PatchBlock：面向嵌入式边缘人工智能设备的轻量级对抗补丁防御方法

Nandish Chattopadhyay[1], * Abdul Basit[1], * Amira Guesmi[1],Muhammad Abdullah Hanif[1],Bassem Ouni[2],Muhammad Shafique[11] 阿联酋阿布扎比纽约大学（NYU）工程学院 eBRAIN 实验室[2] 阿联酋迪拜 DakAI

arXiv:2601.00367v1 [cs.CR] 1 Jan 2026

摘要—对抗攻击对边缘人工智能应用中机器学习模型的可靠部署构成了重大挑战，例如自动驾驶和监控；这些应用依赖资源受限设备进行实时推理。在各种攻击中，基于补丁的对抗攻击会将小型恶意补丁（如贴纸）贴在物体上，从而诱骗神经网络做出错误预测，并可能造成严重后果。本文提出 PatchBlock，这是一种轻量级框架，旨在检测并消除图像中的对抗补丁。PatchBlock 利用离群点检测和降维，识别受对抗噪声影响的区域并抑制其影响。它作为传感器层级的预处理模块运行，可在 CPU 上高效并行于 GPU 推理，从而维持系统吞吐量，同时避免额外的图形处理器开销。该框架采用三阶段流水线：将输入分割为块（分块）、通过采用定向切分以加快收敛的改进型孤立森林检测异常区域（分离），以及对已识别离群点进行降维（缓解）。PatchBlock 与模型和补丁无关，可改造后集成到现有流水线中，并能在传感器输入与下游模型之间无缝运行。在多种神经网络架构、基准数据集、攻击类型和不同边缘设备上的评估表明，PatchBlock 能持续提升鲁棒性，在 Google 对抗补丁等强力贴片攻击下，模型准确率最高可恢复至 77%，同时保持高可移植性，并将干净准确率损失控制在最低水平。此外，就计算时间和每样本能耗而言，PatchBlock 的效率优于最先进的防御方法，因此适用于边缘人工智能应用。

索引词—鲁棒性、对抗攻击、对抗补丁、边缘人工智能、轻量级防御、离群点检测、降维

I. I引言

对抗攻击对深度神经网络（DNN）模型在现实应用中的可靠部署构成了重大挑战 [1]。通过向输入数据注入精心构造的扰动，这些攻击利用神经网络架构中的漏洞，可能导致严重的错误分类。其中，基于补丁的对抗攻击 尤其令人担忧，因为它们具有局部性，且易于在实践中实施 [2]。与分布式噪声攻击不同，贴片攻击会操纵图像的特定区域，通常使用醒目的实体贴纸，以极少的对抗努力持续欺骗模型。这类攻击已被证明会破坏分类、检测和深度估计流水线，给自动驾驶、监控和医学诊断等安全关键领域带来严重风险 [3]–[9]。

在边缘人工智能 部署中，这种威胁会进一步加剧，因为模型运行于资源受限设备上。这些系统是自动驾驶汽车、物联网监控和移动机器人的重要组成部分，必须在严格限制下兼顾准确率、延迟和功耗 [10]。包括对抗训练、模型集成和认证鲁棒性 [11]–[13], 在内的传统防御策略会带来高昂的计算成本，在这类环境中并不实用。因此，我们需要轻量且可移植的防御措施，能够在无需重新训练或专用硬件的情况下提供实用的鲁棒性。



图 1. PatchBlock 实时检测和缓解对抗补丁的运行示例：展示了在 INRIA 数据集 [16] 上，AdvYOLO[15] 攻击下 YOLOv4 [14] 的目标检测。

我们对抗补丁行为的研究揭示了其独特的信息论特征。使用互信息（MI） [17]，我们量化图像滑动窗口块中各颜色通道之间的局部依赖关系。与对抗补丁重叠的区域表现出异常高的互信息值，可与干净区域明确区分开来。这一观察促成了我们的防御策略，如图 2 所示，其中对抗区域可以高效地识别为统计离群值。

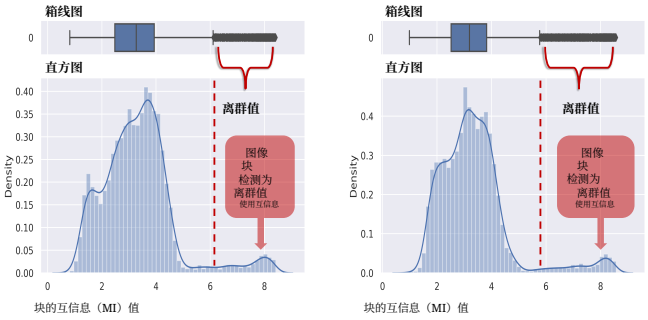

图 2. PatchBlock 的核心洞见：对抗补丁呈现出不同的统计分布，可通过局部互信息分析捕捉。

基于这一发现，我们提出了 **PatchBlock**，这是一种轻量级预处理框架，可在图像输入下游模型之前检测并消除对抗补丁。PatchBlock 包含：(i) 分块，将输入划分为局部窗口；(ii)分离，采用重新设计的孤立森林，通过定向切分实现更快收敛；以及 (iii) 缓解，通过奇异值分解（SVD）进行降维，以消除异常区域。与需要重新训练模型或依赖 GPU 的认证验证的防御方法不同，PatchBlock 可在 CPU 上高效运行

*这些作者对本项工作的贡献相同。

并与 GPU 推理重叠执行，从而在嵌入式部署中保持吞吐量。

本文的主要贡献如下：

- 我们提出**PatchBlock**，这是一种轻量且与模型无关的对抗防御方法，通过定位受影响区域并进行降维来检测和缓解对抗补丁的影响。PatchBlock 被设计为一个预处理模块，可直接集成到现有流水线中，无需重新训练。

- PatchBlock 利用统计方法，结合局部互信息分析、孤立森林和奇异值分解（SVD），高效识别异常区域。它主要依靠 CPU 运行，无需额外的 GPU 计算，因此非常适合部署在资源受限的边缘人工智能环境中。

