in resource consumption in large-scale scenarios. Korthikanti [45] proposes that by integrating sequence parallelism and tensor parallelism, the memory of large-scale parameter Transformers could be reduced and the computational utilization rate could be improved. The HO-QRTP method [46] conducts a parallel low-rank approximation strategy based on QR decomposition, and realizes distributed computing through tensor expansion and logical recombination. However, its accuracy is relatively weak, and the communication overhead is significant in high-order tensor scenarios.

The existing tensor decomposition methods face multiple challenges such as dynamic adaptability, real-time efficiency and feature collaboration in industrial product inspection. The high computational

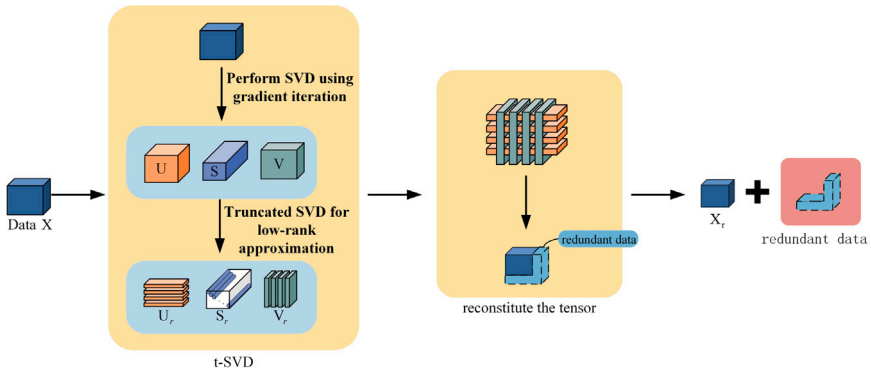


Fig. 3. This is the process of t-SVD which preserves most of the features of the matrix by computing the singular value decomposition and retaining the eigenvectors corresponding to larger eigenvalues. Such a low-rank approximation is able to reduce the data dimensionality, while the removed redundant data will not affect the data processing.

complexity and storage cost of incremental SVD and Tucker limit real-time response to dynamic data streams and multi-modal feature collaborative modeling. Meanwhile, the convergence speed of the existing methods is insufficient, and it is difficult to meet the detection requirements of high-speed production lines. The static feature association mechanism and the feature interaction caused by the lightweight strategy weaken the sensitivity to minor defects. In edge scenarios, the above-mentioned problems are further limited by issues such as poor hardware compatibility.

This paper proposes the Smart Dimensionality Reduction Model (SDRM), which uses truncated singular value decomposition technology t-SVD to compress high-dimensional data, reduces the resources required for storage and processing, and speeds up data transmission and processing while decreasing storage costs. t-SVD is used to extract key features from complex data to improve the efficiency and effect of the learning algorithms. By reducing data dimensions and improving data quality, the model can significantly improve the performance of various machine learning algorithms, and the data after dimensionality reduction can reduce the risk of overfitting and improve the generalization ability of the model. SDRM combines low-rank approximation and dynamic incremental update to reduce computational complexity, and integrates gradient accelerated convergence with computational truncation optimization to adapt to dynamic data streams. Meanwhile, the adaptive truncation threshold and distributed gradient are utilized to balance the accuracy and energy consumption.

3. System model

In order to expedite data processing efficiency and decease data redundancy in large dimension data, we propose the SDRM model which uses the tensor decomposition via gradient iterative learning to reduce the dimensionality of multi-head attention mechanism. Fig. 2 depicts the general framework of the SDRM model which includes the singular value decomposition (SVD), the tensor parallelism of multi-head attention and the attention score calculation.

3.1. Singular value decomposition for low-rank approximation

After entering the image data, we analyze its color space and transform it into tensor form. Then the data is distributed to different edge devices, and the t-SVD process of gradient descent is carried out to complete the low-rank approximation of the data. Finally, the results are output by tensor parallelism of multi-head attention mechanism and recombination of attention scores.

For a given three-dimensional tensor input $X \in \mathbb{R}^{b\times t\times d}$, X is the three-dimensional tensor data, where b is the batch size, t is the time step, and d is the characteristic dimension, expressed as follows:

$$X = \begin{bmatrix} X_0 \\ \cdots \\ X_i \\ \cdots \\ X_b \end{bmatrix}, X_i = \begin{bmatrix} \hat{x_0} \\ \cdots \\ \hat{x_i} \\ \cdots \\ \hat{x_t} \end{bmatrix},$$

$$\begin{bmatrix} \hat{x_0} \\ \cdots \\ \hat{x_i} \\ \cdots \\ \hat{x_t} \end{bmatrix} = \begin{bmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,d} \\ \cdots & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{t,0} & x_{t,1} & \cdots & x_{t,d} \end{bmatrix}$$

After data preprocessing, singular value decomposition is performed on the input data X. According to Algorithm 1, the tensor is expanded into a matrix $b \times (t \cdot d)$. That is, the time steps and feature dimensions of each sample are combined into one large feature vector. We chunk the input data X to get X_i for parallel singular value decomposition.

$$Y_i \approx USV^T \tag{1}$$

The singular value decomposition proceeds as indicated in Eq. (1), where we decompose the tensor expansion matrix Y_i into U, S, V matrices. For ease of presentation, we set up $\Psi \in U, S, V$.

Fig. 3 illustrates the process of the truncated singular value decomposition. We obtain a low-rank approximation by a truncation method that preserves only the largest r singular values. According to the singular value matrix S, we can obtain singular value $\sigma_1, \sigma_2, \ldots, \sigma_n$ of X. The singular value matrix is truncated, and only r singular values and corresponding singular vectors are kept. The SVD is truncated to retain the largest r singular values. We can use the energy retention strategy to determine the value of r. To calculate the total energy information, we need find the sum of squares of all singular values. The sum of the squares is added until the sum of the squares of the r singular values reaches $\phi\%$ of the total value, as depicted in Eqs. (2) and (3).

$$E = \sum_{i=1}^{n} \sigma_i^2 \tag{2}$$

$$\sum_{i=1}^{r} \sigma_i^2 \geq \phi\% \times E \tag{3}$$

After weighting matrix singular value decomposition, we can acquire truncated matrices U_r, S_r, V_r^T. When the redundant data is reduced, the data process of low rank approximation is just completed. This process decreases the data dimension, which makes the training and inference process of the model more efficient, and significantly reduces the computational complexity. The reduced-rank data saves storage space due to decreased dimensionality, and we give Theorem 1 and its proof.

Theorem 1. *When processing image data with large data size, the storage space required for the matrix can be lessened due to decreased redundant data of the images through the method of t-SVD.*

Proof. We can build a mathematical framework from the perspective of mapping space and matrix dimensions. There is an input three-dimensional tensor $X \in \mathbb{R}^{b\times t\times d}$. According to Algorithm 1, the tensor is expanded into a matrix $b \times (t \cdot d)$.

Set Y be the two-dimensional matrix $Y \in \mathbb{R}^{(b\cdot t)\times d}$ reconstructed by X. SVD is applied to Y. We can get the matrix U, S, V, where $U \in \mathbb{R}^{(b\cdot t)\times r}, S \in \mathbb{R}^{r\times r}, V \in \mathbb{R}^{r\times t}$. The storage requirements for raw data that has not been processed are:

$$\mathrm{S}_{\text{original}} = b \cdot t \cdot d \cdot \text{size\_of(element)}$$

The matrix after t-SVD processing only retains r eigenvectors with the largest eigenvalues, and the storage requirements are as follows:

$$\mathrm{S}_{\text{t-SVD}} = ((b \cdot t) \cdot r + r + r \cdot d) \times \text{size\_of(element)}$$

By comparison, we can get the difference in storage requirements before and after data compression. Because of the truncated $r < d$ and $r < (b\cdot t)$, the storage requirements of SVD are significantly smaller than the original storage requirements. Storage compression ratio is shown as follows.

$$\mathrm{CR} = \frac{\mathrm{S}_{\text{original}}}{\mathrm{S}_{\text{t-SVD}}} = \frac{btd}{btr + r + rd} > 1$$

However, when considering the boundary case, we find that truncation does not occur for $r = d$ and the storage requirement is $btr+r+rd = btd + d + d^2$, which is instead higher than the original storage. At this point the compression ratio is $\frac{btd}{btd+d+d^2} < 1$, indicating that $r < d$ is a prerequisite for this method. When the value of r is small (we take $r = 1$ for ease of computation), the storage requirement is $bt + 1 + d$, and the compression ratio is $\frac{btd}{bt+1+d} \approx d$. The compression is significant at this point, but a too low rank may bring a big reconstruction error. □

From Theorem 1, it can be known that the compression ratio $\mathrm{CR} = \frac{btd}{btr+r+rd}$. Similarly, according to the energy retention strategy, We can get information retention $\phi = \frac{\sum_{i=1}^{r}\sigma_i^2}{\sum_{i=1}^{\min(b,t,d)}\sigma_i^2}$. By setting the minimum acceptable average precision $mAP(\phi)$ according to the task requirements, the minimum precision retention rate $\phi_{\min} = \phi_0 - \frac{1}{k}\ln(1 - \frac{90\%}{mAP(\phi)})$ can be obtained. In this study, in order to ensure that the loss of defect identification accuracy is not significant, we limit the information retention rate.

t-SVD, through truncating singular values and matrix block decomposition, has lower computational complexity and storage requirements than existing tensor decomposition methods (Tucker decomposition and CP decomposition). The t-SVD is an adaptive adjustment model based on truncated rank, which optimizes the balance of computing, communication and storage.

