一种用于智能汽车的硬件友好对抗样本在线防御方法

范仁昊[1]，庞 猛[1]，王明羽[2]，李明钊[3]，张悠慧[1]，李兆麟[1]

（1. 清华大学 计算机科学与技术系，北京 100084；2. 中山大学 微电子科学与技术学院，广州 510275；
3. 无锡太昊慧芯微电子有限公司，江苏，无锡 214063）

摘　要：提出了一种针对对抗样本攻击的硬件友好的在线防御方法。该方法由三部分组成，一个使用自编码器作为检测器来逼近自然样本流形分布的广谱检测算法，一个适用于深度神经网络（Deep Neural Network，DNN）瓷片加速器架构的高效层调度方案以减少数据访问开销，以及一个软硬件协同设计方法以达到检测精度和算法开销的平衡。试验表明，基于自编码器的广谱在线检测方法能够达到与已有算法相当的检测精度，提出的层调度方案将推理网络与检测器耦合的联合网络的DRAM访问量减少了43%，进而降低了能耗，提高了吞吐量。此外，软硬件协同设计方法在保证检测精度不降低的情况下，将耦合网络的能耗和运行时间分别降低了58%和54%。

关键词：神经网络；对抗样本攻击；在线防御；软硬件协同设计

中图分类号：TP183　　**文献标志码：**A　　**DOI：**10.3969/j.issn.2095–1469.2022.05.03

Hardware-Friendly Online Defense Against Adversarial Attacks for Smart Cars

FAN Renhao[1]，PANG Meng[1]，WANG Mingyu[2]，LI Mingzhao[3]，ZHANG Youhui[1]，LI Zhaolin[1]

（1. Department of Computer Science and Technology，Tsinghua University，Beijing 100084，China；
2. School of Microelectronics Science and Technology，Sun Yat-sen University，Guangzhou 510275，China；
3. WUXI TAIHAOHUIXIN Microelectronics Corporation，Wuxi 214063，Jiangsu，China）

Abstract: This paper proposes a hardware-friendly online defense scheme called Auto-defense against adversarial attacks. Auto-defense is composed of a broad-spectrum detection algorithm which uses autoencoders to approximate manifolds of natural samples, a tiled DNN accelerator architecture with an efficient layer scheduling scheme to reduce data access overhead and a hardware/software co-design method to reach the balance of overhead and detection accuracy. The experimental evaluation shows that the broad-spectrum detection method achieves the state-of-the-art accuracy. The proposed layer scheduling scheme reduces the amount of DRAM access of the DNN coupled with detectors by more than 43%, thus resulting in lower energy consumption and higher throughput. Furthermore, the co-design method reduces the energy and execution time of the coupled network by 58% and 54% respectively without accuracy degradation.

Keywords: neural netwoks; adversarial attacks; online defense; software/hardware co-design

收稿日期：2022–03–25　**改稿日期：**2022–04–27

基金项目：国家重点研发计划项目（2020YFB1600202）

参考文献引用格式：

范仁昊，庞猛，王明羽，等.一种用于智能汽车的硬件友好对抗样本在线防御方法[J].汽车工程学报，2022，12(5)：583–592.

FAN Renhao，PANG Meng，WANG Mingyu，et al.Hardware-Friendly Online Defense Against Adversarial Attacks for Smart Cars [J].Chinese Journal of Automotive Engineering，2022，12(5)：583–592.(in Chinese)

近年来，随着计算机算力的快速增长和数据集规模的扩大，DNN已在人脸识别[14]、自动驾驶[1]等智能任务中取得压倒性优势。但是现有工作表明，DNN具有高维稀疏性质和不可解释性，导致DNN很容易受到对抗样本攻击[5, 8, 15]，即通过添加人眼不可见的微小扰动，输入图片就会被错误分类，这给DNN的广泛使用带来了巨大的安全风险。特别对于自动驾驶等涉及人身安全的应用领域，微小的错误都有可能导致严重的后果，因此，研究针对深度神经网络攻击的防御方法至关重要。

许多研究工作已经提出了针对神经网络对抗样本攻击的防御方法，它们可以分为两类：模型防御方法和样本检测方法。模型防御方法通过增强推理网络模型的鲁棒性来增加构建对抗样本的难度，而在线检测方法利用样本和推理网络隐藏层数据的信息来分辨输入样本是否属于对抗样本。由于增强推理网络的鲁棒性通常会导致推理网络的精度下降，样本检测方法是更好的解决方案。现有的样本检测方法已经取得了很高的检测精度，例如MagNet[18]、NIC[19]和Feature Squeezing[25]。但是，这些方法仅在算法层面解决了对抗样本的在线检测问题，当考虑到DNN模型的实际运行环境时，已有的样本检测方法将会面临两个新的挑战。

首先，为了提高实际运行场景中的能效和吞吐量，DNN通常运行在专用的神经网络加速器上，如Eyeriss[3]和Tangram[10]。这些神经网络加速器使用特别设计的计算阵列来加速DNN中常用的算子，如矩阵乘法、卷积和激活函数。然而，现有的基于支持向量机[19, 23]和空间平滑降噪[25]的在线防御算法包含大量不规则的计算操作，因而很难在神经网络加速器中得到良好的支持。其次，现有工作更注重优先提高检测精度，而忽略了对运行时开销的评估。一些样本检测方法的计算量甚至达到推理网络的数倍，这在实际应用中是无法接受的。因此，如何选择或设计防御算法以很好地适应神经网络加速器的硬件计算能力（简称为硬件友好）是实现检测精度和运行时开销权衡的关键。

