带 L2 惩罚的张量神经网络模型及其应用研究

向科聿[1]，黄靖翔[1*]，于卓熙[1]，孙丛婷[2]
1. 辽宁大学 数学与统计学院，辽宁 沈阳 110036
2. 辽宁大学 环境学院，辽宁 沈阳 110036
DOI:10.61369/ASDS.2025050019

摘　　要： 传统的卷积神经网络由卷积层、池化层、扁平化层和全连接层组成。为了保持原线性结构，减少过拟合的同时提高模型的泛化能力，本文在张量链式回归网络层的训练过程中，添加 L2 惩罚项，提高模型的泛化性和稳定性，并将这种方法应用于三个案例研究，实验结果表明，加入惩罚项后比没有惩罚项的张量链式网络，在测试集中均方差 (MSE) 表现更好，模型的鲁棒性得以提高。最后，我们将模型应用到用胸部癌症 CT 扫描预测乳腺癌，结果显示该模型表现出快速的训练速度，这表明我们提出的方法有效。

关 键 词： 机器学习；张量神经网络；张量 TT 分解；卷积神经网络；医疗图像处理

Tensor Neural Network Model with L2 Penalty and Its Application

Xiang Keyu[1], Huang Jingxiang[1*], Yu Zhuoxi[1], Sun Congting[2]
1.School of Mathematics and Statistics, Liaoning University, Shenyang, Liaoning 110036
2.School of Environment, Liaoning University, Shenyang, Liaoning 110036

Abstract: Traditional convolutional neural networks consist of a convolution layer, a pooling layer, a flattening layer, and a fully connected layer. To maintain the original linear structure and reduce overfitting, while enhancing the model's generalization ability, an L2 penalty is added during the training of the tensor chain regression network layer. The results of applying this method to three case studies show that, compared to tensor chain networks without the penalty term, the model with the penalty term performs better in terms of mean squared error (MSE) on the test set, improving its robustness. Finally, we applied the model to predict from chest cancer CT scans, and the breast cancer model demonstrated rapid training speed.

Keywords: machine learning; tensor neural network; tensor TT decomposition; convolutional neural network; medical image processing

引言

在过去的研究中，数据分析主要关注的是低维数据，即数据中的特征量较少。然而随着科技的发展，数据规模逐渐扩大（Si Y, Zhang Y, Cai Y, et al. 2022[1]）学者们逐渐将研究的重心放在高维数据之中。高维数据包含了更多的信息，更加深入的了解数据集的特征，发现其中的相关性和潜在特征，对于提高模型的精度，减少预测值和真实值之间的误差具有重要意义。例如，在图像识别领域，传统的线性分类器只能处理二维图像，并且还会出现准确率过低，预测效果较差的情况。对于不同的图像要重新采用特征工程并重新建模。但是，通过使用深度学习方法，使用卷积神经网络 (CNN) 通过卷积层处理高维图像数据，将图像数据转化为神经网络可以学习到的数值特征，从而实现更准确的图像识别，如：El Sakka M,Mothe J,Ivanovici M. 2024[2]。同样，在自然处理领域，传统的分类方法只能处理单词频率等简单特征，如 TF-IDF，而 Zhang D,Xiao B,Gao C,el al.(2024)[3] 通过循环神经网络 (RNN)，捕获隐藏神经元，实现长短时间序列的处理，这样方法可以处理高维的文本数据。除此之外，在图像分类中，例如：An s,Oh T J,Kim S W,et al.[4] 利用对抗神经网络 (GAN) 也获得了不错的成绩。

虽然高维数据为我们提供了更丰富的信息，能够让我们深入了解数据集、了解数据集中的隐藏特征等等。但是，高维数据也带来了

基金项目：2023年度国家自然社科基金一般项目“高维张量分位数回归模型及应用研究”（23BTJ063）。
作者简介：
向科聿，辽宁大学数学与统计学院，理学学士；
于卓熙，辽宁大学数学与统计学院，教授，博士生导师，研究方向为回归分析、非参数统计、计量经济、统计学习；
孙丛婷，辽宁大学环境学院，教授，博士生导师，研究方向为环境与资源统计学。
通讯作者：黄靖翔，辽宁大学数学与统计学院，博士研究生，研究方向为张量分位数回归，通讯邮箱：huangjingxiang01@126.com。

一些挑战，如计算资源需求增加，内存、显存的增加，神经网络过度学习参数，在零样本和少样本训练中过拟合风险提高。为了充分利用高维数据的优势，我们需要采用合适的方法来降低数据维度，减少数据参数，提高模型性能。为了解决高维数据带来的问题，人们提出了很多降维的方法，其中 Turchetti C,Falaschetti L(2023)[5] 利用主成分分析 (Programmable Counter Array) 进行降维、Zhu W, Zhang L, Jiang X, et al.(2024)[6] 使用线性判别分析 (LDA)，Kingeski R, Henning E (2024)[7] 使用独立成分分析 (ICA)，Tomov M, Sadinov S, Arsov B.(2024)[8] 使用随机森林进行过滤特征 (RF) 等等。

张量作为高阶数据的统称，其受关注的程度在近些年逐渐增加，核心原因在于它的研究不仅仅包含了高维向量和矩阵，还在现实各种领域内有着广泛的应用。例如：音频图、光谱图可以理解为2阶张量，图像可以认为是3阶张量，其中维度对应与长度，宽度和颜色，Li X, Marcus D, Russell J, et al.(2024)[9] 处理 MRI 图像就是3阶张量的应用，视频也可以被描述为4阶张量，并且大量的多模态信息能够被编译为张量的形式，因此对于张量的研究以及如何将张量应用于现实生活中成为了众多学者们的关注对象。

对于机器学习和神经网络方面，众多学者已经将张量数据引入其中，在此之前，高维向量数据已在这两个领域取得不错的成果。向量作为一阶张量，人们常常将张量展平成向量输入已有的模型之中，包括对多维数据进行清洗，通常采用 PCA，Bagging 等方法对多维数据中的特征进行提取，或者进行主成分分析，找出对模型和需要预测值影响力最大的维度，并进行挑选。目前张量在机器学习中的研究方法依然是将张量张开成向量或矩阵，输入机器学习的模型中，包括利用癌症数据对癌症进行预测，分析影响经济的因素等等。向量和矩阵的分解在数学理论有良好的解释性，人们在证明的模型的时候常常采用矩阵分解，例如 Aboutaleb A, Torabi M, Belzer B, et al.[10] 矩阵奇异值分解 (SVD) 的应用。在神经网络方面，由于计算机算力的提升以及工业界对数据预测的精准度的提升，神经网络便成为当代工业界和学术界不可替代的一环。其中神经网络包括卷积神经网络（CNN），图神经网络（GNN）、循环神经网络（RNN）。时间序列与各种神经网络的结合便成为主要的模型，其中包括 Li Z, Li B, Jahng S G, et al.(2024)[11] 使用 VGG 块，Wang S, Gai K, Zhang S.(2024)[12] 使用 ResNet 块，Transformer。并且，在神经网络中，向量扮演着不可替代的角色，包括在卷积层，池化层，全连接层等等，这是由于矩阵可以完美的进行运算，并且能够降低运算成本，而且能够完好的识别数据中的信息，然后传递给神经网络，并作为信息进行输入和输出，最后反馈给用户。

