文章编号：2095-2163(2022)11-0127-07 中图分类号：TP391.41 文献标志码：A

基于稀疏化训练和聚类降低 IR-Drop 影响的方法

王子杰
（合肥工业大学 计算机与信息学院，合肥 230601）

摘 要： 忆阻器阵列（Memristor based Crossbar）在加速神经网络计算上有很好的效果。然而，忆阻器阵列会受到 IR-Drop 的影响，导致忆阻器阵列的计算精度下降。为此，提出一种方案来提高计算精度，该方案是基于对权值矩阵稀疏化以及对权值矩阵的行向量进行聚类实现的。该方案首先通过分析 IR-Drop 对忆阻器阵列的影响，根据忆阻器阵列和权值矩阵的映射关系，对权值矩阵进行稀疏化训练，将受到较大 IR-Drop 影响的权值置零。然后对权值矩阵的行向量进行聚类，找到近似全零行向量将其权值置零，在保证零权值不变的前提下重新训练权值矩阵，接着删除全零行向量和全零列向量降低矩阵规模。最后在 IR-Drop 影响下计算权值矩阵行向量的权值损失，根据损失大小降序排列行向量得到新的权值矩阵，并映射到忆阻器阵列上。实验表明，经过此方案处理后，忆阻器阵列受到的 IR-Drop 显著降低，有效地提高了计算精度并且降低了硬件规模。

关键词： 忆阻器阵列；神经网络；IR-Drop

A method to reduce the influence of IR-Drop based on sparse training and clustering

WANG Zijie
（School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230601, China）

【Abstract】 Memristor based Crossbar has a good effect on accelerating neural network calculation.However, the Memristor based Crossbar will be affected by IR - Drop, resulting in the decline of the calculation accuracy of the Memristor based Crossbar. Therefore, a scheme is proposed to improve the calculation accuracy. The scheme is based on sparse training of weight matrix and clustering the row vectors of the weight matrix. Firstly, by analyzing the influence of IR - Drop on Memristor based Crossbar, according to the mapping relationship between Memristor based Crossbar and weight matrix, the weight matrix is sparsely trained, and the weight affected largely by IR-Drop is set to zero. Then the row vectors of the weight matrix are clustered, the nearly all zero row vector is found, its weight is set to zero, the weight matrix is retrained on the premise of ensuring that the zero weight remains unchanged, and then all zero row vector and all zero column vector are deleted to reduce the scale of the matrix. Finally, under the influence of IR - Drop, the weight loss of the row vector of the weight matrix is calculated, the row vectors are arranged in descending order according to the loss, a new weight matrix is gotten, and it is mapped to the Memristor based Crossbar. Experiments show that after this scheme, the IR - Drop of Memristor based Crossbar is significantly reduced, which effectively improves the calculation accuracy and reduces the hardware scale.

【Key words】 Memristor based Crossbar; neural network; IR-Drop

0 引 言

目前，卷积神经网络已经广泛应用于深度学习中。研究表明，在卷积神经网络中乘法和累加运算占整个操作的 90%以上[1]。但随着神经网络不断加深，利用传统 CMOS 器件组成的神经网络由于存在着计算时间过长和能耗过大的问题[2]，规模已经很难增大。新型器件忆阻器为实现矩阵乘法提供了一种更高效的方式[3]，能够以 $O(1)$ 的时间复杂度实现矩阵乘法，且具有极低的能耗[4]。基于忆阻器[5-7]构建的神经网络[8]可以加速神经网络的计算，加快神经网络对于大规模数据的处理速度。

通过忆阻器阵列[9]加速神经网络计算，首先需要在软件上训练神经网络得到权值矩阵，然后通过施加电压改变忆阻器的阻值将权值矩阵映射为忆阻器电导矩阵。但由于 IR-Drop 的存在，在运算的过程中，实际施加在忆阻器上的电压和预期的电压不同，计算结果和理想中预期的结果存在偏差，造成计算精度低的情况出现。

在忆阻器阵列中距输入端越远的忆阻器受到 IR-Drop 的影响越大，因为随着导线长度增加、导线电阻也会变大，而距施加电压的输入端越近的忆阻器受到的影响越小，所以忆阻器阵列规模越大，受到的 IR-Drop 影响就越大。随着神经网络的规模增加，相应的忆阻器阵列规模增大，计算精度也会下降。例如当忆阻器阵列规模从 16×16 增大到 128×

作者简介：王子杰（1996-），男，硕士研究生，主要研究方向：基于忆阻器阵列的卷积神经网络。

收稿日期：2022-03-12

128 时,在 IR-Drop 的影响下,忆阻器阵列的计算精度降低了 35%[10]。为了解决这一问题,在硬件和软件方面上,研究人员都提出了一些有效的解决方案。

硬件方面,Huang 等人[11]提出补偿输出电流的方法来提高计算精度。该方法首先求出在理想情况下忆阻器阵列中每一列输出的电流大小,然后在忆阻器阵列中新增行,对新增行施加电压来补偿电流输出,使得每一列的输出电流达到理想的电流大小。但是这种方式会增加过多的硬件开销。为了减少硬件开销, Zhu 等人[12]提出了一种新的补偿电流的方法来提高计算精度。该方法在忆阻器阵列每一列的输出端新增读出放大器(sense amplifier),并对每一列的输出电流调整放大的倍数使得电流输出达到理想的电流大小。文献[11-12]中的方法都会额外增加硬件的开销,而软件的解决方式则不会额外增加硬件开销。