针对上述问题，本文提出了一种硬件友好的对抗样本在线防御方法，该方法由基于自编码器（Auto Encoder）的广谱检测算法、适用于瓷片架构神经网络加速器的层调度方法和相应的软硬件协同设计方法。硬件友好的解决方案可以使神经网络加速器统一地支持推理网络和检测网络，而不需要引入异构的功能模块，从而使加速器设计更加实用、可靠和简单。

具体来说，本方法包括以下贡献：

（1）基于自编码器的在线检测算法。自编码器是一种简单的神经网络，可以从推理网络的隐藏层激活值（Activations）中提取自然样本激活值的流形分布。自编码器不仅可以在不引入对抗样本的条件下拟合自然样本的流形分布，而且能够高效地映射到神经网络加速器，从而使其成为一种广谱和硬件友好的算法。

（2）一种适用于瓷片架构神经网络加速器的高效层调度方案，能够减少数据访问开销。本文使用可扩展瓷片架构的神经网络加速器设计方案Tangram[10]作为基准，提出了一种新的层调度方法来减少推断–检测联合网络的数据访问开销。

（3）能够最小化运行时开销的软硬件协同设计方案。本文对一组检测器集合使用剪枝–搜索的融合方法，以优化检测网络配置，在不损失检测精度的前提下最小化推断–检测联合网络的运行时开销。

1 背景

1.1 对抗样本攻击

对抗样本可以通过在自然样本上添加大小有限的特定扰动来生成。形式上，DNN分类器可以描述为函数f：$X \to Y$，其中X表示输入空间，Y表示类别集合。对于标签为$c \in Y$的自然样本$x \in X$和阈值ϵ，对抗样本x'满足$f(x') \neq f(x) \wedge \Delta(x, x') < \epsilon$，其中$\Delta(x, x')$是$x$和$x'$之间的距离函数。在现有工作中，距离函数通常定义为两个输入样本的0–范数、2–范数或∞–范数。

接下来，简要描述一些对抗样本攻击作为本文

所提算法的检测目标。它们的范数和生成函数在表1中展示。在表1中，x，x'，δ 和 c 表示原始样本、对抗样本、扰动和真实类别。$f(\cdot)$表示推理网络，$L(\cdot,\cdot)$表示损失函数。快速梯度符号方法（Fast Gradient Sign Method，FGSM）[8] 将 x 的每个像素向损失函数的梯度方向移动一小段固定距离来增加损失函数的值，是一种无穷范数攻击方法，而快速梯度方法（Fast Gradient Method，FGM）是FGSM的二范数版本。基本迭代方法（Basic Iteration Method，BIM）[15] 是FGSM或FGM的迭代版本，通过多步移动在原始样本的 ϵ–邻域中找到更好的对抗样本。C&W攻击 [5] 将对抗样本攻击公式转化为优化问题，并定义正则化函数 $g(\cdot)$进行扰动大小和效果的权衡。

表1　一些对抗样本攻击算法

攻击算法	范数	生成函数
FGSM	∞	$x'=x+\epsilon\cdot\mathrm{sign}\left(\nabla L\left(f(x),\ c\right)\right)$
FGM	2	$x'=x+\epsilon\cdot\nabla L\left(f(x),\ c\right)$
BIM	2，∞	$x'_{i+1}=\mathrm{clip}_{\epsilon}\left(x'_i+\alpha\cdot\mathrm{sign}\left(\nabla L\left(f(x),\ c\right)\right)\right)$
C&W攻击	0，2，∞	$\min\|\|\delta\|\|+\alpha\cdot g(x+\delta),\ s.t.\,x+\epsilon\in[0,\ 1]^n$

1.2　现有对抗样本检测算法与架构

已有工作从两个维度尝试解决对抗样本的在线检测问题。其一是设计能够识别对抗样本的算法，其二是设计能够同时支持神经网络算法和在线检测算法运行的神经网络加速器架构。

现有的样本检测算法从作用原理上可以分为两类方法：检测器方法和预测不一致方法。

检测器方法构造各种检测器，从输入样本或推理网络隐藏层激活值中提取攻击特征，从而区分自然样本和对抗样本。检测器通常基于统计模型或机器学习方法构建。例如，Deepfense [21] 通过微调推理网络的参数，使某个隐藏层的激活值服从混合高斯分布模型，如果样本离对应类别的分布中心太远，就会被认为是对抗样本。NIC [19] 使用单分类支持向量机在推理网络每个隐藏层处进行异常值检测，并认为离群值（Outlier）是对抗样本。Deep Neural Rejection [23] 在推理网络靠后的某个隐藏层使用多类别支持向量机进行分类，得到每个类别的置信度，如果所有类别置信度都小于给定阈值，则该样本被判定为对抗样本。

预测不一致方法利用降噪器去除对抗样本扰动，然后测量降噪前后推理网络分类结果之间的差异。Feature Squeezing [25] 对样本进行颜色深度量化和空间平滑降噪，而MagNet [18] 将输入样本通过自编码器实现降噪。事实上，所有模型增强方法都可以进一步转换为预测不一致方法。利用原始模型获得较为准确的分类结果，并运行增强后的模型进行不一致检查，就可以实现样本检测。然而，预测不一致的方法需要运行多个模型或运行多次模型，从而导致成倍的运行时开销。

