Available online at sjuoz.uoz.edu.krd

Science Journal of University of Zakho

Vol. 13, No.1 pp1–6 January-March,2025

journals.uoz.edu.krd

p-ISSN: 2663-628X
e-ISSN: 2663-6298

DEEP NEURAL NETWORK-BASED APPROACH FOR COMPUTING SINGULAR VALUES OF MATRICES

Diyari A. Hassan*

Faculty of Engineering & Computer Science, Qaiwan International University Sulaymaniyah, Kurdistan Region-Iraq

*Corresponding author: diyari.hassan@uniq.edu.iq

Received: 29 Jul 2024 / Accepted:5 Nov., 2024 / Published: 5 Jan., 2025. https://doi.org/10.25271/sjuoz.2024.12.3.1345

ABSTRACT:

Matrix factorization techniques, such as Singular Value Decomposition (SVD), Eigenvalue Decomposition (EVD), and QR decomposition, have long been pivotal in computational mathematics, particularly for applications in signal processing, machine learning, and data analysis. With the growing size and complexity of data, traditional methods of matrix factorization face challenges in efficiency and scalability. This paper investigates the implementation of Convolutional Neural Networks (CNNs) for computing the singular values of both real and complex matrices. By leveraging the hierarchical feature extraction capabilities of CNNs, this approach aims to enhance the accuracy, efficiency, and scalability of SVD calculations. The proposed CNN-based SVD method is evaluated against the conventional SVD algorithm, demonstrating superior performance in terms of computational time and accuracy.

KEYWORDS: Singular Value Decomposition, Matrix Factorization, Convolutional Neural Networks, Computational Compelxity.

1. INTRODUCTION

At the core of computational mathematics, matrix factorization stands out among others as the basis for many applications. The singular value decomposition (SVD), Eigen value decomposition (EVD) and the QR decomposition have been the basic building blocks of popular matrix factorization methods for the last decades due to of their reliability and versatility (Golub & Van Loan, 2013). SVD breaks a matrix into three other matrices retaining the core of the matrix and has wide uses in various fields like data compression, signal processing and machine learning mainly for reducing its dimension and removing noises. In PCA, SVD plays the most important part to recognize the major patterns hidden in vast dimensional vectors . EVD, on the other hand, diagonalizes a square matrix into eigenvalues and eigenvectors; which are useful in solving systems of linear equations, stability analysis and optimisation problem. It is also used in quantum mechanics, vibration analysis and the analysis of network dynamics (Hogben 2006). Both the SVD and EVD play a significant role in promoting research in large dataset analysis by offering methodologies that can help to extract significant information while alternately reducing huge dataset's intricacy, or improving algorithms' performance in data-based science. On the other hand, as the size and complexity of data expands, it becomes more essential to use new and innovative methods of calculating matrix factorization, particularly for matrices of large size (Zhang, 2017).

Recently, there has been a remarkable surge in using deep neural networks (DNNs), and specifically convolutional neural networks (CNNs) to solve machine learning problems such as signal recognition (Mala & Mohammad, 2022), image analysis (Anwar et. al., 2018), and natural language processing (Hassan & Taher,2022). The domains frequently work with the complex datastores that are arranged in the form of grids (Rawat & Wang, 2017). The ability of deep CNNs to outperform traditional mathematical models and achieve over 98% prediction accuracy can be credited to the fact that they were initially created for image classification task (LeCun, 1998). Surprisingly, they have overcome human-level performance in lots of sectors. As an example, in speech recognition, conventional methods have been overshadowed by statistical learning methods which incorporate convolutional neural network architecture, which recently achieved human-level accuracy (Finol *et al*., 2019).

Traditional matrix factorization is computationally intensive, particularly when solving problems involving high-dimensional datasets, making it less suitable for real-time applications or big data. Using neural networks for computing matrix decompositions helps mitigate these issues by take advantage of the parallelism and efficiency of deep learning architectures. Moreover, the scalability of neural network approaches allows for efficient handling of larger datasets without the exponential increase in resource requirements typically associated matrix factorization (Nguyen *et al*., 2018). In particular, the CNN-based approach significantly reduces computational complexity while improving the accuracy of the matrix decomposition process.

