一种面向忆阻器平台的神经网络模型保护方法

靳 松 汪宇航

(华北电力大学电子与通信工程系 河北 保定 071003)

(华北电力大学河北省电力物联网技术重点实验室 河北 保定 071003)

摘 要 基于忆阻器交叉阵列的神经网络加速器具有存算一体化的优势,能够有效提升神经网络的执行性能.但忆阻器的非易失性特点容易被攻击者利用,通过实施模型偷窃攻击获取神经网络的权重参数.加速器作为用户端设备时,攻击者还可以实施模型复制攻击,通过收集输入-输出样本集合训练替代网络.上述攻击方法为神经网络模型在忆阻器平台的安全部署以及模型知识产权的保护带来严峻的挑战.为解决上述问题,本文提出一种基于权重混排和模糊化的模型保护方法.为抵御模型偷窃攻击,在将权重参数映射到忆阻器平台之前,以虚操作单元粒度对权重进行随机混排,打乱权重矩阵中元素的位置.同时,设计密钥保护的输出重定向模块以支持权重混排后神经网络正确的推理计算.另一方面,借助忆阻器的电阻漂移效应抵御模型复制攻击.一旦检测到恶意输入样本,系统自动进入自毁模式,加快忆阻器的电阻漂移,造成权重值模糊化,使网络模型的分类精度迅速下降到不可接受的程度,破坏攻击者收集输入-输出样本的努力.实验结果表明,与已有工作相比,本文提出的混排方法能够实现相同的安全保证,但硬件开销和功耗降低了两个数量级.当系统进入自毁模式,提出的权重模糊化方法只需几十个输入样本即可将网络模型的分类精度降低到40%以下,有效阻值攻击者的模型复制努力.

关键词 忆阻器平台;深度神经网络;模型保护;模型偷窃攻击;模型复制攻击;权重混排与模糊化

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2022.02377

On Protecting Neural Network Model in Memristor Platforms

JIN Song WANG Yu-Hang

(*Department of Electronic and Communication Engineering, North China Electric Power University, Baoding, Hebei* 071003)

(*Hebei Key Laboratory of Power Internet of Things Technology, North China Electric Power University, Baoding, Hebei* 071003)

Abstract The memristor crossbar based neural network accelerators have the advantage of in-memory computing, which can effectively improve the performance of neural network. However, the non-volatile characteristics of memristor are easy to be used by attackers to obtain the weight parameters of neural network by performing model stealing attack. When the accelerator is a user side device, the attacker can also conduct model replicating attack and train the alternative network by collecting input-output samples. The above attack methods bring severe challenges to the security deployment of neural network model on memristor platform and the protection of model intellectual property rights. In order to solve the above problems, a model protection method based on weight obfuscation and blurring is proposed in this paper. To resist the model stealing attack, before mapping the weight parameters to the memristor platform, the weights are randomly permuted with the granularity of virtual operation unit to obfuscate the position of elements in the weight matrix. At the same time, the output redirection module with key protection is designed to support the correct inference operations of neural network after weight obfuscation.

收稿日期:2021-12-23;在线发布日期:2022-07-21.本课题得到河北省自然科学基金"考虑忆阻器静态缺陷与动态偏差的存算一体化神经网络加速器可靠性优化方法研究"(F2021502006)资助.**靳 松**,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为深度学习安全、硬件安全和可靠性设计.E-mail:jinsong@ncepu.edu.cn.**汪宇航**,硕士研究生,主要研究方向为深度学习安全、硬件安全和可靠性设计.

On the other hand, the resistance drift effect of memristor is utilized to resist the model replicating attack. Once one or more malicious input queries are detected, the system automatically enters the self-destruction mode which speeds up the resistance drift of the memristors. This will result in the blurring of the weight value, rapidly degrading the classification accuracy of the network model to an unacceptable level, and undermining the attacker's efforts to collect input-output samples. The experimental results show that compared with the existing work, the weight obfuscation method proposed in this paper can achieve the same security guarantee, but the hardware overhead and power consumption are reduced by two orders of magnitude. When the system enters the self-destruction mode, the proposed weight blurring method only needs dozens of input samples to reduce the classification accuracy of the network model to less than 40%, which effectively blocks the attacker's model replicating efforts.

Keywords memristor; deep neural network; model privacy protection; model stealing attack; model replicating attack; weight obfuscation and blurring

1 引 言

近些年来,基于卷积和深度神经网络的机器学习算法在图像分类[1]、视频处理[2]和机器翻译[3]等领域取得了巨大的成功,并开始在手机、个人电脑、平板电脑等用户端设备上普及.然而,运行神经网络需要底层硬件平台具有较高的计算能力.一些大规模的神经网络需要几百万次浮点运算才能完成一次推理计算[4-5].传统的基于冯·诺依曼体系架构的计算平台不能提供满意的计算性能.人们因而开始寻找新的解决方案.

基于忆阻器交叉阵列(Memristor crossbar)的神经网络加速器由于具备存算一体化(In-memory computing)的优点,被认为是较有前途的下一代高性能计算平台[6].忆阻器交叉阵列不仅能够存储神经网络的权重参数,还能够以常数计算复杂度($O(1)$)实现矩阵-向量的乘法运算.与传统的 CPU、GPU 和定制电路相比,忆阻器交叉阵列能将执行神经网络运算时的计算能效(Energy efficiency)提升上百倍[7-8].

忆阻器交叉阵列的上述优点虽然有助于加速神经网络的执行,但也存在着一些安全漏洞[9].由于忆阻器的非易失性特点,芯片即使断电后,编程在忆阻器上的权重参数仍会保留.攻击者能够使用微探测技术来执行模型偷窃攻击,从忆阻器中获取权重参数[10-12].当加速器部署在用户端时,即使没有探测能力的攻击者仍然可以通过调用机器学习服务的 API 来执行模型复制攻击[13-14].攻击者向模型发送大量精心设计的输入样本,并观察网络相应的分类或预测输出.这些输入输出样本集合可以用来训练攻击者自己的替代模型.

高精度神经网络模型的训练往往需要大量的时间和金钱投入.网络模型被窃取会侵犯模型拥有者的知识产权并造成收益损失.更为严重的是,拥有神经网络模型的确切参数后,攻击者可以发起白盒对抗攻击(Adversarial attack),从而导致模型做出错误的分类决策[15].这会给一些安全关键应用,如自动驾驶汽车或无人机等带来重大威胁.因此,保护部署在忆阻器交叉阵列上的神经网络模型具有十分迫切的需求.

为了抵御模型偷窃攻击,一个自然的想法是加密存储在忆阻器中的权重,在神经网络计算开始时解密它们,并在计算完成后重新加密这些权重[16].然而,实现加/解密功能需要复杂的硬件设计、引入较大的硬件开销;实时加/解密操作还会降低神经网络的执行性能.此外,权重参数映射到忆阻器交叉阵列后,重复对某些权重进行加/解密操作意味着对这些忆阻器的反复编程.这会降低设备耐久性(Endurance)并导致可靠性问题.一些研究人员还提出行维度的权重矩阵混排方法[17],通过随机打乱权重行实现无需加/解密的模型保护.然而,交叉阵列中忆阻器行的数量较多(如 128 行).行混排后,为实现输入神经元到交叉阵列相应字线的正确路由需要非常大的硬件开销.

在抵御模型复制攻击方面,在线检测恶意查询(Query)和模型预测结果添加扰动(Perturbation)等策略通常应用于服务器级应用场景中,如机器学习即服务(MLaaS)[18-19].现有的方案计算复杂度高,只有当攻击者的查询预算(Budget)有限时才能

有效的工作. 然而，对于资源受限的用户端设备而言，计算密集型的防御措施并不可取. 当网络模型部署在用户端时，能够访问设备的攻击者在理论上具有无限的查询预算. 当攻击者获得足够多的输入输出样本，就可以训练他们自己的替代模型[19].

考虑上述安全威胁，本文提出基于权重混排和模糊化的低开销神经网络模型参数保护方法. 为抵御模型偷窃攻击，本文以虚操作单元(Virtual Operational Unit，VOU)为粒度对权重矩阵进行行列维度的随机混排. 通过密钥保护的输出重定向模块保证权重混排后神经网络仍可以进行正确的推理计算. 若无密钥，直接使用混排后的神经网络模型，其分类精度非常低，接近于随机猜测. 攻击者获得这样的混排网络模型也就毫无意义.

当神经网络模型部署在用户端时，攻击者可以通过身份验证访问该模型提供的机器学习服务. 此时，理论上攻击者有无限的查询预算来收集足够的输入输出样本以发起模型复制攻击. 在现有的方案中，例如对网络预测结果添加扰动，都只能延缓攻击者复制模型的速度，但不能完全消除这种攻击. 因此，从高度安全性角度出发，本文利用忆阻器的电阻漂移来实现神经网络模型的自毁，以抵御模型复制攻击. 由于模型复制攻击时的恶意查询通常与正常查询的分布有很大的偏差[20]，在网络模型进行推理计算时通过检查查询类型可以识别潜在的攻击. 当少量恶意查询被检测到，加速器将切换到自毁模式. 此时，通过在交叉阵列计算时复用幅值更大、持续时间更长的编程电压脉冲加快忆阻器的电阻漂移. 相应的，存储在忆阻器中的权重就会很快被模糊化(即权值发生较大变化)，网络分类精度也会在很短的时间内降低到不可接受的程度. 这使得攻击者无法获取有用的输入输出样本，也就无法进一步实施模型复制攻击. 此后，合法用户可以通过再次验证、重新编程忆阻器来恢复基于神经网络模型的服务. 本文的主要贡献包括：

(1) 本文提出基于虚操作单元的权重混排方法以抵御模型偷窃攻击. 提出的混排方法有足够大的排列空间，防止攻击者将混排后的权重矩阵恢复至原始权重矩阵. 由于不需要对权重执行加/解密操作，对神经网络的执行性能和设备耐久性没有负面影响. 此外，提出的方法只需要一个轻量级的输出重定向模块来支持混排权重矩阵的计算，具有较低的硬件开销.