在软件方面,主要是采取缩减忆阻器阵列规模来降低 IR-Drop 的影响,忆阻器阵列规模越小,受到的 IR-Drop 影响就越小,计算精度就越高。Wang 等人[13]提出通过 PCA 分解矩阵的方式来提高计算精度。通过将一个大的忆阻器阵列分解成若干个小的忆阻器阵列的方式来减少忆阻器阵列规模[14]。但分解过后的忆阻器阵列计算精度较低,所以需要对神经网络进行重新训练来提高计算精度。而 Liu 等人[10]提出通过 SVD 分解矩阵的方式[15]来提高计算精度,这种方式不需要对神经网络进行重新训练,相对于 PCA 分解矩阵的方式计算精度较高。文献[10,14]中采用分解矩阵的方法得到的忆阻器阵列规模较大,仍然会受到较大的 IR-Drop 影响,忆阻器阵列的计算精度较低。

为了解决 IR-Drop 对计算精度的影响,本文提出基于对权值矩阵稀疏化以及对权值矩阵的行向量进行聚类的方案(Sparse Training Clustering,STC)来提高计算精度,为了描述方便,本文余下部分对所提方案简称 STC 方案。首先,对神经网络进行稀疏化训练,将神经网络的权值矩阵上行号、列号之和较大的权值置零且保证计算精度大于阈值 p。然后对矩阵的行向量进行聚类,将全零行向量和近似全零行向量聚集在一起,将其权值置零并且在保证零权值不变和神经网络的计算精度大于阈值 γ 的前提下重新训练神经网络得到新的权值矩阵,接着删除权值矩阵的全零行向量和全零列向量减少矩阵规模。最后在 IR-Drop 下计算行向量的权值损失,将行向量按照损失大小降序排列得到新的权值矩阵并且映射到忆阻器阵列上。STC 方案可以有效地降低忆阻器阵列的规模,同时仍然保持较高的计算精度。

1 忆阻器阵列

利用忆阻器阵列实现矩阵乘法, 首先需要将训练好的神经网络的权值矩阵 $\boldsymbol{W}_{n\times m}$ 映射到忆阻器阵列上,忆阻器阵列如图 1 所示。用忆阻器阵列上的忆阻器电导值 g_{ij} 来代表权值矩阵上的权值 w_{ij},对每一行施加电压 $V=\{V_1, V_2, \cdots, V_n\}$,第 j 列的输出电流的计算公式可表示为:

$$I_j=\sum_{i=1}^{n} g_{ij}\cdot V_i \tag{1}$$

其中,V_i 为第 i 行的输入,I_j 为第 j 列输出。因为施加电压在电阻上可以得到一个电流,并且在导线上输出端会输出电流之和,这个速度比起传统的计算要快,所以忆阻器阵列可以加速神经网络的计算速度。为了描述方便,本文余下部分对于忆阻器阵列上的电导值统称为忆阻器阵列的权值。

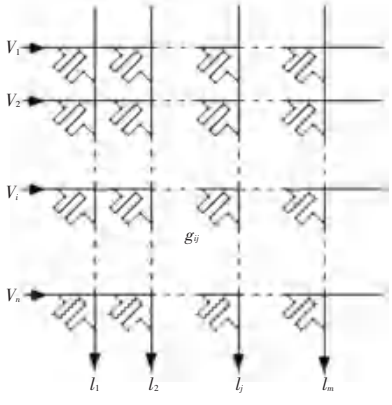

图 1 忆阻器阵列

Fig. 1 Memristor based Crossbar

2 STC 方案流程

STC 方案的流程图如图 2 所示。STC 方案首先需要输入模拟忆阻器阵列上 IR-Drop 影响的矩阵 $\boldsymbol{I}_{n\times m}$ 和神经网络的权值矩阵 $\boldsymbol{S}_{n\times m}$,$\boldsymbol{I}_{n\times m}\cdot\boldsymbol{S}_{n\times m}$ 代表 $\boldsymbol{I}_{n\times m}$ 和 $\boldsymbol{S}_{n\times m}$ 对应位置相乘得到的矩阵。用神经网络的权值矩阵 $\boldsymbol{I}_{n\times m}\cdot\boldsymbol{S}_{n\times m}$ 的计算精度来代表在 IR-Drop 下矩阵 $\boldsymbol{S}_{n\times m}$ 映射到忆阻器阵列上的计算精度。然后稀疏化训练将 $\boldsymbol{S}_{n\times m}$ 上行号、列号之和较大的权值置零,因为行号、列号之和较大的权值在忆阻器阵列上距离输入端较远, 受到的 IR-Drop 影响较大。紧接着得到在 IR-Drop 影响下计算精度高于阈值 p 的神经网络权值矩阵 $\boldsymbol{C}_{n\times m}$。接下来,对 $\boldsymbol{C}_{n\times m}$ 的行向量进行聚类,将全零行向量和近似全零行向量

