ISSN 2616-5775 Vol. 7, Issue 4: 29-35, DOI: 10.25236/AJCIS.2024.070405

Importance of matrix operations in artificial neural networks

Jinghan Kang[1,a], Junxi Li[2,b], Nana Guo[3,c], Xin Guo[1,d,*]

[1]Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russian Federation
[2]Herzen University, St. Petersburg, Russian Federation
[3]Gansu Agricultural University, Gansu, China
[a] kan.ts@edu.spbstu.ru, [b]lijunxi111@yandex.com, [c]NnG.cn@outlook.com, [d]go7.s@edu.spbstu.ru
[*]Corresponding author

Abstract: *In this paper, the importance of matrices in artificial neural networks is studied. It is mainly argued from three architectures, traditional convolutional neural network, attention mechanism network, and generative adversarial network. The results show that various mathematical operations on matrices and transformations of matrices are particularly important in artificial neural networks.*

Keywords: *Linear Planning; Matrix Operations; Artificial Neural Networks*

1. Introduction

The importance of matrices is self-evident, and the mainstream directions of artificial intelligence are mainly natural language processing, computer vision and speech processing, which are all very closely related to matrices. Currently there are three kinds of neural networks used most: convolutional neural network (Cnn & Rnn) and attention mechanism network (Transform) generative adversarial network (Gan) [1]. For traditional convolutional neural networks, if we focus more on structural features when designing the model, it is better to choose Cnn networks; on the contrary, if we focus more on temporal features, we are better to choose Rnn networks. If we are facing a machine translation task, we need to choose an attention mechanism network. If we need to complete the generative task, we need to choose Generative Adversarial Network.

Matrices are essentially two-dimensional arrays, a form of organization of data, common deformations of matrices are diagonal decomposition of matrices, triangular decomposition, triangular diagonal decomposition, feature extraction, diagonalization of matrices, these operations are based on the equivalence of matrices, the contract of matrices, and the similarity of matrices, and have a wide range of applications in artificial intelligence [2].

The matrix can already be seen in the most primitive neuronal structures [3]. Matrices are gradually coming into the AI arena along with the concept of neural networks. Matrices are widely used in several disciplines, such as matrix theory in mathematics, medical image processing, and face recognition. In short matrices exist in all corners of our lives.

2. Matrix Operations for Convolutional Neural Network and Recurrent Neural Network

In the field of CNN networks and image processing, matrix convolution is commonly employed to calculate image features. There exist two types of matrix convolution: full convolution and RMS convolution [4].

The full convolution is defined as:

$$z(u,v) = \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} x_{ij} \cdot k_{u-i,v-j} \tag{1}$$

Suppose X is an m x m matrix, K is an n x n matrix, and K_{rot} is obtained by rotating K by 180°. The effective value convolution is defined as:

ISSN 2616-5775 Vol. 7, Issue 4: 29-35, DOI: 10.25236/AJCIS.2024.070405

$$z(u,v) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} x_{i+u,j+v} \cdot k_{rot\,i,j} \cdot X(i,j) \tag{2}$$

$$X(i,j) = \begin{cases} 1, & 0 \le i,j \le n \\ 0, & others \end{cases} \tag{3}$$

The convolution operation can be performed in two steps: first, the convolution filter is multiplied by each element of the input matrix slice; and in the second step, the sum is computed over all values in the resulting product matrix.

The 5x5 input matrix is shown in Figure 1:

128	36	52	221	134
35	27	25	200	187
37	27	24	28	181
37	25	28	193	178
42	56	132	192	177

Figure 1: 5*x*5 matrix

Now, we use the 2x2 convolutional filter shown in Figure 2:

1	0
0	1

Figure 2: 2x2 Convolutional Filter

Each convolution operation involves a single 2x2 slice of the input matrix. For example, if we consider a 2x2 slice starting from the upper left corner of the input matrix, the convolution operation performed on this slice is shown in Figure 3:

128	36	X	1	0	=>	128	0	=	128 + 27 =155
35	27		0	1		0	27		

Figure 3: Convolution operation

The convolutional layer consists of a series of convolutional operations, each of which acts on a different slice of the input matrix.