在张量的机器学习和神经网络领域，部分学者已经取得了一定的成果。例如：Sidiropoulos N D, De Lathauwer L, Fu X, et al.(2017)[13] 利用张量分解进行数据挖掘的融合等。在图像处理中，张量的应用更为广泛（Chen G, Bai J, Ou Z, et al.(2024)[14]，Pashaian M, Seyedin Sz(2024)[15]）例如，人脸识别将图像分解和压缩，是三阶张量的应用。在图像识别中，在卷积层中的输入层和激活都使用了三阶张量，在传统的机器学习中，Zhou H, Sarkar R.(2023)[16] 使用了张量处理分类和回归问题。在神经网络层中，Al Olaimat M, Bozdag S(2024)[17] 利用神经网络挖掘用户的潜在信息。

对于高维数据我们首先想到的处理方法是如何降维，张量自然也有其独特的将为方式—张量分解，发展到目前，张量分解的方法已经有好多种：CP 分解、Tucker 分解等等，本文主要采用的是张量链式分解（Tensor Train decomposition, TT decomposition）。张量链式分解是 SVD 矩阵分解的一种形式，其将高维张分解为多个低维张量，再反解 SVD 矩阵分解，这样的做法使得张量链式分解保护了原本张量的链式结构，实现了端到端的多模态结构。但是过去的研究中，人们更多的是关注如何通过 CP 分解和 Tucker 分解来解决张量问题，并将这些方法融入深度学习之中，取得了显著的效果。Bharadwaj V, Malik O A, Murray R, et al.（2023）[18] 利用随机 CP 分解来对分解进行加速；Yuan S, Huang K.（2023）[19] 利用 CP 分解提高了计算精度； Baghershahi P, Hosseini R, Moradi H.(2023)[20] 将 CP 分解融合到图神经网络中 (GNN) 中提高了信息结合的效率； Xiang L, Yin M, Zhang C, et al.（2023）[21]，将 Tucker 分解引入到卷积神经网络之中，并在 gpu 上实现加速过程。Zhang Y, Zhu Y N, Zhang X.(2024)[22] 利用 Tucker 分解和压缩实现了高效储存信息。Novikov A, Podoprikhin D, Osokin A, et al.（2015）[23] 使用 TT 分解，对全连接层采用低秩张量结构，以便用来压缩。 Kossaifi J, Lipton Z C, Kolbeinsson A, et al.（2020）[24] 对 TT 分解进行改良，实现了端到端的网络架构。Liu Y, Chakraborty N, Qin Z S, et al.(2023)[25] 将张量回归网络在遗传学中提高了模型的精度，Liu Y, Liu J, Long Z, et al.（2023）[26] 将张量回归网络运用到各个方面。

现如今的研究中，大部分的卷积神经网络都聚焦于对卷积核、池化层进行优化，来提高模型的准确率和减少真实值和预测值之间的误差，如：Zhou Y, Tan K, Shen X, et al. (2024)[27] 利用扁平层对张量进行压缩，改变了其原有的结构，舍弃大量的线性结构，丢失了多模态信息。由于全连接层需要将张量展平为向量，此时通过全连接层的参数便有了几何倍的增长，此时就会有一个不可避免的问题，即：维度爆炸问题，影响了模型的精度和增加了计算机算力的难度。再加上逐渐增多的数据集和数据样本，如何提高模型的精度，使得模型的预测值尽可能接近原始数据，如何使得模型更加具有泛化性，如何减少错误率，如何减少模型的误差，也是需要考虑的地方。因此，如何提高精准度，如何节省数据资源，减少卷积网络中的参数就成为了要解决的问题。

已有的相关研究中，通常是利用线性回归对数据继进行处理，随着数据集规模的不断扩大，参数个数的增多，模型的拟合效果差强人意，这时就需要适当的选择加入惩罚项来提高模型的泛化能力。

本文主要的贡献是采用张量链式回归层 (TTRL) 代替传统卷积网络的扁平层和全连接层，即采用张量链式分解（TT decomposition）来分解高维张量，这样保持模型原有的多模态结构，减少由于扁平层张量压缩所带来的过多的参数，并且在原有的张量链式回归层 (TTRL) 基础上，加入 L2 惩罚项，加速模型的训练速度，并且减少模型的过拟合情况，提高模型的泛化性 . 我们将模型应用于胸部平面 CT 数据集中进行预测，获得了良好的结果。

一、方法与模型介绍

我们现在假设一个张量$X \in R^{I_0 \times I_1 \times \ldots I_N}$，将此时的张量展开为一个矩阵为$X_{[n]} \in R^{I_n \times I_M}$（其中$I_M = \prod_{K=0,\ K \neq n}^{N} I_K$），将这个张量做一个映射，将（$i_0,...,i_N$）映射到（$i_0, j$），其中 j=$\sum_{k=0}^{N}$，$i_k \times \prod_{l=k+1}^{N}$，$I_l$.（Kossaifi J, Lipton Z C, Kolbeinsson A, et al. (2020)[24]）。

我们假设一个张量，$X \in R^{I_0 \times I_1 \times \ldots I_N}$，和一个矩阵 M $\in R^{R \times I_n}$，如果一个 n 维张量的模的大小为（$I_0 \times \ldots \times R \times \ldots I_N$），之后我们可以将 $X \times_n M$ 表示为 M $X_{[n]} \in R^{I_0 \times \ldots I_{n-1} \times R \times I_{n+1} \times \ldots I_N}$。

如果张量X，$Y \in R^{I_0 \times I_1 \times \ldots I_N}$，且具有相同大小的形状，我们可以定义

$$\langle X, Y \rangle = \sum_{i_0=0}^{i_0-1} \sum_{i_1=0}^{i_1-1} \ldots \sum_{i_n=0}^{i_n-1} x_{i_0} \ldots x_{i_n} y_{i_0} \ldots y_{i_n}, \quad (1.1)$$