聚集在一组，将近似全零行向量的权值置零、并且在保证零权值不变的前提下重新训练神经网络得到计算精度高于阈值 γ 的神经网络权值矩阵 $\boldsymbol{D}_{n\times m}$。同时，删除矩阵 $\boldsymbol{D}_{n\times m}$ 的全零行向量和全零列向量得到矩阵 $\boldsymbol{K}_{l\times h}$。最后，在 IR-Drop 影响下计算矩阵 $\boldsymbol{K}_{l\times h}$ 行向量的权值损失，将行向量按照损失大小降序排列得到新的权值矩阵 $\boldsymbol{F}_{l\times h}$，并将权值矩阵 $\boldsymbol{F}_{l\times h}$ 映射到忆阻器阵列上。对 STC 方案流程，本文拟展开研究分述如下。

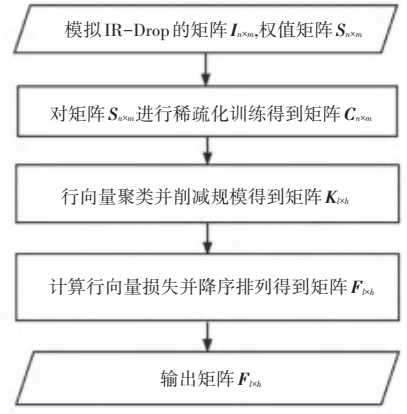

图 2　STC 方案流程图

Fig. 2　Flow chart of STC scheme

2.1　稀疏化训练

IR-Drop 影响大小是由忆阻器阻值以及导线电阻决定的[10]，随着忆阻器规模的增大，导线电阻增加，IR-Drop 影响也会增加。如果忆阻器的电阻设置为最大、即权值为零，此时导线电阻分压的影响就会降低，IR-Drop 影响就会降低，计算精度也随之提高。在 STC 方案中，经过稀疏化训练的权值矩阵有大量的零权值，映射在忆阻器阵列上受到的 IR-Drop 影响较小，计算精度显著提升。关于稀疏化的训练过程，其代码表述具体如下。

输入　权值矩阵 $\boldsymbol{S}_{n\times m}$，模拟忆阻器阵列上 IR-Drop 影响的矩阵 $\boldsymbol{I}_{n\times m}$，阈值 q，阈值 p

输出　稀疏化后的矩阵 $\boldsymbol{C}_{n\times m}$

1：$q = 40, p = 92\%$ //q 为稀疏化范围，计算精度高于 p 则停止训练

2：$\boldsymbol{C}_{n\times m} = \boldsymbol{S}_{n\times m}$;

3：While(神经网络的权值矩阵 $\boldsymbol{I}_{n\times m} \cdot \boldsymbol{C}_{n\times m}$ 的计算精度小于等于 p)

4：　$\boldsymbol{C}_{n\times m} = \boldsymbol{S}_{n\times m}$;

　　// 按照新稀疏化范围重新训练

5：　For ($i = 1 \rightarrow n$) //i 代表矩阵的行号

6：　　For ($j = 1 \rightarrow m$) //j 代表矩阵的列号

7：　　If ($i + j > q$) // 如果行列号之和大于 q，那么代表距输入端较远

8：　　$C[i][j] = 0$;// 置零

9：　　end

10：　end

11：　Retrain C // 重新训练

12：　q ++;// 缩小稀疏化范围

13：end

综上可知，输入为神经网络的权值矩阵 $\boldsymbol{S}_{n\times m}$，模拟忆阻器阵列上 IR-Drop 影响的矩阵 $\boldsymbol{I}_{n\times m}$，阈值 q 和 p。第 1 行给阈值 q 和 p 赋值，权值的行号、列号之和超过 q 代表在忆阻器阵列上此权值与输入端相距较远，受到的 IR-Drop 影响较大，p 为神经网络计算精度阈值。第 2 行将 $\boldsymbol{S}_{n\times m}$ 赋值给 $\boldsymbol{C}_{n\times m}$，第 3 行 $\boldsymbol{I}_{n\times m} \cdot \boldsymbol{C}_{n\times m}$ 代表 $\boldsymbol{I}_{n\times m}$ 和 $\boldsymbol{C}_{n\times m}$ 对应位置相乘得到的矩阵，神经网络的权值矩阵 $\boldsymbol{I}_{n\times m} \cdot \boldsymbol{C}_{n\times m}$ 的计算精度代表在 IR-Drop 下矩阵 $\boldsymbol{S}_{n\times m}$ 映射到忆阻器阵列上的计算精度，计算精度超过 p 则停止训练。第 4 行重新将 $\boldsymbol{S}_{n\times m}$ 赋值给 $\boldsymbol{C}_{n\times m}$ 用于稀疏化训练。第 5 ~ 10 行当矩阵 $\boldsymbol{C}_{n\times m}$ 上的权值行号、列号之和大于 q 时置零，第 11 行在保证零权值不变的情况下重新训练神经网络。第 12 行缩小稀疏化范围，因为神经网络的计算精度还未超过 p，通过改变稀疏化范围使得神经网络计算精度超过 p。第 14 行输出计算精度高于 p 的神经网络权值矩阵 $\boldsymbol{C}_{n\times m}$。