- 为降低计算开销，我们通过定向切分策略重新设计了孤立森林，相较于标准实现可加快收敛速度。此外，我们优化了互信息（MI）的计算方法，利用局部依赖关系和并行处理，提升了其实时防御的实用性。

- 我们在多种神经网络模型（ResNet-50 [18],VGG-19 [19],视觉变换器 [20]，YOLOv4 [14]）、数据集（ImageNet[21], INRIA [16], CASIA [22]）以及对抗贴片攻击（Google 对抗补丁（GAP） [6],AdvYOLO [15]）上对 PatchBlock 进行了全面评估。结果表明，PatchBlock 能够持续提升不同架构和边缘设备上的鲁棒性，面对强力贴片攻击时，准确率恢复最高可达 77%，同时仅造成极小的干净性能下降。

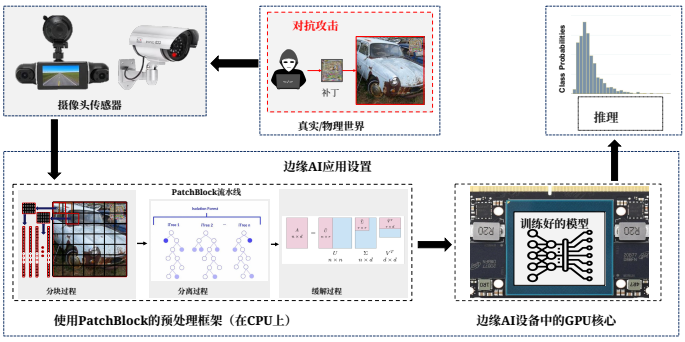

图 3. PatchBlock 防御流水线概览，该流水线部署在嵌入式边缘人工智能设备的传感器输入与下游推理引擎之间。

II. 背景

本节概述构成理论基础的内容以及设计选择，这些内容和选择共同推动了 PatchBlock 防御框架的设计。

A. 威胁模型：对抗贴片攻击

对抗贴片攻击是一种局部扰动，攻击者会在图像中插入醒目的图案（例如贴纸），以误导深度神经网络（DNNs）。研究表明，这些攻击会破坏分类、检测和深度估计流水线，给自动驾驶、监控和医学诊断带来严重影响 [2], [3], [5]–[9], [15], [23]。我们采用与以往研究一致的白盒威胁模型 [24]–[26], ，即攻击者了解模型架构和防御策略。尽管自适应攻击者可能会降低鲁棒性裕度，PatchBlock 仍展现出抵御强贴片攻击的能力。

B. 互信息

互信息（MI）用于量化两个随机变量之间共享的信息量。对于 X 和 Y，MI 定义如下：

$$I(X;Y) = \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) \tag{1}$$

其中，$p(x,y)$ 是联合分布，$p(x)$ 和 $p(y)$ 是边缘分布。

在图像中，MI 捕捉局部区域之间的统计依赖关系。对抗补丁通常会引入异常相关性，导致 MI 值偏离干净区域的 MI 值。这为检测提供了有用的统计信号。形式化地，对于补丁 $\mathbf{P}_{i,\ j}$ 及其邻域 $\mathbf{N}_{i,\ j}$ 在图像 $\mathbf{I}$ 中：

$$I(\mathbf{P}_{i,j};\mathbf{I}) = I\left(\mathbf{P}_{i,j}; \bigcup_{(k,l)} \mathbf{P}_{k,l}\right) \tag{2}$$

尽管计算全局 MI 的计算开销较大，PatchBlock 采用了一种高效的局部近似方法，能够捕捉这些异常，同时仍可实现实时部署。

C. 离群点检测

孤立森林（IF）是一种基于集成的异常检测算法，通过使用随机生成的决策树递归划分特征空间 [27], [28]。其原理是，异常通常较为稀少且具有明显差异，因此只需较少的划分即可将其孤立。在 PatchBlock 中，对抗补丁正是此类异常：它们仅占据图像中的小块区域，同时在统计特征上不同于干净内容。因此，IF 是进行补丁定位的自然选择，因为它无需监督、不依赖标注数据，且计算开销较低。为进一步提升效率，我们对 IF 引入了定向切分修改。不同于随机划分，定向切分会使划分过程偏向具有统计信息的区域，从而减少树深度并加快收敛。这种设计使 IF 更适用于资源受限的边缘人工智能环境。

D. 降维

降维方法通过将数据投影到低维子空间，同时保留关键信息，从而减轻对抗噪声的影响。PatchBlock 使用奇异值分解（SVD） [29], [30], ，并以奇异值比率估算信息保留程度 [31]。通过削弱异常成分，SVD 能有效抑制补丁的影响，同时保留干净特征。虽然也可以采用置零或模糊等更简单的操作，但 SVD 在鲁棒性与保真度之间实现了有依据的平衡。采用这一方法有助于 PatchBlock 消除补丁，同时避免大幅降低干净样本上的性能。

E. 边缘设备与应用

NVIDIA Jetson AGX Orin、AGX Xavier 和 Orin Nano 等边缘人工智能平台 [32] 广泛应用于自动驾驶、机器人技术和监控系统。这些设备必须在计算、内存和能源限制严格的条件下处理高吞吐量的传感器数据。典型应用包括：

- **自动驾驶：** 实时分析多模态数据（摄像头、激光雷达、雷达），以实现环境感知、目标检测和导航。

- **监控：** 在设备端进行视频分析，用于异常检测、目标跟踪和人脸识别，同时减少对云端资源的依赖。

对于此类应用场景，防御机制必须在提升鲁棒性的同时，避免引入高昂的延迟或能耗。PatchBlock

专为满足这一要求而设计：它可在 CPU 上高效运行，与 GPU 推理并行执行，并且只会带来可忽略不计的系统级开销。

III. PATCHBLOCK 算法

PatchBlock 算法的实现分为三个阶段。第一阶段是使用滑动窗口将输入图像分割成多个块，这一过程称为分块过程。该步骤由核的大小和滑动窗口的步长决定。第二阶段使用孤立森林检测图像中可能包含对抗补丁的异常图像块，详情见算法2、3和4。最后一步是使用奇异值分解进行降维，以缓解可能包含补丁的区域。该三阶段流程如算法1所示，相应的示意图见图4。