3.2. AdaMatrix gradient iteration for rank reduction

In the calculation of large-scale data, directly solving the singular value decomposition matrix will cause a huge amount of computation. This algorithm uses gradient iteration to solve the decomposed U, S, V matrix, which reduces the required computation amount. SDRM combines Adaptive Moment Estimation [47] and Nesterov Accelerated Gradient [48] for gradient iteration of t-SVD. Then, a novel iteration optimization method named as *AdaMatrix* is proposed, which is an iterative update method for singular vector matrix based on Adam optimizer. We make a forecast update, which can provide more accurate gradient information closer to the optimal point, to the parameters based on momentum. The optimal solution of the loss function can be reached faster when dealing with large-scale data. *AdaMatrix* incorporates a dynamic adjustment strategy for the momentum term for the low-rank decomposition problem and designs element-wise adaptive weights for the regularization term of singular matrices. To find the U, V^T that could best represents the raw data, we can define the loss function as Eq. (4). Using gradient descent to optimize U_k, S_k, V_k, and perform further regularization $\lambda_1\|S_k\|_1$ to S_k to compress model. Where $\|U\|_*$ and $\|V\|_*$ matrices are respectively U and V nuclear norms, which are used to control low rank of matrix. We prove the convergence of *AdaMatrix* in Theorem 3.

$$L = \left\|Y - USV^T\right\|_F^2 - \lambda_1\|S\|_1 + \lambda_2\|U\|_* + \lambda_3\|V\|_* \tag{4}$$

Eqs. (5), (6) and (7) uses Hadamard product (i.e., a type of matrix calculation) to calculate the partial derivatives of L with respect to the three matrices U, S and V. Using this approach allows each element to be individually weighted and can be adjusted to the actual problem.

$$\frac{\partial L}{\partial U} = 2\left((USV^T - Y) \odot W\right)V_k S_k^T + \lambda_2\|U\|_* \tag{5}$$

$$\frac{\partial L}{\partial S} = 2U^T(USV^T - Y)V + \lambda_1 \cdot \mathrm{sign}(S) \tag{6}$$

$$\frac{\partial L}{\partial V} = 2\left((USV^T - Y)^T \odot W\right)US + \lambda_3 \cdot \|V\|_* \tag{7}$$

In Eq. (5), the first term is the reconstruction error gradient, and W is the element-by-element weight matrix. The element-by-element weight matrix assigns an independent weight to each element in the matrix during the gradient calculation process. The value of W is dynamically determined through the evaluation of data feature importance and gradient iterative optimization. According to the core strategy of the energy retention strategy in Eqs. (2) and (3), we initialize W based on the energy distribution. For each singular value σ_i, calculate its normalized energy as shown in Eq. (8). w_i reflects the importance of the characteristic necklace corresponding to the ith singular value in the total retained energy. We map w_i to the elements of the corresponding matrices U and V. The value of the W element in the high-energy feature region (i.e., the first r singular values retained) is larger, and its reconstruction error contributes more significantly to the gradient. The value of the W element in the low-energy or noisy region is relatively small, which suppresses its gradient update and avoids the interference of redundant information on the iteration.

$$w_i = \frac{\sigma_i^2}{\sum_{j=1}^{r}\sigma_j^2} \tag{8}$$

At the same time, *AdaMatrix* introduces adaptive learning rate to improve computing efficiency. It can dynamically adjust the learning rate and fit it to different singular value components. *AdaMatrix* performs the update of first-order moments m and second-order moments υ in gradient descent, as described in Eqs. (9) and (10). In this way, we achieve smooth gradient update and adaptive learning rate by weighted moving average of first-order and second-order moments, as described in Eq. (11).

$$m_\Psi = \beta_1 m_\Psi + (1 - \beta_1)\nabla_{\tilde{\Psi}} L \tag{9}$$

$$\upsilon_\Psi = \beta_2 \upsilon_\Psi + (1 - \beta_2)(\nabla_{\tilde{\Psi}} L)^2 \tag{10}$$

$$\Psi \leftarrow \Psi - \eta\frac{\hat{m}_\Psi}{\sqrt{\hat{\upsilon}_\Psi} + \epsilon} + m\Delta\Psi_{prev} \tag{11}$$

Learning rate η is the key to control each parameter update, and proper learning rate can ensure the model convergence and stability. m is the momentum coefficient, and $\Delta\Psi_{\text{prev}} \in \Delta U_{\text{prev}}, \Delta S_{\text{prev}}, \Delta V_{\text{prev}}$ represents the parameter update of the last iteration, that is, the momentum of the last round. The introduction of momentum from the previous round can accelerate the process and reduce oscillations during successive gradient descent, and we further present Theorem 2 and its proof.

Theorem 2. *The computational efficiency of the singular value decomposition algorithm optimized by using gradient iteration in SDRM is much higher than that of the direct computational singular value decomposition method when the data size is large, and the truncated and retained matrix dimension r is much smaller than the original dimension n.*

Proof. We can make a comparative analysis by comparing the complexity of the two computational methods.

For the generalization of this proof, we set the dimension of the input matrix X to $m \times n$, and calculate the covariance $X^T X$. Since this is a multiplication of $m \times n$ matrices, the complexity is $O(mn^2)$

$$X^T X = \begin{pmatrix} x_{11}x_{11} & x_{21}x_{12} & \cdots & x_{m1}x_{1n} \\ x_{12}x_{21} & x_{22}x_{22} & \cdots & x_{m2}x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1n}x_{m1} & x_{2n}x_{m2} & \cdots & x_{mn}x_{mn} \end{pmatrix}$$

An eigenvalue decomposition on $X^T X$ is then performed to obtain $X^T X = V \Lambda V^T$. $V \in \mathbb{R}^{n \times n}$ is the right singular matrix containing the right singular vectors of X. Λ is the diagonal matrix containing eigenvalues of X. Eigenvalue decomposition requires the calculation of eigenvalues of $n \times n$ matrices, so the complexity of eigenvalue decomposition is $O(n^3)$. S^{-1} is the inverse of the singular value matrix, and the matrix $U = XVS^{-1}$ is computed from the already obtained V and S^{-1}.

$$S^{-1} = \begin{pmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sigma_n} \end{pmatrix}$$

We need to conduct matrix multiplication of X with two matrices of dimension $n \times n$, so the computational complexity of this process is $O(mn^2)$. According to the above proof, the computational complexity of direct SVD calculation is about $O(mn^2)$.

In SVD using ordinary gradient iteration, we approximate the best low-rank approximation of X through iterative optimization, $X \approx USV$. Where U is a $m \times k$ matrix, S is a $k \times k$ matrix, and V^T is a $k \times n$ matrix.

Initializing U, S, V, we can obtain the computational complexity $O(mk + nk + k)$ based on the dimensions of the matrix. According to the matrix dimension, the complexity of calculating USV^T is $O(mnk)$. On the basis of Eq. (4), the dimension of matrix USV^T and X is $m \times n$, computing $USV^T - X$ computational complexity for $O(mn)$. Therefore, the overall computational complexity of the reconstruction error calculation is $O(mnk)$. Suppose that the gradient descent requires T_{GD} iterations, the total computational complexity is $O[T_{GD} \times (mnk + mnk)] = O(T_{GD} \cdot mnk)$. In order to maintain the numerical stability of the matrices, we perform QR decomposition of the matrix U and V after the iterative update to ensure their orthogonality. The computational complexity of orthogonalizing U and V is $O(mk^2)$ and $O(nk^2)$, respectively. Thus, the total complexity of the truncated singular value decomposition optimized using gradient descent is $O(T_{GD} \cdot mnk + mk^2 + nk^2)$. Since $r \ll n$, the overall complexity is mainly $O(T_{GD} \cdot mnr)$.

This study introduces *AdaMatrix* for gradient iteration, and accelerates convergence through adaptive learning rate and Nesterov momentum. By using this method, the number of iterations T_{Ada} required to achieve the same accuracy has been reduced. The single gradient complexity of *AdaMatrix* is the same as that of ordinary gradient descent, both being $O(nmr)$. The updates of the first-order momentum m_Ψ and the second-order momentum υ_Ψ are element-by-element operations, as well as the parameter updates in Eq. (8), all with a complexity of $O(nmr)$. The orthogonalization constraint is also the same as that of the ordinary gradient iteration. Therefore, the complexity of a single iteration of *AdaMatrix* is consistent with that of the ordinary gradient iteration, and $T_{Ada} \ll T_{GD}$:

$$O(T_{Ada} \cdot nmr + mr^2 + nr^2) \approx O(T_{Ada} \cdot nmr), T_{Ada} \ll T_{GD}$$

AdaMatrix reduces oscillations, and adapts to the importance of different characteristics through momentum accumulation and adaptive learning rate adjustment.