2.2　聚类并削减矩阵规模

经过了神经网络的稀疏化训练后，STC 方案对矩阵 $\boldsymbol{C}_{n\times m}$ 上的行向量进行聚类，找到全零行向量和近似全零行向量的集合。首先定义一个阈值 γ，在神经网络的计算精度高于 γ 的前提下，将权值矩阵的近似全零行向量的权值置零，然后保证零权值不变并重新训练神经网络，使得权值矩阵有更多的全零行向量，最后删除矩阵的全零行向量和全零列向量，降低权值矩阵的规模。对聚类并削减矩阵规模过程，可给出设计描述如下。

输入　稀疏化后的矩阵 $\boldsymbol{C}_{n\times m}$，忆阻器阵列 IR-Drop 分布信息矩阵 $\boldsymbol{I}_{n\times m}$，半径 h，阈值 γ

输出　经过删减的新矩阵 $\boldsymbol{K}_{l\times h}$

1：$h = 2, \gamma = \boldsymbol{I}_{n\times m} \cdot \boldsymbol{C}_{n\times m}$ 在神经网络上的计算精度 - 1% //h 越大近似全零行向量越多

2：$\boldsymbol{Row}_{n\times m} = \boldsymbol{C}_{n\times m}$; // 将 $\boldsymbol{C}_{n\times m}$ 赋值给 $\boldsymbol{Row}$，$\boldsymbol{Row}$ 用于聚类重训练

3：While(神经网络的权值矩阵 $\boldsymbol{I}_{n\times m} \cdot \boldsymbol{Row}_{n\times m}$ 的计算精度小于 γ)

4：　$\boldsymbol{Row}_{n\times m} = \boldsymbol{C}_{n\times m}$;

5: ***result*** = *mean - shift*(*h*,***Row***)// 带入 *h* 对 ***Row*** 进行 mean - shift 运算得到数组 *result* ,*result* 内为分类集合

6: Int *flag*; // 记录 *result* 第几组为全零行向量和近似全零行向量的集合

7: For (*i* = 1 → ***result*** 的分类数量)

8: For (*j* = 1 → ***result*** 的第 *i* 组行向量的数量)

9: If (***result***[*i*][*j*] 为全零行向量)
//找到全为零的行向量组

10: *flag* = *i*
// 找到全零行向量和近似全零行向量集合

11: break;

12: end

13: end

14: For (*i* = 1 → ***result*** 的第 *flag* 组行向量的数量)

15: ***Result***[*flag*][*i*] 的权值置零 //增加全零行向量数量

16: end

17: ***result*** → $\boldsymbol{Row}_{n\times m}$; // 将 ***result*** 赋值给 ***Row*** 进行重新训练

18: Retrain $\boldsymbol{Row}_{n\times m}$; // 重新训练神经网络得到新矩阵 ***Row***

19: *h* -= 0.1// 不同半径会导致训练出来权值矩阵计算精度不同,*h* 越小则计算精度越高,近似全零行向量越少

20:end

21: $\boldsymbol{D}_{n\times m}$ = $\boldsymbol{Row}_{n\times m}$ // 得到在 IR - Drop 下计算精度高于 γ 的矩阵 $\boldsymbol{D}_{n\times m}$

22:For (*i* = 1 → ***D*** 的行号)

23: If (***D***[*i*] 是全零行向量) delete ***D***[*i*];
// 删除全零行向量

24:end

25:For (*i* = 1 → ***D*** 的列号)

26: If (***D***[*i*] 是全零列向量) delete ***D***[*i*];
// 删除全零列向量

27:end

28:得到新矩阵 $\boldsymbol{K}_{l\times h}$;

29:输出 $\boldsymbol{K}_{l\times h}$

综上可知,输入为经过稀疏化的矩阵 $\boldsymbol{C}_{n\times m}$, 模拟忆阻器阵列上 IR-Drop 影响的矩阵 $\boldsymbol{I}_{n\times m}$,半径 *h*,阈值 γ。第 1 行给 *h* 和 γ 赋值,若行向量与全零行向量距离 *h* 以内的则为近似全零行向量,γ 为神经网络计算精度阈值。例如:

$$\boldsymbol{C}_{6\times 6}=\begin{bmatrix}0.1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0.9 & 0 & 0\\ 1 & 1 & 0 & 0 & 0.9 & 0\\ 0 & 0 & 1 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