算法1： PatchBlock 算法

IN：I: 未处理图像样本，T: 孤立森林中的树集成大小，
c: 离群值分数，$info$: SVD参数，k: 核大小，str: 步长大小
OUT：I：已处理图像样本
/*$\boldsymbol{ChunkingProcess}$*/
从图像 I 生成 n 图像块 $X \leftarrow (x_1, \ldots\ldots, x_n)$，尺寸为
核 = k 和步长大小 = str
/*$\boldsymbol{SeparatingProcess}$*/
设置 $s = 0.3 \times size\,(X)$
$iForest = FastIsolationForest(X,\ T,\ s)$
initialize $Y = (y_1, \ldots, y_n)$
对于 $i = 1$ 到 n，执行:
$y_i = AnomalyScore(x_i)$，其中 $x_i \in X$ [来自算法 2]
结束循环
获取按降序排列的 Y
选取异常（z_1、.....、z_r）$\rightarrow Z$作为（$1 - c$）$* n$ 个顶部 y_is
/*$\boldsymbol{MitigatingProcess}$*/
对于 $j = 1$ 到 r 执行:
$M \leftarrow \{m_1, \ldots, m_r\}$
使得 $m_j \leftarrow x_i$ 对于匹配对（$i,\ j$）
使用奇异值分解进行降维，如下:
对于 $k = 1$ 到 r 执行:
$\hat{}m_k \leftarrow SVD(m_k, info)$
将图像 I 中的 m_k-s替换为$\hat{}m_k$-s，得到$\hat{I}$
return $\hat{I}$

为了检测异常，PatchBlock 使用快速孤立树集成，即快速孤立森林，详见算法2。具体而言，我们采用了一种效率显著更高的孤立树版本，称为快速孤立树。它使用针对数据集属性的定向切分，而非随机切分，从而加快收敛并更快地检测潜在异常 [33]。快速孤立树算法见算法3，其中使用了一个名为$GradientSplit$的函数来执行定向切分。

算法2： $FastIsolationForest(X,\ T,\ size)$

IN：$X = (x_1, \ldots\ldots, x_n)$: 输入数据，$T$: 森林中的树的数量，$s$: 样本大小
OUT：一个 $FastIsolationForest$（$FastIsolationTree$集成）
初始化：$FastIsolationForest$
设置 $height_{max} = ceiling\,(log_2\ s)$
对于 $i = 1$ 至 T，执行:
$X' \leftarrow Sample\,(X,\ s)$
$FastIsolationForest \leftarrow$
$FastIsolationForest \cup FastIsolationTree\,(X',\ 0,\ height_{max})$
结束循环
返回 $FastIsolationForest$

A. 孤立树中的定向切分

属性值的分布可作为考察两类实例之间差异的基础。

算法 3： $FastIsolationTree(X, height, height_{max})$

IN：$X = (x_1, \ldots, x_n)$: 输入数据，$height$: 树的高度，$height_{max}$: 树的最大高度，Q：X的属性
OUT：一个 $FastIsolationTree$
如果 $height \geq height_{max}$ 或 $|X| \leq 1$，则:
返回 $externalNode\{Size \leftarrow |X|\}$
else
从 Q中随机选择 k（$k \leq q$）个不同的属性
选出属性 $\hat{q}$ 使用可分性指数最高的
算法 4
$X_{left} \leftarrow \{x | x < bestX, x \in X\}$
$X_{right} \leftarrow \{x | x \geq bestX, x \in X\}$
return $internalNode$
{
$LeftTree \leftarrow FastIsolationTree(X_{left}, height + 1, height_{max})$
$RightTree \leftarrow FastIsolationTree(X_{right}, height + 1, height_{max})$
$attribute \leftarrow bestX$
$value \leftarrow highestSep$
}
end if

其基本概念是：不同属性的数值分布累积差异可以反映异常实例与正常实例之间的差异程度。属性的可分性受两个因素影响：峰值之间的距离，以及每类实例中数值的离散程度。在这种情况下，对每类实例的数值求平均（记作 $E()$），得到代表数值中心的均值。方差函数（$Var()$）再次用于衡量数值的离散程度。这两个指标用于构建计算可分性指数 $Separation(X^q,\ v)$ 的函数，公式如下：

$$\frac{\sqrt{(E(x|x \in X^q, x < v) - E(x|x \in X^q, x > v))^2 Var(X^q)}}{Var((x|x \in X^q, x < v)) + Var((x|x \in X^q, x > v))}$$

针对给定属性搜索分割点，提出了一种近似最优方法。通过计算相邻属性值之间可分性指数的梯度，该方法可以跳过不太可能被选为分割点的属性值，从而加快搜索过程。梯度计算函数G如下：

$$G = \frac{Separation(X^q, x_{i+1}) - Separation(X^q, x_i)}{x_{i+1} - x_i} \tag{3}$$

用于更新 $step$ 参数的函数定义如下：

$$step = \begin{cases} \frac{3}{1+e^{G * log_{10}|X^q|}} * \frac{|X^q|}{100} & \text{if } G < 0 \\ 0.7 - \frac{1.3}{1+e^{G * log_{10}|X^q|}} * \frac{|X^q|}{100} & \text{if } G \geq 0 \end{cases} \tag{4}$$

这两个函数用于以下梯度分割算法，以找出可分性指数最高的属性。

B. 图像块的高效互信息计算

为防御基于补丁的对抗攻击，我们的防御机制需要识别并重建表现异常的图像块。在这一过程中，一个关键步骤是计算图像块与整幅图像之间的互信息（MI），这有助于检测可能已被对抗补丁篡改的区域。然而，计算每个图像块与整幅图像之间的互信息需要大量计算，因而成为实时应用的瓶颈。

