文章编号：2095-2163(2023)03-0051-08 中图分类号：TP389.1 文献标志码：A

一种基于权值缩减克服 IR-Drop 的忆阻器阵列神经网络训练方法

缪伟伟
（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)能够有效地加速神经网络中的矩阵运算。然而，忆阻器阵列会受到 IR-Drop 的影响，降低到达忆阻器的计算电压，导致计算精度下降。为减轻 IR-Drop 对忆阻器阵列计算精度的影响，提出了一种基于权值缩减的神经网络训练方法。首先，在网络训练中添加 *L2* 正则化，使训练后的神经网络权值尽可能分布在较小值范围，以此提高计算精度对 IR-Drop 的鲁棒性。然后，利用基于行列约束的映射算法将大权值映射到受 IR-Drop 影响小的忆阻器上，减小忆阻器阵列精度损失。最后，迭代减小受到 IR-Drop 影响大的大权值，再通过重训练调整被减小值的附近权值，提升忆阻器阵列的计算精度。实验结果表明，所提方法能够有效地提高忆阻器阵列的计算精度，最多可以将忆阻器阵列计算精度提升至接近理想状态，精度损失小于 1%。

关键词：忆阻器阵列；神经网络训练；IR-Drop；映射算法

A network training method of memristor-based crossbar by weight reduction to overcome IR-Drop

MIAO Weiwei

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

【Abstract】 Memristor-based crossbar can effectively accelerate matrix-vector multiplication in neural network. However, the accuracy of crossbar may seriously decrease due to IR-Drop. To reduce the impact of IR-Drop on the memristor-based crossbar, this paper proposes a network training method of memristor-based crossbar by weight reduction. Firstly, *L2* regularization is added to the network training to shrink the weight distribution to a small value range, which increase the resistance of memristor in crossbar to reduce the impact of IR-Drop. Then, a mapping algorithm with the constraint of row and column is used to map the weight to the memristor less affected by IR-Drop, which avoids that large weights mapped the memristors with high IR-Drop, increasing the accuracy loss of memristor-based crossbar caused by IR-Drop. Finally, the weights mapped to the memristor with high IR-Drop are reduced iteratively, and the accuracy of the memristor-based crossbar can be restored through retraining. The experimental results show that the proposed technique can pull the classification accuracy up close to ideal level with the loss of less than 1%.

【Key words】 memristor-based crossbar; neural network training; IR-Drop; mapping algorithm

0 引 言

深度神经网络(DNN)中存在大量的矩阵乘法运算。然而随着神经网络层数的不断增加，利用传统处理器实现矩阵乘法会造成计算时间过长和能耗过大。新型器件忆阻器(memristor)为实现矩阵乘法提供了一种更高效的方式[1]，能够以 $O(1)$ 的时间复杂度实现矩阵乘法。并且与传统的 CMOS ASIC 和 GPU 解决方案相比，忆阻器阵列可以将能效提高 100 倍以上[2-3]。忆阻器阵列实现矩阵乘法的结构如图 1 所示。

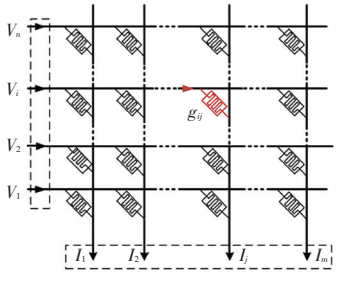


图 1 利用忆阻器阵列实现矩阵乘法

Fig. 1 The implementation of matrix multiplication using memristor-based crossbar

图 1 中，忆阻器的电导值 g_{ij} 表示神经网络的权值，为忆阻器电阻值的倒数。对忆阻器阵列的第 i

基金项目：国家自然科学基金(U1613217, 62174048)。

作者简介：缪伟伟(1997-)，男，硕士研究生，主要研究方向：人工神经网络计算机。

收稿日期：2022-04-28

行施加一个电压矢量 V_i,流经第 i 行第 j 列忆阻器的电流为 $g_{ij} \cdot V_i$,第 j 列的输出电流 $I_j = \sum_{i=1}^{n} g_{ij} \cdot V_i$,即为输入向量与权值矩阵第 j 列的乘积结果。

尽管忆阻器具有很好的应用前景,但是由于 IR-Drop 问题,会导致忆阻器阵列计算精度下降。IR-Drop 会造成输入端电压与实际到达忆阻器的计算电压之间存在偏差,导致忆阻器的实际输出偏移理想输出,忆阻器阵列计算精度降低。在本文中,将输入端电压称为理想计算电压,实际到达忆阻器的计算电压称为实际计算电压。忆阻器离输入端和输出端越远,IR-Drop 造成的理想计算电压和实际计算电压的偏差越大,忆阻器的理想输出电流和实际输出电流的偏差也越大[4-5]。并且随着忆阻器阵列规模的增大,IR-Drop 对忆阻器阵列计算精度的影响也越明显。例如,当忆阻器阵列规模从 16×16 增大到 128×128 时,计算精度降低了 35%[5]。