第 2 行是将矩阵 $\boldsymbol{C}_{n\times m}$ 赋值给 $\boldsymbol{Row}_{n\times m}$。第 3 行的 $\boldsymbol{I}_{n\times m}\cdot\boldsymbol{Row}_{n\times m}$ 为 $\boldsymbol{I}_{n\times m}$ 和 $\boldsymbol{Row}_{n\times m}$ 对应位置相乘得到的矩阵,神经网络的权值矩阵 $\boldsymbol{I}_{n\times m}\cdot\boldsymbol{Row}_{n\times m}$ 的计算精度代表在 IR - Drop 下矩阵 $\boldsymbol{Row}_{n\times m}$ 映射到忆阻器阵列上的计算精度,当计算精度超过 γ 时停止训练。第 4 行重新将 $\boldsymbol{C}_{n\times m}$ 赋值给 $\boldsymbol{Row}_{n\times m}$, 第 5 行对 ***Row*** 采用 mean - shift 算法聚类并得到 ***result*** 分类集合。mean - shift 算法的聚类中心处于最高密度处,由于矩阵 $\boldsymbol{C}_{n\times m}$ 经过稀疏化训练,所以全零行向量这一类型必定最多,所以得到的聚类中心相对来说比较准确。mean - shift 算法中的 *h* 表示如果 2 个向量距离 *h* 以内则为同一类,将 *h* 设为 0.1 可得:

$$\boldsymbol{result}[1]=\begin{Bmatrix}0.1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\end{Bmatrix}$$

$$\boldsymbol{result}[2]=\{1\quad 1\quad 0\quad 0\quad 0.9\quad 0\}$$

$$\boldsymbol{result}[3]=\begin{Bmatrix}0 & 0 & 1 & 0.9 & 0 & 0\\ 0 & 0 & 1 & 1 & 0 & 0\end{Bmatrix}$$

第 6~16 行寻找 ***result*** 内拥有全零行向量和近似全零行向量的集合,并将集合编号赋值给 *flag*,再将 ***result*** 编号为 *flag* 的集合内所有行向量权值置零,***result***[*i*] 代表第 *i* 个集合,***result***[*i*][*j*] 代表第 *i* 个集合的第 *j* 个行向量。第 17 ~ 18 行将 ***result*** 赋值给 $\boldsymbol{Row}_{n\times m}$,同时保证零权值不变的前提下重新训练神经网络。第 19 行减少 *h*,因为神经网络计算精度还没超过 γ,*h* 越小、近似全零行向量越少,则进行重训练后的神经网络计算精度越高,不断改变 *h* 的大小,直到神经网络的计算精度超过 γ。第 21 行得到计算精度高于 γ 的神经网络权值矩阵 $\boldsymbol{Row}_{n\times m}$,并将其赋值给矩阵 $\boldsymbol{D}_{n\times m}$,推导后得到的运算结果如下:

$$\boldsymbol{D}_{6\times 6}=\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0.9 & 0 & 0\\ 1 & 1 & 0 & 0 & 0.9 & 0\\ 0 & 0 & 1 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

在 22~29 行删除矩阵 $\boldsymbol{D}_{n\times m}$ 的全零行向量和全零列向量得到新矩阵 $\boldsymbol{K}_{l\times h}$ 并输出，并可表示为：

$$\boldsymbol{K}_{3\times5}=\begin{bmatrix}0 & 0 & 1 & 0.9 & 0\\ 1 & 1 & 0 & 0 & 0.9\\ 0 & 0 & 1 & 1 & 0\end{bmatrix}$$

2.3　行向量排序

对于删减后的权值矩阵 $\boldsymbol{K}_{l\times h}$，STC 方案通过对其行向量采用排序的方式来提高计算精度。采用如下步骤：

(1)设定一个向量 $\boldsymbol{b}$，利用 $\boldsymbol{b}$ 来代表 IR-Drop 的影响下权值矩阵上行向量的每个权值损失比例。

(2)利用 $\boldsymbol{b}$ 计算权值矩阵行向量的权值损失大小，将行向量按照损失大小降序排列得到新的矩阵。

这样将容易受到 IR-Drop 影响的行向量放到在忆阻器阵列上受 IR-Drop 影响小的位置，而将不易受到 IR-Drop 影响的行向量放到在忆阻器阵列上受 IR-Drop 影响大的位置上，使得整体受到的 IR-Drop 影响降低。行向量排序过程可做完整描述如下。

输入　经过删减的矩阵 $\boldsymbol{K}_{l\times h}$，向量 $\boldsymbol{b}$

输出　经过排序的新矩阵 $\boldsymbol{F}_{l\times h}$

1：For ($i\rightarrow\boldsymbol{K}$ 的行号)

2：　$map[\boldsymbol{K}[i]\cdot\boldsymbol{b}]=\boldsymbol{K}[i]$ // 这里 $\boldsymbol{K}[i]\cdot\boldsymbol{b}$ 为该行向量的权值损失，$\boldsymbol{K}[i]$ 为行向量，建立匹配

3：$Sort(map.first)$ // 针对损失降序排序

4：$map.second\rightarrow\boldsymbol{F}_{l\times h}$ // 将排序好的行向量赋值给矩阵 $\boldsymbol{F}$