First, we need to understand sequence data, sequence data is a sequence of points organized in a temporal way, it does not necessarily need a time scale, it is only organized in a chronological relationship to the data. For example, our common DNA data and voice data, as well as our text data and the K-line in the inventory, are all sequence data.

A recurrent neural network is a type of neural network that is designed to be run multiple times, with the output of each run serving as input for the next run. In an RNN, the hidden layer from the previous run contributes to the input of the same hidden layer in the next run [5].

When RNNs networks deal with tasks of temporal order, they generally go to step t-1 first, then deal with the task at step t, and finally deal with the task at step t+1.

$$f_{RNN} = \tanh\left(W_{ih}x_t + b_{ih} + W_{hh}h_{(t-1)} + b_{hh}\right) \tag{4}$$

ISSN 2616-5775 Vol. 7, Issue 4: 29-35, DOI: 10.25236/AJCIS.2024.070405

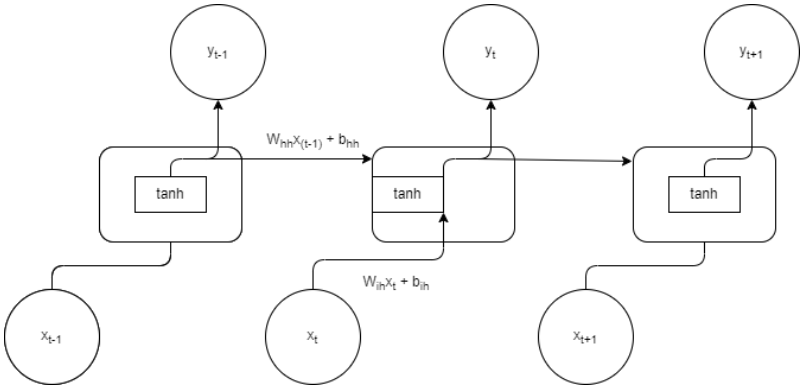

Figure 4: Recurrent neural network

As shown in Figure 4, there is a sequence of data processing at each time point., but it is not independent, the input of the current node is based on the temporal characteristics of the output of the previous node, which also reflects the memory function of the recurrent neural network [6].

$$\begin{bmatrix} 0.5 \\ 3.6 \\ 15. \end{bmatrix} \times \begin{bmatrix} 0. \\ 0.9 \\ 0.1 \end{bmatrix} = \begin{bmatrix} 0. \\ 3.24 \\ 1.5 \end{bmatrix} \tag{5}$$

The first matrix represents the original data and the second matrix represents the matrix processed by the sigmoid function; the data processed by the sigmoid function will all be between [0, 1]. Of course, the multiplication of the two matrices has a special meaning, the result of the first row indicates that the signal is lost or forgotten; the result of the second row of the pair indicates that the signal is enhanced; the data in the last row indicates that the signal is suppressed [7-8].

3. Matrix Operations for Attention Neural Network

In Attention Mechanism Neural Networks, the first thing we are going to explore is the matrix inner product operation that underlies the formulation of the self-attention mechanism [9].

$$Softmax(XX^T) \tag{6}$$

Equation 6 above is the original form of the self-attention mechanism, and the XX^T in parentheses can be seen as a matrix multiplied by its own transpose matrix, which of course we can see as the inner product of the vectors of the current matrix with the vectors of its transpose matrix respectively.

The geometric meaning of the inner product of vectors can characterize the angle between two directions, and when one vector is a unit vector and the other is not, it is expressed as the projection of the other vector onto the unit vector [10].