为减轻忆阻器阵列中 IR-Drop 的影响,文献[5]、文献[6]分别提出主成分分析(Principal Component Analysis, PCA)和奇异值分解(Singular Value Decomposition, SVD)的方法将大矩阵分解为 2 个小矩阵的乘积。通过减小忆阻器阵列规模,降低IR-Drop 对忆阻器阵列计算精度的影响。文献[7]、文献[8]分别在忆阻器阵列每列的输出端添加对应的平均电流偏移量以及调整每列跨阻放大器(TIA)的阻值,以此直接减小列输出的偏差。文献[4]、文献[9]在网络训练中加入忆阻器阵列的 IR-Drop 模型,使训练出的权值对 IR-Drop 具有更好的鲁棒性。

IR-Drop 会降低忆阻器的实际计算电压,进而影响忆阻器的输出结果。但忆阻器的输出结果等于忆阻器的实际计算电压与权值的乘积。忆阻器的权值越小,IR-Drop 造成的输出结果偏差也越小。假设理想计算电压为 1 V,忆阻器的实际计算电压为 0.8 V。当忆阻器权值为 5 时,忆阻器的输出结果偏差则为 $|1-0.8|\times5=1$。而当忆阻器权值为 1 时,忆阻器的理想输出结果与实际输出结果的偏差为 $|1-0.8|\times1=0.2$。同时映射到忆阻器的权值越小,忆阻器阻值越大,IR-Drop 对忆阻器实际计算电压的影响也就越小,造成忆阻器输出结果偏差也越小[8]。因此,小权值会使忆阻器输出结果对 IR-Drop 有更好的鲁棒性。

为减小 IR-Drop 对忆阻器阵列计算精度的影响,本文提出了一种基于权值缩减的神经网络训练方法(A Network Training Weight Reduction),为叙述方便在后续部分中简称为 NTWR。首先,在网络训练中添加 $L2$ 正则化,以此使训练出的权值尽可能小,从而提高忆阻器阵列计算精度对 IR-Drop 的鲁棒性。然后,本文通过基于行列约束的映射算法将大权值映射到离输入端和输出端较近的位置,避免大权值映射到 IR-Drop 影响较大的忆阻器上,产生较大的输出结果偏差。在确定权值与忆阻器的映射关系后,可能仍存在部分大权值映射到离输入端和输出端较远处的忆阻单元上,导致忆阻器阵列计算精度降低。最后,减小映射到离输入端和输出端较远处的大权值,再利用重训练调整附近权值以恢复由于减小权值带来的计算精度损失。不断迭代减小权值和重训练,直到忆阻器阵列的计算精度无法提升为止。

1 基于权值缩减的神经网络训练方法

1.1 NTWR 方法的整体流程

NTWR 方法的整体流程如图 2 所示。由图 2 可看到,第一步是在神经网络训练中添加 $L2$ 正则化,使训练出的权值分布在较小值的范围,从而减小 IR-Drop对忆阻器阵列计算精度的影响。第二步是在得到训练好的网络权值后,执行行列映射算法将大权值映射到离输入端和输出端较近的位置,避免大权值映射到 IR-Drop 影响较大的位置,造成较大的计算精度损失。第三步是在执行映射算法后,减小映射在 IR-Drop 影响较大处的权值,以此降低 IR-Drop 造成的输出结果偏差。再执行重训练恢复由于权值减小造成的计算精度损失,直到忆阻器阵列计算精度无法提升为止。

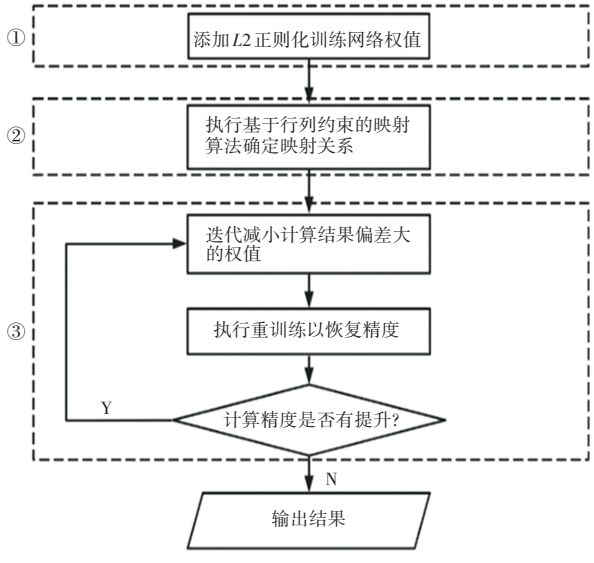


图 2 NTWR 的总体流程

Fig. 2 The overall process of NTWR

1.2　*L2* 正则化

在本节中，将详细探讨 *L2* 正则化在神经网络权值训练中的作用。研究时在训练中增加 *L2* 正则化，是因为 *L2* 正则化在训练过程中对大权重具有更大的偏向性，可以将权重分布缩小到较小的值范围[10]，进而提高忆阻器阵列的计算精度对 IR-Drop 的鲁棒性。因此本文提出在损失函数中增加 *L2* 正则化的惩罚项，利用 *L2* 正则化使训练出的权值尽可能小，从而降低 IR-Drop 的影响。这里需用到的数学公式为：

$$\hat{L}(\boldsymbol{W}) = L(\boldsymbol{W}) + \lambda \cdot \|\boldsymbol{X}\|_2 \tag{1}$$