After the above proof, we can compare the computational complexity of direct SVD and gradient iterative SVD. When the data scale is large, that is, the values of m and n are massive, the SVD algorithm using gradient iteration method is more efficient. □

Theorem 3. *The gradient iteration process of AdaMatrix is convergent.*

Proof. Suppose the loss function $L(\Psi)$ is quadratic differentiable at its parameter Ψ. According to Taylor expansion, the loss change after parameter update is as follows. Among them, $\tilde{\Psi}$ is located within the interval $[\Psi_t, \Psi_{t+1}]$.

$$L(\Psi_{t+1}) = L(\Psi_t) + \nabla L(\Psi_t)^T (\Psi_{t+1} - \Psi_t) - \frac{1}{2}(\Psi_{t+1} - \Psi_t)^T \nabla^2 L(\tilde{\Psi})(\Psi_{t+1} - \Psi_t)$$

We utilize the Lipschitz continuous gradient assumption to perform upper bound control on the second-order terms of Taylor expansion. Under the Lipschitz continuous gradient assumption, the spectral norm of the Hessian matrix $\nabla^2 L(\tilde{\Psi})$, denoted as $\|\nabla^2 L(\tilde{\Psi})\| \le L$. Constraining the upper bound of the quadratic $(\Psi_{t+1} - \Psi_t)^T \nabla^2 L(\tilde{\Psi})(\Psi_{t+1} - \Psi_t)$ to be $\frac{L}{2}\|\Psi_{t+1} - \Psi_t\|^2$, according to matrix quadratic property, the second-order term of the Taylor expansion satisfies the following equation.

$$\frac{1}{2}(\Psi_{t+1} - \Psi_t)^T \nabla^2 L(\tilde{\Psi})(\Psi_{t+1} - \Psi_t) \le \frac{L}{2}\|\Psi_{t+1} - \Psi_t\|^2$$

By using the Lipschitz continuous gradient assumption and replacing the second-order term of the Taylor expansion with an upper bound, the inequality of the single-step loss change is obtained as follow. Among them, the first-order term represents the loss descent along the gradient direction, and the second-order term represents the penalty of the parameter update amount.

$$L(\Psi_{t+1}) \le L(\Psi_t) + \nabla L(\Psi_t)^T (\Psi_{t+1} - \Psi_t) + \frac{L}{2}\|\Psi_{t+1} - \Psi_t\|^2$$

The parameter update amount of *AdaMatrix* is $\Delta\Psi_t = \Psi_{t+1} - \Psi_t = -\eta \frac{\hat{m}_t}{\sqrt{\hat{\upsilon}_t} + \epsilon} + \mu \Delta\Psi_{t-1}$. Among them, $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ and $\hat{\upsilon}_t = \frac{\upsilon_t}{1-\beta_2^t}$ are the momentum terms after correcting the deviation, respectively reflecting the first-order moment and second-order moment of the gradient. Substituting $\Delta\Psi_t$ into the inequality of the single-step loss variation above results in:

$$L(\Psi_{t+1}) \le L(\Psi_t) - \eta \nabla L(\Psi_t)^T \frac{\hat{m}_t}{\sqrt{\hat{\upsilon}_t} + \epsilon} + \nabla L(\Psi_t)^T \Delta\Psi_t - \frac{L}{2}\left\| \frac{\hat{m}_t}{\sqrt{\hat{\upsilon}_t} + \epsilon} + \mu \Delta\Psi_{t-1} \right\|^2$$

During the gradient iteration process, in order to handle the influence of noise in the stochastic gradient, we will assign the stochastic gradient $g_t = \nabla L(\Psi_t; \xi_t)$ decomposes into the true gradient and the noise term $g_t = \nabla L(\Psi_t) + \zeta_t$, where the expectation of the noise term $\mathbb{E}[\zeta_t] = 0$. Variance $\mathbb{E}[\|\zeta_t\|^2] \le \sigma^2$. Using the momentum update rule $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$, calculate the expectation for the deviation-corrected $\hat{m}_t$. Get $\mathbb{E}[\hat{m}_t] = \nabla L(\Psi_t) + \frac{\beta_1}{1-\beta_1^t} \sum_{i=1}^{t} \beta_1^{t-i} \zeta_i$. Since the noise term is expected to be zero, the accumulation of the noise term tends to zero under the long-term expectation, and the momentum term approaches the true gradient, reducing the noise interference. Taking the expectation of the loss change inequality and comprehensively considering the combined influence of noise, momentum and regularization terms on the loss expectation, the expected expression of the loss change is obtained as follows.

$$\mathbb{E}[L(\Psi_{t+1})] \le \mathbb{E}[L(\Psi_t)] - \eta \mathbb{E}\left[\frac{\|\nabla L(\Psi_t)\|^2}{\sqrt{\hat{\upsilon}_t} + \epsilon} \right] + \mu \mathbb{E}[\nabla L(\Psi_t)^T \Delta\Psi_{t-1}] + \frac{L}{2}\mathbb{E}[\|\Delta\Psi_t\|^2]$$

Suppose the parameter is bounded $\|\Psi_t\| \; leD$, recursively calculate the loss changes at all time steps. The total loss reduction is dominated by the gradient norm, while noise and momentum terms are secondary influences. Based on the above conditions, if the learning rate satisfies $\eta \le \frac{\epsilon(1-\beta_1)}{\sqrt{2L(1+\beta_2)}}$, we can obtain the upper bound of the convergence rate of the iterative sequence:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\|\nabla L(\Psi_t)\|^2\right] \le \frac{2(L(\Psi_1) - L^*)}{\eta T} + \frac{C\sigma^2}{\sqrt{T}}$$

The $C = \frac{\beta_1}{(1-\beta_1)\sqrt{1-\beta_2}}$, L^* for loss function is lower. $\sqrt{\hat{v}_t} + \epsilon$ dynamically adjusts the step size to avoid unstable iteration caused by an overly large step size. It adaptively scales the learning rate based on the second moment of the historical gradient and balances the update speed in different directions. Thus, the actual learning rate $\eta_{eff} = \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}$ is decreased in the direction of the large gradient amplitude, thereby reducing the oscillation. □

Algorithm 1 Combining the computational process of t-SVD of AdaMatirx

Input: $X \in \mathbb{R}^{b\times t\times d}$, η, number of iterations n, convergence threshold ϵ
Output: left singular vector matrix $U \in \mathbb{R}^{m\times r}$, singular value diagonal matrix $S \in \mathbb{R}^{r\times r}$, right singular vector matrix $V \in \mathbb{R}^{n\times r}$

1: Set $\Psi \in U, S, V$
2: $Y \leftarrow \text{reshape}(X, (b \cdot t, d))$
3: $U \leftarrow \text{random\_normal}((b \cdot t), r)$
4: $S \leftarrow \text{random\_normal}(r, r)$
5: $V \leftarrow \text{random\_normal}(d, r)$
6: $U_{\text{prev}}, S_{\text{prev}}, V_{\text{prev}} \leftarrow 0, 0, 0$
7: $U, \_ \leftarrow \text{QR}(U)$
8: $V, \_ \leftarrow \text{QR}(V)$
9: **for** k **from** 1 **to** n **do**
10: **if** $\min_{U,V} \|Y - USV^T\|_F^2 - \lambda\|S_k\|_1 \geq \epsilon$ **then**
11: $\hat{X} \leftarrow USV^T$
12: $\Psi, \_ \leftarrow \Psi - m\Delta\Psi_{\text{prev}}$
13: $\hat{X} \leftarrow U, \_ \cdot S, \_ \cdot V, \_^T$
14: $\text{grad\_}U = \left(\hat{X} - Y\right) V_k S_k^T$
15: $\text{grad\_}S = U_k^T \left(\hat{X} - Y\right) V_k$
16: $\text{grad\_}V = \left(\hat{X} - Y\right)^T U_k S_k$
17: $m_\Psi = \beta_1 m_\Psi + (1-\beta_1)\nabla_{\tilde{\Psi}} L$
18: $v_\Psi = \beta_2 v_\Psi + (1-\beta_2)(\nabla_{\tilde{\Psi}} L)^2$
19: $\hat{m}_U = \frac{m_U}{1-\beta_1}$
20: $\hat{v}_U = \frac{v_U}{1-\beta_2^t}$
21: $\Delta\Psi \leftarrow \eta \frac{\hat{m}_\Psi}{\sqrt{\hat{v}_\Psi}+\epsilon} + m\Delta\Psi_{prev}$
22: $[U, S, V] \leftarrow [U - \Delta U, S - \Delta S, V - \Delta V]$
23: $[\Delta U_{\text{prev}}, \Delta S_{\text{prev}}, \Delta V_{\text{prev}}] \leftarrow [\Delta U, \Delta S, \Delta V]$
24: $U, \_ \leftarrow \text{QR}(U)$
25: $V, \_ \leftarrow \text{QR}(V)^T$
26: **else**
27: break
28: **end if**
29: **end for**
30: **return** U_k, S_k, V_k

We design a corresponding algorithm for t-SVD combined with gradient descent, as shown in Algorithm 1. Lines 1 to 6 are the initialization phase. Line 2 expands the input 3D tensor into a 2D matrix Y in order to apply SVD. Lines 3 to 5 use random initialization methods to assign initial values to the U, S, V matrix. Line 6 initializes the momentum term of the previous round, setting the initial value to 0. The momentum term is used in the NAG estimation step. Lines 7 and 8 perform QR decomposition of U and V to ensure that their initial values are orthogonal matrices. Next, lines 9 to 30 are iterative singular value decomposition. Lines 9 and 10 are the conditions for iterating the loop and terminating. Lines 11 to 13 update the NAG momentum estimate. Lines 14 to 16 perform gradient calculations. Lines 17 to 20 update *AdaMatrix*'s first-order moments and second-order moments. After completing the above operation, the parameters U, S, V are updated in lines 21 to 23, and the momentum term of this cycle is updated and stored. Lines 24 to 25 perform the QR decomposition of the updated U and V again. Finally, the algorithm outputs the truncated matrix U_k, S_k, V_k. In this algorithm, '$U, \_$' indicates all columns of a particular row, '$V, \_$' indicates all rows of a particular column, and so on. Where the '_' indicates that a particular dimension of the matrix is selected for the operation on the matrix, and this dimension notation is used to emphasize the dimensionality in the gradient update.