The primary contribution of this paper is the development of a CNN model capable of computing the singular values for matrices with real and complex scalar entries. The performance of this model is evaluated in terms of accuracy and computational complexity, and is compared to that of the conventional SVD algorithm.

The structure of this paper is as follows: Section 2 reviews the related works. Section 3 discusses the conventional (SVD) method, while Section 4 introduces the neural network approach for computing SVD. The simulation results are presented in Section 5, and the conclusions are outlined in Section 6.

* Corresponding author

Related Works

The applicability of neural network-based algorithms for matrix eigenvalue calculations was explored by (Yi, & Tang, 2004), where they introduced a continuous time recurrent neural networks (RNN) which can be applied only to symmetric matrices. There are two main limitation of their proposed approach; firstly, the approach is valid only for non-symmetric matrices and secondly, the computational complexity of the task is still high. Similarly, in another study conducted by (Zou *et al*., 2013), a recurrent neural network (RNN) was presented for the Eigendecomposition of actual normal matrices. As the proposed method is primarily focused on real normal matrices, its applicability to other types of matrices can be somewhat restricted. Despite the fact that the method can find n-dimensional complex eigenvectors, the computational complexity remains a problem especially when dealing with large matrices. Building on these ideas, (Hang *et al*., 2013) extend the use of neural networks for estimating the generalized EVD (GEVD) algorithm. However, the proposed method does not applicable for non-singular matrices or rank-deficient cases. In addition, this method is relevant for computing only the greatest and smallest eigenvalues.

In a different approach, Han, Lu, and Zhou, (2020) used deep neural network to develop a new method to solve high-dimensional eigenvalue problems by transform the problem into parabolic equation. This approach is the hybrid of the diffusion Monte Carlo (DMC) and the neural network learning of the eigenfunctions, which allows for efficient and accurate estimation of the potential energy eigenvalue and the corresponding eigenfunction or their gradient. Also, the direct use of eigenvalue calculation has been explored by Finol et al, (2019), where propose deep CNNs designed for solving eigenvalue problems used in mechanics in the fields of one-dimensional and two-dimensional phononic crystals. As this paper shows, CNNs are more accurate and data efficient for predicting eigenvalues than fully connected neural networks (MLPs). Furthermore, Hu, (2022) pointed out that deep learning network-based matrix eigenvalue algorithms are able to calculate much faster than traditional algorithms (17% faster in the matrix range of 4×4).

Conventional Singular Value Decomposition (Svd)

SVD is a very useful matrix factorization technique that is used in multiple computational tasks in varied domains. It gives a fascinating and in-depth picture of a matrix. It helps to explore its inner structure and thus to be applied in different fields like signal processing, image analysis, machine learning and scientific computing. SVD has become the most efficient and robust numerical way for diagonalization of any size matrices, including $M \times N$ and having complex scalar coefficients, and the transformation to the canonical basis is the final point. Different from the EVD, the SVD doesn't require the input matrix to be Hermitian for the basis of diagonal decomposition (Golub & Van Loan, 2013).

At its core, SVD decomposes a given matrix $\boldsymbol{A}$ into three constituent matrices, each offering valuable insights into the original matrix:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

where:

- $\boldsymbol{U}$ is an $m \times m$ orthogonal matrix containing the left singular vectors of $\boldsymbol{A}$.
- $\boldsymbol{\Sigma}$ is an $m \times n$ diagonal matrix with non-negative real numbers (singular values) on its diagonal, arranged in descending order.
- $\boldsymbol{V}^T$ is the transpose of an $n \times n$ orthogonal matrix $\boldsymbol{V}$, containing the right singular vectors of $\boldsymbol{A}$.