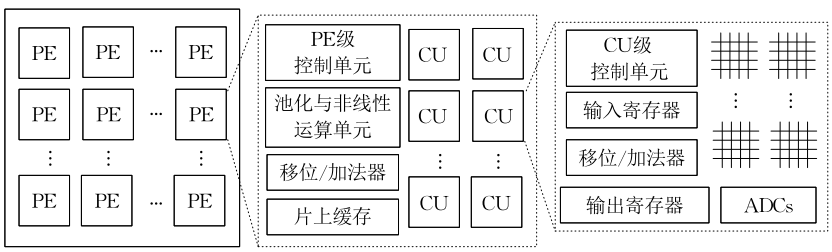
(2) 本文提出通过加速忆阻器的电阻漂移实现权重模糊化以抵抗模型复制攻击. 通过复用忆阻器交叉阵列的编程电压脉冲，可以加速忆阻器的电阻漂移，改变忆阻器表示的权重参数. 神经网络的分类精度在很短的时间内也会下降到不可接受的程度，使得攻击者无法进一步地实施模型复制攻击.

本文第 2 节介绍一些背景知识；第 3 节介绍安全威胁场景和提出的防御方案概述；第 4 节介绍基于虚操作单元的权重混排方法；第 5 节介绍权重模糊化方法；第 6 节给出实验结果；第 7 节介绍相关研究现状；第 8 节对全文进行总结.

2　背景知识介绍

2.1　基于忆阻器交叉阵列的神经网络加速器

图 1 给出了基于忆阻器交叉阵列的神经网络加速器的一般化结构示意图. 考虑到页面限制，图 1 中省略了主 CPU 和片外 DRAM. 图 1 所示的加速器具有层次化的组织结构. 多个处理引擎(PE)通过片上网络实现互连. 每个 PE 由多个计算单元(CU)组成. 每个计算单元都含有多个忆阻器交叉阵列. PE 级控制单元负责协调 CU 的操作. 片上缓存用于临时存储输入和输出特征图. PE 中还包括池化和非线性功能单元，以支持神经网络中的相应功能. CU 级控制单元用于协调忆阻器交叉阵列的操作. 输入神经元存储在与交叉阵列字线相连的输入寄存器中，而计算得到的输出神经元存储在输出寄存器中. 经过移位和加法电路以及多个数/模转换模块(ADC)处理，可以将模拟点积计算结果转换为数字量.



(a) 基于忆阻器的神经网络加速器　(b) 处理单元(PE)　(c) 计算单元(CU)

图 1　基于忆阻器交叉阵列的神经网络加速器结构示意图

2.2 操作单元

最近的一些研究成果表明，受限于计算精度，在单个周期内，忆阻器交叉阵列中只有有限数量的字线和位线可以被同时激活[5,21-22]. 可同时激活的字线和位线的最大数量取决于 ADC 的精度限制. 例如，在为卷积神经网络加速而设计的忆阻器宏模块中，大小为 512×256 的交叉阵列内仅可同时激活 9 条字线和 8 条位线[21]. 因此，一个大的交叉阵列可以被划分成许多小块，每个小块就被称为操作单元(Operation Unit，OU). 在运行时，这些 OU 被逐个激活，每个 OU 计算输出神经元的部分结果. 在累加部分结果之后，可以得到神经元最终的输出值. OU 的提出权衡了忆阻器交叉阵列的性能和精度. 然而，本文作者发现这种执行模式对于权重混排也很有价值，具体细节将在第 4 节中介绍.

2.3 忆阻器的电阻漂移

忆阻器交叉阵列在执行计算时，每次向忆阻器施加一个小电压，该忆阻器的阻值相应的都会有一个微小的漂移(Drift). 漂移的程度取决于施加的电压脉冲的幅值和持续时间[23-24]. 随着电阻漂移的累积，映射到忆阻器上的权重将随之改变，最终降低网络的分类精度[25]. 因此，忆阻器需要定期进行刷新操作. 在第 5 节中，我们将利用电阻漂移实现神经网络模型的自毁.

3 防御方法概述

在本节中，我们将阐述部署在忆阻器平台上的神经网络模型可能面临的安全威胁. 此外，我们还对提出的防御方法进行概述.

3.1 威胁模型

对于部署在用户端的忆阻器神经网络加速器，可能面临的安全威胁主要包括模型偷窃攻击和模型复制攻击.

(1) 模型偷窃攻击. 攻击者可以通过微探测或微复制方法探测存储在忆阻器中的权重[11-12]. 窃取模型参数后，他们可以免费使用，或者出售模型以获取非法利润. 更严重的是，凭借获取的模型参数信息，攻击者可以执行白盒对抗攻击[15]，通过在输入中添加不可察觉的噪声扰动，导致神经网络产生错误的分类结果. 这对一些安全关键应用(如自动驾驶汽车或无人机)构成了很大的威胁.

(2) 模型复制攻击. 攻击者不需要了解神经网络模型的详细信息. 他们可以通过应用程序 API 调用部署在加速器上的网络模型的机器学习服务[18-19]. 通过向网络发送大量精心合成的输入样本，并观察网络的预测输出，可以形成攻击者自己的标签训练集. 然后，攻击者可以使用这些训练集训练他们自己的替代模型.

本文不考虑各种侧信道攻击[26-27]和芯片中的木马植入攻击[28]. 处理这些攻击的对策与本文提出的防御方法是正交的. 可以将这些策略与本文方法结合起来，以获得更广泛的安全保障. 此外，本文不考虑拒绝服务(DoS)攻击，因为作者认为攻击者的首要目的是获得神经网络模型. 与部署在服务器端的神经网络应用程序不同，阻止部署在用户端的神经网络加速器执行正常的操作对攻击者来说意义不大.

3.2 防御方法概述

图 2 为本文提出的防御方法的示意图. 通过执行权重混排以抵御模型偷窃攻击. 交叉阵列首先被

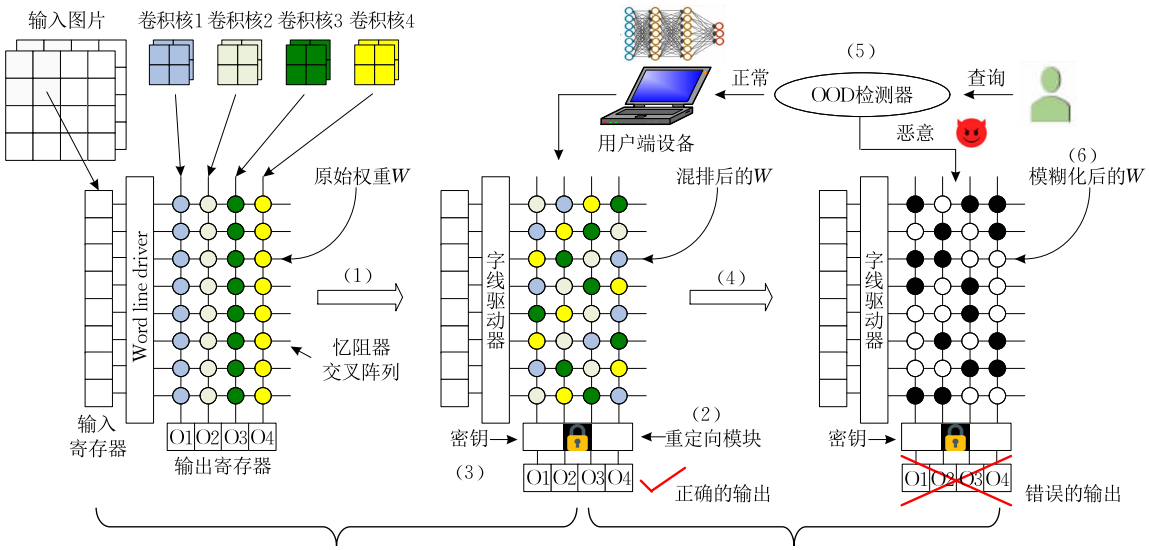


图 2 本文防御方法概览

随机划分为许多大小相等的虚操作单元(VOU).然后,沿列维度执行 VOU 权重的随机混排(标示(1)).输出重定向模块(标示(2))可以支持混排后的权重矩阵仍能实现正确的点积运算.与混排权重相关的重定向向量被设置为密钥(标示(3)).没有这些密钥,加速器无法执行网络正确的推理操作.

为了抵御模型复制攻击,本文借助忆阻器的电阻漂移来实现权重值模糊化,造成神经网络模型的自毁(标示(4)).恶意查询与正常查询的分布有很大的区别.因此,可以在神经网络运行时检查输入的查询(标示(5)).如果连续检测到恶意查询,加速器将切换到自毁模式.在这种模式下,复用具有更高幅值和更长持续时间的编程电压脉冲作为交叉阵列的输入电压.这会加速忆阻器的电阻漂移,导致权重快速模糊化,并在很短的时间内将模型的分类精度降低到不可接受的程度(标示(6)).攻击者因此无法进一步执行模型复制攻击.之后,合法用户通过再次进行身份验证,重新编程忆阻器,以恢复基于神经网络模型的服务.

4　抵御模型偷窃攻击的权重混排方法

本节介绍基于 VOU 的权重混排方法以及相关的硬件设计方案.

4.1　虚操作单元(VOU)

正如在第 2 节中提到的,为了达到理想的计算精度,可以将忆阻器交叉阵列划分为许多大小相等的块,称为操作单元(OU).当忆阻器交叉阵列执行神经网络的计算时,这些 OU 依次被激活.最后将同列 OU 的计算结果累加,得到输出神经元的最终计算结果.

