Applied Intelligence (2024) 54:6624–6636
https://doi.org/10.1007/s10489-024-05472-4

Image classification based on tensor network DenseNet model

Chunyang Zhu[1] · Lei Wang[2] · Weihua Zhao[1] · Heng Lian[3]

Accepted: 19 April 2024 / Published online: 21 May 2024

Abstract
Image classification, the primary domain where deep neural networks significantly contribute to image analysis, requires a substantial amount of computer memory to train. This is particularly true in the fully connected layer, which accounts for 90% of the total memory. Moreover, the flattening operation could potentially result in the loss of the multi-linear structure of the image data. The tensor regression network, however, minimally impacts the performance of the neural network while achieving a high compression rate. This effectively mitigates the issue of large memory occupation in the neural network model. The DenseNet model, in particular, can alleviate the vanishing-gradient problem and strengthen feature propagation and outperform other existing networks. This article proposes a novel tensor network model that embeds the tensor regression layer into the DenseNet model. The framework of this tensor DenseNet model has been established, and its estimation procedure is developed. Tensor network model is applied to the classification of the following datasets: Fruits 360, 100 Sports Image, ASL Alphabet, and Mini-ImageNet. The experimental results indicate that the combination of the DenseNet model with the tensor regression layer not only conserves a significant amount of memory but also maintains a high accuracy of classification, compared with existing tensor network models.

Keywords Tensor decomposition · Tensor regression layer · Tensor Networks · Densely connected convolutional network · Image classification

1 Introduction

Over the last decade, there has been a growing interest in applying tensor methods in machine learning. In various scientific fields such as image analysis, signal processing, and space-time analysis, tensors have demonstrated their capability to represent data with a multi-modal structure while preserving the original data structure. Unlike traditional machine learning approaches that often convert natural data into vector form, which can result in information loss regarding the natural data structure, tensor-based machine learning methods have become essential for preserving and utilizing the inherent structures of the data [1].

With the development of tensor learning methods, deep neural networks have demonstrated advanced performance in various large-scale machine learning tasks, including computer vision, speech recognition, and text processing. For instance, the convolutional neural network (CNN) [2, 3] has shown significant advantages in image classification tasks. These network models consist of thousands of nodes and millions of learnable parameters, and are trained using millions of images on powerful graphics processing units (GPUs) [4]. However, the expensive hardware requirements and long training time limit the widespread utilization of these models on traditional desktop computers and portable devices. Consequently, researchers have conducted extensive research to explore various methods for reducing hardware requirements, memory consumption, and training time.

The fully connected layer is one of the most commonly used layers in convolutional networks, responsible for performing linear transformations from high-dimensional input data to high-dimensional output data. The traditional approach employs matrices to define this transformation. For

✉ Lei Wang
lwangstat@nankai.edu.cn

✉ Weihua Zhao
zhaowhstat@163.com

1 School of Mathematics and Statistics, Nantong University, Jiangsu Nantong 226019, People's Republic of China

2 School of Statistics and Data Science, KLMDASR, LEBPS and LPMC, Nankai University, Tianjin 300071, People's Republic of China

3 The City University of Hong Kong, Kowloon Tong, Hong Kong

example, in a typical CNN, the dimensions of the input and output data of the fully connected layer are both in the thousands, resulting in millions of parameters. This complexity hinders the simplicity of the model structure. However, since the input and output of the convolutional layer are tensor data, it is natural to introduce tensor methods into convolutional networks to optimize the overall model structure, which has become an important research issue.

The main idea of this paper is to apply tensor decomposition to deep learning and then reparameterize the existing layers of deep convolutional networks to accelerate computation or reduce the number of parameters. Several studies have explored tensor decomposition in deep convolutional networks. Lebedev [5] proposed using tensor CANDECOMP/PARAFAC (CP) [6, 7] decomposition to accelerate feature extraction in convolutional layers. Similarly, Tai [8] introduced a new algorithm for computing low-rank tensor decomposition to eliminate redundancy in convolutional kernels. Kim [9] used the pre-trained network, applied tensor Tucker decomposition on the convolution kernels and finally fine-tuned the resulting network. Yang [10] proposed weight sharing in multi-task representation learning framework to learn cross-task sharing structure at every layer in a deep network. Chen[11] proposed sharing the remaining units and proposed a new architecture, Collective Residual Unit (CRU), to improve the parameter efficiency of deep neural networks through collective tensor factorization. Novikov [12] used Tensor-Train (TT) decomposition to apply the low rank tensor structure to the weight of the fully connected layer such that the number of parameters is reduced and at the same time the expressive power of the layer is preserved. However, these studies still retain the full connection layer of the network such that these models have a large number of parameters to be trained and optimized. In addition, the deformation of high order data can not maintain the multilinear structure of data.

Additionally, the flattening operation used in higher-order data dimension cannot preserve the multi-linear structure of the data. In contrast, Kossaifi [13] proposed the tensor regression layer to replace the vectorization operation and fully connected layer in CNN with high-order multiple regression. This layer is embedded into popular models such as visual geometry group (VGG, [3]) and residual network (ResNet, [14]). The advantage of this replacement is the ability to compress the model while preserving the multi-modal information of the dataset. Vectorization of high-dimensional datasets leads to the loss of the multi-modal information. For example, applying a flattening operation to a color image (a third-order tensor) eliminates the relationship between channels. The tensor regression layer can address this problem by performing a multi-linear regression task between the output of the final convolutional layer and the classification layer, enabling the capture of multi-modal information.

To further enhance the performance of tensor networks, Gao [15] proposed a quantized tensor neural network (QTNN) that combines the powerful learning ability of neural networks with the simplicity of tensor networks. QTNN is a generalized multi-layer nonlinear tensor network that effectively extracts low-dimensional features from data while preserving the original structural information. While tensor methods have been widely employed in supervised learning, researchers have also turned their attention to unsupervised learning. For example, Oldfield [16] used tensor regression to address the problem of finding interpretable directions in the potential space of pre-trained Generative Adversarial Networks (GANs), promoting controllable image synthesis. [17] proposed auto-weighted multiple kernel tensor clustering (AMKTC) to capture the essential high-order correlations between multiple base kernels by leveraging tensor-singular value decomposition (t-SVD)-based tensor nuclear norm constraint on a 3-order graph tensor.

Other researchers have uncovered valuable insights through exploration of 3D convolutional neural network (CNN) models and 3D filtering CNNs [18–20], as well as deep CNNs for image classification. Additionally, there are other relevant studies, such as the multi-objective deep CNN model proposed by Lu [21], among others [22–26].