为了解决这一问题，我们提出了一种高效的互信息（MI）计算方法，在保持防御机制有效性的同时，显著降低计算负担。我们不再

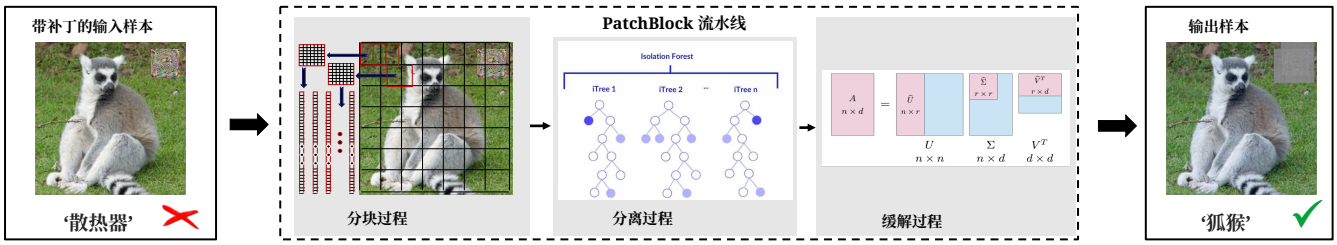

图 4. PatchBlock 流水线：分块（使用滑动窗口获取核，并将其转换为向量）、分离（使用互信息和通过梯度分割进行定向切分的快速隔离森林）和缓解（使用奇异值分解）这三个过程。

算法 4: *GradientSplit*(X)

IN: $X = (x_1, \ldots, x_n)$: 输入数据，X^q: 第 q 个属性的排序值
OUT: $bestX$: 具有最高可分性指数的属性，
$highestSeparation$: 最大可分性指数值
初始化: $step$ 为 $ceiling(|X^q| * 0.001)$
令 $highestSeparation = Separation(X^q, x_1)$
令 $bestX = (x_1 + x_2)/2$ 和 $i = 0$
当 $i < |X_q|$ 执行:
设置 $i = i + step$
设置 $currentSeparation = Separation(X^q, x_i)$
如果 $currentSeparation > highestSeparation$ **则**:
设置 $highestSeparation = currentSeparation$
设置 $bestX = (x_i + x_{i+1})/2$
结束如果
使用公式 4 更新 $step$，公式 4 使用公式 3
结束循环
返回 $(bestX, highestSeparation)$

计算每个块与整幅图像之间的互信息，而是计算每个块与其相邻块之间的互信息。该方法基于这样一个观察：局部依赖对于检测对抗补丁引入的异常更为重要，因为这些补丁通常只出现在局部区域，并会扰乱图像的局部统计属性。

1) 局部互信息计算： 我们提出计算每个图像块 $\mathbf{P}_{i,j}$ 与其紧邻邻居 $\mathbf{N}_{i,j}$之间的互信息：

$$I_{\text{local}}(\mathbf{P}_{i,j}; \mathbf{N}_{i,j}) = \sum_{\mathbf{p} \in \mathcal{P}} \sum_{\mathbf{n} \in \mathcal{N}} p_{\mathbf{PN}}(\mathbf{p}, \mathbf{n}) \log \left(\frac{p_{\mathbf{PN}}(\mathbf{p}, \mathbf{n})}{p_{\mathbf{P}}(\mathbf{p}) p_{\mathbf{N}}(\mathbf{n})} \right), \quad (5)$$

其中， $\mathcal{P}$ 和 $\mathcal{N}$ 分别是该图像块及其邻居中可能像素强度值的集合。

这种局部互信息计算利用了自然图像因空间连续性而呈现强局部依赖这一事实。对抗补丁引入的异常会破坏这些局部依赖，导致图像块与其邻居之间的互信息显著偏离。

2) 局部互信息的理论依据： 在图像建模中，马尔可夫随机场（MRF）的性质为使用局部互信息提供了坚实的理论依据。在马尔可夫随机场中，像素（或图像块）的强度概率主要取决于其局部邻域。因此，通过图像块与其邻居之间的互信息捕获的统计依赖关系足以检测异常。

此外，由于共享信息趋于饱和，互信息会随着邻域大小增加而呈现边际效益递减的特性。这意味着，在计算中加入距离更远的补丁，对互信息值的贡献微乎其微。数学上，对于更大的邻域 $\mathbf{N}'_{i,j} \supseteq \mathbf{N}_{i,j}$:

$$I(\mathbf{P}_{i,j}; \mathbf{N}'_{i,j}) \approx I(\mathbf{P}_{i,j}; \mathbf{N}_{i,j}) \quad (6)$$

对于足够大的图像，远距离块之间的互信息极小。

3) 计算效率：通过将互信息计算限制在相邻块之间，我们将复杂度从 $O(N^2)$ 降至 $O(N)$，其中 N 是图像中的块数。我们还采用并行处理，同时计算所有块的互信息分数。设 K 为每个块的邻居数。互信息计算总次数为 $O(N \cdot K)$，远低于将每个块与整幅图像进行比较时所需的原始$O(N^2)$次计算。

IV. 实验结果

我们在多种模型、数据集和对抗贴片攻击下评估了所提出的 PatchBlock 防御的有效性。我们的目标是在嵌入式边缘人工智能设备上，展示其在图像分类和目标检测任务中的鲁棒性、适应性和可移植性。所有选用的基准和攻击策略均广泛应用于先前的研究，确保比较公平且结果可复现。

A. 实验设置

PatchBlock 部署为运行在 CPU 核心上的预处理模块，而底层深度学习模型则运行在 GPU 核心上。这种设置反映了真实的部署场景：轻量级防御与高吞吐量推理引擎协同运行。

图像分类： 我们使用 ImageNet 数据集 [21], 评估分类鲁棒性，该数据集包含多样化的类别和具有挑战性的变化。骨干模型包括 ResNet-152 和 ResNet-50 [18], VGG-19 [19],以及视觉变换器（ViT） [20],，涵盖卷积架构和基于变换器的架构。