本文作者发现,以列维度对 OU 进行混排可以在实现高安全性的同时显著降低硬件开销(将在第 4 节中对此进行详细说明).但是,对物理 OU 进行混排将会损害安全性,因为物理 OU 内可能包含一个卷积核的全部权重.这里,物理 OU 表示由多个连续相邻的字线和位线组成的 OU.图 3(a)给出了包含 8 个物理 OU 的交叉阵列,每个 OU 包含 4 个相邻字线上的权重.注意,为简单起见,图 3 未显示需要将多个忆阻器列组合在一起表示权重的实际情况.如图 3(b)所示,如果我们在物理 OU 粒度中混排权重,例如,将 OU1 与 OU3 交换,将 OU2 与 OU4 交换等,混排后的 OU 仍然包含卷积核的全部权重(假设卷积核为 2×2).也就是说,攻击者仍然可以获得有关卷积核的完整信息,从而降低了混排的安全性.

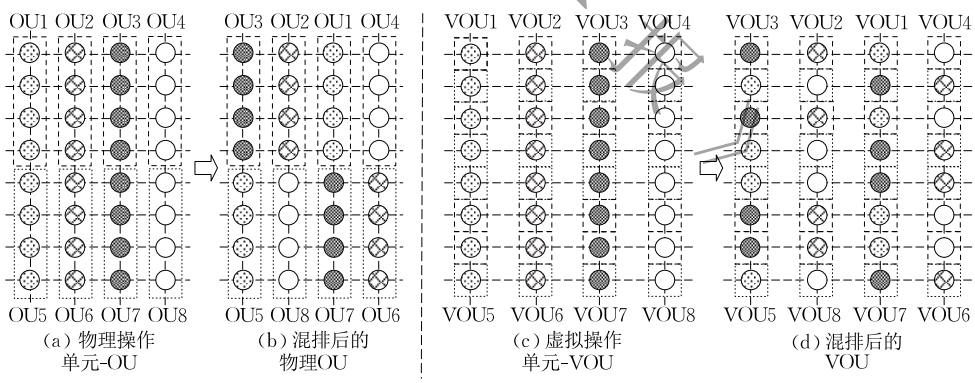


图 3　物理 OU 粒度和 VOU 粒度下的混排对比

因此,本文提出将交叉阵列划分为许多大小相等的虚操作单元(VOU).如图 3(c)所示,VOU 的大小与物理 OU 相同,但由多个非相邻字线组成(例如,VOU1 由 4 个权重组成,权重位于 4 条非相邻字线上).我们执行与图 3(b)中相同的混排处理.此时位于相邻字线上的权重是交错的,如图 3(d)所示.这样,权重混排不仅可以在卷积核之间实现,还可以在单个卷积核内实现.安全性也相应得到提升.

4.2　基于 VOU 的权重混排算法

算法 1 给出了基于 VOU 的权重混排算法伪代码.基于经典的 Knuth Shuffle 算法[29],我们对 VOU 执行随机混排.算法 1(第 1～7 行)中的函数 *KnuthShuffle*()说明了该算法的细节.为了将整数数组 a 里的 n 个元素进行完全随机混排,Knuth Shuffle 算法以迭代方式执行.每次迭代中,该算法生成一个随机整数.使用该整数索引 a 中的元素,然

后将该元素与数组中最后一个元素互换位置.理论分析证明,Knuth Shuffle 算法可以实现无偏置完全随机混排.

算法 1. 基于 VOU 粒度的权重混排算法.

```
1.  FUNCTION KnuthShuffle(array a, length n)
2.    FOR i=n→2 DO //索引值由 1 开始
3.      j←生成随机整数 j,1≤j≤i
4.      交换 a[j] and a[i]
5.    END FOR
6.    返回 a //返回混排后的数组 a
7.  END FUNCTION
8.  FUNCTON WeightObfuscation(row id set R)
9.    RPM←empty //初始化混排的行号组
10.   RPM=KnuthShuffle(R,length(R)) //混排行
11.   RPM 中每 8 行组成一个行组 RG,一共 16 个行组
12.   每个行组 RG 划分为 16 个 VOU,一共 16 个 VOU
13.   FOR i=1→16 DO //混排 RG 中的 VOU
14.     RGPM(i)←empty
15.     RGPM(i)=KnuthShuffle(RG(i),length(RG(i)))
16.   END FOR
17.   返回 RGPM //返回混排后的 VOU
18. END FUNCTION
```

基于 KnuthShuffle 算法,本文实现了基于 VOU 的随机全混排(权重混排函数见算法 1,第 8~18 行).接下来以 128×128 的忆阻器交叉阵列为例对算法进行详细说明.假设 VOU 大小为 8×8,其中包含 8 条非相邻字线和 8 条相邻位线(假设需要 8 个忆阻器来表示一个 16 位的权重).首先,以随机方式将交叉阵列划分为许多大小相等的 VOU.具体地说,生成一个数组 R,R 中元素为交叉阵列的行序号(例如,$R(1,2,\cdots,128)$).然后调用 KnuthShuffle 算法对 R 执行随机全混排,例如,$R(1,2,\cdots,128)$重排为 $RPM(9,126,\cdots,15)$(算法 1 的第 9~10 行).值得注意的是,这里只是排列行序号,而不是交叉阵列里的物理行.接下来,对于获得的混排数组 RPM,每 8 个行序号(即 8 个字线行)形成一个行组,我们总共可以获得 16 个行组(第 11 行).对于每个行组,通过将每 8 个相邻位线分组在一起形成 16 个 VOU.因此,整个交叉阵列可以随机划分为 256 个 VOU(第 12 行).对于每个包含 16 个 VOU 的行组,我们再次调用 Knuth Shuffle 算法对这些 VOU 执行全混排.对所有行组进行混排后,函数返回 VOU 混排结果(第 13~17 行).

4.3 输出重定向模块设计方案

本文修改了忆阻器交叉阵列的部分外围电路以保障权重矩阵在混排后仍能得到正确的点积运算结果.修改主要包括两部分:第一部分是行线(字线)、列线(位线)激活模块,用于选择和激活 VOU;第二部分是输出重定向模块,用于支持属于同一个输出神经元的 VOU 的计算结果进行正确的累加.图 4 给出了硬件设计示意图.在该图中,交叉阵列尺寸为 128×128.假设每个忆阻器可以代表 2 位,这个交叉阵列包含 16 个 VOU 行组,每个组包含 16 个 VOU,因此可以计算 16 个神经元.假设 VOU 已被混排.出于方便显示的目的,图中的 VOU 与物理 OU 类似,但实际上它们由非相邻字线组成.如图 4 所示,通过逐一激活 VOU 来执行计算.在每个执行周期中,CU 级控制器通过获取相应的激活向量并将其发送到行和列线激活模块,使 VOU 能够启动计算.VOU 的激活顺序取决于激活向量的配置.例如,可以按从左到右的顺序逐个激活 VOU.

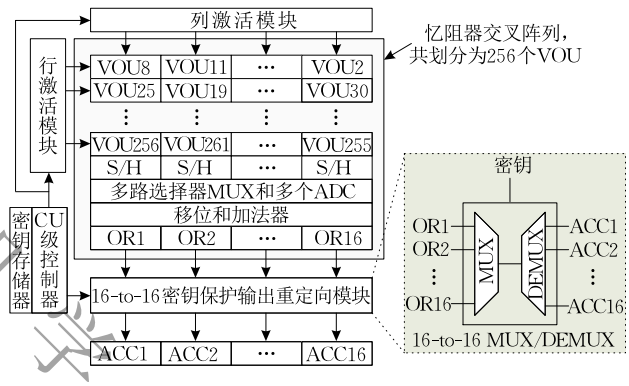


图 4 支持基于 VOU 的权重混排方法的硬件设计方案

由于 VOU 已被混排,原本属于同一输出神经元的 VOU 也变为交错的.因此,应正确累加这些 VOU 的计算结果,以获得输出神经元的最终值.为了达到这一目的,本文提出为 CU 中的每个交叉阵列附着一个输出重定向模块.当交叉阵列进行计算时,在每个执行周期中激活 1 个 VOU.此 VOU 的计算结果通过输出重定向模块送入到该 VOU 所属输出神经元的累加器中(图 4 中的 ACC_i).输出重定向模块由 16 选 1 的数据选择器(MUX)和 1 到 16 的数据分配器(DEMUX)组成,可支持 16 到 16 的数据路由.MUX 和 DEMUX 的通道选择信号被设置为密钥.每次激活 VOU 时,CU 级控制器获取存储在 SRAM 密钥存储器中的密钥,将密钥发送到重定向模块,对 VOU 的计算结果进行正确路由.相反,如果没有密钥,交叉阵列就无法实现网络的正确计算,导致模型分类精度极低.获取这样的模型对攻击者来说是没有意义的.

考虑到硬件开销,一个 CU 中的所有交叉阵列可以对映射的权重矩阵采用相同的混排方案.在这

种情况下，每个 CU 只需要一个控制器，用于协调所有阵列的操作. 然而，不同的 CU 可以采用不同的排列. 这可以通过为各个 CU 设置单独的激活向量和重定向向量来实现.

4.4　混排方法破解难度分析

针对权重混排矩阵，攻击者可以尝试使用基于暴力搜索的破解方法恢复原始权重矩阵[11,17]. 不过，本文提出的混排方法有足够大的排列空间来阻止攻击者的破解努力. 以单个 128×128 的交叉阵列为例，假设 VOU 为 8×8. 此交叉阵列中包含 16 个 VOU 行组，每个组包含 16 个 VOU. 根据算法 1，Knuth Shuffle 算法对每个 VOU 行组执行随机全混排. 一个 VOU 行组所有可能的排列数目达到 16!. 因此，对于所有 16 个 VOU 行组，VOU 的可能排列的总数是 $(16!)^{16}$，远远超过了暴力搜索的极限能力. 此外，如果不同的 CU 采用不同的排列，可能的排列总数将会更多. 由此可见，采用暴力搜索方式破解本文提出的混排方案是无效的. 第 6 节的实验结果还会证明，攻击者试图基于混排后的权重矩阵，通过网络重训练来恢复模型精度也是无效的.