5：输出 $\boldsymbol{F}_{l\times h}$

分析可知，输入为经过删减的矩阵 $\boldsymbol{K}_{l\times h}$ 和向量 $\boldsymbol{b}$。例如：

$$\boldsymbol{K}_{3\times5}=\begin{bmatrix}0 & 0 & 1 & 0.9 & 0\\ 1 & 1 & 0 & 0 & 0.9\\ 0 & 0 & 1 & 1 & 0\end{bmatrix}$$

$$\boldsymbol{b}=(0.1\quad 0.2\quad 0.3\quad 0.4\quad 0.5)$$

第 1~3 行，用 $\boldsymbol{b}$ 计算矩阵 $\boldsymbol{K}_{l\times h}$ 的行向量的权值损失，并且和行向量建立匹配关系 map，对 map 的行向量的权值损失进行降序排序。$\boldsymbol{K}[i]$ 为矩阵 $\boldsymbol{K}_{l\times h}$ 的第 i 个行向量，$\boldsymbol{K}[i]\cdot\boldsymbol{b}$ 为第 i 个行向量和向量 $\boldsymbol{b}$ 的内积结果，***map.first*** 存放行向量的权值损失，***map.second*** 存放对应的行向量。为此，可以得到：

$$\boldsymbol{map}=\begin{bmatrix}0.75: & 1 & 1 & 0 & 0 & 0.9\\ 0.7: & 0 & 0 & 1 & 1 & 0\\ 0.66: & 0 & 0 & 1 & 0.9 & 0\end{bmatrix}$$

第 4~5 行，将 ***map*** 上的行向量赋值给矩阵 $\boldsymbol{F}_{l\times h}$ 并输出，求得的结果可写为：

$$\boldsymbol{F}_{3\times5}=\begin{bmatrix}1 & 1 & 0 & 0 & 0.9\\ 0 & 0 & 1 & 1 & 0\\ 0 & 0 & 1 & 0.9 & 0\end{bmatrix}$$

3　实验结果以及分析

3.1　实验参数设置

本实验在 Pytorch 上构建了一个 3 层卷积神经网络，采用 MINST 数据集对其进行仿真测试。卷积神经网络主要包括 2 个卷积层、2 个池化层和 1 个全连接层。输入图像的大小为 28×28。忆阻器的电阻范围为 10 kΩ~1 MΩ，忆阻器阵列中忆阻单元之间的导线电阻为 2.5 Ω，忆阻器阵列规模分别为 64×64、96×96 和 128×128，参数配置见表 1。在 IR-Drop 的影响下，施加到忆阻器两端的电压受到忆阻器处于忆阻器阵列的位置的影响，根据文献[10]中的忆阻器阵列的实际退化情况得到模拟结果，按照模拟结果来表示 IR-Drop 的影响。

表 1　实验参数设置

Tab. 1　Experimental parameters setting

参数	参数值
Maximum Crossbar Size	128×128
Minimum Crossbar Size	64×64
High Resistance State	1 MΩ
Low Resistance State	10 KΩ
R_{wire} for single cell	2.5 Ω

3.2　实验结果分析

本实验将 STC 方案和 SVD 方案在不同规模的忆阻器阵列上的计算精度和硬件规模开销上做了对比。STC 方案和 SVD 方案在计算精度的对比如图 3 所示，图 3 展示了不同规模的忆阻器阵列在 4 种情况下的计算精度，分别是权值矩阵在有 IR-Drop 影响下和无 IR-Drop 影响下的计算精度，采用 SVD 方案和 STC 方案的计算精度。在 IR-Drop 的影响下，随着忆阻器阵列规模的增加，计算精度会不断下降，STC 方案和 SVD 方案有效地提升了在 IR-Drop 影响下忆阻器阵列的计算精度，并且 STC 方案与 SVD 方案相比计算精度较高。这主要是因为采用 STC 方案得到的矩阵有大量零权值，零权值受到 IR-Drop 的影响很小，所以忆阻器整体受到的 IR-Drop 影响较小。而 SVD 方案得到的矩阵规模较大，所以仍然会受到较大的 IR-Drop 影响。可以看出 STC 方案在忆阻器规模较小的时候计算精度的提升较小，而对于较大规模的忆阻器阵列计算精度提升较大。

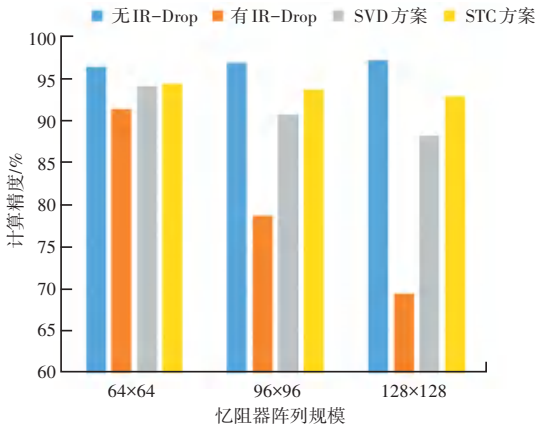

图 3 不同规模的忆阻器阵列的计算精度

Fig. 3 Calculation accuracy of Memristor based Crossbar of different sizes

STC 方案和 SVD 方案在硬件规模开销对比见表 2。表 2 展示了不同规模的忆阻器阵列,采用 STC 方案和 SVD 方案所使用硬件规模的比较。SVD 方案会把矩阵分解成 2 个小矩阵,STC 方案则直接将一个大矩阵缩减成小矩阵。STC 方案在硬件规模的缩减大小上比 SVD 方案要高。这主要是因为 STC 方案采用稀疏化训练的方式使得零权值大量分布于神经网络的权值矩阵上,因此权值矩阵上可以删除许多全零行向量和全零列向量来减少矩阵规模,而 SVD 方案为了提高计算精度,得到的矩阵规模较大。STC 方案对于较大规模忆阻器阵列的硬件规模削减效果更好,例如对于 128×128 大小的忆阻器阵列缩减了 77.29%,而对于 64×64 大小的忆阻器阵列只减少了 23.44%。