The densely connected convolutional network (DenseNet), introduced by Huang [27], stands as a prevalent deep neural network architecture adopted across diverse fields. DenseNet ensures maximum information transmission between network layers by establishing connection relationships among different layers. It explicitly distinguishes between the information added to the network and the information retained. The primary of this paper focus on embedding the tensor regression layer into the densely connected convolutional network for learning purposes. This unique connection method effectively alleviates the issue of gradient disappearance during model training. The integration of this convolutional network with the tensor regression layer offers several advantages:

(1) The tensor regression layer replaces the fully connected layer to optimize the model structure, which can significantly reduce the number of parameters that need to be trained, minimize memory consumption and lessen the hardware requirements for model training while maintaining performance.
(2) The fully connected layer is replaced by a tensor regression layer, which is a special type of regression layer structure designed to handle tensor-formatted data. When processing image data, it can extract multiple features simultaneously and conduct regression prediction. This layer takes the outputs tensors from previous layers as input and is able to preserve the spatial structural features of the data.

(3) The special connection mode of the DenseNet strengthens feature propagation, which leads to alleviate gradient disappearance during training of the network model embedded with the tensor regression layer.
(4) Our proposed tensor network can achieve high classification accuracy while significantly reducing memory usage and computation time.

The remainder of this paper is organized as follows. Section 2 introduces tensor algebra, including the basics and CP decomposition of tensors, as well as the tensor regression layer. Section 3 describes the DenseNet architecture and the tensor network. In Section 4, experiments are conducted to verify the superior performance of our proposed method compared to existing models. Finally, Section 5 provides some conclusions and discussion.

2 Preliminaries

In this Section, some basic concepts and properties of tensors will be mentioned firstly, and then some preliminaries about the tensor regression layer will be provided.

2.1 Tensor algebra

Vectors, also known as first-order tensors, are represented in boldface lowercase letters, e.g., $\mathbf{a}$. Matrices, also known as second-order tensors, are represented in boldface capital letters, e.g., $\mathbf{A}$. Higher-order tensors are represented in boldface Euler script letters, e.g., $\mathcal{X}$. Scalars are represented in lowercase letters, e.g., a. The ith element of the vector $\mathbf{a}$ is represented as a_i. The (i, j)th element of the matrix $\mathbf{A}$ is represented as a_{ij}. The (i, j, k)th element of the 3rd-order tensor $\mathcal{X}$ is represented as x_{ijk}.

The inner product of two tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M}$ with the same dimension is the sum of the products of their corresponding elements:

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_m=1}^{I_M} a_{i_1 i_2 \cdots i_m} b_{i_1 i_2 \cdots i_m}. \tag{1}$$

If $\mathcal{A} = \mathcal{B}$, it can get $\langle \mathcal{A}, \mathcal{A} \rangle = \|\mathcal{A}\|_F^2$, where $\|\mathcal{A}\|_F = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_m=1}^{I_M} a_{i_1 i_2 \cdots i_m}^2}$ denotes the Frobenius norm of a tensor. A tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M}$ is called rank-one if it can be written as the outer product of M vectors: $\mathcal{T} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(M)}$. The n-mode product of a tensor $\mathcal{M} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M}$ and a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ is expressed as $\mathcal{M} \times_n \mathbf{U}$. The result is still a tensor and its dimension is $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \cdots \times I_M$ given as $(\mathcal{M} \times_n \mathbf{U})_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_M} = \sum_{i_m=1}^{I_M} x_{i_1 i_2 \cdots i_m} u_{j i_m}$. Then $\mathcal{Y} = \mathcal{M} \times_n \mathbf{U} \Leftrightarrow \mathbf{Y}_{(n)} = \mathbf{U}\mathbf{M}_{(n)}$.

2.2 Tensor decomposition

The CANDECOMP/PARAFAC decomposition of tensors expresses a tensor as the sum of tensors of a finite number of rank-1 tensors. Figure 1 shows the process of CP decomposition of a tensor.

Given a third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, it can be approximated by a sum of tensors as follows:

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{2}$$

with R being a positive integer, and $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, $\mathbf{c}_r \in \mathbb{R}^K$, for $r = 1, \ldots, R$. The factor matrix is a combination of these vectors, i.e., $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_R]$, and likewise for $\mathbf{B}$ and $\mathbf{C}$. According to this definition, the following equations hold:

$$\mathbf{X}_{(1)} \approx \mathbf{A}(\mathbf{C} \odot \mathbf{B})^\top, \ \ \mathbf{X}_{(2)} \approx \mathbf{B}(\mathbf{C} \odot \mathbf{A})^\top, \ \ \mathbf{X}_{(3)} \approx \mathbf{C}(\mathbf{B} \odot \mathbf{A})^\top. \tag{3}$$

Thus, the CP model can be concisely expressed as

$$\mathcal{X} \approx [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \equiv \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r. \tag{4}$$

Due to the intuitive nature of CP decomposition, it can be easily extended to higher-order tensors. The more detailed theories of tensors can be found in [28].

2.3 Tensor regression layer

Tensor regression layer is a differentiable neural network layer. The full connection layer parameters of CNN account for the majority of the total parameters of the model. In addition to such a large consumption of computing resources,

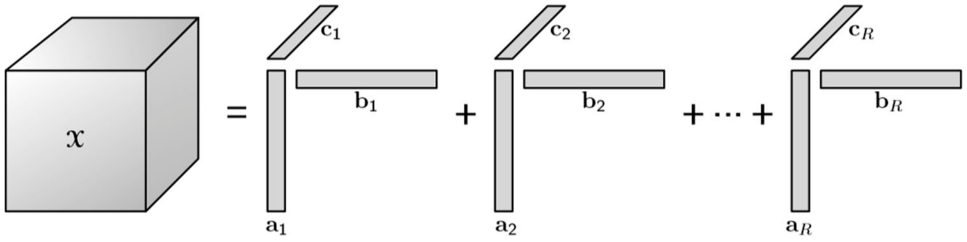

Fig. 1 CP decomposition of a three-way array

data flattening also leads to the loss of rich spatial structure information in the final convolution layer. Tensor regression layer uses multi-linear mapping instead of flattening and full connection layer in the model.

Given $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times I_{N+1}}$ with I_{N+1} being the number of categories in the dataset, the function f can be defined as:

$$f(\mathcal{X}) = \mathcal{W}_{(N+1)} \operatorname{vec}(\mathcal{X}) + \mathbf{b}, \tag{5}$$

where $\mathbf{b} \in \mathbb{R}^{I_{N+1}}$ is the bais vector.

In the past, tensor regression was trained as an independent model. It is a generalization of the least squares regression problem in tensor space, and is often combined with some feature extraction methods. However, tensor regression model is difficult to undertake the task of large-scale data analysis. This tensor structure can be embed into the convolutional network as a trainable neural network layer. Figure 2 shows the visualization structure of the tensor network layer. The main idea behind the tensor regression layer is to implement a low tensor rank structure on $\mathcal{W}$ to reduce memory usage and utilize the multi-linear structure of input $\mathcal{X}$.