The orthonormal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ can reflect the input and output spaces' rotations and reflections, respectively, while $\boldsymbol{\Sigma}$ the diagonal matrix captures the scale factors along each dimension. The SVD has some important properties and interpretations. The magnitudes of singular vectors in $\boldsymbol{U}$ and $\boldsymbol{V}$ are represented by singular values on the diagonal matrix $\boldsymbol{\Sigma}$, which are ordered in decreasing order by importance with the first singular value indicating the major direction of data variation. The rank of matrix $\boldsymbol{A}$ is equivalent to the number of non-zero singular values in $\boldsymbol{\Sigma}$, hence rank computation and low-rank matrix approximation by retaining the largest singular values and their corresponding vectors is possible. Therefore, both $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices and their column vectors remains orthogonal and maintain the original matrix orthogonality. Although exact reconstruction of matrix $\boldsymbol{A}$ from SVD includes multiplying three constituent matrices, many applications involve keeping a subset of the singular values and vectors for an approximation in lower dimensions (Strang, 2022).

The Golub-Reinsch algorithm, also known as the SVD by Bidiagonalization and Golub-Kahan-Reinsch algorithm, is a common and easy-to-use technique that is commonly used in finding the SVD for matrices. This algorithm involves several steps: the process of bidiagonalization starts by applying Householder reflections to transform the initial matrix into bidiagonal form. If the matrix is already square, it further tridiagonalizes by performing Givens rotations making the matrix a tridiagonal form. Secondly, an iterative QR algorithm diagonalizes the tridiagonal matrix by performing orthogonal similarity transformations until a converged diagonal or nearly diagonal form results. At the end, the singular values are pulled out from the elements through the main diagonal of the obtained matrix, and singular vectors can be rebuilt through the unitary transformations implemented during bidiagonalization (Hogben, 2006; Wu & Tsai, 2019).

Neural Network For Computing SVD

This section introduces a deep neural network structure specifically designed for computing SVD, with particular emphasis on its application in CNNs. The details of SVD computation using CNNs are explored, including the selection of data and the configuration of CNN parameters.

Convolutional Neural Networks (CNN)

CNNs are significant in deep learning networks that have been specifically created for the complex process of image and video processing. Due to their hierarchical architecture comprising of layers of cells, the CNNs are uniquely able to automatically detect and extract patterns and features from input data without human involvement. Our native aptitude in this field is the reason why CNNs are highly in demand in different applications like image classification, object detection and facial recognition. CNNs use convolutional layers to extract prominent features from the input data, and pooling layers are responsible for the reduction of spatial dimensions which increases neural network efficiency. Then, the fully connected layers incorporate

these features to finally produce the correct predictions or classifications. The extensive implementation and continued development of CNNs has not only reshaped the outlook of computer vision, but it has also established them as a fundamental technology of deep learning, which is a comprehensive domain (Albawi *et al*., 2017; Li *et al*., 2021).

SVD-Based CNN

The proposed CNN network architecture takes the singular values obtained from the diagonalized matrix $\boldsymbol{\Sigma}$, which are the result of applying the conventional SVD algorithm to a scalar A matrix as inputs. In order to maintain the computational efficiency and avoid the overfitting of the model, the feedforward neural network (FNN) component is applied in the prediction phase that consists of multiple layers such as input, hidden (fully connected layers) and output layers. In the proposed implementation the fully connected layers utilize two hidden layers with 32 and 16 neurons each that help to identify complex dependencies within the matrices and provide for the extraction of features at multiple abstraction levels, facilitating learning. ReLU activation functions are rectified in the hidden layers to introduce non-linearity and increase the model's capacity of cathing complex patterns, and at the output layers, the linear activation functions are used to facilitate regression and singular values estimation. In this CNN-based SVD architecture, the predicted values are iteratively compared to real ones using MSE to minimize error and evaluate the model effectiveness. As shown in Figure 1 below, the block diagram illustrates the proposed CNN-based SVD architecture, used for both training and implementation. The CNN undertakes hierarchical feature extraction using convolutional and pooling layers, whereas FNN uses multiple hidden layers to carry out complex decision-making to predict the singular values as obtained from conventional SVD. The CNN, on the other hand, has two convolutional layers with ReLU activation functions that have 32 and 16 filters, respectively; each layer extracts features from the input matrices A. In addition, max-pooling layers with a size of (2×2) and a stride of (2×2) are applied to convolve the extracted features, thereby enhancing computational efficiency and retaining important information. Algorithm 1 provides the pseudocode for implementing this architecture.