4.5　密钥保护

由于忆阻器交叉阵列依赖于密钥来确保神经网络计算的正确执行，因此密钥的保护也是一个非常重要的问题. 针对密钥，一种常用的攻击方式是冷启动攻击[30]. 尽管冷启动攻击在 DRAM 上有效，但在 SRAM 上效果要差得多，因为 SRAM 在断电后数据保留的时间很短(几毫秒). 因此，可以将密钥存储到片上 SRAM 密钥存储器中. 具体来说，密钥可以被加密并存储在片外非易失性存储器中. 当加速器启动并开始执行基于神经网络模型的服务时，CPU 可以读取密钥、对其解密并写入 CU 中的片上密钥存储器. 控制器获取密钥，控制交叉阵列执行正确的计算. 加速器断电后，密钥立即掉电丢失. 这样，攻击者便无法获取密钥.

对于密钥进行二次加密以便存放于片外非易失性存储器的问题，可以参考如文献[31]提出的密钥存储、分发协议等相关做法. 处于篇幅限制，本文就不做详细说明了.

5　抵御模型复制攻击的权重模糊化方法

本节首先介绍对恶意查询的在线检测方法. 随后详细介绍所提出的权重模糊化方法.

5.1　恶意查询的检测

研究发现，现有的模型复制攻击在实施过程中几乎都不可避免地会产生分布外(Out-of-Distribution，OOD)查询[32-33]，OOD 查询与正常查询存在较大偏差. 因此，通过在运行时检测是否有 OOD 查询，我们可以识别出潜在的攻击.

检测 OOD 查询的有效方法是计算神经网络的 softmax 层输出概率(MSP)的最大值[20]. 对于正常查询，模型输出的 MSP 通常较大；而 OOD 查询将导致模型产生较小的 MSP. 对于输入为 x，K 个输出概率分别为 $\{y_i\}_{i=1}^{K}$ 的 NN 模型，可以通过阈值化 MSP 来进行 OOD 检测[32]：

$$Dec(x)=\begin{cases} ID, & \max_i(y_i)>\tau \\ OOD, & \text{其他} \end{cases} \tag{1}$$

式(1)中 ID 表示正常查询(In Distribution). τ 表示阈值. 正常查询因为与训练集中的样本有着相同的特征和分布，因此神经网络在预测时会对正确的标签给出较大的概率值. 反之，非正常查询，包括恶意查询，因为与训练数据集存在显著差别，神经网络在预测时所有类的标签概率值较为平均. 据此，可以在神经网络训练过程中，对每一类的样本统计网络输出的概率值(如取 top-5 概率值并相加)，并据此确定某一类样本预测结果的阈值 τ. 用于恶意查询的检测时，若神经网络的 softmax 层对某一个输入样本得到的概率和小于或远小于所有标签类训练记录的值，则表明该查询样本不是正常查询，从而被识别出来. 当然，为了尽量减少误判情况，可以在概率和上增加一个小的保护带，比如允许上下浮动 5%.

5.2　权重模糊化方法

一旦检测到恶意查询，被攻击方可以采用不同的防御方案. 然而，现有的防御方案只能延缓攻击者试图复制神经网络模型的努力，而不能最终阻止模型复制攻击. 所以，从高安全性角度考虑，本文提出借助忆阻器的电阻漂移来实现神经网络模型的自毁. 当连续检测到恶意查询时，加速器将切换到自毁模式. 在这种模式下，禁止忆阻器正常的刷新操作. 此外，加速忆阻器的电阻漂移，导致权重值快速模糊化，并在很短的时间内将神经网络的分类精度降低到不可接受的程度.

当加速器执行神经网络的推理操作时，正常馈送到交叉阵列的输入电压脉冲幅值较小(比如 0.3 V)且持续时间较短(比如 100 ns). 因此，忆阻器漂移的累积是一个缓慢的过程. 攻击者仍有可能在这段时间内获取足够数量的输入输出样本. 为了加速权重

模糊化，本文采用具有更大幅值和更长持续时间的忆阻器编程电压来代替正常输入电压.这将迅速恶化忆阻器的电阻漂移，很快将神经网络模型的分类精度降低到不可接受的程度.

电阻漂移在很大程度上取决于外加电压脉冲的幅值和持续时间.基于经典的忆阻器模型[34]，忆阻器的忆阻值可以表示为

$$M(\alpha)=\alpha\cdot R_H+(1-\alpha)\cdot R_L \tag{2}$$

其中 R_H 和 R_L 表示忆阻器的最大和最小电阻.α 是相对的掺杂前沿位置，范围为 0 到 1.可通过求解速度微分方程获得

$$\alpha(t)=\frac{R_H-\sqrt{R_H^2-2\cdot(R_H-R_L)\cdot(A+B)}}{R_H-R_L} \tag{3}$$

其中，$A=\mu_v\cdot\frac{R_L}{h^2}\cdot\int_{t_0}^{t}V(t)\,\mathrm{d}t$，$B=R_H\cdot\alpha_0+\frac{1}{2}\cdot(R_H-R_L)\cdot\alpha_0^2$.$\alpha_0$ 是 α 的初始状态.根据成熟工艺中的忆阻器参数[34]，我们可以设置 $h=50\,\mathrm{nm}$，$\mu_v=10^{-14}\,\mathrm{m^2S^{-1}V^{-1}}$，$R_H=1\,\mathrm{M\Omega}$，$R_L=1\,\mathrm{K\Omega}$，$\alpha_0=0.3$，$t$ 从 $t=0$ 开始变化.图 5 显示了忆阻器在施加不同幅值和持续时间的电压脉冲下电阻的漂移情况.

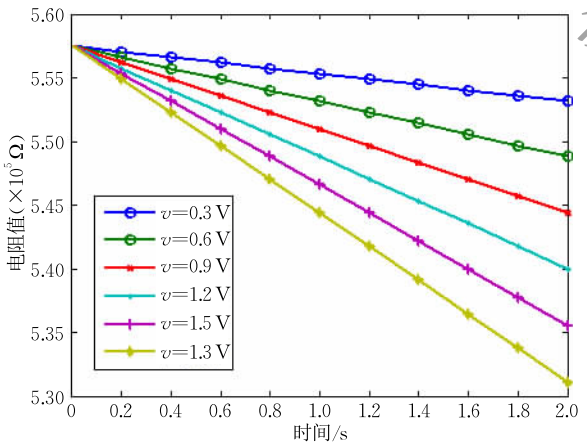


图 5 不同持续时间和电压幅值下的忆阻器电阻漂移情况

如图 5 所示，当增加施加在忆阻器上的电压脉冲幅值和持续时间时，电阻漂移明显变快.基于上述观察，当加速器进入自毁模式时，使用忆阻器的编程电压来加速电阻漂移.例如，电压幅度可以从 0.3 V 增加到 1.8 V，持续时间从 100 ns 延长到 1 μs.第 6 节的实验结果表明，应用编程电压可以加速电阻漂移和权值模糊化，导致神经网络的分类精度在很短的时间内下降到不可接受的程度.

本文采用权重模糊化抵御模型复制攻击的主要考虑有两点：(1) 尽量降低安全性设计的复杂度；(2) 满足较高安全性的要求.本文实现加速电阻漂移的方法是切换忆阻器交叉阵列的输入电压，将交叉阵列正常计算时的输入电压切换为编程电压.这只是复用了本已存在的功能电路，没有额外的电路模块设计要求，实现起来较为简单.同时，对于有着较高安全性要求的应用场景，如无人机等应用，当系统检测到有恶意攻击时，出于安全性考虑，暂时彻底毁掉模型不失为一种较为安全的防御方法，可以避免攻击者采取其他手段获取模型.

模型自毁后，如何恢复服务，则需视具体情况区别对待，通过软件的设置进行处理.例如，如果加速器系统部署在服务器端，则可以通过识别使用恶意查询的用户身份，禁止此用户使用查询功能.若加速器用于客户端且攻击者是通过非法方法访问模型，则需重新进行认证操作，断绝非法访问.但如果攻击者是通过取得合法访问身份，恶意窃取模型，系统甚至可以不恢复模糊化的权重，彻底毁掉部署在客户端的模型.安全认证可采用文献[35]提出的方法.这些方法与本文的模型保护方法正交，可以组合使用.忆阻器重新编程方面，文献[36]提出了对忆阻器交叉阵列进行快速、高精度的编程方法，可在很短时间内实现权重值的恢复，感兴趣的读者可以参考这些文献.

5.3 权重模糊化方法的安全性分析

一个合理的假设是攻击者并不拥有网络模型的训练数据集(否则攻击者可以自行训练高精度模型而没有必要窃取模型了).这种情况下，攻击者实施模型复制攻击时使用的查询样本(恶意查询)与训练数据集中的样本会存在显著区别，从而可以被系统较容易的检测到.攻击者不知道阈值的具体信息，即使知道，攻击者也无法绕过检测方法.因为恶意查询与训练集中的正常查询有着显著区别，在不知道训练集的情况下，攻击者无法伪装恶意查询.当系统检测到恶意查询时(阈值可以由用户设定)，加速器进入自毁模式.这种自毁模式是不可逆的.权重值快速模糊化，在很短时间将模型精度下降到无法接受的程度.此时，实施模型复制攻击已无实际意义.后续，用户必须重新认证、对忆阻器交叉阵列重新编程后才可以再次使用加速器中部署的神经网络模型.所以，将恶意查询与正常查询穿插进行以隐藏恶意查询，避免加速器进入自毁模式是无法实现的.攻击者当然也可以只使用很少量的恶意查询.这样做虽然可以避免加速器进入自毁模式，但恶意查询数量太少的话，获得的输入、输出样本不足以实现模型复制，无法复制出高精度，有实际使用价值的模型.