由于对抗样本检测算法大多使用SVM、随机森林等传统机器学习方法，而这些方法很难在专为神经网络加速而设计的加速器架构上直接运行，所以支持对抗样本检测的神经网络加速器架构的设计重点在于如何支持这些复杂的检测算法。

DNNGuard [28] 是一种异构的DNN加速器架构，该架构将DNN加速器与CPU内核紧耦合到一个芯片中，用DNN加速器运行推理网络，而用CPU内核运行NIC、Feature Squeezing等检测算法，并设计了专用的数据通路和缓存来实现高效的数据传输，可以实现DNN和检测算法的协同执行。然而，试验结果表明，运行在CPU上的检测算法速度慢于部分推理网络（如GoogLeNet和AlexNet），从而会影响推理网络的运行速度。这种异构的设计方案也显著增大了神经网络加速器的设计复杂性和系统开销。

因此，本文的设计思路是设计一种基于自编码器的检测算法。由于自编码器是一种特殊的神经网络，可以在现有的神经网络加速器上运行，而无需额外的硬件支持。同时，由推理网络和多个检测网络耦合在一起形成的多分支复杂神经网络，其在加速器架构上的映射和调度，也有很大的软硬件协同设计空间。可以通过优化映射和调度方案，实现一个硬件友好的对抗样本在线检测方案。

1.3 瓷片架构和数据流调度

为了实现高性能和高能效，许多新的神经网络硬件加速器架构被提出[4, 6, 12]，这些架构在内部数据流上有所不同[26]。如今，规模越来越大的神经网络具有更高的计算和内存要求，导致芯片上集成越来越多的计算和存储资源[27]，这也使瓷片架构的神经网络加速器成为高度可扩展的解决方案[2, 9]。

由于瓷片架构加速器的参数探索空间很大，寻找与之相适应的数据流映射方案成为关键。这方面的一项最先进的工作是Tangram[10]，它提出了对层内并行性和层间流水线的优化，能够找到片上资源映射的最优策略。

在对抗样本攻击的在线防御场景中，推理网络和检测网络的共存带来了更多的数据依赖，使神经网络各层组成的有向无环图更加复杂。在本文提出的在线检测算法（图1）中，多个检测器依赖于推理网络中相应的隐藏层，而所有检测器的输出作为联合分类器的输入。这增加了上述资源映射方案搜索的难度。尽管Tangram已经为一般的神经网络提出了一个最优的数据流方案，但对于这个应用场景来说它仍然有很大的改进空间。基于Tangram的工作，本文将提出并评估一种拥有新的层调度方法的瓷片架构加速器，用于在线防御对抗样本攻击。

2 系统设计与实现

2.1 基于自编码器的检测算法

2.1.1 动机

首先解释对抗样本的产生机制。现有工作[20]表明，所有高维空间的自然样本都位于低维流形中。设样本空间的维数为 n，流形空间的维数为 k，那么有 $k \ll n$。理想情况下，稳健的分类器应该生成垂直于样本流形分布的分类边界。然而，Dimpled Manifold Model[24]表明DNN倾向于生成带“酒窝”的分类边界，这些边界紧紧贴合于样本所在流形的分布，只在样本点附近产生微小的凸起或凹陷，以确保分类的正确性。因此，将自然样本沿着垂直于流形的方向移动一小步以越过分类边界，就能构建出对抗样本。

相应地，这也为对抗样本的检测提供了一个新的方向。只要能够拟合出 k 维流形，就能利用样本到流形的距离判断样本是否属于对抗样本。自编码器是学习流形分布的一种较优选择。自编码器是由编码器和解码器组成的简单神经网络，它尝试用少量隐层神经元提取压缩特征，并在输出端重现输入样本。对于编码器函数 e：$R^n \to R^k$ 和解码器函数 d：$R^k \to R^n$（$n \ll k$），自编码器定义如下的损失函数（也称为重建损失）：

$$\mathrm{Loss}(x) = \left|\left| x - d\left(e(x)\right)\right|\right|_2$$

在使用自然样本进行训练后，编码器可以将 n 维输入向量 x 减少为 k 维向量 $e(x)$，然后解码器能够使用 $e(x)$ 中的 k 个分量重新构建输入向量。令 $v_i \in R^k (i = 1 \cdots k)$ 为第 i 个分量为1且所有其他分量为0的向量，则自编码器会将 n 维输入空间投影到由 $\left\{d(v_1),\ d(v_2),\ \cdots,\ d(v_k)\right\}$ 为基向量组成的 k 维子空间，这个 k 维子空间就对应于数据的流形分布。对分布在流形之外的样本（即对抗样本），经过训练的自编码器将输出其在流形上的投影，因此重建损失恰好是样本与流形之间的欧式距离，于是就可以使用重建损失的大小判断样本是否属于对抗样本。

2.1.2 算法设计

本文假设除了输入样本本身之外，输入样本在推理网络某个隐层的激活值向量也分布在一个低维的流形上。用 $f_i(x)$ 表示 x 在推理网络第 i 层的激活值向量，那么，可以训练一个自编码器来拟合这一层激活向量对应的流形分布 $M_i = \{ f_i(x) | x \in D \}$。对神经网络的每个隐藏层，都可以构建相应的自编码器，进而判断样本的属性，这种类型的自编码器称为单层检测器。