其中$X \in R^{I_X \times I_0 \times I_1 \times \ldots I_N}$，$Y \in R^{I_0 \times I_1 \times \ldots I_N \times I_y}$，其中$\langle X, Y \rangle$称为广义内积。

（一）张量向量化

张量向量化是张量重塑为不同矩阵的操作，其中有两种矩阵化形式，包括模式矩阵化和顺序矩阵化。模式矩阵化是将第 s 个模式设置为行，然后列举其余的模式。让$p_{-s} = \prod_{i=1, i \neq s}^{d} p_i$，这时张量$X$被重塑为$p_s$ −by− p_{-s}矩阵，用$[X]_{(s)}$来表示其中的元素，其中元素$X_{i_1, i_2 \ldots i_d}$被映射为$[X]_{(s)}$中的（i_s, j）−th 元素，

$$j = 1 + \sum_{k=1, k \neq s}^{d} (i_k - 1) J_k, J_k = \prod_{l=1, l \neq s}^{k-1} p_l, \quad (1.2)$$

顺序矩阵化是将张量X重塑为$\prod_{i=1}^{s} p_i, \prod_{i=s+1}^{d} p_i$矩阵，用$[X]_s$来表示，其中我们行从1 到 s 进行列举，列从 s+1 到 d 进行列举，其中元素$X_{i_1, i_2 \ldots i_d}$被映射为$[X]_{(s)}$中的（i_s, j）−th 元素，有：

$$i_{s+1} = (i_s - 1) p_1 \ldots p_{s-1} + (i_{s-1} - 1) p_1 \ldots p_{s-2} + \ldots i_1. \quad (1.3)$$

$$j = (i_d - 1) p_{s+1} \ldots p_{d-1} + \ldots i_{s+1}. \quad (1.4)$$

（二）张量链式分解

张量链式分解与 Tucker 分解一样稳定，同时可以像 CP 分解一样压缩空间 . 我们假设一个张量$X \in R^{p_1 \times p_2 \ldots p_d}$，能分解为：

$$X_{i_1, i_2, \ldots, i_d} = G_1(i_1) G_2(i_2) \ldots G_{d-1}(i_{d-1}) G_d(i_d), \quad (1.5)$$

其中G_k是$r_{k-1} \times r_k$的矩阵，$1 \leq k \leq d$，$r_0 = r_d = 1$。

当 $2 \leq k \leq$ d−1，我们将这些矩阵$G_k(i_k)$叠加为一个张量$G_k \in R^{r_{k-1} \times p_k \times r_k}$，这样可以定义$[G_k]_{(3)} = (G_k^{\intercal}(1), \ldots, G_k^{\intercal}(p_k))$。

此外，我们令 $G_1 = G_1^{\intercal}(1), \ldots G_1^{\intercal}(p_1)^{\intercal} \in R^{p_1 \times r_1}$，和$G_d = G_d^{\intercal}(1), \ldots G_d^{\intercal}(p_d)^{\intercal} \in R^{p_d \times r_{d-1}}$，我们将$G_1, G_2, \ldots G_{d-1}, G_d$，当作 TT 分解的核心。

将G_1的$i_1 - th$向量、G_2的$i_2 - th$矩阵、G_{d-1}的G_2，…，$i_{d-1} - th$矩阵与G_d的$i_d - th$矩阵依次相乘得到X元素。

此外，TT 分解的行列可以被定义为（$r_1, \ldots r_{d-1}$），其中使得$r_i = rank([X_i])$ 其中，$1 \leq i \leq d$ −1. 它的顺序变化还有另一种变换形式：

$$[X]_k = (I_{p_2 \cdots p_k} \otimes G_1) \ldots (I_{p_k} \otimes [G_{k-1}]_2)[G_k]_2 [G_{k+1}]_1 ([G_{k+2}]_1 \otimes I_{p_{k+2}}) \ldots (G_d^{\intercal} \otimes I_{p_{k+2} \cdots p_{d-1}}), \quad (1.6)$$

Sidiropoulos N D, De Lathauwer L, Fu X， et al. (2017)[21] 证明 TT 分解能够应用于张量回归。

（三）张量链式回归

我们考虑张量输入值和输出值：

$$Y_i = \langle A, X_i \rangle + \varepsilon_i, 1 \leq i \leq N, \quad (1.7)$$

其中，$Y_i \in R^{q_1 \times q_2 \times \ldots q_n}$，$X_i \in R^{p_1 \times p_2 \times \ldots p_m}$，$A \in R^{q_1 \times q_2 \times \ldots q_n \times p_1 \times p_2 \times \ldots p_m}$作为系数张量，$\varepsilon_i \in R^{q_1 \times q_2 \times \ldots q_n}$作为随机误差项，N 为样本大小。对于所有的$1 \leq j \leq n$ −1，其中Y_i是具有嵌套或层次结构，其中第 j 个模块是能被（j+1）个模块进行嵌套的。

我们假设系数张量A的秩 R =（$r_1, r_2, \ldots r_{m+n-1}$）。使得$r_i = rank([A]_i)$，对于所有的 i，$1 \leq i \leq m$ +n−1，参数空间可以表示为：

$\theta_i(R) = \{A \in R^{q_1 \times q_2 \times \ldots q_n \times p_1 \times p_2 \times \ldots p_m} : rank([A]_i) \leq r_i, 1 \leq i \leq m+n-1\}$，其中，我们有：

$$A = \left[\left[G_1; G_2, .., \sum, \ldots G_{m+n-1}, G_{m+n}\right]\right], \quad (1.8)$$

$$[A]_n = (I_{q_2 \cdots q_n} \otimes G_1) \ldots (I_{q_n} \otimes [G_{n-1}]_2)[G_n]_2 [G_{n+1}]_1 ([G_{n+2}]_1 \otimes I_{p_1}) \ldots (G_{n+m}^{\intercal} \otimes I_{p_1 \cdots p_{m-1}}), \quad (1.9)$$

其中$G_1^{\intercal} G_1 = I_n, [G_i]_2^{\intercal} [G_i]_2 = I_{r_i}$，$2 \leq i \leq n, [G_i][G_i]_1^{\intercal} = I_{r_{i-1}}$。

对于 n+1 $\leq i \leq n$ +m−1，

$$G_{n+m}^{\intercal} G_{n+m} = I_{r_{n+m-1}}, \quad (1.10)$$

其中，$\sum \in R^{r_n \times r_n}$为对角线，并且 vec($Y_i$) = $[A]_n$ vec(X_i) + vec(ε_i)，