其中，$\boldsymbol{W}$ 是神经网络的权值矩阵；λ 是正则化参数，用来控制 *L2* 正则化对损失函数 $\hat{L}(\boldsymbol{W})$ 的重要性；$\|\boldsymbol{X}\|_2$ 是 *L2* 正则化的惩罚项。

目前，比较流行的为 *L1* 正则化方法和 *L2* 正则化方法，可分别由如下公式进行描述：

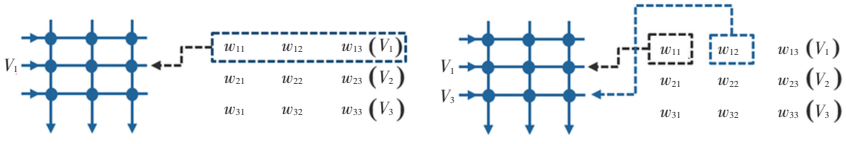
$$\|\boldsymbol{X}\|_1 = \sum_{i=1}^{n} |x_i| \tag{2}$$

$$\|\boldsymbol{X}\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2} \tag{3}$$

如果在网络训练中添加 *L1* 正则化，则会使网络权值为零，导致训练出的神经网络稀疏化。尽管稀疏化使大部分权值为零，不会对神经网络的计算精度造成较大的精度损失[11]，但会使神经网络的计算精度对训练出的非零权值更加敏感。在 IR-Drop 影响下，会造成非零权值的输出结果出现偏差，对忆阻器阵列计算精度产生更大的影响。而由文献[12]分析可知，*L2* 正则化可以降低神经网络的敏感性，从而提高神经网络的鲁棒性。执行神经网络训练后，可以得到 *L2* 正则化后的权值矩阵。下面将对权值矩阵的映射算法进行研究阐述。

1.3　基于行列约束的映射算法

在本节中，将详细介绍基于行列约束的映射算法（Mapping algorithm with the Constraint of Row and Column，MCRC）的具体步骤。MCRC 算法的主要思想是将权值矩阵中未确定映射关系的最大值在行列约束下映射到离输入端和输出端最近的忆阻器上，以此最小化 IR-Drop 对忆阻器阵列计算精度的影响。其中，行列约束指的是权值矩阵同一行和同一列的权值在映射到忆阻器阵列后仍在同一行和同一列。之所以需要令确定的映射关系满足行约束，是因为施加在忆阻器阵列一行的理想计算电压是同一个，而不同行的理想计算电压是不同的。例如，若将权值矩阵整个第 1 行的权值映射到忆阻器阵列第 2 行，则只需要在忆阻器阵列第 2 行施加权值矩阵第 1 行的理想计算电压 V_1，权值矩阵第 1 行第 j 列的输出结果仍为 $V_1 \times w_{1j}$，如图 3(a)所示。但是若权值矩阵第 1 行的 w_{11} 和 w_{12} 分别映射到忆阻器阵列第 2 行和第 3 行，则权值矩阵第 1 行第 2 列的输出结果不再是 $V_1 \times w_{12}$，而是 $V_3 \times w_{12}$，如图 3(b)所示。同理，每一列的输出结果等于该列所有忆阻器的输出结果之和，如果权值矩阵一列的权值被映射到忆阻器阵列的不同列上，同样会导致输出结果出现误差。故权值矩阵与忆阻器阵列的映射关系也需要满足列约束。



（a）满足行约束的映射　　（b）不满足行约束的映射

图 3　基于行约束映射的举例

Fig. 3　The example of mapping with the constraint of row

忆阻器阵列实现多层神经网络时，每层忆阻器阵列的输出都会连接下一层忆阻器阵列的输入，如图 4 所示。如果调整第 n 层权值矩阵与忆阻器阵列的映射关系，例如将权值矩阵第 i 行映射到忆阻器阵列第 j 行，则原连接到第 n 层第 i 行的第 $n-1$ 层的输出也需要重新连接到第 n 层的第 j 行。同理，如果要将第 n 层权值矩阵第 i 列映射到忆阻器阵列第 j 列，则原连接第 n 层第 i 列和第 j 列的第 $n+1$ 层的输入也需要交换连接。因此，如果想要独立映射每层权值矩阵的 M 行或 M 列，而不改变与相邻层的连接，则需要使用 $M \times M$ 的路由模块来连接相邻层的忆阻器阵列，会带来较大的硬件开销[13]。而 MCRC 算法是一种对忆阻器阵列通用的映射算法，无需得知忆阻器阵列的相关信息。因此可以在确定每层权值矩阵与忆阻器阵列的映射关系后再制造忆阻器阵列，在多层神经网络中无需考虑映射带来的硬件开销，只需确定的映射关系满足行列约束。故本文提出的 MCRC 算法可以同时执行行映射和列映射。