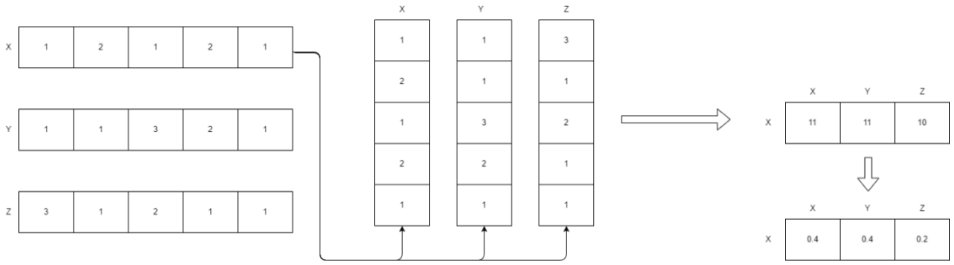

Figure 5: Schematic diagram of matrix calculation

X, Y and Z in Figure 5 represent a Chinese character, X, Y, Z are all vectors after embedding, firstly, X does inner product with itself and the other two vectors respectively to get a new vector. The matrix XX^T is a new matrix expanded from the perspective of inner product of row vectors, the matrix XX^T saves the result of inner product operation of each vector with itself and all the vectors of other matrices.

The softmax function in the attention mechanism neural network performs a normalization operation on our new matrix XX^T, and after the softmax operation, the values of the row vectors sum to 1.

ISSN 2616-5775 Vol. 7, Issue 4: 29-35, DOI: 10.25236/AJCIS.2024.070405

$$Attention(Q,K,V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (7)$$

The Q, K, and V matrices in Eq. 7 above are actually the product of X and the parameter matrices W^Q, W^K, and W^V, which is essentially a linear variation of X.

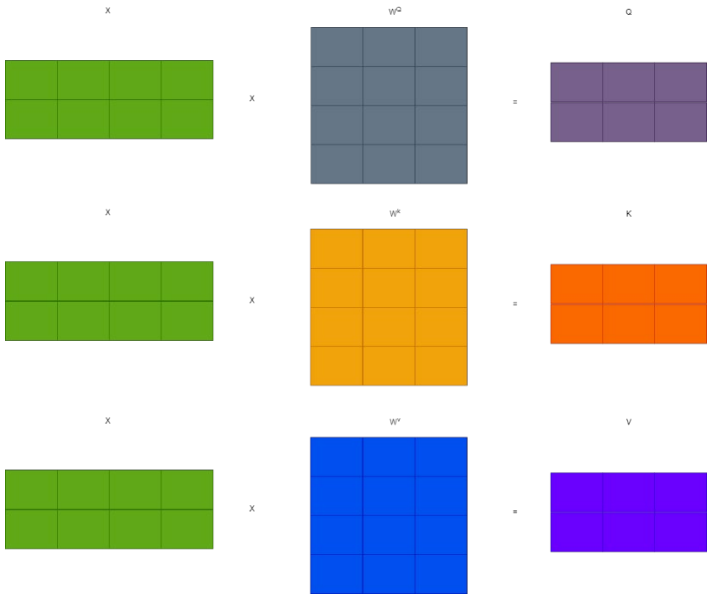

Figure 6: Operations of matrix X with parameter matrices W^Q, W^K, W^V

W^Q, W^K, and W^V of Figure 6 are all linear variations of the matrix X. The reason for not using X directly but using linear variations of X is to improve the fit of the model, and the parameter matrices are all trainable and act as a buffer.

4. Matrix Operations for Generative Adversarial Network

Generative Adversarial Network is a generative model based on game theory, which is widely used in the field of image generation. GAN consists of a generative network and a discriminative network, where the generative network automatically generates data and the discriminative network determines whether the data belongs to the data generated by the generative network [11].

The goal of learning is to build generative networks that can automatically generate data with the same distribution as the training data already given. The process of learning is also the process of gaming, where the generative and discriminative networks are constantly gaming by optimizing their network parameters. When the equilibrium state is reached, the learning ends and the generative network can generate data that is false and the discriminative network has difficulty in determining whether the data is true or false.

Suppose that the training data D has been given to follow the distribution $P_{data}(x)$, where x is a sample, and the generative network is denoted by $x = G(z; \theta)$, where z is the input matrix, x is the output matrix, i.e., the generated data, and θ is the network parameters.