所以对于模型$Y_i = \langle A, X_i \rangle + \varepsilon_i, 1 \leq$ i $\leq N$ 能被改写为：

$$[G_n]_2^{\intercal} (I_{q_n} \otimes [G_{n-1}]_2^{\intercal}) \ldots (I_{q_2 \cdots q_n} \otimes G_1^{\intercal}) \text{vec}(Y_i) = \sum \{[G_{n+1}]_1 ([G_{n+2}]_1 \otimes I_{p_1}) \ldots (G_{n+m}^{\intercal} \otimes I_{p_1 \cdots p_{m-1}}) \text{vec}(X_i)\} + [G_n]_2^{\intercal} (I_{q_n} \otimes [G_{n-1}]_2^{\intercal}) \ldots (I_{q_2 \cdots q_n} \otimes [G_{n-1}]_2^{\intercal}) \text{vec}(\varepsilon_i) \quad (1.11)$$

以上模型称为张量回归模型。

在第一阶段，对Y_i沿着第一种方式，通过G_1将q_1转变为r_1因子，然后压缩堆叠第一个模组和第二个模组，通过G_2将$r_1 q_2$转变为r_2的因子。依此类推，直到我们所提取的r_k因子。相反X_i的因子的提取顺序遵循相反的原则，从最后一个行、列依次迭代到第一个行、列。最后预测器中提取到r_k，第二阶段将r_k当作预测器的响应器，系数矩阵Σ为对角因子，因此只涉及r_k参数。

（四）张量链式分解回归网络层

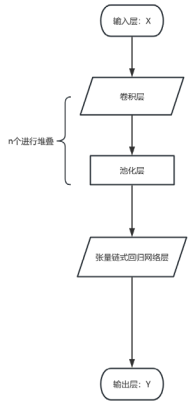

图1 神经网络模型构建

接下来，我们改变传统卷积网络网络层的堆积情况，如图1、2中，我们分别采用卷积层，池化层，卷积层，进行堆叠，之后再采用张量链式回归层输出结果，最后构成我们主体的模型，改变了传统卷积网络 (CNN) 的结构，利用张量链式回归层 (TTRL) 来替换最后一个池化层，扁平层和全连接层，最大程度的保留了原有的线性结构。其中张量回归网络作为新的层，是具有学习能力的，能够学习数据集的特征，识别和输出数据。

通过上述流程图，我们可以发现，在传统卷积网络中，我们将张量线性网络替换了原有的扁平层，全连接层，由于我们之前在引言中讨论过，当一个张量经过扁平层后，会损失原有线性性质和指数级增加神经网络参数，然而，当通过张量链式回归层 (TTRL) 的神经网络将会避免这一情况发生。对于此模型的参数，我们采用反向传播进行调节参数。

每一个回归的权重和每一个因素梯度的核心：

$$\frac{\partial W}{\partial U^{(K)}}=\frac{\partial G\times_0 U^{(0)}...\times_{(N+1)}U^{(N+1)}}{\partial U^{(K)}}, \tag{1.12}$$

我们对回归权重进行张开，并对梯度的核心进行矢量化可得到：

$$\frac{\partial v(W)}{\partial v(G)}=\frac{\partial\left(U^{(0)}\otimes...\otimes U^{(N+1)}\right)v(G)}{\partial v(G)}. \tag{1.13}$$

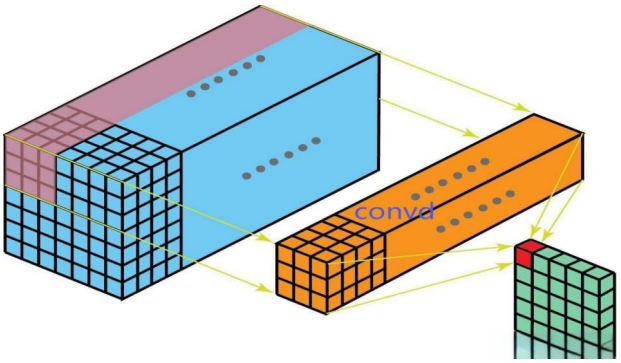

图2 利用卷积层进行卷积

二、模型构建与算法设计

（一）模型构建

基于张量链式分解作为回归层的启发，我们对张量链式回归层添加惩罚项，用来减少模型的复杂度和减少模型过拟合情况。

我们假设输入的张量为 $X\in R^{M\times I_0\times I_1\times...I_N}$，这一个张量对应着一批具有的 M 维的样本，即 ($x_1,...x_m$)，此时存在一个 y，使得 $y\in R^{S\times O}$，其中 O 代表的是输入样本中的标题，我们创建这个回归网络层的权重，然后在这个网络层中加入惩罚项。我们设权重为 W，且有 $W\in R^{I_0\times I_1...I_N\times O}$，b 为随机误差项，$L$ 为损失函数，使得：

$$\begin{cases} Y=\langle X,W\rangle_N+b, \\ L=\dfrac{1}{M}\sum_{i=1}^{M}\left\|Y_i-\overline{Y}_i\right\|^2+\lambda\|W\|_2^2 \\ W=\left[\left[G;U^{(0)},..,U^{(N)},U^{(N+1)}\right]\right]. \end{cases}, \tag{2.1}$$

我们将这个张量链式回归层 (TTRL) 作为神经网络层中的可训练层，在原有的基础上，对原本张量回归网络的回归迭代中，加入 L2 正则化项。我们允许这一层可以进行学习（其中包括卷积层和张量回归）潜在特征 . 我们直接将这个作用于高阶输入张量中的应用，我们利用张量链式回归层 (TTRL) 来代替原有最后一层池化层，扁平层和全连接层，保留了线性结构，直接实现端到端的操作，并且由于正则化的原因，减少了此模型过拟合的风险，增加了模型的稳健性。

（二）张量链式分解算法（TT decomposition）

张量链式分解可以分为 2 个步骤 .

第一步，我们将输入的张量设为 A，

$$A\in R^{q_1\times q_2\times...q_n\times p_1\times...\times p_m},\ p=\prod_{i=1}^{m}p_i, \tag{2.2}$$

我们将张量的维数设为 ranks，ranks=($r_0,...,r_{m+n}$)，有 $r_0=r_{m+n}=1$，$W=A$.。将张量 A 初始化，并令 $C=A$. 之后遍历 i，其中 i=1，2，...，n−1，使用奇异值矩阵分解 (SVD) 对张量 C 进行分解，可以得到：

$$M_1(C)=U_i\sum_i V_i^{\intercal}, \tag{2.3}$$

重塑，$U_i\in R^{(r_{i-1})q_i\times r_i}$，将其的维数改变为 $G_i\in R^{(r_{i-1})\times q_i\times r_i}$，

重塑 $\sum_i V_i^{\intercal}\in R^{r_i\times\left(\prod_{i=i+1}^{n}q_i\right)\times p}$，将其的维数改变为

$$C\in R^{(r_iq_{i+1})q_{i+2}\times...\times q_n\times p_1\times...\times p_m}, \tag{2.4}$$