By applying CP decomposition to the weight tensor of the function and considering the properties of tensor decomposition, equation (5) can be rewritten as:

$$\begin{aligned} f(\mathcal{X}) &= [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(N+1)}]\!]_{(N+1)} \operatorname{vec}(\mathcal{X}) + \mathbf{b} \\ &= \mathbf{A}^{(N+1)} \left(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(1)}\right)^{\top} \operatorname{vec}(\mathcal{X}) + \mathbf{b}. \end{aligned} \tag{6}$$

In order to achieve end-to-end system training, instead of optimizing model (6) as a separate tensor regression, the back propagation method is used to optimize the model. The partial derivatives required for gradient based optimization methods according to equation (6) can be obtained as follows,

$$\frac{\partial f(\mathcal{X})_i}{\partial \left(\mathbf{A}^{(n)}\right)_{jk}} = \frac{\left(\partial \mathbf{A}^{(N+1)} \left(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(1)}\right)^T \operatorname{vec}(\mathcal{X})\right)_i}{\partial \left(\mathbf{A}^{(n)}\right)_{jk}}, \tag{7}$$

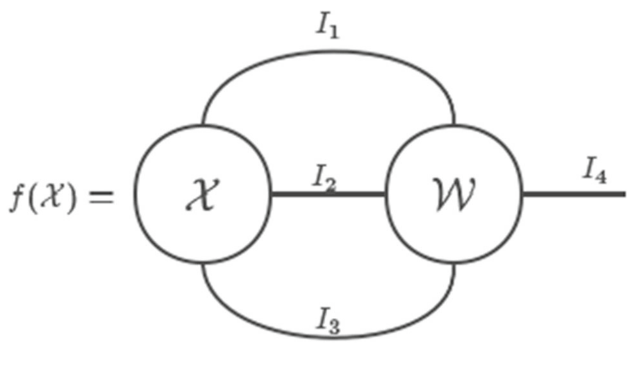

Fig. 2 Visualization of tensor regression layer in tensor networks

where $n = 1, 2, \cdots, N+1$. In addition, for a given mode n, it can naturally arrange these partial derivatives into third-order tensors $\partial f(\mathcal{X})/\partial \mathbf{A}^{(n)}$, and use tensor expansion to obtain their expressions:

$$\begin{aligned} \left(\frac{\partial f}{\partial \mathbf{A}^{(n)}}\right)_{(2)} = (\mathcal{X})_{(n)} \Big(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \\ \odot \cdots \odot \mathbf{A}^{(1)}\Big) \left(\mathbf{A}^{(N+1)} \odot \mathbf{I}_R\right)^T, \end{aligned} \tag{8}$$

for $n = 1, 2, \cdots, N$, and when $n = N + 1$, it becomes

$$\left(\frac{\partial f}{\partial \mathbf{A}^{(N+1)}}\right)_{(1)} = \mathbf{I}_{I_{N+1}} \otimes \left(\operatorname{vec}(\mathcal{X})^T \left(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(1)}\right)\right). \tag{9}$$

3 Tensor network

After introducing the basic theory of tensor and tensor regression layer, A mainstream deep convolution neural network model should be introduced: Densely connected convolutional network (DenseNet). In order to optimize the model structure, the tensor regression layer is embedded into DenseNet to reconstruct a new network model, and study and analyze the new model structure.

3.1 DenseNet

In the field of computer vision, CNN has become the most mainstream method, with architectures like VGG-16/19, GoogLenet [29]. As CNN becomes deeper and deeper, a new research problem arises: when information about input or gradient passes through many layers, it may disappear when it reaches the end (or beginning) of the network. A milestone in the history of CNN is the emergence of ResNet. ResNet can train a deeper CNN model to achieve higher accuracy. The core of ResNet model is to establish a “short circuit connection" between the earlier layer and the later layer, which is helpful for the back propagation of the gradient during the training process and prevents the gradient from disappearing during the propagation process. Such a connection structure can lead to train a deeper CNN network. Building upon ResNet, [27] put forward densely connected convolutional networks. Its basic idea is the same as ResNet, but it establishes a dense connection between all the previous layers and the subsequent layers. DenseNet excels in feature reuse through channel-wise feature concatenation, enabling better performance than ResNet with fewer parameters and computational costs.

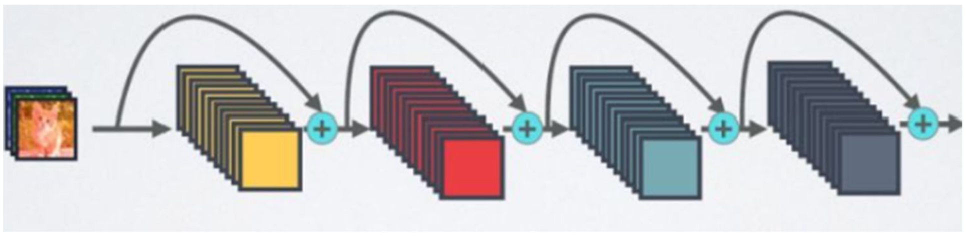

Fig. 3 Short circuit connection mechanism of ResNet ("+" represents feature addition operation)

3.1.1 Structure of DenseNet

Compared with ResNet, DenseNet proposes a more radical dense connection mechanism: it connects all layers. Specifically, each layer will accept all the preceding layers as its additional input. As shown in Fig. 3, the figure depicts the connection mechanism of ResNet. In contrast, Fig. 4 shows the dense connection mechanism of DenseNet. ResNet establishes a short circuit connection between each layer and the previous layer. The connection method is feature addition. In DenseNet, each layer will be concatenated with all the previous layers in the channel dimension and used as the input of the next layer. For an L-layer network, DenseNet contains a total of $\frac{L(L+1)}{2}$ connections. Compared with ResNet, this is a dense connection. Furthermore, DenseNet directly connects feature maps from different layers, which can realize feature reuse and improve efficiency. This feature is the main difference between DenseNet and ResNet. Denote $\mathbf{X}_{L-1}$ as the output of layer $L-1$ of the model, then the output of the traditional deep neural network at layer L can be expressed as:

$$\mathbf{X}_L = H\left(\mathbf{X}_{L-1}\right). \tag{10}$$

For ResNet, the features of the input from the upper layer are added. The output from layer L is:

$$\mathbf{X}_L = H\left(\mathbf{X}_{L-1}\right) + \mathbf{X}_{L-1}. \tag{11}$$

In DenseNet, the features of all previous layers will be connected. The output of layer L is:

$$\mathbf{X}_L = H\left(\left[\mathbf{X}_0, \cdots \mathbf{X}_{L-1}\right]\right), \tag{12}$$

where H represents the nonlinear transformation function. This combined operation includes a series of normalization, activation, pooling, and convolution operations.