行人检测： 对于检测任务，我们聚焦于行人检测这一任务，这是对抗补丁尤其有效的子领域。我们使用 INRIA 数据集 [16], ，该数据集涵盖多样化的现实场景；同时使用 CASIA 数据集 [22], ，其中每帧包含多个补丁实例。YOLOv4 [14] 凭借其广泛的应用和在嵌入式环境中的高效率，作为检测骨干模型。

B. 对抗贴片攻击

我们将 PatchBlock 与最先进水平的对抗补丁攻击进行对比评估。

- **分类：** Google 对抗补丁（GAP） [6]和 LAVAN [7], ，它们代表了强大的白盒攻击策略。
- **检测：** AdvYOLO 补丁 [15], 专为破坏基于 YOLO 的检测器而设计。

这些攻击涵盖了多种补丁设计和目标，为防御机制提供了严格的测试。

C. 边缘人工智能设备

实验在两种广泛使用的嵌入式边缘人工智能平台上进行：

- **NVIDIA Jetson Orin**：12 核 ARM Cortex-A78AE CPU 和配备多达 2048 个 CUDA 核心的 NVIDIA Ampere GPU。
- **NVIDIA Jetson Orin Nano**：6 核 ARM Cortex-A78AE CPU 和配备多达 1024 个 CUDA 核心的 NVIDIA Ampere GPU。
- **Lambda Tensorbook**：Intel Core i7-11800H（8 核，2.3–4.6 GHz），搭载 16 GB 显存的 NVIDIA 3080 Ti 笔记本电脑 GPU。

PatchBlock 仅在 CPU 核心上运行，而分类和检测模型则在 GPU 上执行。这种分离方式表明，PatchBlock 引入的开销极小，因此能够在资源受限环境中实现实时部署。

D. *PatchBlock* 实验结果

本节详细评估了 PatchBlock 防御机制在不同任务、模型和设备上的表现，展示了其鲁棒性、适应性和效率。

表 I PatchBlock 在图像分类任务中针对 Google 对抗补丁的性能分析（224×224）

Lambda Tensorbook 模型	干净准确率（%）	对抗准确率（%）	鲁棒性准确率（%）	PatchBlock 运行时间（秒）
VGG-19	75.4	0.45	62.05	0.0716
ResNet-50	79.45	3.0	68.65	0.0738
ViT-16	83.85	0.0	77.90	0.0729

Jetson Orin 模型	干净准确率（%）	对抗准确率（%）	鲁棒性准确率（%）	PatchBlock 运行时间（秒）
VGG-19	77.25	0.55	64.80	0.2052
ResNet-50	79.60	3.0	67.95	0.2038
ViT-16	84.35	0.05	78.10	0.1984

Jetson Orin Nano 模型	干净准确率（%）	对抗准确率（%）	鲁棒性准确率（%）	PatchBlock 运行时间（秒）
VGG-19	78.15	0.7	65.45	0.3174
ResNet-50	78.55	3.05	66.0	0.3177
ViT-16	83.95	0.05	78.10	0.3160

我们在 Google 对抗补丁（GAP）攻击下，使用三种神经网络模型 VGG-19、ResNet-50 和视觉变换器（ViT-16），在 ImageNet数据集上评估了 PatchBlock。表 I 报告了三种设备——Lambda Tensorbook、Jetson Orin 和 Jetson Orin Nano——上的干净准确率、对抗准确率（应用 GAP 后）、鲁棒准确率（应用 PatchBlock 后）以及**CPU 运行时间** 。PatchBlock 显著提升了模型的鲁棒性：在 Tensorbook 上，ViT-16 的鲁棒准确率最高达到 77.90%，从 0% 的对抗准确率中恢复。尽管存在硬件差异，各设备上的鲁棒准确率仍保持一致；例如，VGG-19 在 Jetson Orin 上达到 64.80%，在 Jetson Orin Nano 上达到 65.45%。此外，PatchBlock 的 CPU 运行时间适合资源受限的环境；对于 VGG-19，从 Tensorbook 上的 0.0716 秒到 Jetson Orin Nano 上的 0.3174 秒不等。

表 II 报告了在 ImageNet数据集上，PatchBlock 结合 LAVAN 补丁执行图像分类任务时的性能。其趋势与 Google 对抗补丁的趋势一致。

我们使用 YOLO-v4 模型，在 Adv-YOLO 补丁攻击下，进一步评估了 PatchBlock 在 INRIA 和 CASIA 数据集上的行人检测表现。表 III 展示了干净准确率、对抗准确率、鲁棒平均精度（AP）和运行时间。PatchBlock 将 INRIA 上的平均精度恢复至 92.86%，将 CASIA 上的平均精度恢复至 100%；此前，对抗 AP 值最低仅为 14.28%。运行时间仍处于实时应用可接受的范围内：在 INRIA 上，从 Tensorbook 的 0.326 秒到 Jetson Orin Nano 的 1.384 秒不等。

表 II PatchBlock 结合 LAVAN 补丁在图像分类任务（224X224）中的性能分析

Lambda Tensorbook 模型	干净准确率（%）	对抗准确率（%）	鲁棒性准确率（%）	PatchBlock 运行时间（秒）
VGG-19	75.65	0	63.9	0.0758
ResNet-50	81.10	6.0	72.3	0.0725
ViT-16	82.40	5.8	73.1	0.0744

Jetson Orin 模型	干净准确率（%）	对抗准确率（%）	鲁棒性准确率（%）	PatchBlock 运行时间（秒）
VGG-19	76.60	0	65.3	0.2068
ResNet-50	79.85	8.6	72.05	0.2007
ViT-16	84.70	6.4	72.85	0.2037

Jetson Orin Nano 模型	干净准确率（%）	对抗准确率（%）	鲁棒性准确率（%）	PatchBlock 运行时间（秒）
VGG-19	75.95	0	63.65	0.3229
ResNet-50	79.60	3.25	70.65	0.3115
ViT-16	84.60	6.5	83.20	0.3161