系统将加速器切换为自毁模式的指标可根据复

制模型需要的输入输出样本数量以及用户对于安全性的要求来确定.根据已有工作和本文实验的评估，对于 VGG-16 或 VGG-19 这种规模的神经网络，攻击者至少需要超过 400 个输入输出样本对，才能使复制模型的精度达到可接受的水平(见实验部分)；而若想实现高精度模型复制，需要的样本数量会更多.参考上述数据，系统可以根据用户对于安全性的要求来确定检测到的恶意查询数量与加速器进入自毁模式的关系.例如，若用户对于安全性有着非常高的要求，则即使检测到一两个恶意查询，系统都可以设置加速器进入自毁模式.反之，若用户对安全性无太高要求，则可以适当放宽进入自毁模式的条件，只要能保证恶意查询的数量远远小于模型复制需要的样本数量即可.

攻击者也可以只使用很少量的恶意查询或者间歇性的插入少量恶意查询来实现隐蔽攻击.这样做虽然可以避免加速器进入自毁模式，但恶意查询数量太少的话，获得的输入、输出样本不足以实现模型复制，无法复制出高精度、有实际使用价值的模型.且系统可以对这种攻击模式进行预防.例如，如果发现间歇性的恶意攻击，则采取一定的应对措施.通过软件的设定不难实现上述措施.

对忆阻器进行编程操作会影响其使用寿命.但需注意的是，一个合理的假设是，通过恶意查询进行模型复制攻击毕竟属于小概率事件，不会经常发生.且一旦检测的恶意查询，在没有确保切断攻击者的攻击途径之前，系统也不会对加速器进行重编程从而被反复攻击.因此，本文提出的加快电阻漂移以及通过重编程恢复权值的做法在实际部署中是很少采用的.因此，可以说本文方法对忆阻器使用寿命的影响较为轻微，且在安全性具有较高优先级的考虑下，是一种可以承受的代价.

6　实验和讨论

6.1　实验配置

本文基于开源模拟器 NeuroSim[37] 开展忆阻器平台上神经网络的推理计算.所有硬件配置均参考 ISAAC[7].加速器包含 168 个 PE，每个 PE 配备 12 个 CU.每个 CU 含 8 个忆阻器交叉阵列，交叉阵列的大小设置为 128×128.VOU 的大小设置为 8×8.CU 级控制器和密钥存储器为 CU 内的所有交叉阵列所共享.使用 CACTI[38] 在 32 nm 工艺下评估密钥存储器的功耗.在 CU 中，每个交叉阵列连接一个输出重定向模块(ORM)，该模块由一个 16 选 1 的 MUX 和一个 1 到 16 的 DEMUX 组成.MUX 和 DEMUX 的面积和功耗数据由 Synopsys Design Compiler 在 TSMC 45 nm 工艺下进行评估并缩放到 32 nm.表 1 给出了一些重要的配置参数.

表 1　硬件配置

芯片参数参考于文献[7]			
组件	数目	面积/mm²	功耗/mW
交叉阵列(C)	8 /CU	0.0016	19.2
CU	12/PE	0.1570	289.0
PE	168/芯片	62.5000	55.4
本文方案额外添加的模块			
MUX(16-1)	1/ORM	2.38e-4	0.232
DEMUX(1-16)	1/ORM	2.86e-4	0.408
重定向模块(ORM)	1/C	5.25e-4	0.641
密钥存储空间	256B/C	7.70e-4	0.230

实验中训练四个神经网络模型，在两个数据集上实现数字或图像分类.其中，LeNet-5 应用于 MNIST 数据集以识别 10 类手写数字.而 VGG-16、VGG-19 和 GoogleNet 则应用于 ImageNet 数据集，以实现 1000 个类别的图像分类.实验中权重值采用 16 位定点数表示，采用 2 的补码形式.表 2 详细说明了四个网络的拓扑结构以及网络训练后的分类精度.

表 2　网络结构参数及原始分类精度

网络名称	精度/%	网络结构(只显示卷积层和全连接层)
LeNet-5	98.8	conv5*6-conv5*16-84-10
VGG16	93.5	conv3*64-conv3*64-conv3*128-conv3*128-conv3*256-conv3*256-conv3*512-conv3*512-conv3*512-conv3*512-conv3*512-4096-4096-1000
VGG19	89.9	conv3*64-conv3*64-conv3*128-conv3*128-conv3*256-conv3*256-conv3*256-conv3*256-conv3*512-conv3*512-conv3*512-conv3*512-conv3*512-conv3*512-conv3*512-conv3*512-4096-4096-1000
Google-Net	94.1	conv7*64-conv3*192-inception(3a)-inception(3b)-inception(4a)-inception(4b)-inception(4c)-inception(4d)-inception(4e)-inception(5a)-inception(5b)-1000

为了评估权重混排对网络分类精度的影响，首先从训练好的网络中提取权重矩阵，然后将这些权重矩阵映射到交叉阵列上.接下来，根据算法 1 实现基于 VOU 的权重混排.随后，将混排后的权重矩阵替换原始权重矩阵，并通过在模拟器上执行网络推理来评估分类精度.

本文采用类似的方法模拟权重模糊化，并评估其对网络分类精度的影响.当网络执行推理时，提取每一层的输入神经元，并将其转换为施加在交叉阵列字线上的输入电压脉冲.基于式(2)和式(3)，计算

忆阻器的电阻漂移及相应权重值的变化，以获得不同时间点的模糊权重矩阵. 然后，将这些权重矩阵替换网络中的原始矩阵并评估分类精度.

6.2 抵御模型偷窃攻击的有效性

实验首先评估在没有正确密钥的情况下，使用混排后权重矩阵的网络的分类精度. 实验中，以两种方式实现权重混排. 一种是对网络中全部层的权重矩阵都进行混排处理. 这种方式被称为完全混排. 另一种是只对单独一层的权重矩阵进行混排处理，称为单层混排. 实验中采用随机生成的密钥来模拟一个不知道密钥的攻击者. 为了进行比较，实验中还模拟了文献[17]中提出的权重行混排方法. 考虑到随机性，为行混排和 VOU 混排各生成了 50 组不同的混排方案. 图 6 显示了采用行混排和 VOU 混排方法分别对目标网络进行完全混排和单层混排后网络的分类精度数据. 箱型图中“All”表示应用完全混排的网络分类精度；而带有层号的数据则表示对该层执行单层混排后网络的分类精度.

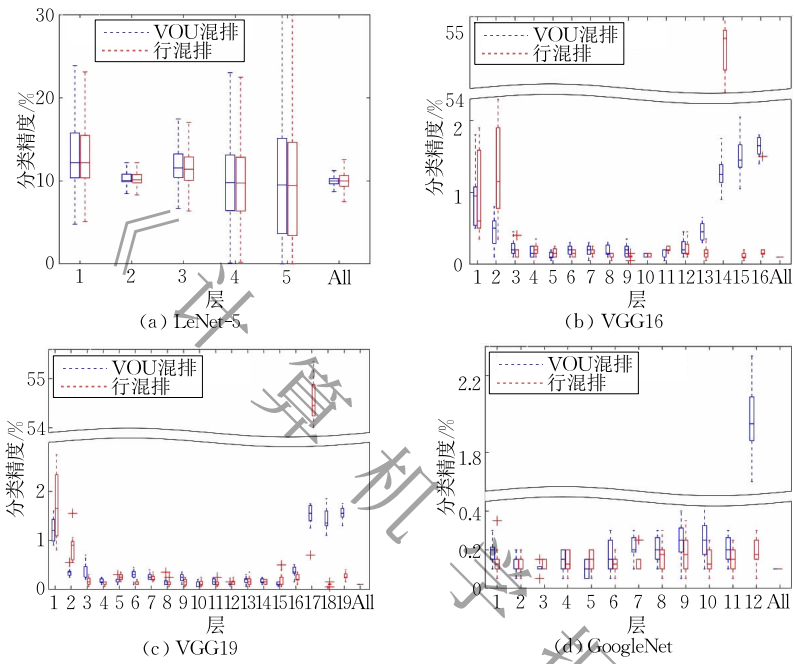


图 6 使用 50 种不同混排方案时完全混排和单层混排的网络分类精度分布数据

6.2.1 完全混排

精度分析：可以看出，对权重矩阵进行完全混排显著降低网络的分类精度. 如图 6(a)所示，平均而言，LeNet-5 的分类精度从 98.8% 降至 11%. LeNet-5 模型对他们来说也不具备实用价值. 没有正确的密钥，攻击者无法借助混排的网络完成任何有意义的工作. 再如图 6(b)、(c)和(d)所示，在对其它三个网络进行的实验中也存在类似的结果. 平均而言，这三个网络的分类精度从 93.5%、89.9% 和 94.1%下降到 0.1%左右. 这也相当于从 ImageNet 数据集中的 1000 个类别中随机猜测一个结果. 因此，在大规模网络上进行权重混排处理具有更好的安全性.

开销分析：如图 6 所示，行粒度下的完全混排处理对网络分类精度的影响与本文提出的 VOU 混排方法相近. 然而，行混排会引入较大的硬件面积和功耗开销，如表 3 所示. 除 LeNet-5 外，三个大型网络中实现行混排的面积开销分别达到 221%、230%和 11.8%；而功耗开销分别为 330%、343%和 17.6%. 这实际上意味着行混排无法承担完全混排的成本. 具体原因在于，行混排为实现输入向量重定向，每个交叉阵列需要 32 个 16 选 1 MUX 和 16 个 1 到 16 的 DEMUX[17]. 然而，与行混排不同，本文方法沿列维度实现基于 VOU 的权重混排. 每个交叉阵列只需要一个行列选择模块和输出重定向模块. 因此，面积和功耗开销远小于行粒度混排. 四个目标网络中，本文方法在 VGG-19 中存在最大的面积和功耗开销，分别为 9.92%和 15.7%. 至于密钥存储器的开销，类似的数据可以在表 3 中找到. 行混排中所需的密钥空间是本文方法的 48 倍. 相应地，他们需要 43.6%的面积开销和 16.9%的功耗开销，而本文方案的开销分别只有 1.81%和 0.7%.