3.1.2 Composition of DenseNet

Dense blocks Deep convolutional networks generally reduce the size of feature maps through pooling or convolution layers, but dense connection mode requires that the size of feature maps to be consistent. To solve this problem, DenseBlock is defined. In the dense block, the feature map of each layer is consistent in size and connected on the channel. Assuming that the number of channels in the input feature map is K_0, the number of channels in layer L is $K_0+(L-1)K$ with K being the growth rate, which is a hyperparameter. As shown in Fig. 5, it is a dense block with 4 layers and the growth rate is 4.

Transition layer The transition layer mainly connects two adjacent DenseBlocks and reduces the size of the feature map. The transition layer consists of a 1×1 convolution kernel and a 2×2 global average pooling layer, which can compress the model.

Bottleneck layer although each layer only generates k output feature maps, the input sample volume of the model is large. Before each convolution operation, we introduce a convolution of size 1×1 as the Bottleneck layer to reduce the number of input feature maps and improve the computational efficiency. The design of the bottleneck layer is particularly effective for DenseNet, called DenseNet-B.

It can be seen from the network structure of DenseNet that it defines dense blocks based on residual neural network, which strengthens the propagation of features and encourages the reuse of features. At the same time, the existence

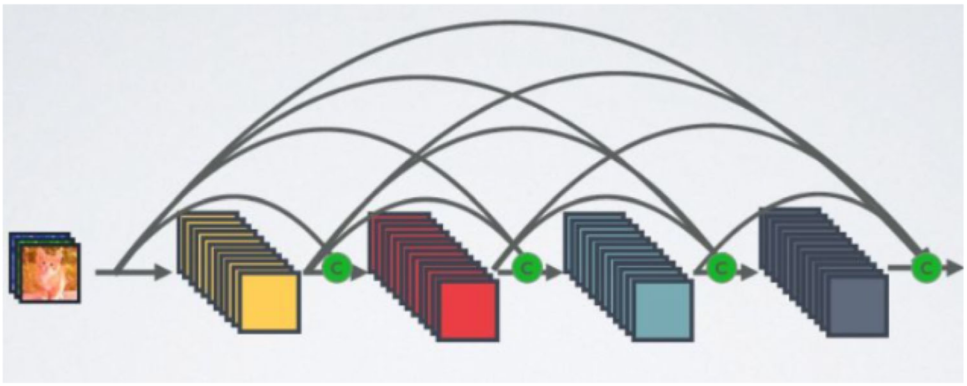

Fig. 4 Dense connection mechanism of DenseNet ("c" represents channel connection operation)

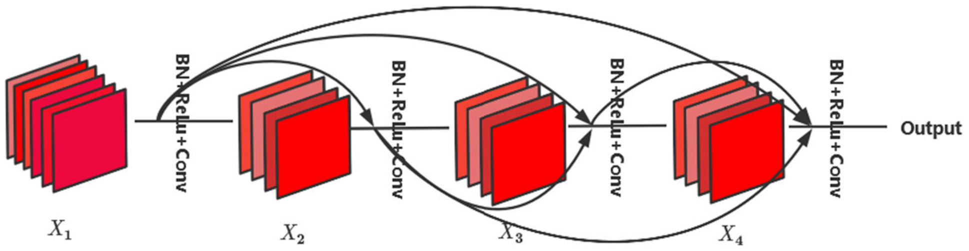

Fig. 5 A dense block with 4 layers and a growth rate of 4

of transition layer and bottleneck layer can make the model simpler, greatly reduce the number of parameters and speed up the training of the model. As shown in Fig. 6, this is a DenseNet with 3 dense blocks.

3.2 Tensor network based on DenseNet

The original DenseNet structure employs global average pooling and fully connected layers for classifying image data after feature extraction, which may compromise the spatial structure information of features. Additionally, the resulting fully connected layer entails a large number of trainable parameters, posing challenges for model training. To address these concerns, we propose optimizing the DenseNet structure by embedding a tensor regression layer as a trainable layer into the network, facilitating joint learning of features for data classification.

To achieve this, we directly substitute the global average pooling and fully connected layers of the original network with the tensor regression layer, while imposing low-rank constraints on the regression weights. Intuitively, the tensor regression layer offers the advantage of effectively utilizing spatial structure information from the data and significantly reducing the model's training parameters.

Figure 7 illustrates the tensor network structure based on DenseNet, showcasing the integration of the tensor regression layer into the network architecture. This optimized structure aims to enhance feature learning and classification accuracy while mitigating the issues associated with traditional classification layers.

3.3 Analysis of the model

The network model uses a simple tensor regression structure to undertake the classification tasks in the model, and can also use gradient back propagation algorithm to optimize the parameters of the model. According to the structure of tensor regression layer, one only need to train a few parameters.

Consider the output tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_0 \times I_1 \times \cdots \times I_N}$ of convolution layer, and assume that the rank of tensor regression layer weight is R, with $R \leq I_k$. The output is an n-dimensional array. For a fully connected layer with only 1-layer, the training parameters required are:

$$n_{\mathrm{FC}} = n \times \prod_{k=0}^{N} I_k. \tag{13}$$

In comparison, the number of parameters to be trained in tensor regression layer is only:

$$n_{\mathrm{TRL}} = \sum_{k=0}^{N} R \times I_k + R \times n. \tag{14}$$

As a result, the memory consumption of the computer is greatly reduced together with greatly reducing the number of parameters. The promising performance of our proposed tensor network model will be examined by experiments in the next Section.

4 Experiments and discussion

In this Section, extensive comparative experiments are conducted to verify the feasibility of the model. To ensure the fairness of the experiment, DenseNet-121, DenseNet-169, DenseNet-201, and DenseNet-264 are utilized as the experimental objects. Subsequently, the tensor regression layer is embedded into each model. The computer memory used for training each model is observed, and the training results of the model are analyzed. All the experiments are carried out under the Linux system, with GPU model GTX-1650 being used, and the program is written based on the Python language.