最后，可以得到我们经过初始化的张量 C 的维数可以改变为：

$$C\in R^{(r_iq_{i+1})q_{i+2}\times...\times q_n\times p_1\times...\times p_m}, \tag{2.5}$$

第二步，之后我们反解矩阵，对于 i = m+n，m+n−1，...，n+2，我们依然采用矩阵分解，可以得到 $M_{-1}(C)$ = $U_i\sum_i V_i^{\intercal}$ 和 rank r_{i-1}，

重塑 $V_i^{\intercal}\in R^{(r_{i-1})\times p_ir_i}$，将其的维数改变为 $G_i\in R^{(r_{i-1})\times p_i\times r_i}$，

重塑 $U_i\sum_i\in R^{r_{n-1}\times\left(\prod_{i=i}^{i-n-1}p_i\right)\times r_{i-1}}$，

将其的维数改变为 $C\in R^{r_{n-1}q_n\times p_1\times...\times p_{i-n-2}\times(p_{i-n-1}r_{i-1})}$，

最后，经过初始化的张量 C 转变为 $C\in R^{r_{n-1}q_n\times p_1\times...\times p_{i-n-2}\times(p_{i-n-1}r_{i-1})}$。我们对整体进行奇异值矩阵分解，可以得到 $C=U\sum V^{\intercal}$,rank = r_n，

重塑 $U\in R^{r_{n-1}q_n\times r_n}$ 矩阵，可以转变为 $G_i\in R^{(r_i-1)\times q_i\times r_i}$

重塑 $V^{\intercal}\in R^{(r_{i-1})\times p_ir_i}$ 矩阵，可以转变为 $G_i\in R^{(r_{i-1})\times p_i\times r_i}$，

最后我们可以得到：

$$\left[\left[G_1,G_2,...,G_n,\sum,G_{n+1},...,G_{m+n-1},G_{m+n}\right]\right]. \quad (2.6)$$

（三）交替方向乘子法（ADMM）

首先我们输入的张量为 $\{(Y_i,X_i),1\le i\le N\}$，训练数据为：$\{(Y_i,X_i)\}_{i=1}^{M}$

张量链式分解的行列为 :R=($r_1,...,r_{m+n-1}$)，迭代 K 次。

首先初始化张量值：$W^0=A^0=\theta$，$\Lambda^{(0)}=0$

遍历 k = 1，2，...，K.

$$W^{(k+1)}=argmin(\sum_{i=1}^{\lambda}\left(Y_i-\langle X_i,W\rangle_N-b\right)^2+\lambda\left\|Z^k\right\|_2+\Lambda^{(k)}\left(W-Z^k\right)$$

$$+\frac{\rho}{2}\left\|W-Z^k\right\|_2^2, \quad (2.7)$$

$$W^{(k+1)}=argmin(\sum_{i=1}^{\lambda}\left(Y_i-\langle X_i,W^k\rangle_N-b\right)^2+\lambda\left\|Z^k\right\|_2+\Lambda^{(k)}\left(W-Z\right)$$

$$+\frac{\rho}{2}\left\|W^k-Z\right\|_2^2, \quad (2.8)$$

$$\Lambda^{(k+1)}=\Lambda^{(k)}+\rho\left(W^{(k+1)}-Z^{(k+1)}\right), \quad (2.9)$$

其中 Λ 是拉格朗日算子，ρ 为惩罚项。

（四）适应性矩估计算法（ADAM）

适应性矩估计算法结合了适应性梯度算法和均方根算法的优点，能够为不同的参数设计独立的自适应率。由于对于图像集的优化算法，并非为凸优化算法。然而传统的梯度下降算法会出现局部最优解，因此传统的梯度下降便不适用于非凸优化，因此我们将采用 Dereich S, Jentzen A. (2024)[28] 的 Adam 算法进行优化，Adam 算法的关键部分来源于指数加权平均值来估计梯度动量和二阶矩，

$$v_t\leftarrow\beta_1 v_{t-1}+\left(1-\beta_1\right)g_t, \quad (2.10)$$

$$s_t\leftarrow\beta_2 s_{t-1}+\left(1-\beta_2\right)g_t^2, \quad (2.11)$$

其中 β_1，β_2 是非负加权参数。

由于方差估计的运算速度远远慢于动量估计的移动速度。可以初始化 v，s，但是会得到初始偏差，因此需要进行标准化。

$$v\_hat_t=\frac{v_t}{1-\beta_1^t},s\_hat_t=\frac{s_t}{1-\beta_2^t}, \quad (2.12)$$

有了正确的估计方式，就可以写出更新方程。

$$g'=\frac{\eta v\_hat_t}{\sqrt{s\_hat_t}+\dot{o}}, \quad (2.13)$$

通常在优化参数的时候，选择$\dot{o}$ =1e−6，为了数值的稳定。

三、数值模拟

由于加入惩罚项之后，我们无法从理论上做出决断，来判断添加惩罚项后与未添加惩罚项之前，在精度和损失值上是否有提升。因此，在这个部分，为了验证所做的惩罚项是有效的，我们分别对我们的模型和原模型在随机数组中进行实验，查看这个数组在已添加惩罚项的张量回归网络和未添加张量回归网络中 MSE 的情况。

（一）随机数组和随机矩阵的生成

为了验证我们加入惩罚项的张量回归网络是否有效，我们利用 python 中的 numpy 库，随机生成高维整数数组，由于像素点的取值范围在0−255之间，为了模拟真实的效果，其中我们将 x 的大小限制在0−255之间，以大小为 (28,28) 生成张量。对于图像，有彩色图像和灰度图像之分，为了模拟更多的噪声，我们将数值的通道设置为3，即 x 的维度为 (3,28,28) 目，$x\in R^{n*x^{28}*x^{28}*x^3}$。之后，我们对生成的 x 进行归一化，使得其被限制在0−1之间，然后再对 y 进行独热编码，对于 y，我们将 y 分为10个类别，即 (10,) 最后把这个数组放入未添加惩罚项和已添加惩罚项的模型之中，分别比较在2种张量网络下的 MSE 好坏。

（二）数值模拟结果

为了使得模型更具稳健性和泛化性，我们改变输入模型样本量，查看不同的数据样本下，添加惩罚项是否对张量链式分解网络有效。我们采用相同的 epochs 和 batch_size 进行训练模型。在这些样本量中，如果添加惩罚项后的张量链式网络均好于未添加惩罚项的张量链式网络，则说明我们添加的惩罚项是有效的，并且可进一步采用此模型进行实际情况的应用，如图3及表1所示。