表 3　输出重定向模块和密钥存储单元的面积、功耗开销

网络名称	输出重定向模块								密钥存储							
	面积开销/%				功耗开销/%				面积开销/%				功耗开销/%			
	行混排		VOU 混排		行混排		VOU 混排		行混排		VOU 混排		行混排		VOU 混排	
	FO	SO	FO	SO	FO	SO	FO	SO	FO	SO	FO	SO	FO	SO	FO	SO
LeNet-5	0.092	0.003	0.004	1e-4	0.136	0.005	0.006	2e-4	0.017	6e-4	7e-4	2e-5	0.006	2e-4	3e-4	9e-6
VGG16	221	0.94	9.55	0.04	330	1.41	15.1	0.06	41.9	0.71	1.74	0.007	16.3	0.07	0.67	0.003
VGG19	230	0.47	9.92	0.02	343	0.70	15.7	0.03	43.6	0.08	1.81	0.003	16.9	0.04	0.70	0.001
GoogleNet	11.8	0.64	0.51	0.03	17.6	0.95	0.81	0.04	2.2	0.12	0.09	0.005	0.87	0.05	0.03	0.002

注：FO 表示完全混排，SO 表示单层混排

6.2.2　单层混排

精度分析：一个有趣的观察结果是，混排单层权重矩阵也会导致网络分类精度的大幅降低. 如图 6(a)所示，对于小规模网络 LeNet-5，单层混排引起的精度下降与完全混排的精度下降相似，都在 10%左右. 这意味着对于 LeNet-5，可以只在网络的一层中执行权重混排，即可达到与完全混排相同的效果，同时还可以大大节省硬件和功耗开销. 然而，图 6(b)、(c)和(d)表明，对于大规模网络，混排不同的网络层，精度下降具有明显的依赖性. 通常，混排卷积层会导致网络精度低于混排全连接层. 精度降低的差异可以达到一个数量级(例如，conv 层为 0.1%，FC 层为 1.5%). 这可能是因为卷积层的功能是提取特征. 卷积层混排处理后，产生的计算误差对网络分类精度影响较大. 此外，如图 6(b)和(c)所示，混排第一个卷积层(Cov1)对分类精度的影响小于混排其他卷积层. 精度降低的差异也达到了一个数量级.

开销分析：上述讨论表明，选择合适的权重层进行单层混排处理能够以较低的硬件开销实现较大的精度降低. 实验中选择导致网络分类精度下降最为严重的网络层作为评估目标. 表 4 给出了对四个网络进行单层混排时选择的目标层以及在这些层的计算中使用的交叉阵列的数量. 表 3 中给出了所需面积和功耗开销. 对于三个大规模网络，单层混排引入的面积开销仅为 0.04%、0.02%和 0.03%. 这些结果表明，如果设计是面积开销敏感的，可以选择单层混排. 尽管单层行混排也可以实现较低的面积开销. 然而，总的来说，单层行混排的面积开销比本文方法要高一个数量级. 在功耗开销方面，本文方法也比行混排方案的数据要好得多. 本文方法的最大功耗开销为 0.03%，而行混排的最大功耗开销为 0.7%. 在密钥存储器的开销评估中也发现了类似的结果. 无论是在面积还是功耗方面，本文方法都远低于行混排方法.

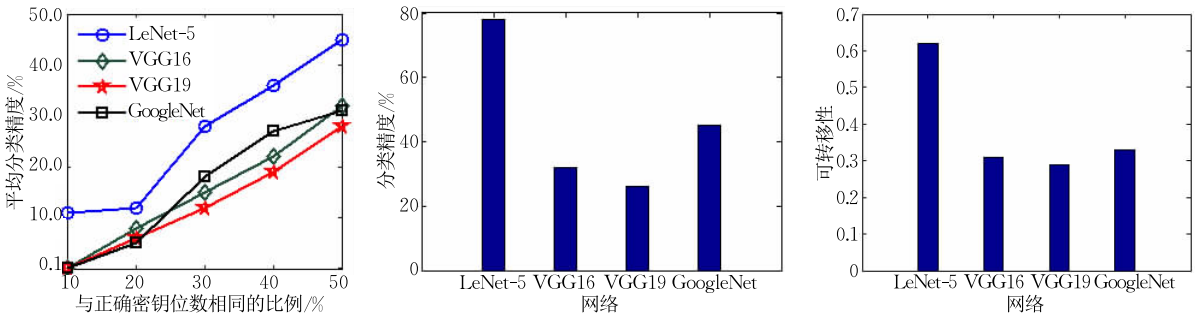
表 4　单层混排的目标层及所需交叉阵列数目

网络名称	目标层及所需交叉阵列数目	
	行粒度混排	VOU 粒度混排
LeNet-5	Cov1，1 crossbar	Cov1，1 crossbar
VGG16	Cov5，144 crossbars	Cov6，288 crossbars
VGG19	Cov7，288 crossbars	Cov5，144 crossbars
GoogleNet	4b，216 crossbars	4b，216 crossbars

6.2.3　防御方法感知的攻击

假定攻击者已经知道加速器采用了权重混排方法保护网络模型，他们可以执行防御方法感知的攻击. 例如，攻击者可以执行基于密钥猜测的攻击. 该攻击试图生成与正确密钥具有相同比特的密钥，并使用它们执行神经网络的计算. 图 7(a)显示了在攻击者能够生成不同比例的与正确密钥有相同比特位时四个网络的分类精度. 可以看出，即使攻击者密钥中有一半比特位与正确密钥相同，四种网络的精度仍均低于 50%，大约是使用正确密钥时网络精度的一半. 请注意，对于用于一个交叉阵列中的 256 字节的密钥，攻击者能够推断出与正确密钥相同的一半比特位的概率非常低，更不用说所有密钥空间的一半比特位了. 显然，如果没有密钥，对手就无法直接使用混排后的神经网络执行任何有用的操作.

另一种防御方法感知的攻击是基于网络重训练的攻击. 攻击者可以将混排后的权值作为初始随机权值，并对网络进行重新训练，以恢复网络的分类精度. 需要注意的是，通常情况下，攻击者没有神经网络模型的完全训练数据集. 因此，实验中从 MNIST 和 ImageNet 数据集随机选取 10%的图像，并使用基于雅可比矩阵的数据集扩充方法[13]生成额外 90%的图像. 采用新生成的训练数据集执行网络重训练. 图 7(b)示出了四个网络在重新训练后的分类精度. 对于小型网络 LeNet-5，重新训练后的精度恢复到 78%，但仍远低于原始网络的精度. 对于其它 3 个大型网络，基于混排权重矩阵的重训练无法将网络精度恢复到可接受的程度，精度均低于 50%，表明本文提出的混排方法对基于重训练的攻击具有较好的防御效果.



(a) 猜测密钥攻击 (b) 网络重训练攻击 (c) 对抗攻击的可转移能力

图 7 防御方法感知的攻击下四种网络的分类精度和对抗攻击下的可转移能力

6.2.4 防御对抗攻击

攻击者窃取模型参数后,可以实施白盒对抗攻击,训练自己的对抗样本.本文通过实验验证了提出的混排方法对此类攻击的防御能力.实验中采用 I-FGSM 方法[39]针对混排后的神经网络生成 1000 个对抗样本(无密钥),并使用它们攻击神经网络模型.采用文献[40-41]中使用的可转移性(Transferability)来评估对抗攻击的有效性.可转移性定义为成功攻击受害者模型的对抗样本数与对抗样本总数的比率.图 7(c)给出了攻击四个网络时的可转移性数据.可以看出,对于 3 个大型网络,可转移性非常低(均低于 35%),即使对于小型网络 LeNet-5,可转移性也仅达到 65%.因此,可以认为本文的混排方法能够有效地抵御对抗攻击.

6.2.5 延迟开销分析

如前述,以操作单元为粒度进行交叉阵列的计算能平衡忆阻器交叉阵列的性能和精度.这种执行模式下,操作单元(OU)被逐个激活,每个 OU 计算输出神经元的部分结果.在累加部分结果之后,可以得到神经元最终的输出值.

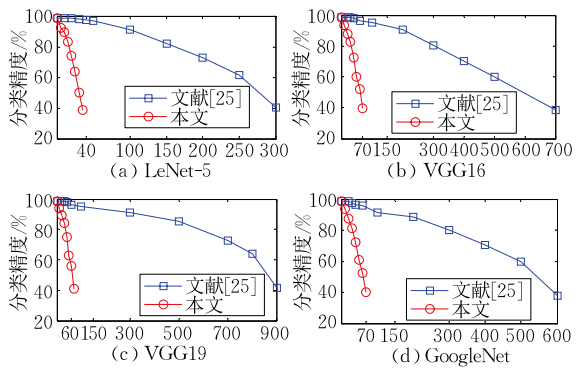
本文中忆阻器交叉阵列的计算同样是基于操作单元粒度.本文考虑精度需求,将交叉阵列划分为许多虚单元.这些虚单元的大小与前述物理操作单元一样,不同之处在于每个虚单元(VOU)包含的是不相邻的字线.因此,在基于操作单元的计算模式下,本文方法不会引入额外的延迟.行混排方法[17]同样考虑了这个问题.他们也是按照操作单元包含的字线数量来分批次实现交叉阵列的计算,但他们默认是激活交叉阵列中全部的位线.在这种情况下,本文方法的延迟会是行混排的数倍.例如,本文以 8 条字线和 8 条位线组成一个 VOU 将一个 128×128 的交叉阵列划分为 256 个 VOU,一共需 256 个单位时间完成整个交叉阵列的计算.行混排[17]则是以 8 条字线为单位将交叉阵列的计算划分为 16 组,需 16 个单位时间完成计算.但需要指出的是,文献[17]这种只激活部分字线,但激活全部位线的方式会降低交叉阵列的计算精度.若文献[17]也采用 8 字线 8 位线的激活方式,则延迟与本文方法相同.更为重要的是,本文方法在面积和功耗开销方面,远远小于已有工作的开销.如表 3 所示,本文方法的面积和功耗开销相比较于已有方法(包括文献[17])降低了一到两个数量级.这些评估数据可以表明本文方法的优势.