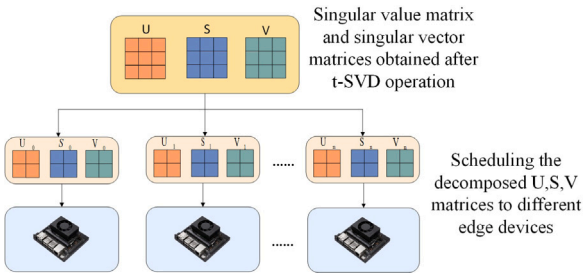


Fig. 4. Scheduling the sliced singular value and singular vector matrices to different edge devices.

Lines 14 to 22 in Algorithm 1 analyze the iterative process of *AdaMatrix*. The complexity of the first-order and second-order momentum updates is $O(mr + r^2 + nr)$, which is proportional to the dimension of the parameter matrix. The complexity of parameter update is $O(mr + r^2 + nr)$. QR decomposition is subjected to orthognalization constraints, with a complexity of $O(mr^2 + nr^2)$. When $r \ll \min(m, n)$, the total single-iteration complexity is approximately $O(mnr)$.

Some data in the data set have very small singular values. These data represent noise or unimportant information. To improve computational efficiency, we utilize a truncation process to filter out these noisy data and retain the most important information. We distribute the decomposed U, S, V matrix blocks to different edge devices, as shown in Fig. 4.

The feature dimension d is divided into h heads, each head has d/h dimensions, for example, X is divided into $X_i \in \mathbb{R}^{b\times t\times d/h}$. After that, truncated singular value decomposition of U_r, S_r, V_r^T is segmentation for $U_{r,i}, S_{r,i}, V_{r,i}^T$.

4. Task scheduling to edge devices and calculation of multi-head attention

According to the computing power and load condition of each edge device, the attention head is allocated to different devices, and the delivery optimal strategy is introduced to optimize the transmission path and speed of data between different devices. Given an objective function to minimize data transfer time as depicted in Eq. (12).

$$\min \sum_{i=1}^{N} \sum_{j=1}^{M} d_{ij} x_{ij} \tag{12}$$

The d_{ij} represents the data transmission volume from device i to j. x_{ij} is a decision variable, which denotes whether choose transmission path from device i to j. By optimizing the value of x_{ij} to find the path which the data takes the least time to travel between all devices. In order to ensure that the total data sent and received on the device i does not exceed the maximum bandwidth B_i of the device, constraints are set as shown in Eq. (13). The first summation term $\sum_{j=1}^{M} d_{ij} x_{ij}$ represents the total amount of data sent by device i to j. The second summation term $\sum_{k=1}^{N} d_{ki} x_{ki} \leq B_i$ represents the total amount of data sent to device i from all other devices k.

$$\sum_{j=1}^{M} d_{ij} x_{ij} + \sum_{k=1}^{N} d_{ki} x_{ki} \leq B_i, \quad \forall i \in \{1, \ldots, N\} \tag{13}$$

Algorithm 2 Feature-driven data distribution in SDRM

```
1: function FORWARD(X, feature, heads)
2:     V ← empty list of size len(heads)
3:     for i = 1 to len(heads) do
4:         Q_i ← X[:, feature]
5:         V_i ← heads[i](Q_i)
6:         V[i] ← V_i
7:     end for
8:     return concatenate(V)
9: end function
10: function BACKWARD(grad_output, feature, heads)
11:     grad_output ←zero matrix with dimensions of X
12:     for i = len(heads) downto 1 do
13:         grad_V_i ← grad_output corresponding to output from heads[i]
14:         grad_Q_i ← heads[i].backward(grad_V_i)
15:         grad_input[:, feature[i]]+ = grad_Q_i
16:     end for
17:     return grad_input
18: end function
19: function CONCATAENATE(V)
20:     return concatenate(V, axis = 1)
21: end function
```

Algorithm 2 describes the distribution of data in SDRM. Line 2 initializes an empty list V, which store the processing results of each header in preparation for subsequent matrix block splicing. Line 3 iterates through all the attention heads one by one. Line 4 and 5 extract the feature columns in the matrix and input them to the ith attention head for forward propagation. Line 8 consolidates all header data to form the final output. Lines 10 to 18 show the implementation of backpropagation. Line 11 initializes a zero matrix to store the gradients generated during backpropagation. Lines 13 to 15 extract the gradient of the current attention head. Line 17 returns the input gradient, thus updating the model parameters.

The edge devices process each set of matrix blocks in parallel. Each edge device is responsible for calculating the Q, K, V of its corresponding fragment as shown in Eq. (14). Among them, the matrices Q_i, K_i, and V_i are respectively the query matrix, the key matrix, and the value matrix. After t-SVD decomposition, three low-rank matrix blocks, namely $U_{r,i}$, $S_{r,i}$, and $V_{r,i}^T$, are obtained. The W matrix here is the weight matrix of the linear transformation. Projecting the low-rank matrix onto different subspaces generates the corresponding vectors.

$$(Q_i, K_i, V_i) = U_{r,i} S_{r,i} V_{r,i}^T (W^Q, W^K, W^V) \tag{14}$$

At this point, the input matrix of each attention head has undergone low-rank approximation through t-SVD. Q_i, K_i, and V_i are reconstructed through low-rank tensors. The parameter scale of each head is reduced from $O(d^2)$ to $O((r/h)^2)$, and the computational complexity is reduced from $O(n^2)$ to $O(nr)$. The decomposed low-rank matrix is allocated to different edge devices, and the attention score of each head is calculated in parallel using multiple devices. Storage requirements after a low rank approximation is reduced. The low-rank decomposition of the input tensor significantly reduces the computational complexity and storage requirement.

Algorithm 3 is the task scheduling algorithm of SDRM that schedules data to edge devices. The algorithm is based on the resource load of the devices and the optimal transmission capacity. Lines 1 to 4 are the initialization phase of the algorithm. Line 1 initializes M, which indicates the number of available edge devices. Line 2 defines the maximum task queue size for each edge device, ensuring that the capacity of each device is not exceeded when performing task allocation. Line 3 assigns the input data X to each queue according to the size of the task queue. Line 4 initializes flag to false, which is used to keep track of whether the task scheduling is completed or not. We assume that N data blocks are allocated to the queue and the t-SVD operation is performed on N elements, with a time complexity of $O(N)$. Lines 6 through 17 describe the process of task assignment and execution. Line 6 iterates through all tasks X_m to determine the optimal execution device for each task. Based on Eqs. (12) and (13), line 7 determines the optimal distribution path for the tasks through an optimization problem. Then edge device e_m is facilitated to monitor the current resource load of each device. In lines 11 to 16, the load of the edge device is analyzed to see if it is above the threshold T. If it is below T, the task is assigned to the change edge device for the multi-head attention mechanism. If it is above the threshold, the system waits for the load of the device to decrease and then tries to assign the task to the device again. This nested loop operation has an outer layer for N tasks and an inner layer for M edge devices, with a time complexity of $O(N \cdot M)$. The results are reconstructed on lines 21 through 28. We traverse the edge device and check the device load. If the load is below T, the result is reconstructed on the device. If the load is high, the system waits for the congestion to decrease before reconstructing the results on the device. The time complexity of result reconstruction for M devices is $O(M)$. Line 29 caches the final result of the processing to the edge device. Lines 30 through 31 set $flag$ to $true$ and return the flag bit. Overall, the time complexity of Algorithm 3 is $O(N \cdot M)$.