表 III 使用 ADV-YOLO 补丁进行物体（人员）检测任务（416X416）的 PatchBlock 性能分析

Lambda Tensorbook 数据集	干净准确率（%）	对抗准确率（%）	鲁棒性平均精确率（%）	PatchBlock 运行时间（秒）
INRIA	100	14.28	92.86	0.326
CASIA	100	17.07	100	0.315

Jetson Orin 数据集	干净准确率（%）	对抗准确率（%）	鲁棒性平均精确率（%）	PatchBlock 运行时间（秒）
INRIA	100	14.28	92.86	0.782
CASIA	100	17.07	100	0.764

Jetson Orin Nano 数据集	干净准确率（%）	对抗准确率（%）	鲁棒性平均精确率（%）	PatchBlock 运行时间（秒）
INRIA	100	14.28	92.86	1.384
CASIA	100	17.07	100	1.352

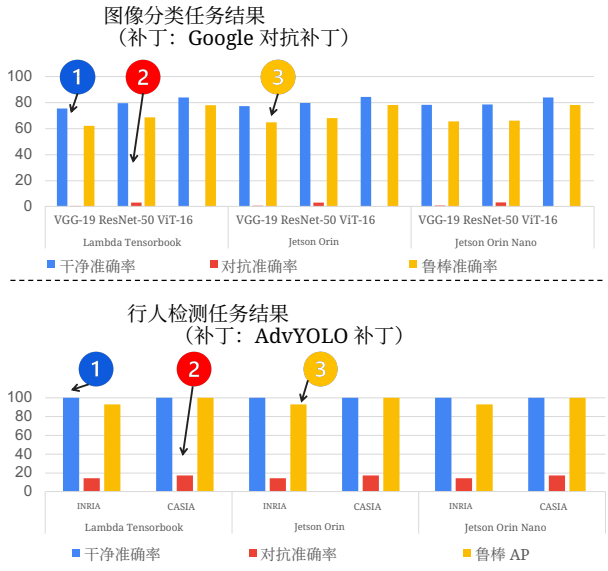

图 5. PatchBlock 在对抗贴片攻击下的图像分类和行人检测任务中的表现。结果展示了干净准确率、对抗准确率（使用 GAP 或 AdvYOLO 补丁）以及应用 PatchBlock 作为预处理防御后获得的鲁棒准确率。PatchBlock 在不同模型、数据集和设备上始终能够恢复准确率，展现出其可移植性和有效性。

E. 主要发现

我们在图5中展示了所有实验的关键发现。在不同机器学习任务、神经网络模型、对抗补丁和边缘人工智能设备上，结果一致表明 PatchBlock 能够有效提升鲁棒性，获得较高的鲁棒准确率（黄色指针 3），显著高于对抗准确率（红色指针 2），并几乎缩小了与干净基线准确率（蓝色指针 1）之间的差距。

F. PatchBlock 与最先进水平的比较

为了将我们的工作与最先进水平进行比较，我们回顾了针对对抗补丁的对抗防御技术，这些技术大致可分为两类：认证防御和经验防御。认证防御包括去随机化平滑（*DS*） [24] 和 *PatchGuard* [25]。经验防御包括降维 [31], *Jujutsu* [34], *ODDR* [35], 异常检测[36], *FNC* [37] 以及局部梯度平滑（*LGS*） [26]等方法。

表IV 使用ResNet-50模型在ImageNet数据集上针对GAP[6] 攻击时，我们提出的防御方法与四种最先进防御方法的性能比较。

防御技术	鲁棒性准确率（%）	防御技术	鲁棒性准确率（%）
局部梯度平滑 [26]	53.86%	Jujutsu [34]	60%
去随机化平滑 [24]	35.02%	FNC [37]	59.6%
PatchGuard [25]	30.96%	Anomaly Unveiled [36]	67.10%
降维 [38]	66.2%	**PatchBlock（我们的方法）**	**68.65%**

PatchBlock 的鲁棒准确率达到 68.65%，优于 Jujutsu（60%）和特征范数裁剪（59.6%）等现有防御方法。除了准确率提升之外，PatchBlock 的计算需求也更低，因此与 ODDR 等更复杂的防御方法相比，更适合部署在资源受限的 EdgeAI设备上。表 V 中总结的运行时间结果进一步凸显了 PatchBlock 的效率优势。

效率对比。在所有防御方法中，PatchBlock（PB）的运行时间和能耗最低。每个样本的执行时间：PB = 0 317 = 0 431 [39] = 1 133 。s，ODDR . s，JEDI . s。每批次的能耗：PB = 15.83 J，ODDR = 20.57 J，JEDI = 48.87 J。能耗值根据测得的设备功率（通过 Tensorbook 上的板载传感器获取）随时间积分估算得出。因此，在这两项效率指标上，PatchBlock 均显著优于 ODDR 和 JEDI。

V. 讨论

本节将探讨 PatchBlock 超参数的调优，并分析其优化对运行效率的影响。

A. PatchBlock 的超参数调优

以下是 PatchBlock 使用的主要超参数：核大小 k：核大小决定了分块过程中生成的块的尺寸。较大的核大小会减少块的数量，从而加快处理速度，但可能降低检测准确率。相反，较小的核大小能提供更精细的粒度，但会增加计算开销。通过实验发现，50 像素的核大小能够有效平衡准确率与运行时间。

信息保留 $info$：在缓解过程中，对已识别的异常图像块应用奇异值分解（SVD）进行降维。$info$ 参数指定了降维后保留的变化比例。保留原始信息的 85% $-$ 90% 是最佳选择，既能充分缓解对抗补丁的影响，又能保持干净数据的完整性。

离群值分数阈值 c：离群值分数用于在分离过程中利用孤立森林算法确定将图像块识别为异常的阈值。最终选定 1% 的最佳阈值，从而可靠地检测对抗区域，同时避免过多噪声。