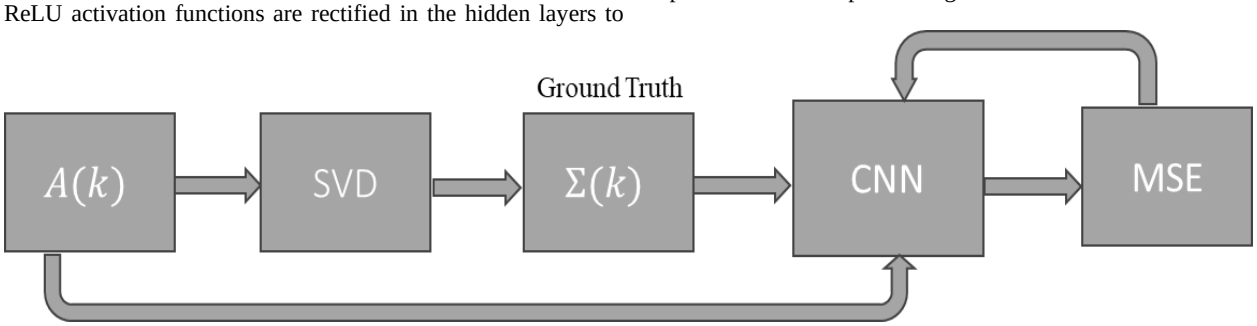

Figure 1: Proposed CNN- based SVD architectures

As shown in Figure 1, this process starts with a K set of matrices. SVD applied to each matrix, $\boldsymbol{A}(k)$ separately in the data set for the purpose of extraction of singular values $\boldsymbol{\Sigma}(k)$. Furthermore, the CNN is trained using these matwrices as inputs, and their singular values as outputs. The network "learns" to reduce the Mean Square Error (MSE) until it converges to the expected level. Trained with this network, it can efficiently evaluate the singular values for any matrix $\boldsymbol{A}$ applied.

Algorithm 1: CNN-based SVD for calculating Singular Value Decomposition for matrices with real or complex entries.

Input:

- Matrix $\boldsymbol{A}_{nx\,m}$ with real or complex values
- Training data (matrix, singular value pairs)
- CNN model architecture

Output:

- Singular values of Matrix $\boldsymbol{A}$

1**. Preprocess The Input Matrix**:

a. Normalize Matrix A.

b. Reshape Matrix A into a suitable form for CNN input (e.g., a 2D or 3D tensor).

c. If necessary, augment the input with additional channels (e.g., real and imaginary parts for complex matrices).

2. **Define Cnn Model Architecture**:

a. Input layer: Takes in the reshaped matrix.

b. Convolutional layers:

i. Apply multiple convolutional filters to capture spatial dependencies.

ii. Use ReLU activation after each convolution.

c. Pooling layers:

i. Apply max pooling to reduce dimensionality.

d. Fully connected layers:

i. Flatten the output of the convolutional and pooling layers.

ii. Apply fully connected (dense) layers to map the output to singular values.

e. Output layer:

i. Predict the singular values corresponding to Matrix A.3.

Train The CNN Model:

a. Initialize the model with randomly assigned weights.

b. Use a training dataset of matrix, singular value pairs.

c. Define loss function:

i. Use mean squared error (MSE) between the predicted and true singular values.

d. Choose an optimization algorithm (e.g., Adam or SGD).

e. Train the model by minimizing the loss function using backpropagation.