Algorithm 3 The process of SDRM resource scheduling to edge devices

Input: Input data X, set of edge devices $\{e_1, e_2, \ldots, e_M\}$, threshold for resource utilization T
Output: Completion $flag$ of SDRM task scheduling

```
1: M ← Number of available edge devices
2: QueueNum ← Maximum task queue size for each edge device
3: Distribute data chunks from X into QueueNum task queues
4: Initialize flag flag ← false
5: Execute t-SVD(X_m) on local devices
6: for each task X_m do
7:     Solve the optimization problem by using Equation (12) with the constraint condition Equation (13)
8:     for each e_m ∈ E do
9:         Monitor the resource load on edge device e_m
10:        if Load(e_m) < T then
11:            Assign task X_m to edge device e_m for multi-head attention execution
12:            Execute multi-head attention(X_m) on edge device e_m
13:        else
14:            Wait for Load(e_m) to decrease below T
15:            Retry task assignment to edge device e_m
16:            Execute multi-head attention(X_m) on edge device e_m by invoked Algorithm 2
17:        end if
18:    end for
19: end for
20: Gather the results from multi-head attention processing
21: for each e_m ∈ E do
22:    if Load(e_m) < T then
23:        Perform local reconstruction of results on edge device e_m
24:    else
25:        Wait for Load(e_m) to decrease below T
26:        Perform local reconstruction of results on edge device e_m
27:    end if
28: end for
29: Cache the final results on edge devices for future use
30: Set flag ← true when all tasks are completed
31: return flag
```

For multi-head attention mechanism on the qkv calculation, each head calculates Z_i by Eq. (15).

$$Z_i = V_i \otimes (Q_i \otimes K_i) \tag{15}$$

The first $\otimes$ in the above formula is the process of self-attention. By generating query, key, and value vectors, calculating and normalizing the attention score, and finally summing the value vectors weighted, we can obtain a weighted representation of each position in the input sequence.

When calculating the attention fraction, the multiplication operations of large-scale matrices are computationally intensive. Partitioning the query (Q), key (K), and value (V) matrices can make calculations

simpler. By dividing Q, K, V into smaller blocks, the amount of data processed at each step is also reduced, thereby alleviating the computational pressure of a single operation. Divide Q_i, K_i, V_i into blocks of initial block size b. For Q_i, we can get $\lceil m/b \rceil$ a block. For K_i and V_i, we can get $\lceil n/b \rceil$ block.

Let us initialize block size and monitor key performance metrics, such as processing time, memory usage and processor usage. Based on the collected performance data, we can use a feedback mechanism to adjust the block size. Simple heuristic rules are applied to dynamically adjust strategies. If a decrease in processing speed or resource usage is detected, the size of the data block is automatically reduced so as to decrease the demand on resources. If resource usage is low, you can increase the block size to improve data processing efficiency. For each head i and each pair of blocks $Q_{i,b}, K_{i,b}$, the attention score is calculated as shown in Eq. (16).

$$Attention(Q_{i,b}, K_{i,b}) = softmax\left(\frac{Q_{i,b}K_{i,b}^T}{\sqrt{d_k}}\right) \tag{16}$$

where d_k is the dimension of the key vector, and is used to adjust the dot product result. The attention score matrix multiplied by the corresponding value matrix block $V_{i,b}$ to get the output block $Z_{i,b}$ by Eq. (17). And then, all the output blocks of each head are summed or concatenated to form the final output matrix Z_i by Eq. (18).

$$Z_{i,b} = Attention(Q_{i,b}, K_{i,b}) \times V_{i,b} \tag{17}$$

$$Z_i = concatenate(Z_1, Z_2, \ldots, Z_h) \tag{18}$$

Let us integrate Z_i in each header to get Z', and perform matrix multiplication with the output weight matrix W^O as shown in Eq. (19).

$$Z = concatenate(Z_1, Z_2, \ldots, Z_h) \otimes W^O \tag{19}$$

Algorithm 4 Calculation of piecewise matrix

Input: Query matrix Q, Key matrix K, Value matrix V, Weighting matrix W^Q, W^K, W^V, Output weight matrix W^O, Block size b
Output: Output matrix of Multi Attention Z

for each $i \in [1, n]$ **do**
  $Q_i = U_{r,i}S_{r,i}V_{r,i}^T W^Q$
  $K_i = U_{r,i}S_{r,i}V_{r,i}^T W^K$
  $V_i = U_{r,i}S_{r,i}V_{r,i}^T W^V$
  for each $b_Q \in [1, m/b]$ **do**
    $Q_b = Q_i[:, (b_Q - 1) * b : min(b_Q * b, m)]$
    for each $b_K \in [1, m/b]$ **do**
      $K_b = Q_i[:, (b_K - 1) * b : min(b_K * b, n)]$
      $V_b = Q_i[:, (b_V - 1) * b : min(b_V * b, n)]$
      $Attention_b = \text{softmax}\left(\frac{Q_{block} \cdot K_{block}^T}{\sqrt{d_k}}\right)$
      $Z_b = Attention_b \times V_b$
    end for
    $Z_i = Z_i \oplus Z_b$
  end for
end for
$Z = concatenate(Z_1, Z_2, \ldots, Z_h)W^O$
return Z

Algorithm 4 calculates the Q, K and V matrices in the multi-head attention mechanism, and uses the block-matrix technique to optimize the process.

We analyze its power consumption through the model of SDRM. In *AdaMatrix*, the complexity of gradient calculation for each iteration is $O(mnr)$, and the number of iterations is T. The power consumption per floating-point operation of the hardware is α_{flop}. Meanwhile, our adjustment of the learning rate will introduce an additional overhead factor ω. The power consumption of *AdaMatrix* can be obtained as $P_{AdaMatrix} = \alpha_{\text{flop}} \cdot T \cdot (mnr + \omega r^2)$. In *AdaMatrix*, the momentum coefficients β_1, β_2 and the learning rate η indirectly affect the number of iterations $T \propto \frac{1}{\eta \cdot (1-\beta_1)}$ by adjusting the convergence rate. In the t-SVD decomposition stage, the computational complexity of a single SVD is $O(mnr)$. Considering QR decomposition and iterative processing, the power consumption of this stage is $P_{\text{SVD}} = \alpha_{\text{flop}}(2mnr + 2r^2(m+n))$. Multi-head attention is divided into multiple attention heads, and the computational complexity of each attention head is $O(\frac{bt^2d}{h})$. Therefore, the total power consumption of all heads is $P_{\text{MHA}} = \alpha_{\text{flop}} \cdot b \cdot t^2 \cdot d$. The matrix blocks are distributed to edge devices. We set the number of devices as M and the size of the matrix blocks as $s \times s$. Then the overall transmission energy consumption is $P_{\text{comm}} = \gamma_{\text{bit}} \cdot M \cdot s^2$, where γ_{bit} is the transmission power consumption per bit. Set the power consumption for reading and writing words as δ_{access}. The power consumption for memory access is related to the data dimension. According to the compression ratio in Theorem 1, The obtained storage access power consumption is $P_{\text{mem}} = \delta_{\text{access}} \cdot (btr + r + rd) \cdot \text{size\_of(element)}$. As shown above, the overall power consumption of the SDRM model is:

$$P_{total} = \alpha_{\text{flop}}(2mnr + Tr^2(m+n)) + \gamma_{\text{bit}}Ms^2 + \delta_{\text{access}}(btd - (btr + rd))$$

5. Performance evaluation

In this section, several experiments are conducted to evaluate the proposed rank reduction methods. We focus on comparing the computation and processing time of the data to analyze whether SDRM can effectively speed up the image data processing.

5.1. Experimental settings

5.1.1. Experimental environment and dataset

Our experiments use and extend the Image Transformer (ViT) [1,27] implemented by Python, and the experimental code[1] was programmed based on the ViT. We trained and tested the model through adopting a database of circuit component product images from the open source dataset - Surface Defect Detection.[2] In order to simulate the situation where the number of qualified products is often greater than that of defective products in industrial product defect detection, we preserve the imbalance between the number of qualified products and that of defective products in the open source dataset. To reduce the impact of the imbalance between the two categories in the dataset on the accuracy rate, we preprocess the data. We set the sample weight and pass in the weight through *class_weight* to adjust the sensitivity of the loss function to the class imbalance. The minority class samples were given higher weights and the majority class samples were given lower weights. Setting the weights in this way allows the model to optimize its ability to recognize the minority class.

The NVIDIA Jetson Orin Nano and Xavier NX are utilized as the edge devices for our distributed parallel processing experiments based on homogeneous edge devices connecting to the same local area network through a switch, as shown in Table 1 and Fig. 5.

5.1.2. The brief description of the compared methods

We compared various rank reduction methods and analyzed the processing efficiency of distributed parallelism.

The SDRM in this study uses truncated singular value decomposition for low-rank tensor approximation. The low-rank approximation of high-dimensional data is performed to optimize the data transfer and processing analysis. The CoNoT (Coupled Nonlinear Transform) method [35] enhances low-rank tensor approximation through dual-domain transformations: a 2D spatial convolution with nonlinear activation coupled with a 1D spectral/temporal convolution, forming cascaded feature abstraction. The NRR is the original distributed model without rank reduction, which is a baseline approach to distributed models. The image is divided into blocks and then input into the multi-layer self-attention module to achieve feature extraction. DeiT-B [49]

[1] https://github.com/doriz104/SDRM.
[2] https://github.com/Charmve/Surface-Defect-Detection/tree/master.

Table 1
Hardware type specification.