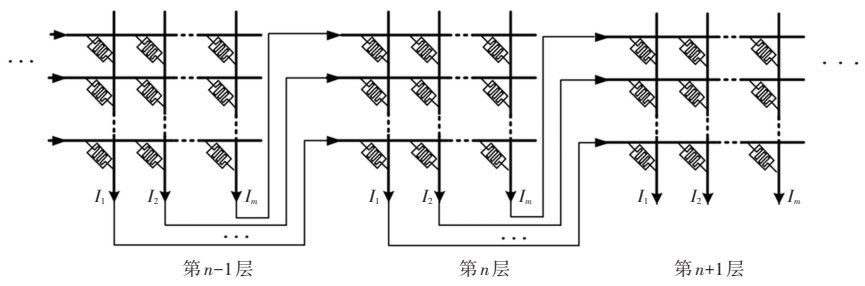


图 4 忆阻器阵列实现多层神经网络

Fig. 4 The implementation of multi-layer neural network using memristor-based crossbar

MCRC 算法的伪代码具体见算法 1。

算法 1 MCRC 算法

输入 权值矩阵 $\boldsymbol{W}_{n\times m}$，距离矩阵 $\boldsymbol{D}_{n\times m}$

输出 输出映射结果

1. $\boldsymbol{T}_{n\times m}$ = | $\boldsymbol{W}_{n\times m}$ |，*mapped_row*[*n*] 和 *mapped_col*[*m*] 全部置为 −1，***occupied***[*n*][*m*] 全部置为 false；

2. While（存在未确定映射关系的权值）

3. 在当前 $\boldsymbol{T}_{n\times m}$ 中找到最大值，将其行序号和列序号分别赋给 *row*，*col*

4. ***visited***[*n*][*m*] = ***occupied***[*n*][*m*]

//将 ***occupied*** 数组中的信息复制到 ***visited*** 数组中

5. *While* (*true*)

6. 在当前 $\boldsymbol{D}_{n\times m}$ 找到 ***visited***[*i*][*j*] = *false* 的最小值，将其行序号和列序号分别赋给 *i*，*j*

7. *If* ((*mapped_row*[*i*] = −1 *or mapped_row*[*i*] = *row*) *and* (*mapped_col*[*i*] = −1 *or mapped_col*[*j*] = *col*))

//判断是否满足行列约束

8. *mapped_row*[*i*] = *row*, *mapped_col*[*j*] = *col*;

// 将确定的映射关系存储到数组中

9. ***occupied***[*i*][*j*] = *true*, ***T***[*row*][*col*] = −1;

// 更新状态

10. *break*; //跳出当前 *while* 循环

11. *End*

12. ***visited***[*i*][*j*] = *true*;

//标记当前位置被访问

13. *End*

14. *End*

15. 输出映射结果

在算法中，输入为权值矩阵 $\boldsymbol{W}_{n\times m}$ 和距离矩阵 $\boldsymbol{D}_{n\times m}$。距离矩阵第 *i* 行第 *j* 列的元素 D_{ij} 表示忆阻器阵列第 *i* 行第 *j* 列的忆阻器离输入端和输出端的距离。以离输入端和输出端最近的忆阻器为原点，字线为横坐标，位线为纵坐标，建立坐标轴，如图 5 所示。以横坐标和纵坐标之和表示忆阻器离输入端和输出端的距离，如 $D_{ij} = i + j$。

算法 1 中，伪代码的第 1 行，对矩阵 $\boldsymbol{T}_{n\times m}$、数组 *mapped_row*、*mapped_col* 和 ***occupied*** 进行初始化。$\boldsymbol{T}_{n\times m}$ 为 $\boldsymbol{W}_{n\times m}$ 的绝对值矩阵，其中 $T_{ij} = |W_{ij}|$。数组 *mapped_row* 和 *mapped_col* 分别用于存储确定的行映射关系和列映射关系。数组 ***occupied*** 用于标记忆阻器阵列中忆阻器是否已确定映射关系，***occupied***[*i*][*j*] = *true* 表示忆阻器阵列第 *i* 行第 *j* 列的忆阻器已确定映射关系。