此外，当同时考虑多个隐藏层时，本文认为自然样本在推理网络多个隐藏层的激活向量的拼接向量也应该分布在高维向量空间的一个子空间中。也就是说，对于某个层编号集合 $\{i_1, i_2, \cdots, i_k\}$，拼

接向量 $cat\left(f_{i_1}(x),\ f_{i_2}(x),\ \ldots,\ f_{i_k}(x)\right)$ 位于高维联合空间 $R^{n_1}\times R^{n_2}\times\cdots\times R^{n_k}$ 的低维子空间中。然而，对抗样本对应的拼接向量分布的区域位于该子空间之外。这是因为，对于自然样本，所有隐层的激活值在它们自己的流形分布中属于同一类别（即该样本的真实类别）的区域；但对于对抗样本，不同层的激活值对应的流行分布的类别不同。具体来说，它们在前几层有较大概率属于真实标签，而在后几层有很高概率属于攻击后的标签。因此，自编码器也可用于逼近自然样本多个隐藏层的联合子空间并据此检测对抗样本。为了方便起见，本文只采用两个相邻层的子空间构建跨层检测器。

算法概述如图 1 所示。在图中，左侧列出了一个 ResNet18 [11] 网络，每个基本块由两个 64 通道的 3×3 卷积层（Conv）组成。在每个基本块的输出端有单层检测器和跨层检测器。算法分为两个阶段：训练阶段和推理阶段。在训练阶段，使用自然训练样本的激活值来训练每个检测器。而在推理阶段，当新样本通过推理网络时，每个检测器都会计算出重建损失，即样本到流形的距离。之后，联合分类器（同样也由一个自编码器组成）收集所有检测器输出的重建损失并做出联合决策以获得最终的判定结果。

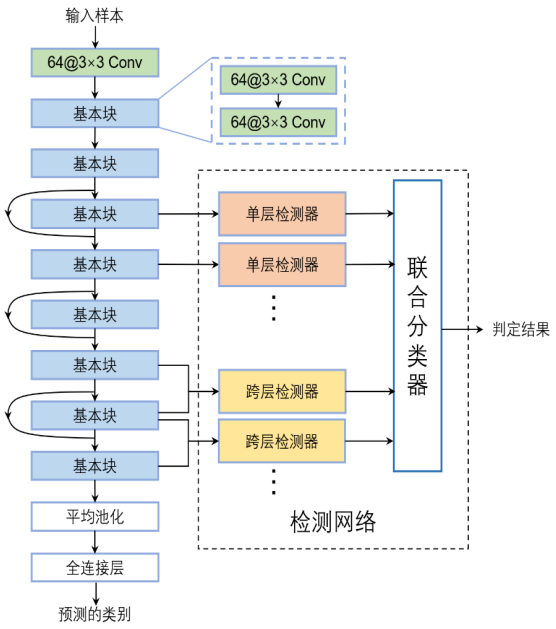

图 1　检测算法示意图

2.2　瓷片架构的高效层调度方案

2.2.1　动机

检测器与推理网络耦合后，整个网络的规模将会变大，随着推理网络模型规模的增大，这一点会越来越明显。本文使用瓷片架构的神经网络加速器作为基本架构，因为它具有良好的可扩展性 [2, 9]。如图 2 所示，瓷片架构加速器由一个二维的运算阵列（Array）组成，其中每个阵列包含一个小型的二维处理引擎（Processing Unit，PE）和一个局部 SRAM 缓冲区。所有的阵列都通过片上网络（Network on Chip，NoC）连接。

为了降低能耗和延迟，一项最新研究（Tangram [10]）提出的层间/层内数据流映射方案用于获得细粒度的数据并行。

为了支持层间流水线，大型神经网络以层为单位划分为段（Segment）。瓷片架构同一时间只运行一个段。图 3 展示了映射和计算的过程：神经网络各层被划分为段序列 S_0，S_1，S_2，S_3，然后依次映射到硬件上执行。在此过程中，DRAM 用于存储段之间的网络权重和临时数据。

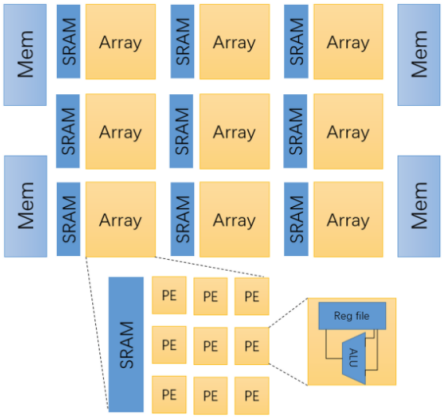

图 2　瓷片架构示意图

当网络拓扑变得复杂时，在满足数据依赖的情况下寻找最优的网络层调度和划分方案将不再是一个简单的遍历搜索任务。高效的调度可以大大减少不同存储层次之间的中间数据传输，从而有效降低能耗，提高性能。Tangram 中提出的层调度方案只考虑了数据依赖关系，不能保证高效地减少段之间的数据传输。例如，在图 3 中，S_1 和 S_3 共享 S_0 的