表 2 不同规模的忆阻器阵列的硬件规模

Tab. 2 Hardware scale of Memristor based Crossbar of different sizes

忆阻器阵列规模	缩减后忆阻器阵列的规模			硬件规模缩减百分比/%	
	SVD 方案		STC 方案	SVD 方案	STC 方案
	矩阵 1	矩阵 2			
64×64	64×25	25×64	56×56	21.88	23.44
96×96	96×30	30×96	61×61	37.50	59.63
128×128	128×37	37×128	61×61	42.19	77.29

在 STC 方案中,通过调整稀疏化范围来得到不同规模的忆阻器阵列的计算精度和硬件规模的开销,针对不同规模的忆阻器阵列的计算精度如图 4 所示。图 4 中,横坐标表示忆阻器阵列上的权值的行号、列号之和大于该值则需要置零。随着神经网络稀疏化范围的减小,忆阻器阵列的计算精度提高。因为在 40~60 的范围内,忆阻器阵列上的大权值受到的 IR-Drop 影响较小。随着权值矩阵稀疏化训练范围的减小,在神经网络训练的过程中可以进行调整的权值数量增加,神经网络的计算精度就会提高,权值矩阵映射到忆阻器阵列上的计算精度也会增加。因此在一定范围内减少神经网络稀疏化训练的范围可以提高忆阻器阵列的计算精度,但是如果神经网络稀疏化范围太小,忆阻器阵列就会受到较大的 IR-Drop 的影响,计算精度反而会下降。

针对不同规模的忆阻器阵列缩减后的硬件规模如图 5 所示。图 5 中,横坐标表示忆阻器阵列上的权值行号、列号之和大于该值则需要置零。随着神经网络稀疏化范围的减小,硬件规模的大小增加。因为随着稀疏化范围的减小,权值矩阵上全零行向量和全零列向量的数量就会变少,权值矩阵的规模增加,所以映射到忆阻器阵列上后的硬件开销就会增加。

从图 4 和图 5 可以看出,通过设置不同的稀疏化范围会导致不同的计算精度和硬件规模的开销,这里 STC 方案以最高计算精度为选取方案。

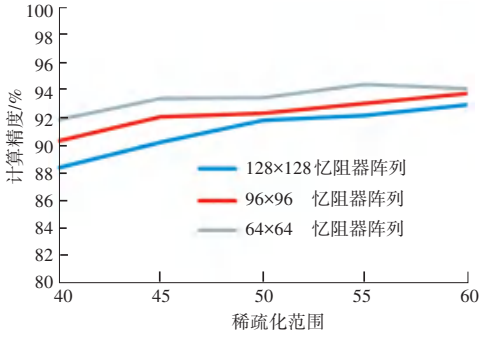

图 4 不同规模的忆阻器阵列计算精度

Fig. 4 Calculation accuracy of Memristor based Crossbar of different sizes

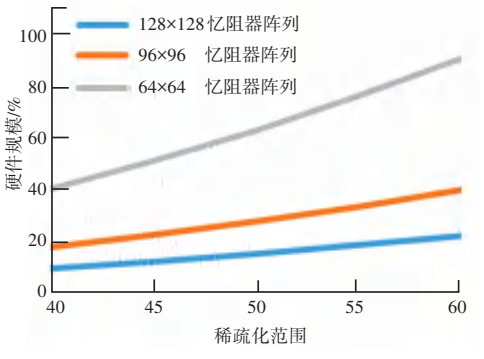

图 5 不同规模的忆阻器阵列硬件规模

Fig. 5 Hardware scale of Memristor based Crossbar with different sizes

4　结束语

忆阻器阵列在神经网络计算加速上有着很好的效果，但是会受到 IR-Drop 的影响，从而造成计算精度的下降。本文提出了 STC 方案用于降低 IR-Drop 的影响，提高计算精度。该方案基于对权值矩阵进行稀疏化以及对权值矩阵的行向量进行聚类来实现，可以有效提高计算精度，并且减少了硬件规模的开销。根据实验结果表明，经过 STC 方案处理的忆阻器阵列在 IR-Drop 的影响下，计算精度显著提高，硬件规模大大降低。

参考文献

[1] XIA Lixue, TANG Tianqi, HUANGFU Wenqin, et al. Switched by input: Power efficient structure for RRAM-based convolutional neural network [C]//2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC). Austin, TX, USA: IEEEE, 2016:1-6.