表1 随机数组中添加 l2惩罚后与未添加惩罚的张量链式回归网络的 MSE 比较

样本数量	添加惩罚项后的张量链式分解回归网络的训练集	添加惩罚项后的张量链式分解回归网络的测试集	未添加惩罚项后的张量链式分解回归网络的训练集	未添加惩罚项后的张量链式分解回归网络的测试集
100	0.593933	0.812683	0.593871	0.812829
200	0.728827	0.564313	0.728750	0.564389
500	0.313030	0.299911	0.312977	0.299944
1000	0.126347	0.123470	0.126343	0.123487
2000	0.100025	0.099998	0.100045	0.100002
5000	0.100082	0.099987	0.100000	0.100000
10000	0.100004	0.099998	0.099999	0.100001

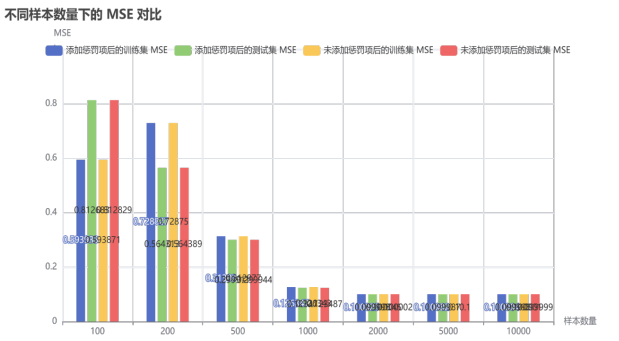

图3 不同样本下两种模型的 MSE

为了验证添加惩罚项后的张量链式网络比未添加惩罚项的张量链式网络，我们分别比较了2种模型在不同样本下的 MSE 的情况。如图4、5所示：

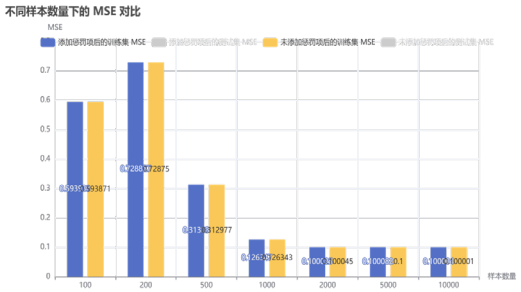

图4 不同样本下两种模型在训练集上的 MSE

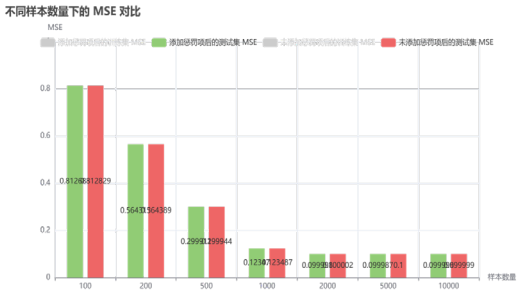

图5 不同样本下两种模型在测试集上的 MSE

模拟结果表明，在训练集中，添加惩罚项后的张量链式网络在各个样本中的 MSE 均高于未添加惩罚项的张量链式网络，在测试集中，添加惩罚后的张量链式网络均好于未添加惩罚项的网络 . 说明添加惩罚项之后，虽然增加了训练集上的 MSE，但是提高了模型的泛化能力，减少了测试集中真实值与预测值之间的误差。为了探究，实验是否出现过拟合的情况，我们分别比较添加惩罚项的张量链式网络中的训练集和测试集中的 MSE。如图6、图7所示。

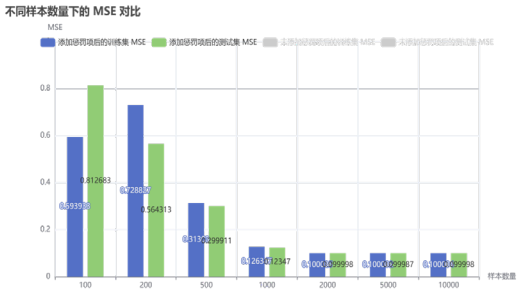

图6 不同样本下添加 L2 惩罚项的张量链式网络在训练集和测试集上的 MSE

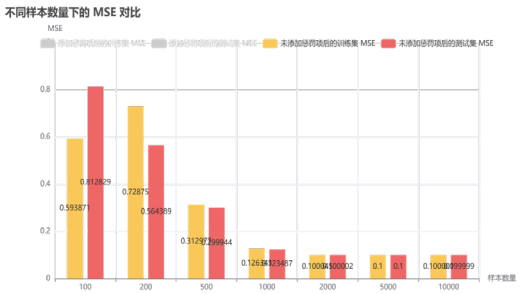

图7 不同样本下未添加惩罚项的张量链式网络在训练集和测试集上的 MSE

实验结果表明，在添加惩罚项的张量链式网络和未添加惩罚项的张量链式网络均未出现过拟合的情况。并且，在测试集中，加入惩罚项的 MSE 比未加入惩罚项 MSE 要好，在样本数据中，尤其是在样本量为100、200、500、1000、2000中均有巨大的提升，其中在100样本集中，在训练集上增加了0.000062的 MSE，在测试集上减少了0.000146的 MSE，减少了模型的过拟合情况。并且在数据集中可以表明，在样本量越小的模型中，使用惩罚项越有效，在大量的样本集中数据差距较小。这是由于神经网络能够通过大量的数据集不断的学习产生的结果，说明在张量链式回归层加入 L2惩罚项是有效的，并且很好的提高了模型在训练集和测试集中的 MSE，使得模型更加具有泛化性，减少了预测值与真实值之间的误差，说明经过 L2惩罚之后，原模型可以很好的减少由于深度神经网络过度学习模型的噪声所带来的过拟合的问题，使得模型更加稳定，因此我们所添加的惩罚项是有效的。

四、张量链式分解在胸部 CT 扫描数据集中应用

为了验证我们在上一部分模拟数据的准确性，以及加入惩罚项之后张量回归网络在实际情况中的可行性，我们将加入惩罚项的张量回归网络用于胸部癌症的预测，并且查看预测值和原数据集的值偏差大小。

数据集来源 kaggle，其中具体的数据集来源于：https://www.kaggle.com/datasets/mohamedhanyyy/chest-ctscan-images.