4.1 Data description

① Fruits 360 is a comprehensive dataset containing 131 types of fruits and vegetables. It comprises approximately

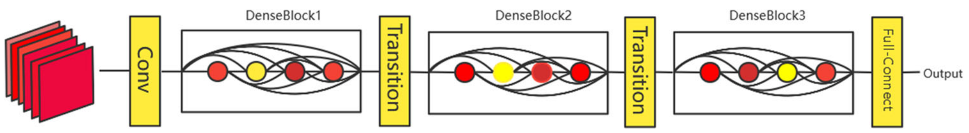

Fig. 6 DenseNet with 3 dense blocks

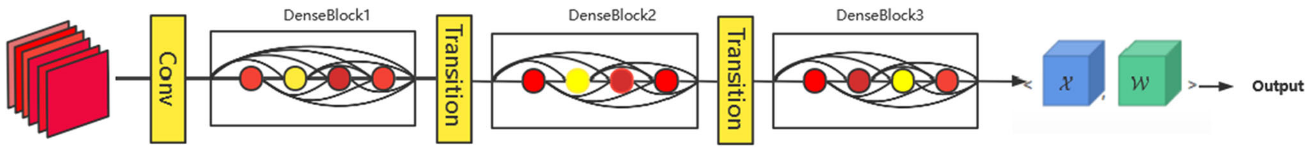

Fig. 7 Tensor network based on DenseNet

90000 images. Existing literature has positioned fruits and vegetables in the shaft of a common motor, recorded a 20-second short film, and then captured the fruits using a camera. As illustrated in Fig. 8, it portrays a fraction of the dataset. It can be observed that the background of all images is white, and the fruits predominantly occupy the images. This aids in better extraction of the characteristics of each type of fruit or vegetable, and simplifies the model training process. This dataset can be accessed at "https://www.kaggle.com/datasets/moltean/fruits".

② 100 Sports Image is a dataset containing 100 different sports types. It consists of approximately 13000 images. The data was collected from the internet, and then organized all the images to obtain a high-quality clean dataset free from duplicate or poor-quality images. As shown in Fig. 9, it displays a portion of the dataset. It can be observed that despite the complex background of each image, each sport type has a symbolic feature. This indicates that the model can accurately extract the main features of each image, and then classify them. This dataset is available at "https://www.kaggle.com/datasets/gpiosenka/sports-classification".

③ASL Alphabet is a dataset for alphabets in the American Sign Language. It comprises 29 classes, with 26 dedicated to the letters A-Z and 3 classes for SPACE, DELETE, and NOTHING. The training dataset contains 87,000 images with dimensions of 200 × 200 pixels. As depicted in Fig. 10, the background of this dataset is relatively simple, which greatly aids model training. This dataset can be accessed at "https://www.kaggle.com/datasets/grassknoted/asl-alphabet".

④ In this study, the Mini-ImageNet dataset is utilized to assess the performance of the proposed model. It consists of thousands of images spanning 100 different object classes. With dimensions of 84 × 84 pixels, each image is associated with a specific label indicating its class category. The dataset is partitioned into three subsets: train set, validation set and test set. The training set is used to train the model, while the validation set aids in hyperparameter tuning and model selection. Lastly, the test set is employed to evaluate the overall accuracy and generalization capability of the model. This dataset is available at "https://www.kaggle.com/datasets/arjunashok33/miniimagenet".

4.2 Experimental results

This Section presents the performance of eight models on two datasets, as depicted in Tables 2 and 3. The Tables 4, 5, 6 and 7 report the test set accuracy, the number of parameters required for model training, and the size of the computer memory band (MB) necessary for training. The percentages highlighted in red indicate a decrease when comparing the tensor regression layer-based method to the existing vectorization-based DenseNet method.

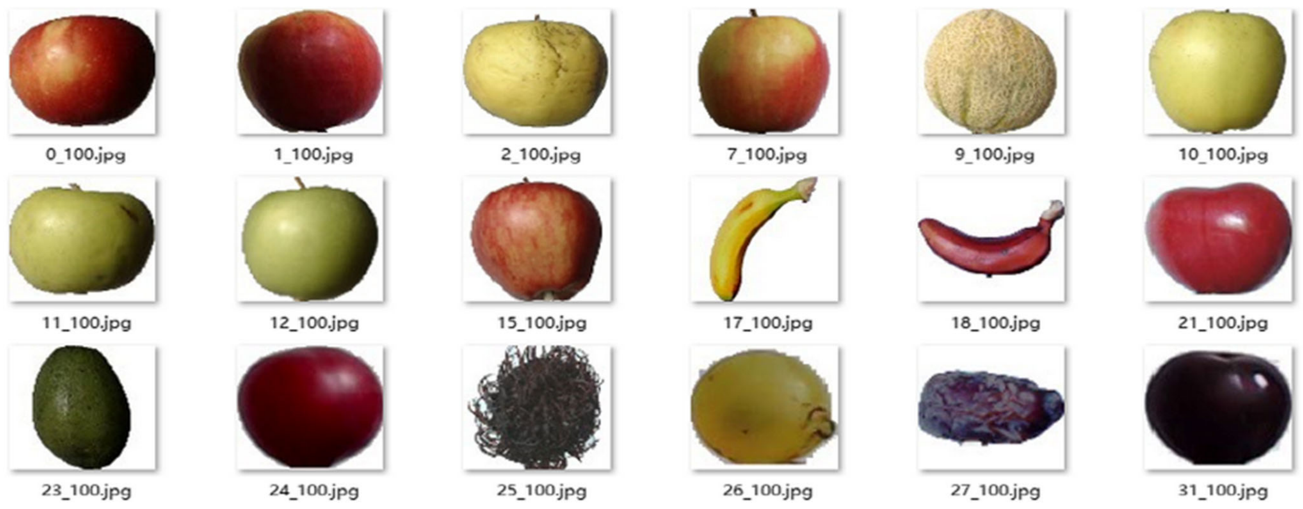

Fig. 8 Fruits 360

Fig. 9 100 Sports Image

In this experiment, all networks were optimized using the adaptive moment estimation (Adam) method. Adam is an adaptive learning rate optimization algorithm that combines momentum method and adaptive gradient method. It adjusts the learning rate by estimating the first-order moment (the mean) and the second-order moment (the uncentered variance) of the gradient, and adaptively updates parameters during the training process. This adaptive adjustment of learning rates helps improve convergence and training efficiency, making Adam a popular choice for optimizing deep neural networks.

For the Fruits 360 dataset, a batch size of 30 was employed and a total of 30 training cycles were conducted. Given the relative simplicity of this dataset, with clear foreground and background images, image enhancement techniques were not applied. The input image size was set to 50×50, determined through observations of the dataset's optimal input size. The initial learning rate was set to 0.0001, which allowed the model to converge within a reasonable timeframe and achieve solid performance. Learning rate decay was not applied due to the dataset's simplicity, yet the model still yielded satisfactory results. In networks featuring fully connected layers, the number of fully connected layers was set to 2, based on experimental observations indicating that this configuration achieved the best performance. This setup ensures efficient feature learning and classification for the Fruits 360 dataset.