Hardware	CPU	GPU	Memory	Quantity
Jetson Orin Nano	6-core ARM Cortex-A78AE @ 1.5 GHz	NVIDIA Ampere architecture 1024 × NVIDIA CUDA Cores 32 × 3rd Gen Tensor Cores	8 GB 128-bit LPDDR5 68 GB/s	2
Jetson Xavier NX	6-core NVIDIA Carmel ARM v8.2 64-bit @ 1.4 GHz	NVIDIA Volta architecture 384 × NVIDIA CUDA Cores 48 × Tensor Cores	8 GB 128-bit LPDDR4x 51.2 GB/s	1
PC	Core Ultra7 155H	RTX 4060	32 GB RAM	1

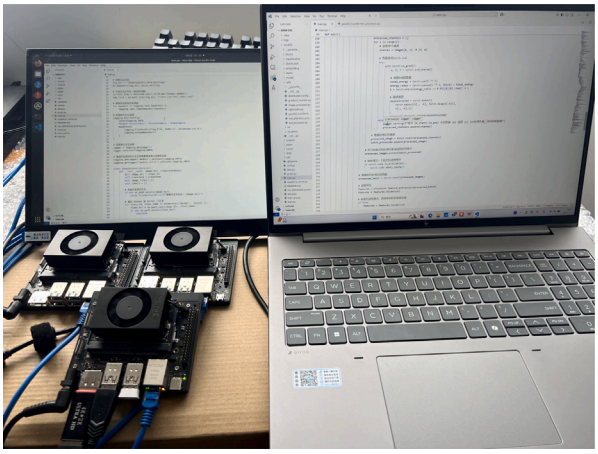


Fig. 5. NVIDIA edge GPU devices and PC used for our experiments.

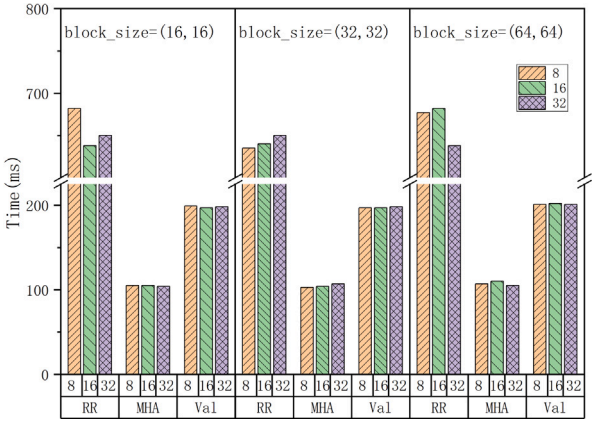


Fig. 6. Comparison of average running time of different phases for different svd_sizes at different block_sizes. (Other parameter standards: Learning rate lr = 1e−4, truncation threshold (energy retention value) ϕ = 90%, regularization coefficients λ_1, λ_2, λ_3 = 1e−3, 1e−4, 1e−4.).

adopts the Transformer architecture to handle image classification tasks and is a variant of the Visual Transformer (ViT, namely NRR). FLORA [50] maps the low-rank selection problem to NAS, combining the methods of eliminating inefficient architectures and weight inheritance. This model realizes the automatic search of fine-grained rank in hypernetwork training. The TRAWL [51] method stacks the weight matrices in the Transformer architecture into high-order tensors and implements low-rank decomposition by using CP decomposition or Tucker decomposition.

5.2. Experimental results

In this experiment, we focus on comparing the running time of each method. This is because the core purpose of this study is to improve the efficiency of parallel processing with multiple heads of attention. We compared the processing time of each processing stage under different parameters of SDRM. The processing time and inter-device transfer time of different methods are also compared. These experiments well illustrate that the use of SDRM for high-dimensional data with reduced-rank processing improves the efficiency of parallel processing.

Through the comparison of various order reduction methods, the efficiency of image processing in terms of running time is reflected. On the basis of no great impact on the accuracy, the processing time is theoretically shortened after image rank reduction. The following validation results prove this point. Note that in order to keep the same context in the experiments, we transferred the rank reduction method in the comparison model to the scenario of distributed multi-head attention processing. We divide the whole processing into three phases: rank reduction (RR), multi-head attention (MHA), and validation (Val) (no time statistics performed in the training phase).

Fig. 6 shows the comparison of the time required for the three phases of RR, MHA, and Val in the SDRM study. This experiment investigated the time caused by different block sizes (block_size) with different SVD batch sizes (svd_batch_size = 8, 16, 32). The experimental

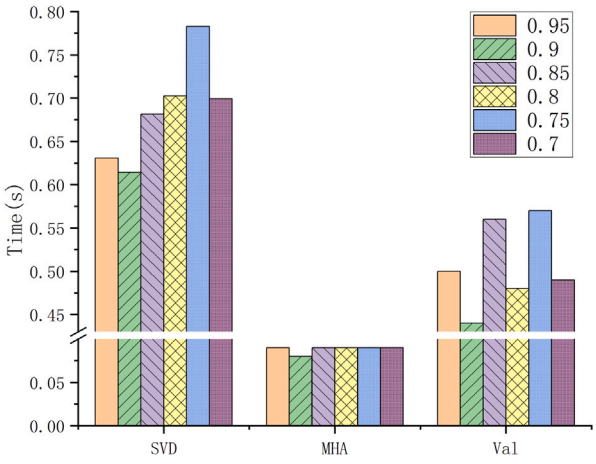


Fig. 7. The processing times of the three processing stages at different energy retention values (0.7 to 0.95).

results show that matrix decomposition and dimensionality reduction are the most time-consuming processes.

Fig. 7 presents the processing times of the three processing stages of SVD, MHA, and Val under different energy retention values. Through the comparative analysis of the time efficiency at each stage, we can find that different energy retention values have little influence on the truncation of multi-head attention processing. In the Val stage, the processing times corresponding to 0.9 (90%) energy retention and 0.85 (85%) energy retention were significantly shorter than those of other energy retention values. When the energy retention value is between

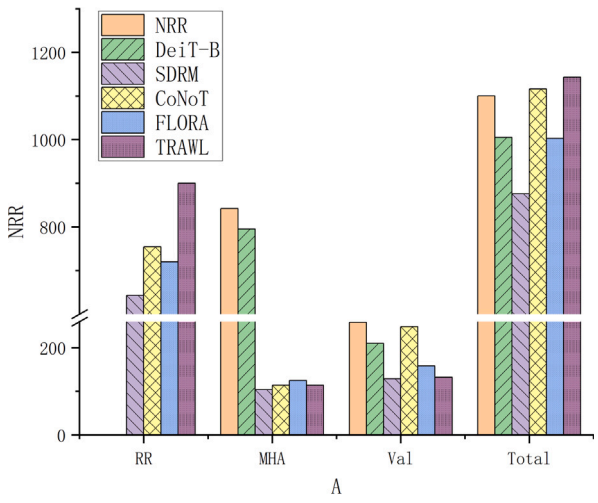


Fig. 8. Comparison of average running time of five phases.

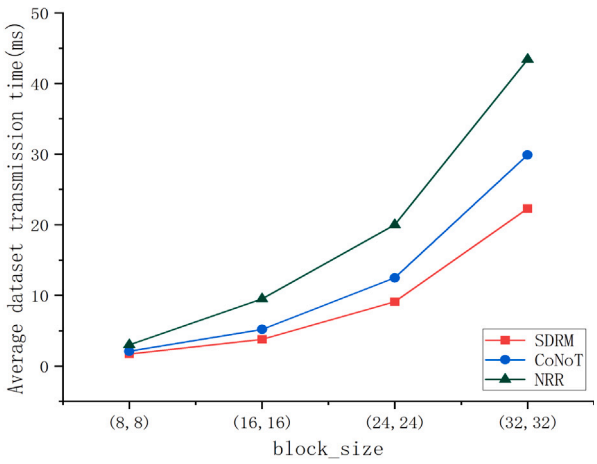


Fig. 9. Compare transfer times for different methods with different block_sizes.

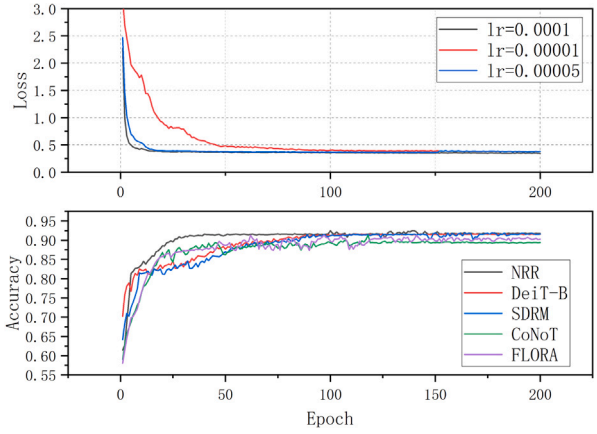


Fig. 10. The training loss of SDRM at different learning rates and the accuracy comparison of several methods.

85% and 90%, it can significantly improve the processing efficiency of SDRM.