6.3 抵御模型复制攻击的有效性

6.3.1 权重模糊化速度

假设检测到恶意查询并且加速器切换到自毁模式.在这种模式下,用编程电压替换施加在交叉阵列字线上的电压脉冲.因此,电压脉冲的幅值从 0.3 V 增加到 1.8 V,持续时间从 100 ns 延长到 1 μs.为了进行比较,还实现了文献[25]中提出的权重模糊化方法.在文献[25]中,反相器介于正负交叉阵列的输入之间.这种设计会导致差分忆阻器的电阻漂移朝相反方向变化,从而加速权重模糊化.然而,它们使用的电压脉冲是交叉阵列正常操作中的电压脉冲.

实验中模拟了权重模糊化进程并获得分类精度.采用导致网络精度下降到不可接受程度的输入-输出对数目来表示权重模糊化的速度.如图 8 所示,本文提出的方法可以有效加快权重模糊化的速度.相应的,网络模型的分类精度迅速下降.一般来说,在施加 70 个输入输出对后,网络分类精度就降低到



(a) LeNet-5 (b) VGG16 (c) VGG19 (d) GoogleNet

图 8 权重模糊化后网络精度随输入输出对数目变化情况

40%以下. 由于此时获取的输入输出对的数量较少，攻击者无法训练自己的高精度替代模型. 相反，使用文献[25]中提出的方案进行权重模糊化的速度明显较慢. 如图 8(a)所示，对于 LeNet-5，在应用 300 个输入输出对后，其精度才降低到 40%. 又如图 8(b)、(c)和(d)所示，对于其它三个网络，降低到该精度需要应用的输入输出对数量分别约为 700、900 和 600. 随着输入输出对数量的增加，攻击者就有更好的机会成功地训练自己的替代模型.

6.3.2　防御方法感知的攻击

实验中还假设攻击者知道权重模糊化的防御方案，并采取对策延迟权重模糊化过程. 例如，攻击者可以按随机顺序重新排列输入特征图. 甚至可以发送一对相反的输入特征图，以部分恢复忆阻器的阻值并延迟权重模糊化进程. 图 9 给出了实施上述两种攻击时网络模型分类精度的变化情况. 图 9 中，有两个观察发现. 首先，在加速器进入自毁模式后，随机排列输入特征图的顺序对权值模糊化的影响很小. 对于所有四个网络，在 90 个输入输出对内，分类精度均降低到 40%以下. 其次，使用一对相反的输入特征图可以延缓权重模糊化的速度，同时这也会延迟网络精度降低到不可接受的程度的时间. 如图 9 所示，对于 GoogleNet，在向加速器发送大约 236 个输入输出对后，分类精度下降到 40%以下. 然而，使用这种数量的输入输出对，攻击者仍然很难训练自己的神经网络模型以达到理想的精度. 这证明了本文防御方法的有效性.

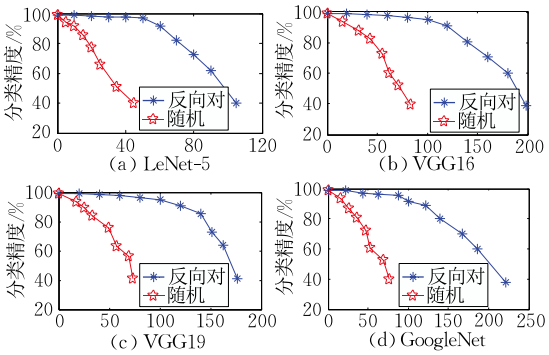


图 9　抵御防御方法感知的攻击的有效性

6.3.3　防御对抗攻击

使用获取的输入输出对，攻击者还可以训练自己的替代模型并发起黑盒对抗攻击. 实验中将输入输出对添加到雅可比增广数据集中. 然后，使用新的数据集对网络进行训练，生成对抗样本，并对原始神经网络模型发起对抗攻击. 图 10 展示了替代网络的精度和对抗攻击的可转移能力. 如图 10(a)所示，由于攻击的黑盒性质，所有替代网络的精度都非常低. 如果不了解数据集和详细的网络参数，攻击者就无法通过训练网络来达到理想的精度. 所以，如图 10(b)所示，当对网络模型执行对抗攻击时，根据替代网络生成的对抗样本是无用的. 综上，可以认为本文的权重模糊化方法能够有效地抵御模型复制攻击.

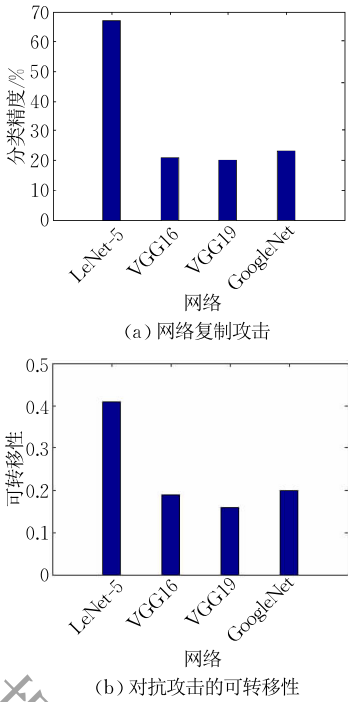


图 10　针对对抗攻击的防御效果

7　相关工作

为了保护存储在非易失性存储器中的数据免受模型偷窃攻击，一些研究人员提出了加密/解密方案. 例如，i-NVMM 仅加密冷页，而将热页保留为明文格式，以减少性能开销[42]. DEUCE 有选择地加密缓存行中修改的数据，以减少写操作导致的设备损耗[43]. 然而，这类方法不适用于基于忆阻器的神经网络加速器，因为网络中的所有权重都涉及到神经网络的计算. 对权重矩阵执行完全加/解密操作将带来巨大的性能开销. Cai 等人[16]提出了部分加/解密方案. 但是，性能开销仍然存在. 他们的方案也降低了设备的耐久性，并导致可靠性问题.

一些研究人员提出混排权重矩阵实现无需加/解密的模型保护. Zou 等人[17]提出沿行维度混排正负交叉阵列之间的行连接. Huang 等人[11]提出设计

一个基于 SRAM 的转置矩阵来混排权重行.然而,由于交叉阵列中行的数目较多,支持输入神经元重定向的硬件设计非常复杂,并且会带来较大的面积和功耗开销.作者曾提出基于 VOU 的权重混排方法以抵御模型偷窃攻击[44].但文献[44]没有详细评估 VOU 混排方法对于不同攻击方法的防御能力.同时,也没有考虑如何抵御模型复制攻击.还有一些研究人员借助芯片制造过程中的工艺偏差(Process variation)及这种偏差的不可克隆性质,将网络模型与加速器设备进行绑定来实现神经网络模型的保护[45].但此类方法破坏了神经网络模型的通用性.且需对每个单独的加速器设备进行网络参数的调节,费时费力.

在服务器端,在线检测 OOD 查询是识别模型复制攻击的常用方法.例如,PRADA 计算用户查询的时间间隔,并在间隔违反正态分布时发出警报[18].在文献[19]和[32]中,作者不仅检测出恶意查询,而且还提出了防御对策.通过在神经网络模型的预测输出中加入噪声,他们试图误导对手训练的复制模型.然而,服务器端防御方案通常需要执行大量的计算,这对于资源受限的用户端神经网络加速器来说是昂贵的.

对于神经网络加速器抵御模型复制攻击来说,直接破坏神经网络模型可能比干扰网络输出效果更优,同时也无需执行高成本计算.在文献[25]中,作者建议利用电阻漂移导致的忆阻器漂移效应来破坏部署在设备(如无人机)上的神经网络模型.然而,他们的方案依赖于自然的电阻漂移来模糊化权重,这通常是一个长时间变化的过程.攻击者有充足的时间收集足够多的输入输出对来训练自己的替代模型.

8 结 论

本文提出了防御方法以阻止针对部署在忆阻器平台上的神经网络模型实施的模型偷窃攻击和模型复制攻击.通过在 VOU 粒度执行随机权重混排处理,攻击者无法在没有密钥的情况下正确执行神经网络的计算.另一方面,本文提出在连续检测到恶意查询后,将加速器切换到自毁模式.在这种模式下,交叉阵列的输入电压脉冲被具有更高幅值和更长持续时间的编程电压脉冲所取代.因此,存储在忆阻器中的权重值很快模糊化,偏离其原始值,网络精度在很短的时间内下降到不可接受的程度.实验结果表明,本文方法能够在低面积开销和低功耗开销的条件下保证高安全性.