For the 100 Sports Image dataset, a batch size of 50 was utilized, and a total of 150 training epochs were conducted to train the model comprehensively. Given the complexity of this dataset's background and its relatively small size, data augmentation techniques were employed. Data augmentation involves generating new training samples by transforming original images (e.g., through rotation, scaling, cropping), effectively enlarging the training set and improving the model's generalization ability. The input image size was set to 264×264 based on observations of the dataset's optimal input size. The initial learning rate was set to 0.001, enabling the model to converge effectively within a reasonable timeframe

Fig. 10 ASL Alphabet

Table 1 Experimental results of Fruits 360

Model	Accuracy	Numbers Of Parameters	Memory
DenseNet-121	0.9824	7,543,811	30.23
DenseNet-121-TRL	0.9799(-0.25%)	6,951,808 (-7.85%)	27.97(-7.48%)
DenseNet-169	0.9876	13,400,835	52.60
DenseNet-169-TRL	0.9871(-0.05%)	12,481,152(-6.86%)	49.09(-6.67%)
DenseNet-201	0.9902	19,139,843	74.55
DenseNet-201-TRL	0.9899(-0.03%)	18,089,088(-5.49%)	70.56(-5.35%)
DenseNet-264	0.9912	32,087,299	123.99
DenseNet-264-TRL	0.9899(-0.13%)	30,643,328(-4.50%)	118.48(-4.44%)

while achieving good performance. Similar to the Fruits 360 dataset, learning rate decay was not applied for the 100 Sports Image dataset due to its manageable complexity.Despite the dataset's challenges, such as complex backgrounds and a relatively small size, the model achieved satisfactory results through a combination of data augmentation techniques and an appropriate network structure. In networks featuring fully connected layers, the number of fully connected layers was set to 3, as experimental observations indicated that this configuration yielded optimal performance. This setup ensures that the model can effectively learn and classify features relevant to the 100 Sports Image dataset.

As shown in Tables 1 and 2, the proposed method demonstrates obvious advantages compared to other existing methods. Specifically, the results in the first row of Table 3 show that our proposed method reduced the number of parameters by 88.20% and memory usage by 78.15%, with only a 2.92% decrease in accuracy. Furthermore, the proposed method has the advantage of yielding greater gains with more complex image backgrounds. This indicates that the method has stronger handling capabilities for images with complex backgrounds, which is critical for many real-world applications.

The robustness and reliability of the proposed method are further demonstrated by its ability to maintain consistently high performance across various initialization schemes and hyperparameter settings. Extensive tuning of hyperparameters, including learning rate, batch size, number of training cycles, and the architecture of fully connected layers, was conducted to ensure optimal performance. The model's stability and reliability were confirmed through multiple runs with different random initializations, all of which consistently yielded high performance. This consistency underscores the effectiveness and generalizability of the proposed approach, enhancing confidence in its applicability to real-world scenarios.

Overall, the proposed method provides an efficient, effective, and robust solution for image classification tasks, especially in scenes with complex image backgrounds. Its superior performance, coupled with its efficiency in parameter usage and memory consumption, suggests that the model is meaningful.

To further evaluate tensor networks, comparisons are made between the performance of DenseNet with tensor regression layer and some currently popular network models. VGG16 and VGG19 are classes of deep networks with smaller convolutional kernels. The proposed network model illustrates that small and deep networks have more advantages than large and shallow ones. ResNet50 and ResNet101 establish connections between layers to realize residual learning. This aids in preventing the gradient from disappearing during propagation, enabling the training of deeper CNN networks. EfficientNet [30] balances the three crucial dimensions of network depth, network width, and image resolution for optimizing network performance. ResNet50-TRL [13] is the ResNet embedded with a tensor regression

Table 2 Experimental results of 100 Sports Image

Model	Accuracy	Numbers Of Parameters	Memory
DenseNet-121	0.7526	58,909,156	253.64
DenseNet-121-TRL	0.7234(-2.92%)	6,951,808(-88.20%)	55.42(-78.15%)
DenseNet-169	0.773	96,551,140	397.80
DenseNet-169-TRL	0.7428(-3.90%)	12,481,152(-87.07%)	77.09(-80.62%)
DenseNet-201	0.7928	115,004,132	496.34
DenseNet-201-TRL	0.7643(-3.39%)	18,089,088(-84.27%)	99.63(-79.93%)
DenseNet-264	0.8016	116,093,540	665.38
DenseNet-264-TRL	0.7952(-0.64%)	36,643,328(-68.44%)	148.67(-77.66%)

Table 3 Fruits 360 dataset classification results comparison

Model	Accuracy	Macro_P	Macro_R	Macro_F1-Score
DenseNet-264-TRL	0.9899	0.9858	0.9901	0.9879
VGG16 [3]	0.9741	0.9763	0.9735	0.9749
VGG19 [3]	0.9706	0.9700	0.9680	0.9690
ResNet50 [14]	0.9862	0.9860	0.9860	0.9860
ResNet101 [14]	0.9941	0.9925	0.9945	0.9935
EfficientNetB0 [30]	0.9967	0.9982	0.9975	0.9978
ResNet50-TRL [13]	0.9726	0.9713	0.9706	0.9709
CRU-NET-56 [11]	0.9858	0.9875	0.9863	0.9869

layer. Unlike this study, the tensor decomposition of this model uses Tucker decomposition. Table 3 presents the accuracy, macro-averaged precision, macro-averaged recall and macro-averaged F1-score of these models, and the calculation formulas for metrics are as follows:

$$\text{Macro\_P (macro-averaged precision)} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FP_i}, \tag{15}$$

$$\text{Macro\_R (macro-averaged recall)} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FN_i}, \tag{16}$$

$$\text{Macro\_F1-Score (macro-averaged F1-score))} = \frac{2 \times \text{Macro\_P} \times \text{Macro\_R}}{\text{Macro\_P} + \text{Macro\_R}}, \tag{17}$$

where N is the number of categories, TP_i is the number of samples belonging to the i-th category and correctly predicted, FP_i is the sample that was erroneously predicted as the i-th category, while FN_i is the number of samples that belong to the i-th category but were erroneously predicted as other categories (Fig. 11).

Additionally, the change chart of accuracy and loss of DenseNet-264-TRL in the training process on the Fruits 360 are provided in Fig. 12.