The rank reduction operation is the most time-consuming in the whole data processing phase. However, a comparison of the runtimes of the different methods was made in Fig. 8. The computational overheads of NRR and DeiT-B are very high when raw image tensor is processed without rank reduction. On the contrary, the systematic rank-reduction operation through t-SVD greatly improves the processing efficiency, SDRM reduces the execution time of the attention mechanism by 87.6% and decreases the time required for the verification phase by 50% compared with NRR. Overall processing time for SDRM was reduced by 20.4% compared to NRR without rank-down processing. CoNoT does not perform as well as SDRM in distributed systems. The rank-down operation of SDRM is 14.7% faster than CoNot, and the verification processing phase saves 47.9% of time. The overall processing time of CoNot is instead longer than that of NRR due to the time consumption of the descending-rank processing. Due to the complexity of CoNoT's rank-reduction operations, it has no significant advantage in time. The running time of SDRM is lower than CoNoT and NRR in all of data processing phases, which demonstrates clearly greater efficiency the industrial product defect detection requiring faster processing speed. The rank reduction stage of TRAWL is the most time-consuming. This is because the Tucker decomposition method is used in this model, which consumes a large amount of storage and computing costs. The time differences of each method in the MHA stage are not significant. During the verification stage, when SDRM, FLORA and TRAWL maintain low consumption, these methods have obvious advantages at this stage. Although the NRR and DeiT-B methods without low-rank operations save some time, they have efficiency shortcomings in subsequent processing. The cost of rank reduction required by CoNoT, FLORA and TRAWL is relatively high. When comparing as a whole, it is found that although these methods carry out rank reduction operations and shorten the verification time required, there is still a certain amount of time consumption overall. The time of each stage of SDRM is balanced, presenting high efficiency throughout the entire process.

As shown in Fig. 9, the time consumed by the three methods for transmission among edge devices is compared with different sizes of data sliced blocks. After the image down-ranking process, our SDRM' transmission time of the same size image is always least under the same network conditions of the same device. The processing time reductions mentioned above are all based on the assurance that there is no significant decrease in verification accuracy after the image rank reduction, as depicted in Fig. 10. The training loss of SDRM converges stably to 0.3 at different learning rates. Under multiple training epochs, SDRM can achieve a similar accuracy to the model without rank reduction decomposition (0.91, with the accuracies of NRR and DeiT-B being approximately 0.92), while CoNoT and FLORA are slightly inferior in terms of accuracy.

Table 2
Comparison of the training accuracy (train_acc), verification accuracy (val_acc), and test accuracy (test_acc) of multiple methods.

	train_acc(%)	val_acc(%)	test_acc(%)
NRR	95.24 ± 0.5	92.13 ± 0.2	91.73 ± 0.5
DeiT-B	94.70 ± 0.5	92.84 ± 0.3	91.53 ± 0.3
CoNoT	92.35 ± 0.5	89.5 ± 0.2	89.39 ± 0.3
FLORA	94.75 ± 0.5	91.50 ± 0.3	90.27 ± 0.3
SDRM	92.87 ± 0.5	91.43 ± 0.3	90.59 ± 0.3

SDRM can bring about a slight effect of reducing overfitting. We can see it from Table 2. Table 2 shows that by comparing the accuracy rates of various methods on different datasets, it can be seen that NRR presents a relatively high accuracy in the data of the training set, but the accuracy decreases most significantly in the validation set and the test set. SDRM and other methods reduce overfitting through low-rank approximation and regularization as well as other methods.

6. Conclusions

In this paper, we propose an intelligent dimensionality reduction model with gradient iterative learning applied to industrial product detection. The model improves the image processing and transmission

rate by low rank approximation to find the defective products at a faster rate. A low-rank approximation is presented by using truncated singular value decomposition based on Image Transformer, which reduces data redundancy while preserving important features of the images. Experiments conducted on edge devices show that the model is stable and efficient. In the future, we will continue to improve the multi-attention computing performance of images on edge devices, and explore the efficient product detection on heterogeneous edge devices.

CRediT authorship contribution statement

Yuze Du: Writing – original draft, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Xiaogang Wang:** Writing – review & editing, Resources, Project administration, Methodology, Funding acquisition. **Haokun Chen:** Validation, Software. **Chenfeng Zhang:** Visualization, Validation, Software. **Jian Cao:** Resources, Methodology, Funding acquisition. **Rajkumar Buyya:** Supervision, Methodology.

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

This work is supported in part by National Natural Science Foundation of China (Granted Number 62072301), in part by Natural Science Foundation of Shanghai Science and Technology Innovation Action Plan (Granted Number 22ZR1425300), and in part by Program of Technology Innovation of the Science and Technology Commission of Shanghai Municipality (Granted Number 21511104700).

Data availability

The research code for this paper is illustrated in the experimental section of the paper, linked to the share base of the author's github account.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[2] J. Wang, G. Xu, F. Yan, J. Wang, Z. Wang, Defect transformer: An efficient hybrid transformer architecture for surface defect detection, Meas. 211 (2023) 112614.

[3] A. Katharopoulos, A. Vyas, N. Pappas, F. Fleuret, Transformers are RNNs: Fast autoregressive transformers with linear attention, in: Proceedings of the 37th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 119, 2020, pp. 5156–5165.

[4] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 11106–11115.

[5] T. Dao, FlashAttention-2: Faster attention with better parallelism and work partitioning, in: The Twelfth International Conference on Learning Representations, 2024, https://openreview.net/forum?id=mZn2Xyh9Ec.

[6] I. Lyngaas, M.G. Meena, E. Calabrese, M. Wahib, P. Chen, J. Igarashi, Y. Huo, X. Wang, Efficient distributed sequence parallelism for transformer-based image segmentation, Electron. Imaging 36 (12) (2024) 1–7.

[7] P. Zeng, G. Hu, X. Zhou, S. Li, P. Liu, S. Liu, Muformer: A long sequence time-series forecasting model based on modified multi-head attention, Knowl.-Based Syst. 254 (2022) 109584.

[8] R. Ran, T. Gao, B. Fang, Transformer-based dimensionality reduction, 2022, http://dx.doi.org/10.48550/arXiv.2210.08288, CoRR abs/2210.08288.

[9] Á. Huertas-García, A. Martín, J. Huertas-Tato, D. Camacho, Exploring dimensionality reduction techniques in multilingual transformers, Cogn. Comput. 15 (2) (2023) 590–612.

[10] A. Anaissi, B. Suleiman, S.M. Zandavi, NeCPD: An online tensor decomposition with optimal stochastic gradient descent, 2020, CoRR abs/2003.08844, https://arxiv.org/abs/2003.08844.

[11] B. Wang, Q. Xu, Z. Bian, Y. You, Tesseract: Parallelize the tensor parallelism efficiently, in: Proceedings of the 51st International Conference on Parallel Processing, 2023, http://dx.doi.org/10.1145/3545008.3545087.

[12] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F.J. R Ruiz, J. Schrittwieser, G. Swirszcz, et al., Discovering faster matrix multiplication algorithms with reinforcement learning, Nat. 610 (7930) (2022) 47–53.

[13] A. Khalesi, P. Elia, Multi-user linearly-separable distributed computing, IEEE Trans. Inform. Theory 69 (10) (2023) 6314–6339.

[14] W. Li, Z. Chen, Z. Wang, S.A. Jafar, H. Jafarkhani, Flexible distributed matrix multiplication, IEEE Trans. Inform. Theory 68 (11) (2022) 7500–7514.

[15] Y. Jin, B. Huang, Y. Yan, Y. Huan, J. Xu, S. Li, P. Gope, L. Da Xu, Z. Zou, L. Zheng, Edge-based collaborative training system for artificial intelligence-of-things, IEEE Trans. Ind. Inform. 18 (10) (2022) 7162–7173.

[16] S. Dash, I. Lyngaas, J. Yin, X. Wang, R. Egele, J.A. Ellis, M. Maiterth, G. Cong, F. Wang, P. Balaprakash, Optimizing distributed training on frontier for large language models, in: ISC High Performance 2024 Research Paper Proceedings, 2024, pp. 1–11, http://dx.doi.org/10.23919/ISC.2024.10528939.

[17] Z. Lai, S. Li, X. Tang, K. Ge, W. Liu, Y. Duan, L. Qiao, D. Li, Merak: An efficient distributed DNN training framework with automated 3D parallelism for giant foundation models, IEEE Trans. Parallel Distrib. Syst. 34 (5) (2023) 1466–1478.

[18] Y. Liu, S. Li, J. Fang, Y. Shao, B. Yao, Y. You, Colossal-auto: Unified automation of parallelization and activation checkpoint for large-scale models, 2023, http://dx.doi.org/10.48550/arXiv.2302.02599, CoRR abs/2302.02599.

[19] P. Chen, W. Zhang, S. He, Y. Gu, Z. Peng, K. Huang, X. Zhan, W. Chen, Y. Zheng, Z. Wang, Y. Yin, G. Chen, Optimizing large model training through overlapped activation recomputation, 2024, http://dx.doi.org/10.48550/arXiv.2406.08756, CoRR abs/2406.08756.

[20] P. Liang, Y. Tang, X. Zhang, Y. Bai, T. Su, Z. Lai, L. Qiao, D. Li, A survey on auto-parallelism of large-scale deep learning training, IEEE Trans. Parallel Distrib. Syst. 34 (8) (2023) 2377–2390.

[21] Y. Gao, B. Hu, M.B. Mashhadi, A.-L. Jin, P. Xiao, C. Wu, US-byte: An efficient communication framework for scheduling unequal-sized tensor blocks in distributed deep learning, IEEE Trans. Parallel Distrib. Syst. 35 (1) (2024) 123–139.

[22] K. Han, A. Xiao, E. Wu, J. Guo, C. XU, Y. Wang, Transformer in transformer, in: Advances in Neural Information Processing Systems, Vol. 34, 2021, pp. 15908–15919.