[2] ASENOV A, KAYA S, BROWN A R. Intrinsic parameter fluctuations in decananometer MOSFETs introduced by gate line edge roughness [J]. IEEE Transactions on Electron Devices, 2003, 50:1254-1260.

[3] SERAFINO N, ZAGHLOUL M. Review of nanoscale memristor devices as synapses in neuromorphic systems[C]//2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS). Columbus, OH, USA:IEEE, 2013: 602-603.

[4] PREZIOSO M, MERRIKH-BAYAT F, HOSKINS B D. Training and operation of an integrated neuromorphic network based on metal-oxide memristors[J]. Nature,2015, 521:61-64 .

[5] YU Shimeng , YI Wu , WONG H S P. Investigating the switching dynamics and multilevel capability of bipolar metal oxide resistive switching memory [J]. Applied Physics Letters, 2011, 98: 103514.

[6] STRUKOV D B, SNIDER G S, STEWART D R, et al. The missing memristor found[J]. Nature, 2008, 453(7191): 80-83.

[7] CHUA L O. Memristor-The missing circuit element[J]. IEEE Transactions on Circuit Theory,1971,18(5):507-519.

[8] SHARAD M, FAN Deliang, ROY K. Ultra low power associative computing with spin neurons and resistive crossbar memory[C]// 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC). Austin, TX, USA:IEEE, 2013, 1-6.

[9] HU Miao, LI Hai, WU Qing, et al. Hardware realization of BSB recall function using memristor crossbar arrays[C]// DAC Design Automation Conference 2012. San Francisco, CA, USA: IEEE, 2012:498-503.

[10] LIU Beiye, LI Hai, CHEN Yiran, et al. Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems [C]//International Conference on Computer - Aided Design. San Jose, CA, USA:IEEE, 2014:63-70.

[11] HUANG Chenglong, XU Nuo, QIU Keni, et al. Efficient and optimized methods for alleviating the impactsof IR-drop and fault in RRAM based neural computing systems[J]. IEEE Journal of the Electron Devices Society,2021,9: 645-652.

[12] ZHU Yujie, ZHAO Xue, QIU Keni. Insights and optimizations on IR - drop induced sneak - path for RRAM crossbar - based Convolutions[C]// Asia and South Pacific Design Automation Conference. Beijing, China:IEEE, 2020:506-511.

[13] WANG Yandan, WEN Wei, LIU Beiye, et al. Group scissor: Scaling neuromorphic computing design to arge neural networks [C]//2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC). Austin, TX, USA:IEEE,2017:1-6.

[14] WOLD S, ESBENSEN K, GELADI P. Principal component analysis[J]. Chemometrics and Intelligent Laboratory Systems, 1987, 2(1/3): 37-52.

[15] HALKO N, MARTINSSON P G, TROPP J A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions[J]. Siam Review, 2011, 53 (2):217-288.

(上接第 126 页)

[3] LANDA J, PROCHAZKA D. Automatic road inventory using LiDAR[J]. Procedia Economics and Finance, 2014, 12:363-370.

[4] ALAM A, SINGH L, JAFFERY Z A, et al. Distance-based confidence generation and aggregation of classifier for unstructured road detection - ScienceDirect [EB/OL]. [2021-09-30]. https://Doi.org/10.1016/J.JKSUCI.2021.09.020

[5] DAVID E H, BLUMENTHAL S, PRASSLER E, et al. Vision-based road boundary tracking system for unstructured roads[C]// 2017 IEEE International Conference on Unmanned Systems (ICUS). Beijing:IEEE, 2017:80-85.

[6] OBRADOVIC D, KONJOVIC Z, PAP E, et al. Linear fuzzy space based road lane model and detection[J]. Knowledge-Based Systems, 2013, 38:37-47.

[7] ARYA D, MAEDA H, GHOSH S K, et al. RDD2020: An annotated image dataset for Automatic Road Damage Detection using Deep Learning[J]. Data in Brief, 2021, 36(1):107133.

[8] LORE K G, AKINTAYO A, SARKAR S. LLNet: A deep autoencoder approach to natural low-light image enhancement[J]. Pattern Recognition, 2017, 61:650-662.

[9] PARK S, YU S, KIM M, et al. Dual autoencoder network for retinex-based low-light image enhancement[J]. IEEE Access, 2018,6:22084-22093.

[10] SOBBAHI R A, TEKLI J. Low-light homomorphic filtering network for integrating image enhancement and classification[J]. Signal Processing: Image Communication, 2022,100:116527.

[11] RONNEBERGER O, FISCHER P, BROX T. U-Net: Convolutional networks for biomedical image segmentation[C]// Proceedings of the Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Cham:Springer, 2015, 9351:234-241.

[12] ZHANG Yuxiao, CHEN Haiqiang, HE Yiran, et al. Road segmentation for all-day outdoor robot navigation [J]. Neurocomputing, 2018, 314: 316-325.

[13] WU Liying, YU QIANG, XU TONGQIANG, et al. An unstructured road detection method with multi-environmental adaptability[J]. International Journal of Simulation Systems, Science &Technology, 2016, 17(12): 16.1-16.6.