数据集中包含腺癌、大细胞癌、鳞状细胞癌 3 种胸癌类型，正常细胞，其中包含训练集0.7，测试集0.2，验证集0.1，其中训练集有613张图片，测试集有215张图片，验证集有72张图片。

肺腺癌：肺腺癌是最一种肺癌形式，在胸部 CT 扫描的数据集中，肺腺癌图像集占所有图像集的30%，约占所有非小细胞肺癌发病率的40%。腺癌存在于几种常见的癌症中，包括乳腺癌、前列腺癌和结直肠癌。肺腺癌位于肺的外部区域，位于分泌粘液并帮助我们呼吸的腺体中。患者的症状包括咳嗽、声音嘶哑、体重减轻和虚弱。

大细胞未分化癌：大细胞未分化癌肺癌生长和扩散迅速，可以在肺部的任何部位发现。在日常的胸部癌症中，这种类型的肺癌通常占所有非小细胞肺癌病例的10% 至15%。

鳞状细胞癌：这种类型的肺癌位于肺部中心，较大的支气管将气管连接到肺部，或位于主要气道分支之一。鳞状细胞肺癌约占所有非小细胞肺癌的30%，通常与吸烟有关。

首先我们对图片集中的训练集中的腺癌、大细胞癌、鳞状细胞癌，正常细胞进行编码，将图片以数据形式输入神经网络层中，使得神经网络可以更好的学习图像方面。由于原数据集是灰度图像，我们首先对灰度图像进行处理，利用图像增广，包括：随机裁剪，随机翻转，中心翻转来剪切图像集如图8所示。

为了更好的提高模型的精度和减少模型的损失 (loss)，我们采用 opencv 对图像集进行处理，首先我们对灰度图像的通道改为 RGB 通道3通道，我们将 RGB 颜色通道改为 HSV 颜色通道，并且为灰度图像集随机上色，用来强化数据集。我们对图像集进行随机裁剪，随机翻转，中心翻转来剪切图像集，增加图像集噪声，在训练神经网络时，我们固定随机种子，卷积核大小，池化层大小使得模型可以复现，最后我们比较测试集中的损失值的大小用来测试带 L2惩罚项的张量链式回归层是否有效，是否能够避免过拟合情况，如图8、图9所示。

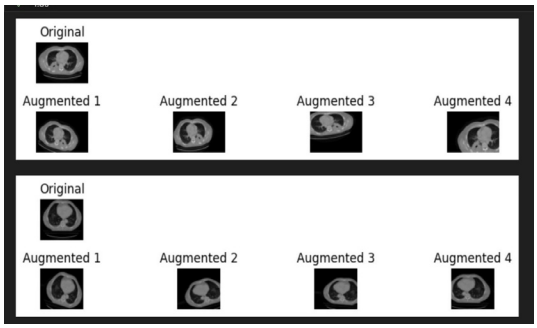

图8 数据增强后的图像 (灰度图像)

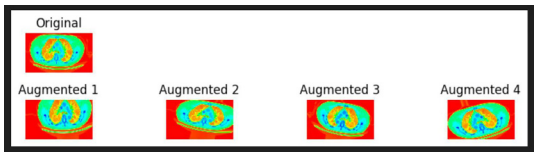

图9 数据增强后的图像 (HSV) 图像

由于图像集的优化过程并非全为凸优化，而且梯度下降（SGD）等方法在非凸优化中表现较差，因此，我们将使用 Adam 为优化器进行训练。在训练过程中，我们采用交叉熵作为我们函数的损失函数进行评估模型的好坏程度以及真实值与预测值之间的误差，如表2，图10所示。

表2 添加 L2惩罚项后的张量链式网络模型下的胸部扫描 CT 的损失值和训练时间

模型的值	训练集中的损失	验证集中的损失	训练时间
添加 l2惩罚项的张量链式网络	0.8537	0.9761	45.17s

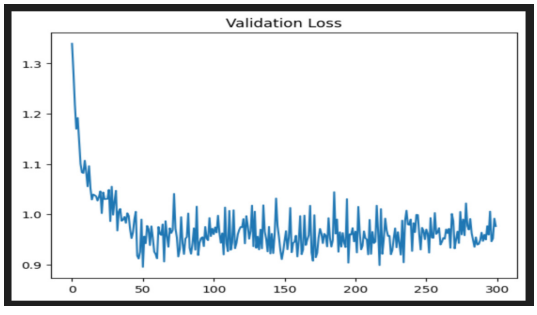

图10 张量链式回归（TTRL）在胸部 CT 平面扫描数据集中的训练过程

实验结果表明，我们的模型在训练速度上和参数，以及在loss上都表现良好，在训练集和验证集中的损失值分别为0.8537，0.9761。在训练集和验证集中的损失值都比较小，说明添加惩罚项后的张量链式网络在 CT 平面扫描中表现良好。

由于在训练集中的损失值和在测试集中的损失值之间相差较小，并且由模型拟合图像可以得到模型是未出现维度爆炸的情况。实验表明模型没有出现过拟合和欠拟合的情况，并且拟合程度较好，说明我们的模型是有效的，并且我们的模型训练时间较少，能够及时的对数据集进行分析。

五、结论与展望

（一）结论

在本文中，我们利用张量回归网络层代替卷积网络中原有的扁平层和全连接层，实现了端到端的操作，通过实验表明，张量回归网络具有学习标签等特征的能力，完全有能力替代传统的扁平层和全连接层，并且张量链式网络保证了原有的线性结构。我们对已有的张量回归网络层中的回归层，加入了 L2惩罚项，使得模型更加具有泛化性和稳健性。为了验证我们加入 L2惩罚项是有效的，我们采用计算机随机生成高维张量，并且将这些张量带入原有的模型和我们更新过后的模型进行比较，发现在评价指标 MSE 上，在测试集中的 MSE 有提升，说明我们的模型在数值模拟的过程中是有效的。

我们将模型应用到 CT 图像中，进行癌症的预测。由于 CT 扫描数据集的大小限制，这对神经网络更具有挑战，如何在较少的数据集中获得更好的训练速度和更小的损失值，便成为了我们需要探讨的问题。我们添加 L2惩罚项之后，实验表明，TTRL 模型在训练集上的损失值为0.8537，并未发现过拟合的情况，在验证集上的损失值为0.9761，说明 TTRL 在验证集中表现良好，并且预测值与真实值之间的差距较小，说明在 TTRL 回归层中添加 L2惩罚项是有效的。

由于计算资源的不足，我们只是进行了卷积层、池化层进行堆叠，并没有考虑在卷积层和池化层进行别的优化，譬如，采用残差神经网络进行处理卷积层和池化层，来提高神经网络的精度和计算时间等等。因此，我们采用堆叠情况会导致模型的精度下降，并且会导致模型的运算时间加快。其次，我们并没有采用多个 GPU 进行小规模训练，因此，我们提出的模型在计算速度会偏低。