[23] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, D. Tao, A survey on vision transformer, IEEE Trans. Pattern Anal. Mach. Intell. 45 (1) (2023) 87–110.

[24] R. Strudel, R. Garcia, I. Laptev, C. Schmid, Segmenter: Transformer for semantic segmentation, in: 2021 IEEE/CVF International Conference on Computer Vision, ICCV, 2021, pp. 7242–7252, http://dx.doi.org/10.1109/ICCV48922.2021.00717.

[25] M. Chen, H. Peng, J. Fu, H. Ling, AutoFormer: Searching transformers for visual recognition, in: 2021 IEEE/CVF International Conference on Computer Vision, ICCV, 2021, pp. 12250–12260, http://dx.doi.org/10.1109/ICCV48922.2021.01205.

[26] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, R. Jin, Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting, in: International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 162, 2022, pp. 27268–27286.

[27] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, 2020, arXiv preprint arXiv:2010.11929, https://arxiv.org/abs/2010.11929.

[28] Z. Song, Y. Zhang, A.A.R.M.A. Ebayyeh, Ednet: Edge-optimized small target detection in UAV imagery - faster context attention, better feature fusion, and hardware acceleration, in: 2024 IEEE Smart World Congress, SWC, 2024, pp. 829–838, http://dx.doi.org/10.1109/SWC62898.2024.00141.

[29] J. Pan, A. Bulat, F. Tan, X. Zhu, L. Dudziak, H. Li, G. Tzimiropoulos, B. Martinez, EdgeViTs: Competing light-weight CNNs on mobile devices with vision transformers, in: European Conference on Computer Vision, 2022, pp. 294–311, http://dx.doi.org/10.1007/978-3-031-20083-0_18.

[30] K. Wu, B. Peng, D. Zhai, Boundary-aware axial attention network for high-quality pavement crack detection, IEEE Trans. Neural Netw. Learn. Syst. (2024) 1–12, http://dx.doi.org/10.1109/TNNLS.2024.3497145.

[31] D. Lee, S.J. Kwon, B. Kim, G. Wei, Learning low-rank approximation for CNNs, 2019, http://dx.doi.org/10.48550/arXiv.1905.10145, CoRR abs/1905.10145.

[32] Y. Jiang, Y. Li, Y. Sun, J. Wang, D. Woodruff, Single pass entrywise-transformed low rank approximation, in: Proceedings of the 38th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 139, 2021, pp. 4982–4991.
[33] J.-L. Wang, T.-Z. Huang, X.-L. Zhao, T.-X. Jiang, M.K. Ng, Multi-dimensional visual data completion via low-rank tensor representation under coupled transform, IEEE Trans. Image Process. 30 (2021) 3581–3596, http://dx.doi.org/10.1109/TIP.2021.3062995.
[34] T. Sarlos, X. Song, D. Woodruff, R. Zhang, Hardness of low rank approximation of entrywise transformed matrix products, in: Advances in Neural Information Processing Systems, Vol. 36, 2023, pp. 52568–52582.
[35] J.-L. Wang, T.-Z. Huang, X.-L. Zhao, Y.-S. Luo, T.-X. Jiang, CoNoT: Coupled nonlinear transform-based low-rank tensor representation for multidimensional image completion, IEEE Trans. Neural Netw. Learn. Syst. 35 (7) (2024) 8969–8983.
[36] S. Ahmadi-Asl, An efficient randomized fixed-precision algorithm for tensor singular value decomposition, Commun. Appl. Math. Comput. 5 (4) (2023) 1564–1583.
[37] Z. Fang, F. Qi, Y. Dong, Y. Zhang, S. Feng, Parallel tensor decomposition with distributed memory based on hierarchical singular value decomposition, Concurr. Comput.: Pr. Exp. 35 (17) (2023) e6656.
[38] M. Amiridi, N. Kargas, N.D. Sidiropoulos, Information-theoretic feature selection via tensor decomposition and submodularity, IEEE Trans. Signal Process. 69 (2021) 6195–6205.
[39] N. Razin, A. Maman, N. Cohen, Implicit regularization in tensor factorization, in: Proceedings of the 38th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 139, 2021, pp. 8913–8924.
[40] S. Cheng, Z. Liu, J. Du, Y. You, ATP: Adaptive tensor parallelism for foundation models, 2023, http://dx.doi.org/10.48550/arXiv.2301.08658, CoRR abs/2301.08658.
[41] Y. Lin, W. Chen, S. Chien, MERIT: Tensor transform for memory-efficient vision processing on parallel architectures, IEEE Trans. Very Large Scale Integr. Syst. 28 (3) (2020) 791–804.
[42] N. Kühl, H. Fischer, M. Hinze, T. Rung, An incremental singular value decomposition approach for large-scale spatially parallel & distributed but temporally serial data–applied to technical flows, Comput. Phys. Comm. 296 (2024) 109022.
[43] X. Chen, E.G. Larsson, K. Huang, On-the-fly communication-and-computing for distributed tensor decomposition, in: GLOBECOM 2023 - 2023 IEEE Global Communications Conference, 2023, pp. 1084–1089, http://dx.doi.org/10.1109/TSP.2023.3329974.
[44] K. Yang, Y. Gao, Y. Shen, B. Zheng, L. Chen, DisMASTD: An efficient distributed multi-aspect streaming tensor decomposition, in: Proceedings of the ACM Turing Award Celebration Conference - China 2023, 2023, pp. 127–128.
[45] V.A. Korthikanti, J. Casper, S. Lym, L. McAfee, M. Andersch, M. Shoeybi, B. Catanzaro, Reducing activation recomputation in large transformer models, in: Proceedings of Machine Learning and Systems, Vol. 5, 2023, pp. 341–353.
[46] M. Beaupère, D. Frenkiel, L. Grigori, Higher-order QR with tournament pivoting for tensor compression, SIAM J. Matrix Anal. Appl. 44 (2023) 106–127.
[47] X. Xie, P. Zhou, H. Li, Z. Lin, S. Yan, Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models, IEEE Trans. Pattern Anal. Mach. Intell. 46 (12) (2024) 9508–9520.
[48] L. Guan, AdaPlus: Integrating nesterov momentum and precise stepsize adjustment on adamw basis, in: ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2024, pp. 5210–5214, http://dx.doi.org/10.1109/ICASSP48485.2024.10447337.
[49] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jegou, Training data-efficient image transformers & distillation through attention, in: Proceedings of the 38th International Conference on Machine Learning, Vol. 139, 2021, pp. 10347–10357.
[50] C.-C. Chang, Y.-Y. Sung, S. Yu, N.-C. Huang, D. Marculescu, K.-C. Wu, FLORA: Fine-grained low-rank architecture search for vision transformer, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, WACV, 2024, pp. 2482–2491.
[51] Y. Luo, H. Patel, Y. Fu, D. Ahn, J. Chen, Y. Dong, E.E. Papalexakis, TRAWL: Tensor reduced and approximated weights for large language models, 2025, URL https://arxiv.org/abs/2406.17261.

Yuze Du was born in Nanjing, Jiangsu, P.R. China. She is currently pursuing the master's degree with the School of Electronics and Information, Shanghai Dianji University, China. Her research interests include distributed machine learning, edge computing, industrial intelligent detection and deep learning.

Xiaogang Wang (Member, IEEE) was born in Nanchang, Jiangxi, P.R. China. He received the Ph.D. degree in computer science and technology from Shanghai Jiao Tong University, China, in 2018. He is currently a Professor with the School of Electronics and Information, Shanghai Dianji University, China. He was also the Visiting Research Scholar with the CLOUDS Laboratory, University of Melbourne, Australia, from 2019 to 2020. He has published more than 50 papers in some journals and conferences such as TC, TSC, JSS, FGCS, CC, APIN, WI-IAT, APSCC. His research interests include distributed machine learning, cloud and edge computing, and industrial intelligent detection.

Haokun Chen was born in Luoyang, Henan, P.R. China. He is currently pursuing the master's degree with the School of Electronics and Information, Shanghai Dianji University, China. His research interests include distributed machine learning, edge computing and deep learning.

Chenfeng Zhang was born in Xinxiang, Henan, P.R. China. She is currently pursuing the master's degree with the School of Electronics and Information, Shanghai Dianji University, China. Her research interests include distributed machine learning, edge computing, video analysis and deep learning.

Jian Cao (Senior Member, IEEE) was born in Yixing, Jiangsu, P.R. China. He received the Ph.D. degree from the Nanjing University of Science and Technology, in 2000. He is currently a professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His main research interests include service computing, cloud computing, cooperative information systems and software engineering. He has published more than 200 papers in prestigious journals.

Rajkumar Buyya (Fellow, IEEE) received the Ph.D. degree in computer science and software engineering from Monash University, Melbourne, Australia, in 2002. He is a Redmond Barry distinguished professor and director of the CLOUDS Laboratory, the University of Melbourne, Australia. He has authored more than 625 publications and seven textbooks. He is one of the highly cited authors in computer science and software engineering worldwide (h-index=155, g-index=334, 126,300+ citations). He served as the founding editor-in-chief of the IEEE Transactions on Cloud Computing. He is currently serving as co-editor-in-chief of Journal of Software: Practice and Experience.