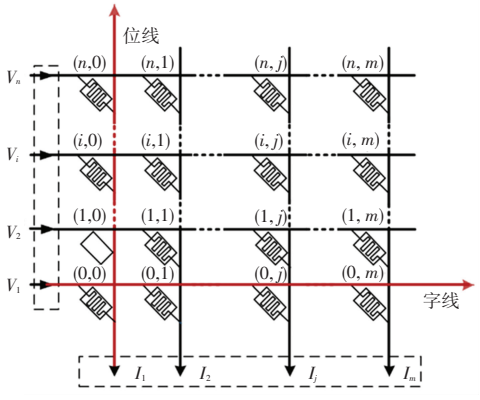


图 5 在忆阻器阵列上建立坐标轴

Fig. 5 The coordinate system based on memristor-based crossbar

下面通过具体的例子来阐释基于 MCRC 算法的具体执行过程。假设矩阵 $\boldsymbol{T}_{n\times m}$、距离矩阵 $\boldsymbol{D}_{n\times m}$ 和数组 ***occupied*** 分别为 $\boldsymbol{T}^{(0)}_{3\times3}$、$\boldsymbol{D}_{3\times3}$ 和 $\boldsymbol{occupied}^{(0)}_{3\times3}$。研究推得的各矩阵值具体如下：

$$\boldsymbol{T}^{(0)}_{3\times3} = \begin{bmatrix} 0.4 & 0.9 & 1 \\ 0.3 & 0.1 & 0.2 \\ 0.5 & 0.6 & 0.7 \end{bmatrix} \quad \boldsymbol{D}_{3\times3} = \begin{bmatrix} 2 & 3 & 4 \\ 1 & 2 & 3 \\ 0 & 1 & 2 \end{bmatrix}$$

$$\boldsymbol{occupied}^{(0)}_{3\times3} = \begin{bmatrix} false & false & false \\ false & false & false \\ false & false & false \end{bmatrix}$$

算法 1 伪代码第 3~4 行，按照 $\boldsymbol{T}_{n\times m}$ 中值大小降序确定权值的映射关系，令 *row* 和 *col* 分别为 $\boldsymbol{T}_{n\times m}$ 中最大值的行序号和列序号。利用数组 ***visited*** 标记

忆阻器是否被访问过或已确定重映射关系。例如在 $\boldsymbol{T}_{3\times3}^{(0)}$ 中，最大值为 $T_{22}=1$，$row=2$，$col=2$。

算法 1 伪代码第 6 行，按照离输入端和输出端的距离升序选择能在行列约束下与权值确定映射关系的忆阻器，令 i 和 j 分别为 $\boldsymbol{D}_{n\times m}$ 中 $\boldsymbol{visited}[i][j]=false$ 的最小值的行序号和列序号。例如在当前 $\boldsymbol{D}_{3\times3}$ 中 $\boldsymbol{visited}[i][j]=false$ 的最小值为 D_{00}，i 和 j 均为 0。

算法 1 伪代码第 7~11 行，判断权值矩阵第 row 行第 col 列权值映射到忆阻器阵列第 i 行第 j 列忆阻器是否满足行列约束。如果满足行列约束，则保存确定的映射关系，更新相应状态。例如在当前 $\boldsymbol{T}_{3\times3}^{(0)}$ 中，$mapped\_row[0]$ 和 $mapped\_col[0]$ 均为 -1，即忆阻器阵列第 0 行和第 0 列都未确定映射关系。因此，权值矩阵第 2 行第 2 列的权值与忆阻器阵列第 0 行第 0 列确定映射关系。$mapped\_row[0]=2$，$mapped\_col[0]=2$，$\boldsymbol{T}_{3\times3}^{(0)}$ 和 $\boldsymbol{occupied}_{3\times3}^{(0)}$ 分别更新为 $\boldsymbol{T}_{3\times3}^{(1)}$ 和 $\boldsymbol{occupied}_{3\times3}^{(1)}$，跳出当前 while 循环。研究推得的各矩阵值具体如下：

$$\boldsymbol{T}_{3\times3}^{(1)}=\begin{bmatrix}0.4 & 0.9 & -1\\ 0.3 & 0.1 & 0.2\\ 0.5 & 0.6 & 0.7\end{bmatrix}$$

$$\boldsymbol{occupied}_{3\times3}^{(1)}=\begin{bmatrix}false & false & false\\ false & false & false\\ true & false & false\end{bmatrix}$$