A discriminative network is a two-class classifier, denoted by $P(1|x) = D(x; \varphi)$, where x is the input matrix, P(1|x) and 1-P(1|x) are the probabilities of the outputs, denoting the probability that the inputs x come from the training data and the generated data, respectively, and φ is a network parameter.

The seed matrix follows a distribution $P_{seed}(z)$, such as a standard normal distribution or a uniform distribution. The data distribution generated by the generative network is denoted as $P_{gens}(x)$, which is jointly determined by $P_{seed}(z)$ and $x = G(z; \theta)$.

ISSN 2616-5775 Vol. 7, Issue 4: 29-35, DOI: 10.25236/AJCIS.2024.070405

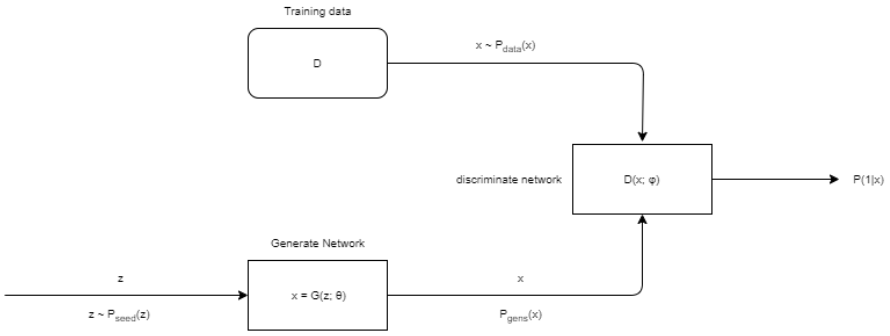

Figure 7: Architecture diagram for generating adversarial networks

If the generative network parameters θ are fixed, the discriminative network parameters φ can be learnt by maximizing the objective function so that it has the ability to discriminate between true and false data.

$$\max_{\varphi} = \{E_x \sim P_{\text{data(x)}} [\log D(x;\ \varphi)] + E_z \sim P_{\text{seed(z)}} [\log(1 - D(G(z;\ \theta);\ \varphi] \} \tag{8}$$

If the discriminative network parameters φ are fixed, then the generative network parameters θ can be learnt by minimizing the objective function, giving it the ability to generate data in a fake way.

$$\min_{\theta} = \{E_z \sim P_{\text{seed(z)}} [\log(1 - D(G(z;\ \theta);\ \varphi] \} \tag{9}$$

The discriminative and generative networks form a game, which can be defined as the very small very large problem, which is the objective function of generating an adversarial network.

$$\max_{\varphi} \min_{\theta} = \{E_x \sim P_{\text{data(x)}} [\log D(x;\ \varphi)] + E_z \sim P_{\text{seed(z)}} [\log(1 - D(G(z;\ \theta);\ \varphi] \} \tag{10}$$

The solutions φ^* and θ^* of the very small and very large problem exist, that is, the Nash equilibrium exists, so the algorithm of GAN network learning is the process of solving the optimal solution of the very small and very large problem.

Using the generative adversarial network architecture in Figure 7, the adversarial training process is used in order to automatically generate image data.

In this context, transposed convolution is particularly important. Transposed convolution is widely used in image generating networks, image autoencoder models, convolution can be used to reduce the size of the image data and dedicated convolution can be used to increase the size of the image data, such processes are also called down sampling and up sampling.

The convolution operation can be expressed as a linear transformation, assuming a kernel matrix W with padding 0 and step size.

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \tag{11}$$

In the process of convolution, its input is the vector of the input matrix expansion and the output is the vector of the output matrix expansion, this linear transformation corresponds to the signaling from the former to the latter layer of the neural network and the upper convolution operation is represented in the process of linear transformation. On the other hand, the linear transformation of the transpose matrix of the original matrix is signaling from the latter to the former layer of the neural network.

$$rot180(W) = \begin{bmatrix} w_{33} & w_{32} & w_{31} \\ w_{23} & w_{22} & w_{21} \\ w_{13} & w_{12} & w_{11} \end{bmatrix} \tag{12}$$

This transposed convolution is a convolution operation with a kernel matrix of rot180(W), a padding of 2, and a step size of 1. Here rot180 means 190 degrees of rotation, and the convolution is computed with full padding of the input matrix.