输出，却分为了两个段，导致数据传输了两次。此外，在联合网络中，具有多个后继的层非常常见。因此，本文针对这种情况提出了一种更好的层调度方法。

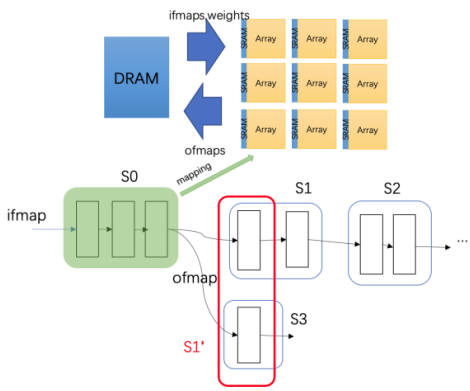

图 3 神经网络的映射过程

2.2.2 层调度方案

由于神经网络是以段为单位依次映射到硬件，段间的数据需要使用 DRAM 存储，带来巨大的性能损失和能耗。为了更有效地调度推理-检测联合网络，本文的想法是找到一个最优的层调度拓扑排序，通过该拓扑序列可以进一步得到段的划分，以最小化对 DRAM 的访问。

Algorithm 1: Layers Topo-Sort Algorithm.

Data: Network topology graph $G(v,e)$, on-chip storage capacity C, parameter index weight α
Result: Topological sequence of layers S

Denote the input layer as v_{in} ;
$S \leftarrow ()$;
$Q \leftarrow \{(v_{in}, 0)\}$;
$I \leftarrow 0$;
while Q *is not empty* **do**
  Extract the vertex v with the smallest key value from Q;
  S.append(v);
  for n *in* v.*nexts*() **do**
    if n.*prevs*() *in* S **then**
      $i \leftarrow$ the largest index of layer has the same parents with n in S;
      $\tau \leftarrow \sum_{k \geq i}$ TotalSize($S[k]$) +TotalSize(n);
      if $\tau > C$ **then**
        $Q \leftarrow Q \cup \{(n, \text{TotalSize}(n) + \alpha I)\}$;
      else
        $Q \leftarrow Q \cup \{(n, -\text{FmapSize}(e(v,n)) + \alpha I)\}$;
      $I \leftarrow I + 1$;

算法 1 神经网络的映射过程

本文提出了一种以中间数据传输量最小化为目标的拓扑排序贪心方法，可以保证神经网络的各层按照某种拓扑顺序有效地划分为段，并最大限度地减少段之间的数据传输。基本思想是：（1）将尽可能多的具有共同数据依赖的层组合在一起（如图 3 中的红色框 S_1'），从而减少公共数据的传输量；（2）将具有较少中间数据传输量的层组合在一起，以便在一个段中包含更多层，从而减少段之间的数据传输。网络被建模为图 $G(v, e)$，其中顶点集 v 表示网络中的层，边集 e 表示层之间的数据依赖关系。算法 1 展示了本文的优化方法，其中一个顶点的 Totalsize 表示该节点计算所需的所有数据量，一条边的 FmapSize 表示要在层之间传输的特征图的大小。

在获得网络中各层的拓扑序列后，通过动态规划算法在拓扑序列上寻找最优的段划分，使整个网络以最短的执行时间或最低的能耗运行。

2.3 软硬件协同设计

根据第 3.1 节中描述的算法，可以在推理网络的每一层都引入单层检测器、在每相邻两层都引入跨层检测器。然而，引入过多的检测器会带来巨大的开销，而且并不是所有的检测器都具有良好的流形表示能力。因此，有必要选择一个高效的检测器集合来减少运行时开销，同时保持检测成功率，从而实现软硬件协同设计。

为了找到最佳的协同设计方案，应该评估每种检测器配置方案的运行时开销和检测精度。在这一步，本文引入一些对抗样本作为验证集来评估检测精度，使用多种对抗样本攻击方法生成对抗样本，混合起来形成验证集，以避免对特定攻击方法的倾向性，保持算法的广谱性。对于运行时开销的评估，相比简单地使用模型计算量或存储量作为指标，更好的方法是衡量实际模型在实际的神经网络加速器架构中的性能指标。本文使用第 3.2 节中设计的瓷片架构评估模型运行的能耗，但这给搜索最佳方案带来了额外的困难。具体来说，由于神经网络在瓷片架构加速器上的调度和划分是一个复杂且离散的问题，添加或删除某个检测器可能会导致完全不同的调度划分方案，因此，传统的基于启发式的局部搜索方法很难获得良好的结果。

因此，本文对检测器集合的所有子集进行穷尽搜索以找到最佳设计。为了避免子集数量随检测器数目的增加而指数增长，本文首先将对自然样本和对抗样本区分度不高的检测器进行剪枝。之后，进行全局搜索并选择具有高精度和低开销的最佳配置。

3 试验分析

3.1 参数设置

本文在3个神经网络模型上进行试验，包括一个4层的简单卷积神经网络（称为 simpleCNN）、ResNet18和ResNet50[11]。它们分别使用MNIST[16]、GTSRB[22] 和 CIFAR10[13] 作为数据集进行训练。本文选择了4种攻击方法来生成对抗样本，即前面提到的FGSM[8]、FGM、BIM[15] 和C&W攻击[5]。神经网络模型的分类准确率和4种攻击方法的攻击成功率见表2。

关于防御的配置，simpleCNN中的每个卷积层和线性层、ResNet18 和 ResNet50 中的每个基本块的输出向量都被用来训练检测器，因此3个推理网络分别有7、19和33个检测器。

本文还实现了NIC[19]，这是一种最先进的对抗样本检测方法，并将其与本文的方法在检测成功率和运行时开销方面进行比较。实验用服务器配置为 Xeon E5-2680 2.40GHz 56核处理器、500 GB RAM 和2个 Tesla P100 GPU。