参 考 文 献

[1] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks//Proceedings of the Neural Information Processing Systems (NIPS). Lake Tahoe, USA, 2012: 1097-1105

[2] Chen G, Parada C, Heigold G. Small-footprint keyword spotting using deep neural networks//Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Florence, Italy, 2014: 4087-4091

[3] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks//Proceedings of the Neural Information Processing Systems (NIPS). Montreal, Canada, 2014: 3104-3112

[4] Osia S A, Shamsabadi A S, Taheri A, et al. Private and scalable personal data analytics using hybrid edge-to-cloud deep learning. Computer, 2018, 51(5): 42-49

[5] Yang Tzu-Hsien, Cheng Hsiang-Yun, Yang Chia-Lin, et al. Sparse ReRAM engine: Joint exploration of activation and weight sparsity in compressed neural networks//Proceedings of the 46th International Symposium on Computer Architecture. Phoenix, USA, 2019: 236-249

[6] Prezioso M, Merrikh-Bayat F, Hoskins B D, et al. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. Nature, 2015, 521(7550): 61-64

[7] Shafiee A, Nag A, Muralimanohar N, et al. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars//Proceedings of the 43rd International Symposium on Computer Architecture. Seoul, Korea, 2016: 14-26

[8] Zhu Zhenhua, Sun Hanbo, Lin Yujun, et al. A configurable multi-precision CNN computing framework based on single bit RRAM//Proceedings of the 56th ACM/IEEE Design Automation Conference (DAC). Las Vegas, USA, 2019: 1-6

[9] Ghosh S, Khan M N I, De A, Jang J-W. Security and privacy threats to on-chip non-volatile memories and countermeasures //Proceedings of the 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). Austin, USA, 2016: 1-6

[10] Hu Xing, Liang Ling, Li Shuangchen, et al. Deep-Sniffer: A DNN model extraction framework based on learning architectural hints//Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems. Online Virtual Conference, 2020: 385-399

[11] Huang Shanshi, Peng Xiaochen, Jiang Hongwu, et al. New security challenges on machine learning inference engine: Chip cloning and model reverse engineering. https://doi.org/10.48550/arXiv.2003.09739, 2020, 3, 21

[12] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, et al. Intriguing properties of neural networks. https://doi.org/10.48550/arXiv.1312.6199, 2013, 12, 21

[13] Papernot N, McDaniel P, Goodfellow I, et al. Practical black-box attacks against machine learning//Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. Dubai, The United Arab Emirates, 2017: 506-519

[14] Tramèr F, Zhang F, Juels A, et al. Stealing machine learning models via prediction APIs//Proceedings of the 25th USENIX Security Symposium. Austin, USA, 2016: 601-618

[15] Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. https://doi.org/10.48550/arXiv.1412.6572, 2014, 12, 20

[16] Cai Yi, Chen Xiaoming, Tian Lu, et al. Enabling secure in-memory neural network computing by sparse fast gradient encryption//Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). Westminster, USA, 2019: 1-8

[17] Zou Minhui, Zhu Zhenhua, Cai Yi, et al. Security enhancement for RRAM computing system through obfuscating crossbar row connections//Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE). Grenoble, France, 2020: 466-471

[18] Juuti M, Szyller S, Marchal S, Asokan N. PRADA: Protecting against DNN model stealing attacks//Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS&P). Stockholm, Sweden, 2019: 512-527

[19] Orekondy T, Schiele B, Fritz M. Prediction poisoning: Towards defenses against DNN model stealing attacks//Proceedings of the International Conference on Learning Representations (ICLR). Addis, ETHIOPIA, 2020: 1-17

[20] Hendrycks D, Gimpel K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. https://doi.org/10.48550/arXiv.1610.02136, 2016, 10, 7

[21] Chen Wei-Hao, Li Kai-Xiang, Lin Wei-Yu, et al. A 65 nm 1 Mb nonvolatile computing in memory ReRAM macro with sub-16 ns multiply-and-accumulate for binary DNN AI edge processors//Proceedings of the 2018 IEEE International Solid State Circuits Conference (ISSCC). San Francisco, USA, 2018: 494-496

[22] Lin Meng-Yao, Cheng Hsiang-Yun, Lin Wei-Ting, et al. DL-RSIM: A simulation framework to enable reliable ReRAM-based accelerators for deep learning//Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). San Diego, USA, 2018: 1-8

[23] Chang Ting, Jo Sung-Hyun, Lu Wei. Short-term memory to long-term memory transition in a Nanoscale memristor. ACS Nano, 2011, 5(9): 7669-7676

[24] Yan Bonan, Yang Jianhua, Wu Qing, et al. A closed-loop design to enhance weight stability of memristor based neural network chips//Proceedings of the 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). Irvine, USA, 2017: 541-548

[25] Yang Chaofei, Liu Beiye, Li Hai, et al. Thwarting replication attack against memristor-based neuromorphic computing system. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39(10): 2192-2205

[26] Batina L, Bhasin S, Jap D, Picek S. CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel//Proceedings of the 28th USENIX Security Symposium (USENIX Security 19). Santa Clara, USA, 2019: 515-532

[27] Wei Lingxiao, Luo Bo, Li Yu, et al. I know what you see: Power side-channel attack on convolutional neural network accelerators//Proceedings of the 34th Annual Computer Security Applications Conference. San Juan, USA, 2018: 393-406

[28] Clements J, Lao Y. Hardware Trojan attacks on neural networks. https://doi.org/10.48550/arXiv.1806.05768, 2018, 6, 14

[29] Knuth D. The art of programming. ITNow, 2011, 53(4): 18-19

[30] Halderman J A, Schoen S D, Heninger N, et al. Lest we remember: Cold boot attacks on encryption keys//Proceedings of the 17th USENIX Security Symposium. San Jose, USA, 2008: 45-60

[31] Buchanan W J, Li S, Asif R. Lightweight cryptography methods. Journal of Cyber Security Technology, 2018, 1(3): 187-201

[32] Kariyappa S, Qureshi M K. Defending against model stealing attacks with adaptive misinformation//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Seattle, USA, 2020: 770-778

[33] Kesarwani M, Mukhoty B, Arya V, Mehta S. Model extraction warning in MLaaS paradigm//Proceedings of the 34th Annual Computer Security Applications Conference. San Juan, USA, 2018: 371-380

[34] Hu Miao, Li Hai, Pino R E. Fast statistical model of TiO_2 thin-film memristor and design implication//Proceedings of the 2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). San Jose, USA, 2011: 345-352

[35] Arafin M T, Qu G. Memristors for secret sharing-based lightweight authentication. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2018, 22(12): 2671-2683

[36] Gao Ligang, Chen Pai-Yu, Yu Shimeng. Programming protocol optimization for analog weight tuning in resistive memories. IEEE Electron Device Letters, 2015, 36(11): 1157-1159

[37] Chen Pai-Yu, Peng Xiaochen, Yu Shimeng. NeuroSim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures//Proceedings of the IEEE International Electron Devices Meeting (IEDM). San Francisco, USA, 2017: 1-6

[38] Muralimanohar N, Balasubramonian R, Jouppi N P. CACTI 6.0: A tool to model large caches. HP Laboratories, 27-28, 2009

[39] Kurakin A, Goodfellow I, Bengio S. Adversarial examples in the physical world. https://doi.org/10.48550/arXiv.1607.02533, 2016, 7, 8

[40] Goodfellow I, McDaniel P, Papernot N. Making machine learning robust against adversarial inputs. Communications of the ACM, 2018, 61(7): 56-66

[41] Liu Yanpei, Chen Xinyun, Liu Chang, Song Dawn. Delving into transferable adversarial examples and black-box attacks. https://doi.org/10.48550/arXiv.1611.02770, 2016, 11, 8

[42] Chhabra S, Solihin Y. i-NVMM: A secure non-volatile main memory system with incremental encryption//Proceedings of the 38th Annual in Ternational Symposium on Computer Architecture(ISCA). San Jose, USA, 2011: 177-188

[43] Young V, Nair P J, Qureshi M K. DEUCE: Write-efficient encryption for non-volatile memories//Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems. Istanbul, Turkey, 2015: 33-44

[44] Wang Yuhang, Jin Song, Li Tao. A low cost weight obfuscation scheme for security enhancement of ReRAM based neural network accelerators//Proceedings of the ACM/IEEE Asia and South Pacific Design Automation Conference (ASP-DAC). Online Virtual Conference, 2021: 499-504

[45] Huang Shanshi, Peng Shanshi, Jiang Hongwu, et al. Exploiting protect machine learning inference engine from chip cloning//Proceedings of the IEEE International Symposium on Circuits and Systems(ISCAS). Daegu, Korea, 2021: 1-6

JIN Song, Ph.D., associate professor. His research interests include security on deep learning, hardware security and reliability-related designs.

WANG Yu-Hang, M. S. candidate. His main research include security on deep learning, hardware security and reliability-related designs.

Background

This paper focuses on protecting the neural network models which are deployed on the memristor platforms. The proposed work belongs to the hardware security domain.

Although the memristor platforms have the advantage of accelerating the computations of the neural networks, the non-volatile characteristics of the memristors introduce some new security vulnerabilities. Since the model's weights will still be maintained in the memristor crossbar after the chip is powered off, the adversarial can conduct the model stealing attack to retrieve the weights of the well trained neural networks. When the accelerators are used as the user side device, the adversarial can also perform the model replicating attack to train their replicated model. Above mentioned attacks can result in violation of the intellectual property or profit loss for the model owners.

To resist a model stealing attack, an intuitive idea is to encrypt the weights stored in the ReRAM, decrypt them when the NN computations start and re-encrypt these weights after the computations are completed. However, implementing encryption/decryption at runtime introduces large hardware overhead and degrades execution performance of the NN model. Moreover, repeatedly encrypting/decrypting some weights means repeated programming on some memristors in the ReRAM which will degrade device endurance and induce reliability issues. Some researchers proposed to shuffle the weight matrices along the row dimension. Nevertheless, there are a large number of rows in a crossbar(e.g., at least 128) and overhead to route the input neurons to the appropriate wordlines of the crossbar are tremendous.

In regarding to resist the model replicating attack, on-line detection of the malicious query and perturbation of the prediction result are commonly used in the server applications such as machine learning as a service (MLaaS). The existing schemes have high computation complexity and only work well when the adversaries have the limited query budget. However, for the resource constrained user end devices, computation intensive defense method is not preferable. Moreover, when the NN model is deployed on the user end, the adversaries who can access the devices have unlimited query budget theoretically. They can obtain the sufficient input-output pairs if they have enough patience, and use these pairs to train their own alternative model.

Targeted ReRAM based hardware accelerators, in this paper we propose a defense scheme to thwart the above mentioned security threats and protect the neural network model. The main contributions of this paper include:

(1) We propose weight obfuscation performed in VOU granularity to resist the model stealing attack. Our scheme has enough large permutation space which prevents the adversaries from restoring the obfuscated weight matrices to the original ones.

(2) We propose weight blurring to implement self-destruction of the NN model to resist the model replicating attack. By multiplexing the programming voltage pulses of the ReRAM crossbar, we can accelerate the memristance drift of the memristors. As a result, the weights blur quickly and the prediction accuracy of the NN model degrades to an unacceptable level in a very short period. This makes the adversaries cannot conduct the model replicating attack further.