From the above figures, it can be clearly seen that: (1) On the whole, the Tensor Network Model based on DenseNet performs well on four datasets. It performs better on the relatively simple Fruits 360 dataset, and the best network can achieve 98.99% accuracy in 30 training cycles. However, for the tensor network based on DenseNet-264, although the depth of the model has increased, the classification accuracy of the model for data has not been significantly improved. On the 100 Sports Image dataset, the tensor network model has more obvious advantages in the number of parameters, but the accuracy of the model is also slightly lower than that of the network model with the full connection layer, and the highest accuracy of 79.52%. This is also an acceptable result. Similarly, for the ASL Alphabet dataset, the accuracy of model can reach 89.48%, and it has an accuracy of 77.90% on Mini-ImageNet dataset. (2) Compared with the popular network models, the performance of our proposed model is slightly better than, or at least comparable to, existing methods. (3) According to the results of our experiment, it is not

Fig. 11 Mini-ImageNet

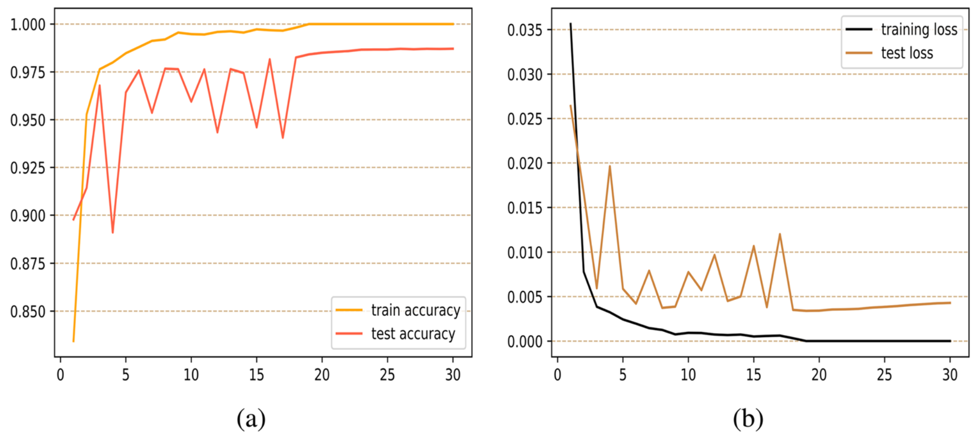

Fig. 12 The changes in accuracy (a) and loss (b) of DenseNet-264-TRL during training on dataset Fruits 360

Table 4 100 Sports Image dataset classification results comparison

Model	Accuracy	Macro_P	Macro_R	Macro_F1-Score
DenseNet-264-TRL	0.7952	0.7864	0.7926	0.7895
VGG16 [3]	0.7530	0.7628	0.7260	0.7444
VGG19 [3]	0.7671	0.7642	0.7568	0.7605
ResNet50 [14]	0.8030	0.7955	0.8100	0.8027
ResNet101 [14]	0.8274	0.8150	0.8234	0.8192
EfficientNetB0 [30]	0.8795	0.8802	0.8764	0.8783
ResNet50-TRL [13]	0.7901	0.7953	0.7826	0.7889
CRU-NET-56 [11]	0.7968	0.7842	0.7964	0.7903

Table 5 ASL Alphabet dataset classification results comparison

Model	Accuracy	Macro_P	Macro_R	Macro_F1-Score
DenseNet-264-TRL	0.8948	0.8924	0.8823	0.8874
VGG16 [3]	0.8492	0.8376	0.8538	0.8457
VGG19 [3]	0.8685	0.8625	0.8518	0.8571
ResNet50 [14]	0.8786	0.8700	0.8725	0.8713
ResNet101 [14]	0.9116	0.9203	0.9087	0.9145
EfficientNetB0 [30]	0.9472	0.9424	0.9360	0.9392
ResNet50-TRL [13]	0.8565	0.8382	0.8418	0.8400
CRU-NET-56 [11]	0.8804	0.8754	0.8680	0.8717

Table 6 Mini-ImageNet dataset classification results comparison

Model	Accuracy	Macro_P	Macro_R	Macro_F1-Score
DenseNet-264-TRL	0.7790	0.7823	0.7684	0.7753
VGG16 [3]	0.7342	0.7451	0.7386	0.7386
VGG19 [3]	0.7536	0.7584	0.7360	0.7473
ResNet50 [14]	0.7867	0.7862	0.7859	0.7860
ResNet101 [14]	0.8046	0.8140	0.8183	0.8169
EfficientNetB0 [30]	0.8119	0.8024	0.8200	0.8109
ResNet50-TRL [13]	0.7231	0.7205	0.7310	0.7242
CRU-NET-56 [11]	0.7748	0.7804	0.7765	0.7771

difficult to see that the accuracy of the tensor network model in data is slightly lower than that of the network model using the full connection layer. However, the number of parameters of tensor network model is relatively small, and the memory occupied by the computer becomes smaller. As expected, the tensor network model can reduce the number of model parameters with less precision cost.

According to the experimental results, the reduction of parameters has a little loss on the classification accuracy of the model, but such a small impact can be accepted. Tensor network model based on DenseNet can greatly reduce the number of model parameters under the condition of small loss of accuracy. Embedding the tensor regression layer into the mainstream DenseNet model can optimize the structure of the model and ensure the performance of the model.

4.3 Ablation study

To further validate the effectiveness of the proposed model, an ablation study is conducted on the Mini-ImageNet dataset. This experiment aims to evaluate the contribution of different components in the proposed model.

The ablation study compared the performance of different variations of the DenseNet-264 model. The full model, DenseNet-264-TRL, achieved an accuracy of 77.90%, with precision, recall, and F1 score at 78.23%, 76.84%, and 0.7753, respectively. This indicates that the proposed model performed reasonably well on the given task. When the Tensor Regression Layer was removed in the DenseNet-264-FC model, the accuracy slightly improved to 79.5%, with precision and recall at 79.0% and 79.2%, and an F1 score of 0.791. This suggests that replacing TRL with fully connected layers may lead to marginal improvements in model performance, although not significantly. Conversely, when DenseNet was excluded in the DenseNet-264-TRL model, the accuracy dropped to 73.2%, along with decreases in precision, recall, and F1 score to 73.5%, 73.0%, and 0.733, respectively. This indicates that DenseNet has a positive impact on the model's performance, and its removal resulted in a noticeable decrease in performance. These findings imply that the increment of fully connected layers may have a positive impact on the model's performance, while the removal of residual learning has a detrimental effect. It is essential to consider these insights for further optimizing the model and enhancing its performance. Additional experiments involving different architectural variations could further validate these observations and provide a deeper understanding of the model's behavior.

4.4 Discussion

It is interesting to embed the tensor regression layer into more robust network models. The structure of the model is optimized through the tensor regression layer, which uses less computer memory and trains fewer parameters, but delivers the same performance as the original model. However, there are still some problems/issues need to be addressed. First, to alleviate model over-fitting, the Dropout method can be applied to the tensor layer to suppress model over-fitting [31]. Second, for different types of datases, how to select suitable optimizers and hyperparameter tuning. Third, divide and conquer and block based modular network/hybridized block modular mode/block combined CNN/evolutionary deep CNN/ also be applied to image classification [32]. Fourth, in our updating model, sensitivity analysis for feature selection and sensitivity analysis of neural networks should be investigated.