算法 1 伪代码第 2~14 行，若还存在未确定映射关系的权值，则继续执行算法。如 $\boldsymbol{T}_{3\times3}^{(1)}$ 中仍存在未确定映射关系的权值，因此继续执行算法。在 $\boldsymbol{T}_{3\times3}^{(1)}$ 中，最大值为 $T_{21}=0.9$，$row=2$，$col=1$。将 $\boldsymbol{occupied}_{3\times3}^{(1)}$ 的信息复制到数组 $\boldsymbol{visited}$ 中。在当前 $\boldsymbol{D}_{3\times3}$ 中 $\boldsymbol{visited}[i][j]=false$ 的最小值为 D_{10}，$i=1$，$j=0$。因 $mapped\_col[0]=2$ 且 $mapped\_col[0]\neq1$，即忆阻器阵列第 0 列已经确定映射关系，且并不是权值矩阵第 1 列确定的映射关系，因此不满足行列约束。将 $\boldsymbol{visited}[1][0]$ 置为 $true$，继续寻找当前 $D_{3\times3}$ 中 $\boldsymbol{visited}[i][j]=false$ 的最小值。此时满足条件的最小值为 D_{01}，$i=0$，$j=1$。$mapped\_row[0]\neq1$ 但 $mapped\_row[0]=2$，并且 $mapped\_col[1]=-1$，满足行列约束。因此，权值矩阵第 2 行第 1 列的权值与忆阻器阵列第 0 行第 1 列确定映射关系。$mapped\_row[0]=2$，$mapped\_col[1]=1$，$\boldsymbol{T}_{3\times3}^{(1)}$ 和 $\boldsymbol{occupied}_{3\times3}^{(1)}$ 分别更新为 $\boldsymbol{T}_{3\times3}^{(2)}$ 和 $\boldsymbol{occupied}_{3\times3}^{(2)}$，跳出当前 $while$ 循环。由于在 $\boldsymbol{T}_{3\times3}^{(2)}$ 中仍有未确定映射关系的权值，因此继续根据上述步骤执行算法，直到所有权值确定映射关系。研究推得的各矩阵值具体如下：

$$\boldsymbol{T}_{3\times3}^{(2)}=\begin{bmatrix}0.4 & -1 & -1\\ 0.3 & 0.1 & 0.2\\ 0.5 & 0.6 & 0.7\end{bmatrix}$$

$$\boldsymbol{occupied}_{3\times3}^{(2)}=\begin{bmatrix}false & false & false\\ false & false & false\\ true & true & false\end{bmatrix}$$

算法 1 伪代码第 15 行，当 $\boldsymbol{T}_{n\times m}$ 中所有值确定映射关系后，输出映射结果。如对 $\boldsymbol{T}_{3\times3}^{(2)}$ 继续执行算法，可以得到映射后的 $\boldsymbol{T}_{3\times3}$ 为 $\boldsymbol{T}_{3\times3}^{mapped}$。推得的矩阵值具体如下：

$$\boldsymbol{T}_{3\times3}^{mapped}=\begin{bmatrix}0.2 & 0.1 & 0.3\\ 0.7 & 0.6 & 0.5\\ 1 & 0.9 & 0.4\end{bmatrix}$$

1.4　重训练算法

尽管利用 MCRC 算法可以尽可能避免大权值映射到离输入端和输出端较远的忆阻器上，但大权值若聚集于一行或一列，则无法避免地会有部分较大权值被映射到离输入端和输出端较远的忆阻器上，导致忆阻器阵列计算精度的下降。因此，本文提出一种重训练算法，通过减小映射到离输入端和输出端较远处的权值，降低 IR-Drop 对输出结果的影响。再通过重训练，恢复权值减小造成的计算精度损失。重训练算法的伪代码具体见算法 2。

算法 2　重训练算法

输入　映射后的权值矩阵 $\boldsymbol{W}_{n\times m}^{mapped}$，距离矩阵 $\boldsymbol{D}_{n\times m}$，偏差矩阵，IR - Drop 影响矩阵 $\boldsymbol{S}_{n\times m}$

输出　输出新的权值矩阵 $\boldsymbol{W}_{n\times m}^{'}$

1.While（忆阻器阵列计算精度仍可提升）

2.　初始化 $\boldsymbol{Modified}[n][m]$，全部置为 $false$

　　// 用于标记修改过的权值

3.　在当前 $\boldsymbol{S}_{n\times m}$ 中找到最大值，将对应的权值赋给 W_{ij}

4.　$W_{ij}=W_{ij}/2$　// 减小权值

5.　$\boldsymbol{Modified}[i][j]=true$

　　// 将权值进行标记，重训练中不更新该权值

6.　执行重训练，得到新的权值矩阵 $\boldsymbol{W}_{n\times m}^{'}$

7.　测试重训练后忆阻器阵列的计算精度

8.　重新计算 $\boldsymbol{S}_{n\times m}$

9. End

10.输出矩阵 $\boldsymbol{W}_{n\times m}^{'}$

在算法中，输入为映射后的权值矩阵 $\boldsymbol{W}_{n\times m}^{mapped}$、距离矩阵 $\boldsymbol{D}_{n\times m}$ 以及 IR-Drop 影响矩阵 $\boldsymbol{S}_{n\times m}$。$\boldsymbol{W}_{n\times m}^{mapped}$

第 i 行第 j 列元素 $\boldsymbol{W}_{ij}^{mapped}$ 表示执行 MCRC 算法后映射到忆阻器阵列第 i 行第 j 列忆阻器的权值。$\boldsymbol{S}_{n\times m}$ 第 i 行第 j 列元素 $\boldsymbol{S}_{ij}$ 表示忆阻器阵列第 i 行第 j 列忆阻器受到的 IR-Drop 影响大小,等于权值绝对值乘以该忆阻器离输入端和输出端的距离,即 $S_{ij} = | W_{ij}^{mapped} | \times D_{ij}$。