B. PatchBlock 优化对运行时间的影响

实验结果表明，在 Lambda Tensorbook（配备 RTX 3080Ti m GPU 和 Intel Core i7-12800H CPU）上，所提出的局部互信息方法可将每张图像的互信息计算时间从约 250 ms 缩短至 60 ms，同时不影响防御机制的有效性。这一改进实现了实时处理，使该防御方法更适合部署。表 V 报告了针对图像分类任务，在不同设备上 PatchBlock 防御及所用相应神经网络模型的绝对运行时间。此处所列数字对应单个样本。对于目标检测这一任务，由于运行时间占比非常相似，我们也采用了相同的批处理策略。

表 V PatchBlock 与各设备上模型运行时间分析

时间（秒）	Lambda Tensorbook		Jetson Orin		Jetson Orin Nano	
模型	PatchBlock	模型	PatchBlock	模型	PatchBlock	模型
VGG-19	0.0716	0.013	0.2052	0.0534	0.3174	0.1125
ResNet-50	0.0738	0.0179	0.2038	0.0613	0.3177	0.0722
VIT-16	0.0729	0.0227	0.1984	0.0947	0.3160	0.0915

值得注意的是，PatchBlock 在 CPU 核心上运行，而模型本身在 GPU 核心上运行。为了简化这一流程，并通过缩短 CPU/GPU 的空闲时间来优化资源利用率，我们每次使用一批 2-4 个样本，在 CPU 核心上通过 PatchBlock 进行处理，然后再将防御后的样本输入运行于 GPU 上的模型。这样做的依据是，PatchBlock 算法的绝对运行时间为神经网络模型绝对运行时间的 2-4 倍。PatchBlock 防御及相应神经网络模型的运行时间趋势在所有机器学习任务和设备上均保持一致，因此采用批处理的策略同样十分有效。

VI. 结论

本文介绍了 PatchBlock，这是一种轻量级预处理防御方法，旨在抵御基于补丁的对抗攻击，同时满足在嵌入式边缘人工智能设备上部署的实用性。PatchBlock 通过由分块、分离和缓解组成的三阶段流程，在保留与任务相关特征的同时，实现对对抗补丁的准确检测与抑制。该方法在孤立树中采用定向切分策略，并将其汇总为孤立森林，以定位可疑区域，随后通过降维有效缓解对抗影响。与需要针对每项具体任务、模型和数据集进行大量训练的对抗补丁不同，PatchBlock 不受这些因素限制，并展现出在不同任务、架构及多种补丁攻击策略下的强泛化能力。

致谢

本研究部分由纽约大学阿布扎比分校网络安全中心（CCS）资助，资金由 Tamkeen 根据纽约大学阿布扎比分校研究所奖 G1104 提供；另有技术创新研究院（TII）通过“CASTLE：机器学习系统物联网的跨层安全”项目提供资助。

参考文献

[1] I. J. Goodfellow、J. Shlens 和 C. Szegedy，“阐释并利用对抗样本”，CoRR，第 abs/1412.6572 卷，2015 年。[2] A. Guesmi、M. A. Hanif、B. Ouni 和 M. Shafique，“基于摄像头的智能系统中的物理对抗攻击：当前趋势、分类、应用、研究挑战与未来展望”，IEEE Access，2023 年。[3] A. Guesmi、M. A. Hanif 和 M. Shafique，“Advrain：用于攻击基于摄像头的智能视觉系统的对抗雨滴”，Information，第 14 卷，第 12 期，第 634 页，2023 年。[4] Y.-C.-T. Hu、J.-C. Chen、B.-H. Kung、K.-L. Hua 和 D. S. Tan，“面向目标检测器的自然物理对抗补丁”，载于 2021年IEEE/CVF国际计算机视觉会议（ICCV），2021 年，第 7828–7837 页。[5] A. Guesmi、R. Ding、M. A. Hanif、I. Alouani 和 M. Shafique，“Dap：用于逃避行人检测器的动态对抗贴片”，载于 IEEE/CVF 计算机视觉与模式识别会议论文集，2024 年，第 24 595–24 604 页。[6] T. Brown，“对抗补丁”，2017 年。[在线]。网址：https://arxiv.org/pdf/1712.09665.pdf [7] D. Karmon，“Lavan：局部可见对抗噪声”，载于国际机器学习会议，2018 年。[8] A. Guesmi、M. A. Hanif、I. Alouani、B. Ouni 和 M. Shafique，“Ssap：一种形状敏感对抗贴片，可全面干扰自主导航应用中的单目深度估计”，arXiv 预印本 arXiv:2403.11515，2024 年。[9] X. Li 和 S. Ji，“生成式动态贴片攻击”，arXiv 预印本 arXiv:2111.04266，2021 年。[10] M. Rohith、A. Sunil 等，“边缘计算与边缘设备的比较分析：物联网和计算机视觉应用中的关键技术”，载于 2021 年电子、信息、通信与技术近期趋势国际会议（RTEICT）。IEEE，2021 年，第 722–727 页。[11] H. Zhang 和 J. Wang，“使用基于特征散射的对抗训练防御对抗攻击”，神经信息处理系统进展，第 32 卷，2019 年。[12] N. Papernot、P. McDaniel、X. Wu、S. Jha 和 A. Swami，“蒸馏作为抵御深度神经网络对抗扰动的防御手段”，载于 2016 年IEEE安全与隐私研讨会（SP）。IEEE，2016 年，第 582–597 页。[13] Z. Yan、Y. Guo 和 C. Zhang，“深度防御：通过改进对抗鲁棒性训练深度神经网络”，神经信息处理系统进展，第 31 卷，2018 年。[14] A. Bochkovskiy、C.-Y. Wang 和 H.-Y. M. Liao，“YOLOv4：目标检测的最佳速度与准确率”，arXiv 预印本 arXiv:2004.10934，2020 年。[15] S. Thys、W. V. Ranst 和 T. Goedem´e，“愚弄自动监控摄像头：用于攻击行人检测的对抗补丁”，CoRR，第 abs/1904.08653 卷，2019 年。 [在线]。网址：http://arxiv.org/abs/1904.08653 。[16] N. Dalal 和 B. Triggs，“用于人体检测的方向梯度直方图”，载于 2005年IEEE计算机学会计算机视觉与模式识别会议（CVPR’05），第 1 卷，2005 年，第 886–893 页，第 1 卷。[17] A. Kraskov、H. St¨ogbauer 和 P. Grassberger，“互信息估计”，《物理评论E》—统计、非线性与软物质物理，第 69 卷，第 6 期，第 066138 页，2004 年。[18] K. He、X. Zhang、S. Ren 和 J. Sun，“用于图像识别的深度残差学习”，载于 IEEE 计算机视觉与模式识别会议论文集，2016 年，第 770–778 页。[19] K. Simonyan 和 A. Zisserman，“用于大规模图像识别的超深卷积网络”，arXiv 预印本 arXiv:1409.1556，2014 年。[20] L. Yuan、Y. Chen、T. Wang、W. Yu、Y. Shi、Z.-H. Jiang、F. E. Tay、J. Feng 和 S. Yan，“Tokens-to-Token ViT：从头开始在 ImageNet 上训练视觉变换器”，载于 IEEE/CVF 国际计算机视觉会议论文集，2021 年，第 558–567 页。[21] J. Deng、W. Dong、R. Socher、L.-J. Li、K. Li 和 L. Fei-Fei，“ImageNet：一个大规模层次化图像数据库”，载于 2009年IEEE计算机视觉与模式识别会议，2009 年，第 248–255 页。[22] S. Yu、D. Tan 和 T. Tan，“一种评估视角、服装及携带条件对步态识别影响的框架”，载于第 18届国际模式识别会议（ICPR’06），第 4 卷。IEEE，2006 年，第 441–444 页。[23] L. Jing、R. Wang、W. Ren、X. Dong 和 C. Zou，“Pad：一种与贴片无关的对抗贴片攻击防御方法”，2024 年。 [在线]。网址：https://arxiv.org/abs/2404.16452