4. **Test The CNN Model**:

a. Input Matrix A into the trained CNN model.

b. Obtain the predicted singular values.

5. **Post-Process The Output:**

a. Convert the predicted output to the correct scale (if necessary, based on normalization).

b. Return the predicted singular values.

End.

2. RESULTS AND DISCUSSION

To demonstrate the benefits of the proposed artificial neural network (ANN) architecture, the results of the SVD-based CNN and the computational load required for their generation will be presented. Using MATLAB's Deep Learning Toolbox, the performance of the envisioned CNN architecture was developed and observed. The model was trained and tested on 10,000 samples from a synthetically generated dataset in the form of 5x5 matrices, created from zero-mean, independent, and identically distributed (i.i.d.) random processes. Conventional singular value decomposition was used to compute the singular values for testing the proposed model. For adaptive learning, the Adagrad (Adaptive Gradient Algorithm) optimizer was applied, with an initial learning rate of 0.01, and the maximum number of epochs set to 50. Training and testing were conducted over 50 epochs, resulting in an average Mean Squared Error (MSE) value of 0.085, as shown in Figure 2.

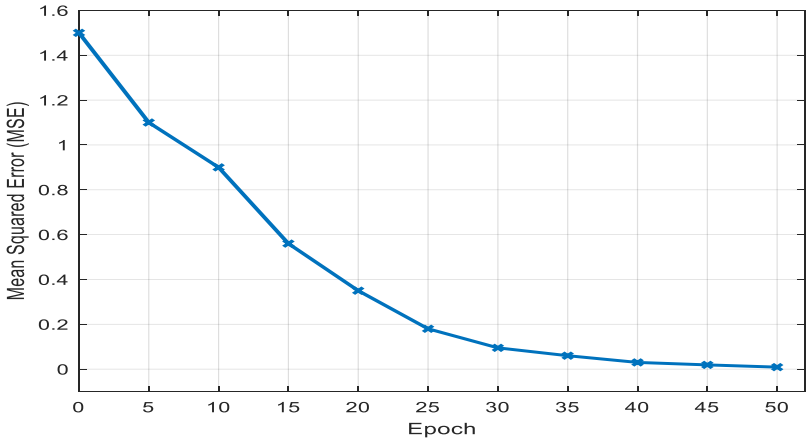

Figure 2: The MSE accuracy of the SVD calculation is assessed using CNN, as a function of the number of epochs.

An important factor that should be taken in consideration is the computational complexity of the presented SVD-based CNN algorithm since it limits its application particularly in the environment that uses a regular CPU. Thus, the results of the SVD-based CNN will be presented, along with an analysis of the number of computations required to obtain this solution.

The evaluation of proposed method covers the CPU time required to solve the problem with the aid of the proposed method in contrast to standard SVD methods. CPU time in seconds denotes the time, which the processor requires to perform all the necessary computations to execute the algorithm on the given data set or problem size. This measure considers the computational cost, continuity of the operations and the hardware resources used (Hsu & Kremer, 2003; Hsia et al., 2021).