算法 2 伪代码第 2~4 行是迭代将 IR-Drop 影响最大的权值减小为原来的 1/2,再通过标记该权值,使其在后续重训练中不更新。由于减小权值会造成计算精度的降低,因此通过重训练来恢复计算精度,参见伪代码第 6 行。在重训练时,为避免权值出现较大变化,造成映射到离输入端和输出端较远处的小权值突然更新为大权值的情况,可以以较小的学习率训练神经网络。较小的学习率可以使权值以小细粒度进行调整,避免权值出现较大的变化[14]。伪代码第 7 行测试重训练后忆阻器阵列的计算精度,更新矩阵 $\boldsymbol{S}_{n\times m}$。如果相较于重训练前,忆阻器阵列计算精度有所提升,则继续执行算法。如果忆阻器阵列的计算精度无法继续提升,则结束算法,输出映射后的权值矩阵。

2 实验结果和分析

2.1 实验设置

为了验证 NTWR 算法的有效性,在 2 个数据集上进行了实验。首先,在 Pytorch 上构建了一个 4 层卷积神经网络,采用 MINST 数据集对其进行测试,输入图像的大小为 28×28,由 3 个卷积层和 1 个全连接层组成。然后,在 Pytorch 上构建了 LeNet,采用 CIFAR-10 数据集进行测试,即由 2 个卷积层、3 个全连接层组成。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,参数配置见表 1。

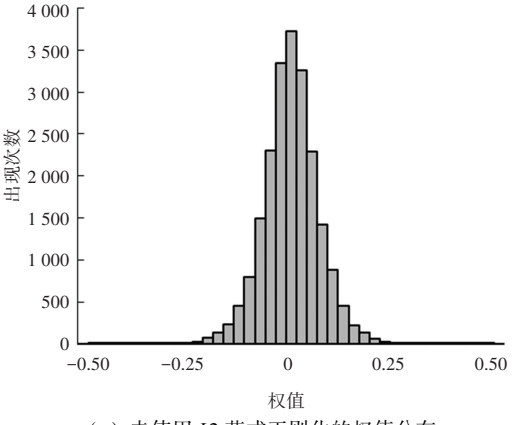
表 1 实验参数设置

Tab. 1 Experimental parameters setup

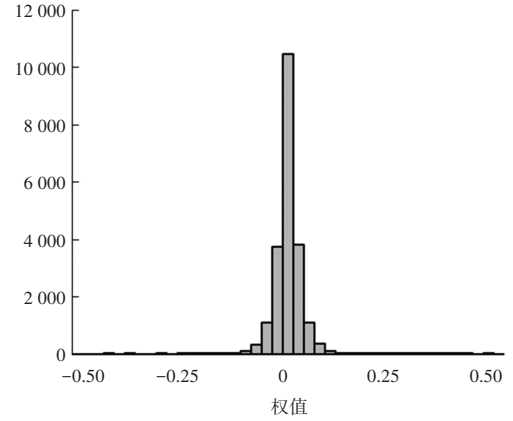
Parameters			Values
忆阻器单元间线电阻 R_{wire}			2.5 Ω
忆阻器阵列规模	MINST	Conv1	8×9
		Conv2	16×72
		Conv3	32×144
		FC	10×1 568
	CIFAR	Conv1	16×75
		Conv2	36×144
		FC1	120×1 296
		FC2	84×120
		FC3	10×84
忆阻器		最大可编程电阻	1 MΩ
		最小可编程电阻	10 KΩ

2.2 实验结果及分析

图 6 为 MINST 数据集上的 4 层卷积神经网络未添加 *L2* 正则化和添加 *L2* 正则化的权值分布图。从图 6 中可以明显看出,与不添加 *L2* 正则化相比,添加 *L2* 正则化可以将权值分布约束在一个较小值的范围内。图 7 为只考虑未添加 *L2* 正则化和添加 *L2* 正则化的忆阻器阵列计算精度对比图。从图 7 中可以看出,添加 *L2* 正则化后,MINST 数据集和 CIFAR-10 下忆阻器阵列的计算精度可以分别提升 15.30%和 11.40%。可以得出相较于未添加 *L2* 正则化,添加 *L2* 正则化虽然会有部分计算精度损失,但是训练出的权值对 IR-Drop 的鲁棒性有显著提高。



(a) 未使用 *L2* 范式正则化的权值分布



(b) 使用 *L2* 范式正则化后的权值分布

图 6 MINST 数据集上未添加 *L2* 正则化和添加 *L2* 正则化的权值分布图

Fig. 6 The weight distribution with and without *L2* regularization on MINST

图 8 为在 MINST 和 CIFAR 数据集下执行 NTWR 方法前后的计算精度对比图。从图 8 中可以看出执行 MCRC 算法和重训练后,可以明显提高忆阻器阵列的计算精度。在 MINST 和 CIFAR 数据集下,执行 NTWR 方法可以分别使忆阻器阵列的计算精度接近理想水平,计算精度损失分别小于 1%和 3%。