本文对包含 16×16 个阵列的瓷片架构进行建模。每个瓷片都是一个Eyeriss NN引擎[3]，其中包含一个8×8 PE 阵列和一个32 kB的私有SRAM缓冲区。假设采用28 nm技术，引擎以500 MHz运行。PE面积和功率从文献[6]中缩放产生，假设每个16位乘加运算单元的面积和单次操作能耗为 0.004 mm^2和1 pJ。NoC功率估计为每跳0.61 pJ/bit[17]。

Tangram中的搜索工具可以支持瓷片架构加速器的数据流方案。它以神经网络拓扑和硬件配置作为输入，并输出运行时间和能耗。本文修改这个搜索工具来实现本文的设计方案，评估本文提出的层调度方法。

3.2 检测算法评估

本文的检测器只使用自然样本进行训练。这些检测器估计重建损失，并将其发送到联合分类器中进行最终决策。用户需要指定一个阈值用来进行分类。当阈值从零移动到一个大数时，假阳性率（FPR，所有自然样本中被错误分类为对抗样本的比例）和真阳性率（TPR，所有对抗样本被正确分类为对抗样本的比率）将从0变为1。TPR与FPR折线图的曲线下面积（Area Under Curve，AUC）可以用来衡量检测成功率。具体来说，AUC是一个介于0和1之间的值，AUC越大意味着检测准确率越高。表3中列出了不同攻击方法和检测算法的AUC分数。在该表中，“全模型”表示在每个卷积层（或基本块）后插入检测器的模型，“协同设计模型”表示剪枝和搜索后的协同设计模型。协同设计模型的细节将在后文进行描述。

表2　模型分类准确率和攻击算法的攻击成功率

神经网络	分类准确率	攻击成功率			
		FGSM	FGM	BIM	C&W攻击
simpleCNN	99.47%	89.23%	75.03%	95.83%	96.08%
ResNet18	98.90%	89.14%	82.11%	100.00%	100.00%
ResNet50	94.95%	82.37%	71.09%	100.00%	100.00%

表3　本文方法与NIC方法的AUC分数比较

模型	检测算法	AUC分数			
		FGSM	FGM	BIM	C&W攻击
simpleCNN	NIC	0.9246	0.9032	**0.8578**	0.9411
	全模型	0.9502	0.8539	0.8541	0.9616
	协同设计模型	**0.9600**	**0.9142**	0.8535	**0.9870**
ResNet18	NIC	0.9361	0.9251	**0.9215**	0.9316
	全模型	**0.9538**	0.9390	0.8891	0.9466
	协同设计模型	0.9533	**0.9419**	0.8842	**0.9553**
ResNet50	NIC	0.9803	**0.9801**	0.9488	0.7003
	全模型	**0.9844**	0.9770	**0.9565**	0.8676
	协同设计模型	0.9791	0.9685	0.8607	**0.8742**

表 3 的结果表明，尽管 NIC 在 BIM 攻击中的表现略好一些，但在 12 个样例中有 9 个样例本文方法的表现优于 NIC，这表明本文方法具有和 NIC 相当的检测精度。本文的方法通常在较大的网络中具有更好的检测精度，这是因为可以集成更多的检测器进行联合判断。此外，除了受到 BIM 攻击的 ResNet50 网络外，协同设计模型的 AUC 分数都高于或几乎等于全模型，表明协同设计不会导致准确率的下降。

3.3 瓷片架构层调度方案的评估

在相同的硬件资源条件下，本文比较了两种调度方式。本文评估了运行一批（32 个）样本的能耗和运行时间。图 4 显示了两种调度方法下对 DRAM 和 SRAM 的数据访问量。本文提出的算法将 DRAM 访问量减少了 43% 以上，而作为代价，SRAM 访问有所增加。这是因为本文的算法将更多的数据依赖划分在同一个段内，这意味着段之间的数据传输被转移到了段内，与此对应的，DRAM 上的数据移动就被转化成了 SRAM 上的数据移动。评估还发现 70% 的系统能耗是由内存访问引起的，因此，通过降低 DRAM 访问可以进一步减小能耗。

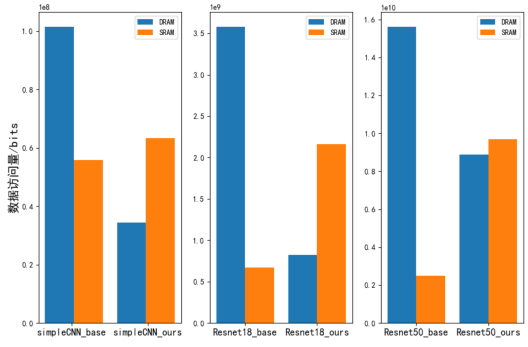

图 4 两种调度方式的内存访问量

表 4 显示了 3 个推理网络与所有检测器联合网络的能耗和性能。与 Tangram 中的方法相比，本文提出的方法在运行所有 3 个模型时都可以节省超过 28% 的能耗并减少超过 41% 的运行时间。能耗和运行时间的节省主要来自于本文所提调度方法对 DRAM 访问量的优化。

表 4 两种调度方式的运行开销

模型	优化方法	能耗/（$\times 10^{10}$ pJ）	运行时间/ms
simpleCNN	Tangram	3.37	0.967
	算法 1	1.65	0.399
ResNet18	Tangram	111.0	35.5
	算法 1	47.9	10.8
ResNet50	Tangram	482	147.0
	算法 1	347	85.8