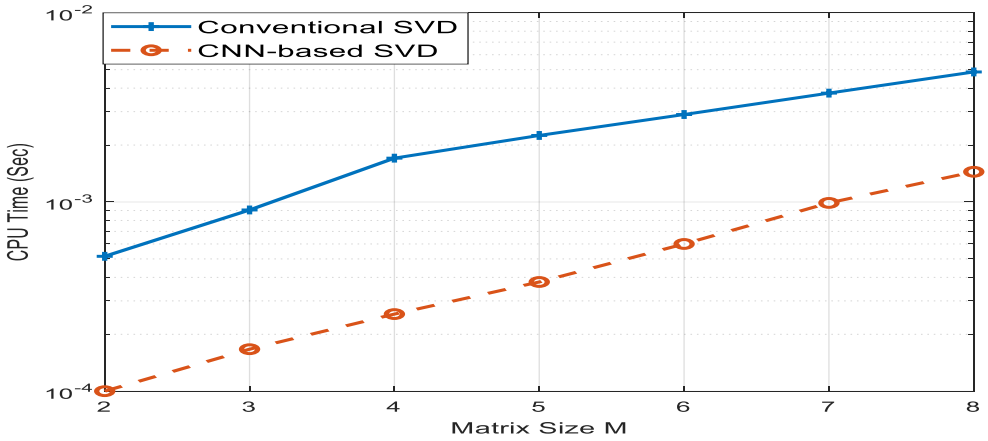

Figure 3: Time complexity analysis for calculating singular values using both CNN-based SVD and the conventional SVD algorithm across varying input matrix sizes.

Figure 3 presents an evaluation of the time complexity involved in calculating the singular values of matrices of varying sizes. The data clearly shows that larger matrices require more computational time. Furthermore, it is apparent that the traditional SVD algorithm takes significantly longer to compute singular values compared to the neural network-based approach.

As observed, the majority of the previous research has centered on the application of neural networks for approximating the eigenvalues of matrices. However, the algorithm proposed in this paper turns its attention to the estimation of singular values, which is a different challenge and offers unique applications. Table 1 indicates a summary of the distinguish methods used in various studies on the selected themes. This table provides a comparison of neural network models employed in matrix decomposition strategies, reflecting the continuously increasing attention of researchers to the machine learning-based calculations of matrices.

Table1: Neural Network Techniques and Matrix Decomposition Types Used in Various Studies

Paper	Neural Network Technique	Type of Matrix Factorization
Yi, Z., Fu, Y., and Tang, H. J., 2004	Continuous-Time Recurrent Neural Networks	Eigenvalue Decomposition
Zou, X., *et al*., 2013	Recurrent Neural Network	Eigenvalue Decomposition
Hang, T., *et al*., 2013	Neural Network for Generalized EVD	Generalized Eigenvalue Decomposition
Han, J., Lu, J., and Zhou, M., 2020	Deep Neural Networks with Diffusion Monte Carlo	High-Dimensional Eigenvalue Problems
Finol, D., *et al*., 2019	Deep Convolutional Neural Networks	Eigenvalue Decomposition for 1D and 2D phononic crystals in mechanics
Hu, Z., 2022	Deep Learning-Based Matrix Eigenvalue Algorithm	Eigenvalue Decomposition
Proposed Approach	Deep Convolutional Neural Networks	Singular Value Decomposition for matrices with real or complex entries.

CONCLUSION

This paper presents a new approach to SVD using CNNs, addressing the limitations of traditional SVD methods in handling large and complex matrices. The CNN-based SVD method exhibits significant improvements in computational efficiency and accuracy, as demonstrated through rigorous testing on synthetic datasets. The proposed architecture effectively leverages CNNs' hierarchical feature extraction capabilities and advanced optimization techniques to achieve high precision in singular value calculations. These findings highlight the potential of integrating deep learning frameworks into classical matrix factorization tasks, paving the way for more efficient and scalable computational methods in various applications such as signal processing, machine learning, and data analysis. Future work will focus on further optimizing the CNN architecture and exploring its applicability to other matrix factorization techniques.

REFERENCE

Albawi, S., Bayat, O., Al-Azawi, S., & Ucan, O. N. (2018). Social touch gesture recognition using convolutional neural network. Computational Intelligence and Neuroscience,2018(1),6973103.DOI:https://doi.org/10.1155/2018/6973103

Anwar, S. M., Majid, M., Qayyum, A., Awais, M., Alnowami, M., & Khan, M. K. (2018). Medical image analysis using convolutional neural networks: a review. *Journal of medicalsystems*,42,1-13.DOI: https://doi.org/10.1007/s10916-018-1088-1

Finol, D., Lu, Y., Mahadevan, V., & Srivastava, A. (2019). Deep convolutional neural networks for eigenvalue problems in mechanics. *International Journal for Numerical Methods in Engineering*, 118(5), 258-275. DOI: https://doi.org/10.1002/nme.6012