The primitive convolutional kernel transposed convolution is a reciprocal but not inverse operation, and between the two layers of a convolutional neural network, both forward and backward propagation are convolutional operations, corresponding to each other and in opposite directions. Given any convolution with W as the kernel matrix, a transposed convolution with rot180(W) as the kernel matrix

ISSN 2616-5775 Vol. 7, Issue 4: 29-35, DOI: 10.25236/AJCIS.2024.070405

can be constructed.

So, we can get that rot180(rot180(W)) = W holds between convolution kernel and transposed convolution kernel, and $(W^T)^T = W$ holds between the corresponding matrix and transposed matrix.

5. Application of Matrices in Federated Architectures

We propose a novel architecture shown in Figure 8 for handwritten digit recognition and generation tasks, which integrates convolutional neural networks, recurrent neural networks, attentional mechanisms, and the matrix operation characteristics of generative adversarial networks.

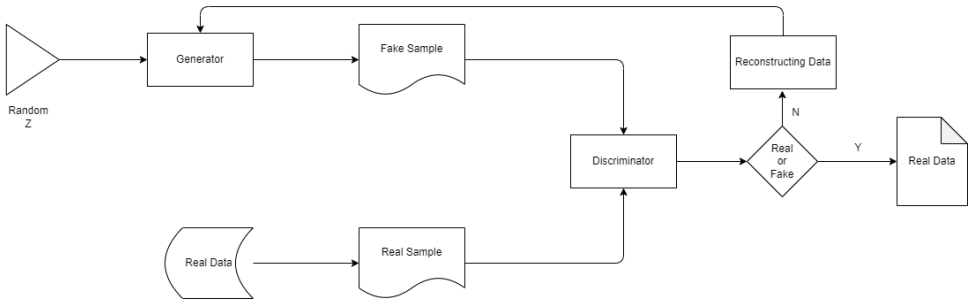

Figure 8: Joint Architecture

Our real data input here from the MNIST dataset is randomly screened for samples that serve as criteria for the discriminator's real data. On the other hand, we input random noise after passing through the generator after screening some samples to the discriminator. The discriminator determines whether the data is true or false, if it is true, then output this is the real data, if it is false, you need to enter the reconstruction of the data module, the reconstruction of the data module through a variety of matrix operations, and then continue to send the results of the operation to the generator, and finally the generator and the discriminator at the same time for training to achieve the Nash equilibrium state, to stop the training, the end of the task.