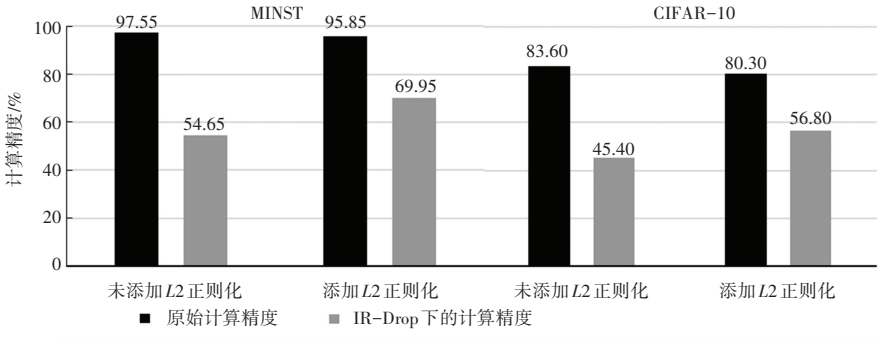


图 7　未添加 *L2* 正则化和添加 *L2* 正则化的计算精度对比图

Fig. 7　The accuracy comparison of network with and without *L2* regularization

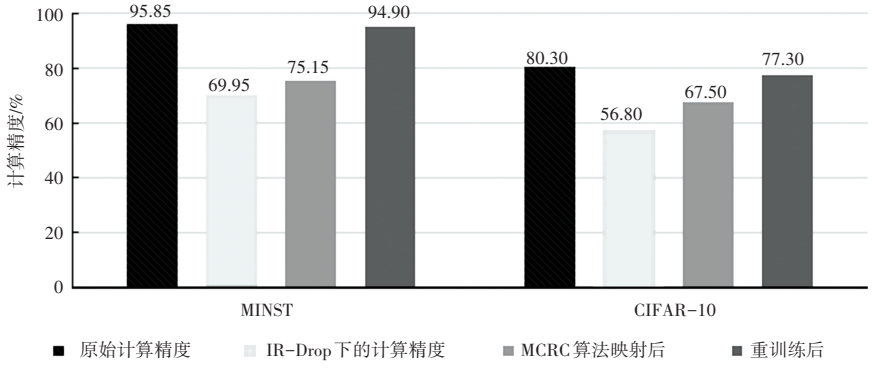


图 8　执行 NTWR 方法的计算精度对比图

Fig. 8　The accuracy comparison of memristor-based crossbar with NTWR

3　结束语

本文提出了一种基于权值缩减的神经网络训练方法，以此减小 IR-Drop 造成的计算精度损失。首先，通过 *L2* 正则化，将权值尽可能训练为较小值，以此减轻 IR-Drop 对忆阻器输出结果的影响。然后，执行基于行列约束的映射算法，将大权值尽可能映射到离输入端和输出端较近的忆阻器上，避免 IR-Drop 产生较大的输出结果偏差。最后，减小映射到离输入端和输出端较远处的大权值，再利用重训练恢复权值减小造成的计算精度损失。实验结果显示，本文所提方法最多可以将忆阻器阵列计算精度恢复到接近理想状态，计算精度损失小于 1%。

参考文献

[1] SUNG H J, CHANG T, EBONG I, et al. Nanoscale memristor device as synapse in neuromorphic systems[J]. Nano Letters, 2010,10(4):1297-1301.

[2] SHAFIEE A, NAG A, MURALIMANOHAR N, et al. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars[C]//ACM/IEEE International Symposium on Computer Architecture (ISCA). Seoul, Korea (South):IEEE, 2016:1-13.

[3] ZHU Zhenhua, SUN Hanbo, LIN Yujun, et al. A configurable multi-precision CNN computing framework based on single bit RRAM [C]//ACM/IEEE Design Automation Conference (DAC). Las Vegas, NV, USA:IEEE, 2019:1-6.

[4] HE Zhezhi, LIN Jie, EWETZ R, et al. Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping [C]//ACM/IEEE Design Automation Conference (DAC). Las Vegas, NV, USA:IEEE, 2019:1-6.

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

[6] WANG Yandan, WEN Wen, LIU Beiye, et al. Group scissor: Scaling neuromorphic computing design to large neural networks [C]// ACM/ IEEE Design Automation Conference (DAC). Austin, Texas, USA :ACM, 2017:1-6.

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

[8] 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 (ASP-DAC). Beijing, China: IEEE, 2020: 506-511.

[9] FOUDA M E, LEE S, LEE J, et al. IR-QNN framework: An IR drop-aware offline training of quantized crossbar arrays[J]. IEEE Access, 2020,8:228392-228408.

[10] WANG Junpeng, XU Qi, YUAN Bo, et al. Reliability-driven neural network training for memristive crossbar-based neuromorphic computing systems [C]// IEEE International Symposium on Circuits and Systems (ISCAS). Seville, Spain: IEEE, 2020: 1-4.

(下转第 63 页)