Golub, G. H., & Van Loan, C. F. (2013). Matrix computations. *JHUpress*.DOI: https://doi.org/10.1137/1.9781421407944

Han, J., Lu, J., & Zhou, M. (2020). Solving high-dimensional eigenvalue problems using deep neural networks: A diffusion Monte Carlo like approach. *Journal of ComputationalPhysics*,423,109792.DOI: https://doi.org/10.1016/j.jcp.2020.109792

Hang, T., Yang, G., Yu, B., Liang, X., & Tang, Y. (2013). Neural network based algorithm for generalized eigenvalue problem. *In 2013 International Conference on Information Science and Cloud Computing Companion* (pp.446-451).IEEE.DOI: https://doi.org/10.1109/ISCC-C.2013.93

Hogben, L. (Ed.). (2006). Handbook of linear algebra. *CRC press*. DOI: https://doi.org/10.1201/9781420010572

Hassan, M. M., & Taher, S. A. (2022). Analysis and classification ofautismdata using machine learning algorithms. *Science Journal of University of Zakho*, *10*(4),206-212.DOI: https://doi.org/10.25271/sjuoz.2022.10.4.1036

Hsia, S. C., Wang, S. H., & Chang, C. Y. (2021). Convolution neural network with low operation FLOPS and high accuracy for image recognition. *Journal of Real-Time Image Processing*, 18(4), 1309-1319 DOI: https://doi.org/10.1007/s11554-021-01140-9

Hsu, C. H., & Kremer, U. (2003). The design, implementation, and evaluation of a compiler algorithm for CPU energy reduction. *In Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*(pp.38-48).DOI: https://doi.org/10.1145/781131.781137

Hu, Z. (2022). Estimation and application of matrix eigenvalues based on deep neural network. *Journal of Intelligent Systems*,31(1),1246-1261.DOI: https://doi.org/10.1515/jisys-2022-0126

LeCun, Y. (1998). The MNIST database of handwritten digits. http://yann. lecun. com/exdb/mnist/.

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12), 6999-7019. DOI: 10.1109/TNNLS.2021.3084827

Mala, Y. H., & Mohammad, M. A. (2022). Brain Waves Signal

Modeling for Object Classification Using Random Forest Method. *Science Journal of University of Zakho*, 10(1), 16-23. DOI:https://doi.org/10.25271/sjuoz.2022.10.1.876

Nguyen, D. M., Tsiligianni, E., & Deligiannis, N. (2018). Matrix factorization via deep learning. *arXiv preprint* arXiv:1812.01478.DOI:https://doi.org/10.48550/arXiv.1812.01478

Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9), 2352-2449. DOI: 10.1162/neco_a_00990

Strang, G. (2022). Introduction to linear algebra. *Wellesley-Cambridge Press*. DOI: 10.4236/jamp.2020.87104

Wu, C. H., & Tsai, P. Y. (2019). An SVD processor based on Golub–Reinsch algorithm for MIMO precoding with adjustable precision. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(7), 2572-2583. DOI: 10.1109/TCSI.2019.2899211

Yi, Z., Fu, Y., & Tang, H. J. (2004). Neural networks based approach for computing eigenvectors and eigenvalues of symmetric matrix. *Computers & Mathematics with Applications*,47(8-9),1155-1164.DOI:https://doi.org/10.1016/S0898-1221(04)90110-1

Zhang, X. D. (2017). Matrix analysis and applications. *CambridgeUniversityPress*.DOI:https://doi.org/10.1017/9781108277587

Zou, X., Tang, Y., Bu, S., Luo, Z., & Zhong, S. (2013). Neural-network-based approach for extracting eigenvectors and eigenvalues of real normal matrices and some extension to real matrices. *Journal of Applied Mathematics*, 2013. DOI:https://doi.org/10.1155/2013/597628