（二）展望

由于计算机算力的问题，我们只是将输入层，卷积层，池化层，和张量网络层进行简单的堆叠。我们只说明了加入惩罚项后的张量链式网络比未添加惩罚项后的张量链式网络更好，能够避免过拟合的情况。为了更好的提高神经网络的准确率，可以采用目前比较流行的多模态模型，如 Transformer，自注意力回归，VGG 块和 ResNet 模型。

如果采用这样的框架，会提高模型的复杂程度。但是，在提高模型的复杂程度的同时，也会提高神经网络学习图像的效率

和精度，更好地提升模型的精度和减少预测值与真实值之间的误差。

在未来的工作中，可以采用更加复杂的数据集，或者采用大型数据集，添加更多的噪声来评判此模式是否比未添加惩罚项的张量回归网络更具有泛化性，也可以采用 L1 正则化项改变此模型的惩罚项，利用弹性网络或许也是一种不错的方法。当然，不断调整卷积层，池化层，张量回归网络层的层数，使模型能够深层次的学习数据集中的特征，提高模型的准确率也是一种不错的方法。

参考文献

[1]Si Y, Zhang Y, Cai Y, et al. An efficient tensor regression for high-dimensional data[J].arXiv e-prints, 2022, arXiv:2205.13734.

[2]El Sakka M, Mothe J, Ivanovici M. Images and CNN applications in smart agriculture[J]. European Journal of Remote Sensing, 2024, 57(1): 2352386.

[3]Zhang D, Xiao B, Gao C, et al. Modeling Bilingual Sentence Processing: Evaluating RNN and Transformer Architectures for Cross-Language Structural Priming[C]// Proceedings of the Fourth Workshop on Multilingual Representation Learning (MRL 2024). 2024: 127-136.

[4]An S, Oh T J, Kim S W, et al. Self-clustered GAN for precipitation nowcasting[J]. Scientific Reports, 2024, 14: 9755.

[5]Turchetti C, Falaschetti L. Tensor PCA from basis in tensor space[J]. arXiv e-prints, 2023, arXiv:2305.02803.

[6]Tomov M, Sadinov S, Arsov B. Impedance Matching Optimization of RF Networks[J]. Engineering Proceedings, 2024, 70(1): 46.

[7]Dereich S, Jentzen A. Convergence rates for the Adam optimizer[J]. arXiv e-prints, 2024, arXiv:2407.21078.

[8]Li Z, Li B, Jahng S G, et al. Improved vgg algorithm for visual prosthesis image recognition[J]. IEEE Access, 2024, 12: 45727-45739.

[9]Li X, Marcus D, Russell J, et al. Weibull parametric model for survival analysis in women with endometrial cancer using clinical and T2-weighted MRI radiomic features[J]. BMC Medical Research Methodology, 2024, 24(1): 107.

[10]Aboutaleb A, Torabi M, Belzer B, et al. Deep Learning-based Auto-encoder for Time-offset Faster-than-Nyquist Downlink NOMA with Timing Errors and Imperfect CSI[J].IEEE Journal of Selected Topics in Signal Processing, 2024, 18(7): 1178-1193.

[11]Li Z, Li B, Jahng S G, et al. Improved vgg algorithm for visual prosthesis image recognition[J]. IEEE Access, 2024, 12: 45727-45739.

[12]Wang S, Gai K, Zhang S. Progressive feedforward collapse of resnet training[J]. arXiv e-prints, 2024, arXiv:2405.00985.

[13]Sidiropoulos N D, De Lathauwer L, Fu X, et al. Tensor decomposition for signal processing and machine learning[J]. IEEE Transactions on signal processing, 2017, 65(13): 3551-3582.

[14]Chen G, Bai J, Ou Z, et al. PSFHS: intrapartum ultrasound image dataset for AI-based segmentation of pubic symphysis and fetal head[J]. Scientific Data, 2024, 11(1): 436.

[15]Pashaian M, Seyedin S. Speech Enhancement Using Joint DNN - NMF Model Learned with Multi - Objective Frequency Differential Spectrum Loss Function[J]. IET Signal Processing, 2024, 2024(1): 8881007.

[16]Zhou H, Sarkar R. Leveraging Graph Machine Learning for Moonlighting Protein Prediction: A PPI Network and Physiochemical Feature Approach[J]. bioRxiv, 2023: 2023.11. 13.566879.

[17]Al Olaimat M, Bozdag S, Alzheimer's Disease Neuroimaging Initiative. TA-RNN: An attention-based time-aware recurrent neural network architecture for electronic health records[J]. Bioinformatics, 2024, 40: i169-i179.

[18]Bharadwaj V, Malik O A, Murray R, et al. Distributed-memory randomized algorithms for sparse tensor cp decomposition[C]//Proceedings of the 36th ACM Symposium on Parallelism in Algorithms and Architectures. 2024: 155-168.

[19]Yuan S, Huang K. Exploring Numerical Priors for Low-Rank Tensor Completion with Generalized CP Decomposition[J]. arXiv e-prints, 2023,arXiv: 2302.05881.

[20]Baghershahi P, Hosseini R, Moradi H. Efficient relation-aware neighborhood aggregation in graph neural networks via tensor decomposition[J]. arXiv e-prints,2022, arXiv:2212.05581.

[21]Xiang L, Yin M, Zhang C, et al. Tdc: Towards extremely efficient cnns on gpus via hardware-aware Tucker decomposition[C]//Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming. 2023: 260-273.

[22]Zhang Y, Zhu Y N, Zhang X. Compressing MIMO Channel Submatrices with Tucker Decomposition: Enabling Efficient Storage and Reducing SINR Computation Overhead[J]. arXiv e-prints, 2024,arXiv:2401.09792.

[23]Novikov A, Podoprikhin D, Osokin A, et al. Tensorizing neural networks[J]. Advances in neural information processing systems, 2015, 28.

[24]Kossaifi J, Lipton Z C, Kolbeinsson A, et al. Tensor regression networks[J]. Journal of Machine Learning Research, 2020, 21(123): 1-21.

[25]Liu Y, Chakraborty N, Qin Z S, et al. Integrative Bayesian tensor regression for imaging genetics applications[J]. Frontiers in Neuroscience, 2023, 17: 1212218.

[26]Liu Y, Liu J, Long Z, et al. Tensor regression[M]. Springer International Publishing, 2022.

[27]Zhou Y, Tan K, Shen X, et al. A protein structure prediction approach leveraging transformer and CNN integration[C]//2024 7th International Conference on Advanced Algorithms and Control Engineering (ICAACE). IEEE, 2024: 749-753.

[28]Dereich S, Jentzen A. Convergence rates for the Adam optimizer[J]. arXiv e-prints, 2024, arXiv:2407.21078.