[24] A. Levine 和 S. Feizi，“用于抵御贴片攻击的（去）随机平滑认证防御”，神经信息处理系统进展，第 33 卷，第 6465–6475 页，2020 年。[25] C. Xiang、A. N. Bhagoji、V. Sehwag 和 P. Mittal，“{PatchGuard}：通过小感受野和掩蔽实现可证明鲁棒的对抗补丁防御”，载于第 *30* 届 *USENIX* 安全研讨会（*USENIX Security 21*），2021 年，第 2237–2254 页。[26] M. Naseer、S. Khan 和 F. Porikli，“局部梯度平滑：抵御局部对抗攻击”，载于 *2019* 年 *IEEE* 冬季计算机视觉应用会议（*WACV*）。IEEE，2019 年，第 1300–1307 页。[27] F. T. Liu、K. M. Ting 和 Z.-H. Zhou，“孤立森林”，载于 *2008* 年第八届 *IEEE* 国际数据挖掘会议。IEEE，2008 年，第 413–422 页。[28] S. Hariri、M. C. Kind 和 R. J. Brunner，“扩展孤立森林”，*IEEE* 知识与数据工程汇刊，第 33 卷，第 4 期，第 1479–1489 页，2019 年。[29] S. Garg、N. Chattopadhyay 和 A. Chattopadhyay，“利用降维实现自动驾驶汽车的鲁棒感知”，载于*2022* 年 *IEEE* 计算与通信中的信任、安全与隐私国际会议（*TrustCom*）。IEEE，2022 年，第 1516–1521 页。[30] D. Freedman、R. Pisani 和 R. Purves，《统计学》，2007 年，*ISBN*：*0-393970-833*，1978 年。[31] N. Chattopadhyay、A. Guesmi、M. A. Hanif、B. Ouni 和 M. Shafique，“利用降维抵御对抗补丁”，载于设计自动化会议。ACM，2024 年，第 222:1–222:6 页。[32] M. Ditty，“英伟达 Orin 片上系统”，载于*2022* 年 *IEEE Hot Chips* 第 *34* 届研讨会（*HCS*）。IEEE 计算机学会，2022 年，第 1–17 页。[33] Z. Liu、X. Liu、J. Ma 和 H. Gao，“孤立森林的优化计算框架”，工程中的数学问题，第 2018 卷，第 1 期，第 2318763 页，2018 年。[34] Z. Chen、P. Dash 和 K. Pattabiraman，“Jujutsu：一种针对深度神经网络对抗贴片攻击的两阶段防御方法”，载于 *2023* 年 *ACM* 亚洲计算机与通信安全会议论文集，丛书：ASIA CCS ’23。美国纽约：计算机协会，2023 年，第 689–703 页。[35] N. Chattopadhyay、A. Guesmi、M. A. Hanif、B. Ouni 和 M. Shafique，“ODDR：基于离群点检测与降维的对抗补丁防御”，载于*IEEE/CVF* 国际计算机视觉会议论文集，2025 年，第 22 999–23 008 页。[36] N. Chattopadhyay、A. Guesmi 和 M. Shafique，“Anomaly Unveiled：保障图像分类免受对抗贴片攻击”，*arXiv* 预印本 *arXiv:2402.06249*，2024 年。[37] C. Yu、J. Chen、Y. Xue、Y. Liu、W. Wan、J. Bao 和 H. Ma，“通过裁剪特征范数抵御通用对抗贴片”，载于*IEEE/CVF* 国际计算机视觉会议论文集，2021 年，第 16 434–16 442 页。[38] N. Chattopadhyay、A. Guesmi、M. A. Hanif、B. Ouni 和 M. Shafique，“利用降维抵御对抗补丁”，载于设计自动化会议。ACM，2024 年，第 222:1–222:6 页。[39] B. Tarchoun、A. Ben Khalifa、M. A. Mahjoub、N. Abu-Ghazaleh 和 I. Alouani，“JEDI：基于熵的对抗补丁定位与移除”，载于*IEEE/CVF* 计算机视觉与模式识别会议论文集，2023 年，第 4087–4095 页。