5 Conclusion

By integrating the tensor regression layer and DenseNet network organically, this paper proposes a new tensor network DenseNet model. The flattening operation and full connection layer are replaced by a tensor regression network with low rank structure, enabling it to directly receive tensor data. Instead of discarding the spatial information structure of features extracted by the network in the convolution layer, it retains and utilizes the multi-linear structure of data while reducing a significant number of parameters. Experiments reveal that the tensor network model can essentially accomplish image classification. The performance of the proposed model is acceptable for both simple datasets and those with relatively complex backgrounds. Compared with the fully connected layer network model, our proposed model can achieve the goal of image data classification with fewer parameters and less computer memory. Numerical examples demonstrate that the higher computational efficiency of the DenseNet tensor network is desirable over the fully connected layer network model.

Table 7 Ablation Study Results

Model	Accuracy	Macro_P	Macro_R	Macro_F1-Score
DenseNet-264-TRL (Full Model)	0.7790	0.7823	0.7684	0.7753
DenseNet-264-FC (Without TRL)	0.7950	0.7900	0.7920	0.7910
CNN-TRL (Without DenseNet)	0.7325	0.7350	0.7300	0.7330

Acknowledgements We are grateful to the Editor, an Associate Editor and four anonymous referees for their insightful comments and suggestions on this article, which have led to significant improvements. This work was supported in part by the National Social Science Fund (22BTJ025) and in part by the National Natural Science Fund (12271272).

Author Contributions Chunyang Zhu contributed to the conception of the study; Lei Wang performed the experiment; Weihua Zhao contributed significantly to analysis and manuscript preparation; Heng Lian helped perform the analysis with constructive discussions.

Data availability and access All data, models, or code generated or used during the study are available from the corresponding author by request.

Declarations

Ethical and informed consent for data used There is no ethical conflict and no need to informed consent for data.

Competing Interests There are no competing interests to declare.

References

1. Xiaowu D, Yuanquan S, Dunhong Y (2023) Theories, algorithms and applications in tensor learning. Appl Intell 53:20514–20534
2. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Commun ACM 60:84–90
3. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. CoRR
4. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein MS, Berg AC, Fei-Fei L (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis 115:211–252
5. Lebedev V, Ganin Y, Rakhuba M, Oseledets I, Lempitsky VS (2015) Speeding-up convolutional neural networks using finetuned cp-decomposition. CoRR
6. Kiers H (2000) Towards a standardized notation and terminology in multiway analysis. J Chemom 14:105–122
7. Mocks J (1988) Topographic components model for event-related potentials and some biophysical considerations. IEEE Trans Biomed Eng 35(6):482–484
8. Tai C, Xiao T, Wang X, Weinan E (2016) Convolutional neural networks with low-rank regularization. 4th International Conference on learning representations, ICLR
9. Kim Y-D, Park E, Yoo S, Choi T, Yang L, Shin D (2015) Compression of deep convolutional neural networks for fast and low power mobile applications. CoRR
10. Yang Y, Hospedales TM (2017) Deep multi-task representation learning: A tensor factorisation approach. In: International conference on learning representations
11. Chen Y, Jin X, Kang B, Feng J, Yan S (2018) Sharing residual units through collective tensor factorization to improve deep neural networks. In: Twenty-seventh international joint conference on artificial intelligence IJCAI-18
12. Novikov A, Podoprikhin D, Osokin A, Vetrov D (2015) Tensorizing neural networks. Neural Inform Process Syst
13. Kossaifi J, Lipton ZC, Kolbeinsson A, Khanna A, Furlanello T, Anandkumar A (2020) Tensor regression networks. J Mach Learn Rese 21(123):1–21
14. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE Conference on computer vision and pattern recognition (CVPR), pp 770–778
15. Gao Y, Yang LT, Zheng D, Yang J, Zhao Y (2021) Quantized tensor neural network. ACM/IMS Trans Data Sci (TDS) 2:1–18
16. Oldfield J, Georgopoulos M, Panagakis Y, Nicolaou MA, Patras I (2021) Tensor component analysis for interpreting the latent space of gans. In: British machine vision conference
17. Xiaowu D, Yuanquan S, Dunhong Y (2023) Auto-weighted multiple kernel tensor clustering. Complex Intell Syst 9:6863–6874
18. Chen L, Luo X (2023) Tensor distribution regression based on the 3D conventional neural networks. IEEE/CAA J Autom Sin 10(7):1628–1630
19. Zou B-J, Guo Y-D, He Q, Ouyang P-B, Liu K, Chen Z-L (2018) 3D filtering by block matching and convolutional neural network for image denoising. J Comput Sci Technol 33:838–848
20. Arvanitis G, Lalos AS, Moustakas K (2020) Image-based 3D MESH denoising through a block matching 3D convolutional neural network filtering approach. In: 2020 IEEE international conference on multimedia and expo (ICME), IEEE, pp 1–6
21. Lu Z, Whalen I, Dhebar Y, Deb K, Goodman ED, Banzhaf W, Boddeti VN (2020) Multiobjective evolutionary design of deep convolutional neural networks for image classification. IEEE Trans Evol Comput 25(2):277–291
22. Lu Z, Liang S, Yang Q, Du B (2022) Evolving block-based convolutional neural network for hyperspectral image classification. IEEE Trans Geosci Remote Sens 60:1–21
23. Yang J, Xiao L, Zhao Y-Q, Chan JC-W (2023) Unsupervised deep tensor network for hyperspectral–multispectral image fusion. IEEE Trans Neural Netw Learn Syst
24. Wang D, Zhao G, Chen H, Liu Z, Deng L, Li G (2021) Nonlinear tensor train format for deep neural network compression. Neural Netw 144:320–333
25. Kolbeinsson A, Kossaifi J, Panagakis Y, Bulat A, Anandkumar A, Tzoulaki I, Matthews PM (2021) Tensor dropout for robust learning. IEEE J Sel Top Signal Process 15(3):630–640
26. Nie C, Wang H (2022) Tensor neural networks via circulant convolution. Neurocomputing 483:22–31
27. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: 2017 IEEE Conference on computer vision and pattern recognition (CVPR), pp 2261–2269
28. Kolda T (2009) Tensor decompositions and applications. Siam Rev 51(3):455–500
29. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: 2015 IEEE Conference on computer vision and pattern recognition (CVPR), pp 1–9
30. Tan M, Le QV (2019) Efficientnet: Rethinking model scaling for convolutional neural networks
31. Kolbeinsson A, Kossaifi J, Panagakis Y, Bulat A, Anandkumar A, Tzoulaki I, Matthews P (2021) Tensor Dropout for Robust Learning. IEEE J Sel Top Signal Process 15:630–640
32. Zhang S, Zhao J, Zhou Z, Du X (2018) Hybridized block modular mode for image classification. Pattern Recog 83:77–87