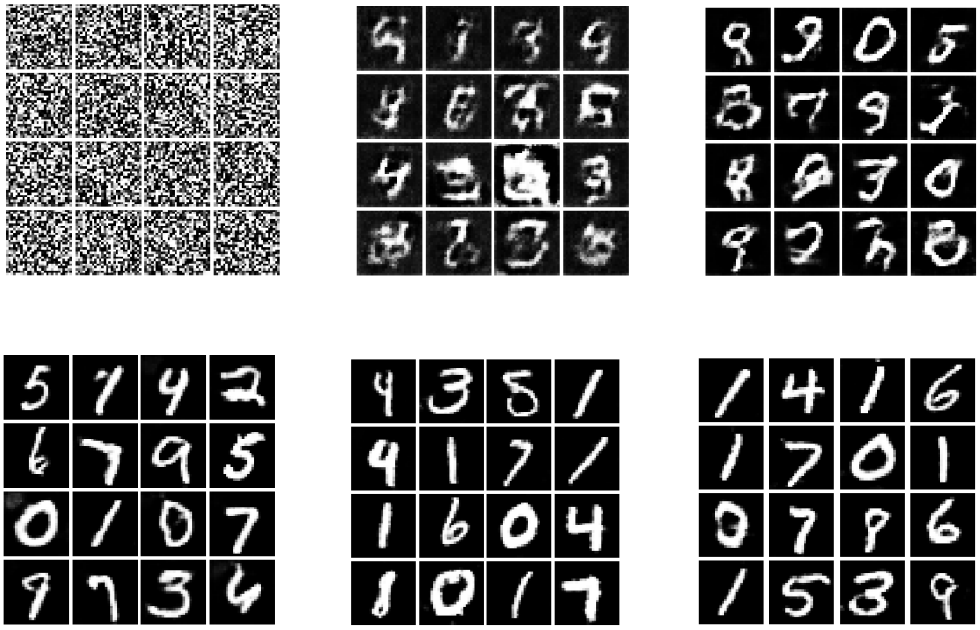

Figure 9: Handwritten digits generated via generative models

Figure 9 shows a portion of the generated image selected during the training of our model. We can clearly see the very blurry image at the beginning. After continuous training by the generation network and the discriminant network, we finally got a clearer handwriting. digital image.

ISSN 2616-5775 Vol. 7, Issue 4: 29-35, DOI: 10.25236/AJCIS.2024.070405

6. Conclusion

Artificial neural networks fundamentally rely on matrix operations for calculations. These operations include matrix multiplication, addition, and inversion, which are critical for propagating input through the network, adjusting weights during training, and performing backpropagation. The efficiency of these operations can significantly impact neural network performance and training time.

Using structured matrices in neural networks can improve computational efficiency. Structured matrices, such as semi-separable matrices, low-shift rank matrices, hierarchical matrices, and products of sparse matrices, can reduce the computational complexity of neural networks. These structures allow for more efficient matrix-vector multiplication, which is a major computational bottleneck in neural networks.

This article examines the importance of matrices in artificial neural networks. The importance of matrix operations in the model is discussed in the mainstream neural network structure. Finally, the importance of matrix operations is demonstrated through an example of generating handwritten digit recognition.

References

[1] Chintalapudi N, Battineni G, Hossain MA, et al. Cascaded Deep Learning Frameworks in Contribution to the Detection of Parkinson's Disease. Bioengineering (Basel). 2022.
[2] Fawzi, A. et al. Artificial intelligence finds faster algorithms for multiplying matrices. Nature. 2022; doi: 10.1038/d41586-022-03023-w
[3] Hofmann M, Mader P. Synaptic Scaling-An Artificial Neural Network Regularization Inspired by Nature. IEEE Trans Neural Netw Learn Syst. 2022; 33 (7):3094-3108. doi:10.1109/TNNLS. 2021. 3050422
[4] Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 3642–3649. IEEE (2012).
[5] Jay Alammar. Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention). 2018.
[6] Wang X, Liu J, Zhang C, et al. SSGraphCPI: A Novel Model for Predicting Compound-Protein Interactions Based on Deep Learning. Int J Mol Sci. 2022.
[7] Hemati W, Mehler A. LSTMVoter: chemical named entity recognition using a conglomerate of sequence labeling tools. J Cheminform. 2019.
[8] Zhou Y, Rosen MC, Swaminathan SK, et al. Distributed functions of prefrontal and parietal cortices during sequential categorical decisions. Elife. 2021.
[9] Nozomu Miyamoto, Masaru Isonuma, Sho Takase, Junichiro Mori, and Ichiro Sakata. 2023. Dynamic Structured Neural Topic Model with Self-Attention Mechanism. In Findings of the Association for Computational Linguistics: ACL 2023, pages 5916–5930, Toronto, Canada. Association for Computational Linguistics.
[10] Chengshen Xu, Lin Li, Xiao Hu, et al. The Cross Products of M Vectors in N-dimensional Spaces and Their Geometric Significance. arXiv preprint arXiv:2206.13809, 2022.
[11] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative Adversarial Networks. In: Advances in Neural Information Processing Systems. 2014; 2672-2680.