3.4 软硬件协同设计的评估

本文以 ResNet18 为例来说明协同设计过程。首先，使用一个验证集来评估每个检测器的检测成功率，保留成功率最高的 8 个检测器，而舍弃其他检测器。然后遍历这些检测器的所有组合（总共 255 个方案）并评估准确率和开销。图 5 是一个散点图，其中每一个点表明每个检测器组合的 AUC 分数和能耗。左上区域的点具有高 AUC 分数和低能耗，是较优的选择。因此，本文选择图中的红点作为协同设计方案。它包含一个单层检测器和两个跨层检测器。同样，可以通过分别部署 1 个和 3 个检测器来找到 simpleCNN 和 ResNet50 的协同设计模型。

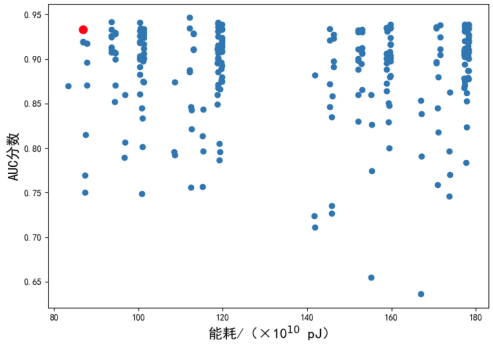

图 5 ResNet18 模型不同检测器组合的 AUC 分数-能耗散点图

表 5 展示了推理网络在无检测模型、使用全模型和协同设计模型的情况下的运行时开销。结果表明，相比原始的推理网络，全模型在每个卷积层（或基本块）引入一个检测器，造成的能耗和运行时间的开销是非常大的。这是由于推理网络和检测网络的网络结构差异引起的。执行图像分类任务的深度神经网络中需要很多卷积层来实现局部特征的

提取，卷积层的参数数目相对较少，而计算量偏大；执行在线检测任务的自编码器网络主要由全连接层组成，参数数目很多，而计算量偏小。对于神经网络加速器而言，主要的能耗和时间开销在访存而非计算，因此，引入检测器会显著增大整个模型的能耗和时间开销。这正是本文提出软硬件协同设计的剪枝方案的初衷。

表 5　不同检测配置下联合网络的运行开销

	推理网络	无检测模型	全模型	协同设计模型
能耗/（$\times 10^{10}$pJ）	simpleCNN	0.41	1.65	0.32
	ResNet18	19.2	47.9	19.7
	ResNet50	53.9	347.0	68.5
运行时间/ms	simpleCNN	0.048	0.399	0.158
	ResNet18	4.6	10.8	4.9
	ResNet50	8.8	85.8	13.9

比较协同设计模型和全模型，可以发现，软硬件协同设计的作用是明显的，由于对检测器的数目和位置进行了筛选，去除了检测能力较差的检测器，所以获得了较大的性能提升。协同设计模型与全模型相比，能耗降低了 58% 以上，执行时间减少了 54% 以上。

4　结论

深度神经网络已广泛用于自动驾驶等，本文针对深度神经网络攻击的防御，提出了一种新型的硬件友好的在线防御方案。本文的设计方案采用多个自编码器来检测推理网络的每一层，然后使用本文提出的新型层调度方法在瓷片架构加速器中运行推理-检测联合网络。最后，本文设计了一种软硬件协同设计方法，以在检测成功率和运行时开销之间取得平衡。

试验表明，本文提出的防御方案不仅达到了相当的检测成功率，而且通过最小化 DRAM 访问量，使能耗降低了 28%，运行时间降低了 41%。软硬件协同设计在不降低检测成功率的前提下，进一步减少了 58% 和 54% 的能耗和时间。

参考文献（References）

[1] BOJARSKI M，TESTA D D，DWORAKOWSKI D，et al. End to End Learning for Self-Driving Cars［Z］. arXiv Preprint arXiv：1604.07316，2016：1-9.

[2] CHEN Yunji，LUO Tao，LIU Shaoli，et al. DaDianNao：A Machine-Learning Supercomputer［C］//2014 47th Annual IEEE/ACM International Symposium on Microarchitecture，Dec. 13-17，2014，Cambridge，UK.Piscataway NJ：IEEE，c2014：609-622.

[3] CHEN Y H，SZE V，EMER J S，et al. 2016. Eyeriss：A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks［J］. ACM SIGARCH Computer Architecture News，2016，44(3)：367-379.

[4] CHEN Y H，KRISHNA T，EMER J S，et al. Eyeriss：An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks［J］. IEEE Journal of Solid-State Circuits，2017，52(1)：127-138.

[5] CARLINI N，WAGNER D. Towards Evaluating The Robustness of Neural Networks［C］//2017 IEEE Symposium on Security and Privacy（SP），May 22-26，2017，San Joes，CA ，USA.Piscataway NJ：IEEE，c2017：39-57.

[6] DU Zizhong，FASTHUBER R，CHEN Tianshi，et al. 2015. ShiDianNao：Shifting Vision Processing Closer to the Sensor［C］//Proceedings of the 42nd Annual International Symposium on Computer Architecture（ISCA），June 13-17，2015，Portland，OR，USA. Piscataway NJ：IEEE，c2015：92-104.

[7] DING Yifan，WANG Liqiang，ZHANG Huan，et al. Defending Against Adversarial Attacks Using Random Forest［C］//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops（CVPRW），June 16-17，2019，Long Beach，CA，USA. Piscataway NJ：IEEE，c2019：105-114.

[8] GOODFELLOW L J，SHLENS J，SZEGEDY C. Explaining and Harnessing Adversarial Examples［Z］. arXiv Preprint arXiv：1412.6572，2014：1-11.

[9] GAO Mingyu，PU Jing，YANG Xuan，et al. Tetris：Scalable and Efficient Neural Network Acceleration with 3d Memory［J］. Computer Architecture News，2017，45(1)：751-764.

[10] GAO Mingyu，YANG Xuan，PU Jing，et al. Tangram：Optimized Coarse-Grained Dataflow for Scalable Nn Accelerators［C］//Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems，2019：807-820.

[11] HE Kaiming，ZHANG Xiangyu，REN Shaoqing，et al.

Deep Residual Learning for Image Recognition［C］// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition（CVPR）, June 27−30, 2016, Las Vegas, NV, USA.Piscataway NJ: IEEE, c2016: 770−778.
［12］JOUPPI N P, YOUNG C, PATIL N, et al. In-Datacenter Performance Analysis of a Tensor Processing Unit［C］// Proceedings of the 44th Annual International Symposium on Computer Architecture（ISCA）, June 24−28, 2017, Toronto, ON, Canada.Piscataway NJ: IEEE, c2017: 1−12.
［13］KRIZHEVSKY A. Learning Multiple Layers of Features from Tiny Images［Z］.2009: 1−58.
［14］KRIZHEVSKY A, SUTSKEVER I, HINTON G, et al. Imagenet Classification with Deep Convolutional Neural Networks［J］. Advances in Neural Information Processing Systems, 2012, 25: 1097−1105.
［15］KURAKIN A, GOODFELLOW I, BENGIO S.Adversarial Examples in the Physical World［Z］. arXiv: 1607.02533, 2016: 1−14.
［16］LECUN Y, BOTTOU L, BENGIO Y, et al. 1998. Gradient-Based Learning Applied to Document Recognition［J］. Proceedings of The IEEE, 1998, 86（11）: 2278−2324.
［17］LI Sheng, AHN J H, STRONG R D, et al. Mcpat: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures［C］//Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture（MICRO）, Dec.12−16, 2009, New York, NY, USA.Piscataway NJ: IEEE, c2009: 469−480.
［18］MENG Dongyu, CHEN Hao. Magnet: A Two-Pronged Defense Against Adversarial Examples［C］// Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017: 135−147.
［19］MA Shiqing, LIU Yingqi, TAO Guanhong, et al. Nic: Detecting Adversarial Samples with Neural Network Invariant Checking［C］//Proceedings of the 26th Network and Distributed System Security Symposium（NDSS 2019）, Feb. 24−27, 2019, San Diego, CA, USA.
［20］POPE P, ZHU Chen, ABDELKADER A, et al. The Intrinsic Dimension of Images and Its Impact on Learning［Z］. arXiv preprint arXiv: 2104.08894, 2021.
［21］ROUHANI B D, SAMRAGH M, JAVAHERIPI M, et al. Deepfense: Online Accelerated Defense Against Adversarial Deep Learning［C］// Proceedings of the IEEE/ACM International Conference on Computer-Aided Design（ICCAD）, Nov. 5−8, 2018, San Diego, CA, USA. Piscataway NJ: IEEE, c2018: 1−8.
［22］STALLKAMP J, SCHLIPSING M, SALMEN J, et al. Man Vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition［J］. Neural Networks, 2012, 32（1）: 323−332.
［23］SOTGIU A, DEMONTIS A, MELIS M, et al. Deep Neural Rejection Against Adversarial Examples［J］. EURASIP Journal on Information Security, 20201: 1−10.
［24］SHAMIR A, MELAMED O, BENSHMUEL O. The Dimpled Manifold Model of Adversarial Examples in Machine Learning［Z］.arXiv Preprint arXiv: 2106.10151, 2021.
［25］XU Weilin, EVANS D, QI Yanjun. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks［Z］. arXiv Preprint arXiv: 1704.01155, 2017.
［26］YANG Xuan, GAO Mingyu, LIU Qiaoyi, et al. Interstellar: Using Halide's Scheduling Language to Analyze DNN Accelerators［C］//Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, 2020: 369−383.https://doi.org/10.1145/3373376.3378514.
［27］JIA Zhe, TILLMAN B, MAGGIONI M, et al. Dissecting the Graphcore IPU Architecture via Microbenchmarking［Z］. arXiv: 1912.03413［cs.DC］, 2019.
［28］WANG Xingbin, HOU Rui, ZHAO Boyan, et al. Dnnguard: An Elastic Heterogeneous DNN Accelerator Architecture Against Adversarial Attacks［C］//Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, 2020: 19−34. https://doi.org/10.1145/3373376.3378532.

作者简介

李兆麟（1973−），男，黑龙江大庆人，博士，教授，主要从事高性能处理器、智能芯片、嵌入式系统等关键技术研究。
Tel：13701227116
E-mail：lzl73@mail.tsinghua.edu.cn

通信作者

范仁昊（2000−），男，河南辉县人，博士研究生，主要研究方向为神经网络的安全性、神经网络加速器架构等。
Tel：18811307870
E-mail：frh21@mails.tsinghua.edu.cn