

PatchBlock: A Lightweight Defense Against Adversarial Patches for Embedded EdgeAI Devices

Nandish Chattopadhyay^{1,*}, Abdul Basit^{1,*}, Amira Guesmi¹, Muhammad Abdullah Hanif¹,
Bassem Ouni^{2,†}, Muhammad Shafique¹

¹ eBRAIN Lab, Division of Engineering, New York University (NYU) Abu Dhabi, UAE
² DakAI, Dubai, UAE

Abstract—Adversarial attacks pose a significant challenge to the reliable deployment of machine learning models in EdgeAI applications, such as autonomous driving and surveillance, which rely on resource-constrained devices for real-time inference. Among these, patch-based adversarial attacks, where small malicious patches (e.g., stickers) are applied to objects, can deceive neural networks into making incorrect predictions with potentially severe consequences. In this paper, we present PatchBlock, a lightweight framework designed to detect and neutralize adversarial patches in images. Leveraging outlier detection and dimensionality reduction, PatchBlock identifies regions affected by adversarial noise and suppresses their impact. It operates as a pre-processing module at the sensor level, efficiently running on CPUs in parallel with GPU inference, thus preserving system throughput while avoiding additional GPU overhead. The framework follows a three-stage pipeline: splitting the input into chunks (Chunking), detecting anomalous regions via a redesigned isolation forest with targeted cuts for faster convergence (Separating), and applying dimensionality reduction on the identified outliers (Mitigating). PatchBlock is both model- and patch-agnostic, can be retrofitted to existing pipelines, and integrates seamlessly between sensor inputs and downstream models. Evaluations across multiple neural architectures, benchmark datasets, attack types, and diverse edge devices demonstrate that PatchBlock consistently improves robustness, recovering up to 77% of model accuracy under strong patch attacks such as the Google Adversarial Patch, while maintaining high portability and minimal clean accuracy loss. Additionally, PatchBlock outperforms the state-of-the-art defenses in efficiency, in terms of computation time and energy consumption per sample, making it suitable for EdgeAI applications.

Index Terms—Robustness, adversarial patch attacks, EdgeAI, lightweight defense, outlier detection, dimension reduction

I. INTRODUCTION

Adversarial attacks represent a significant challenge to the reliable deployment of deep neural networks (DNNs) in real-world applications [1]. By injecting carefully crafted perturbations into input data, these attacks exploit vulnerabilities in neural architectures and can trigger severe misclassifications. Among these, *patch-based adversarial attacks* are concerning due to their localized nature and practical feasibility [2]. Unlike distributed noise attacks, patch attacks manipulate specific regions of an image, often using conspicuous physical stickers, that can consistently deceive models with minimal adversarial effort. Such attacks have been demonstrated to undermine classification, detection, and depth estimation pipelines, raising serious risks for safety-critical domains such as autonomous driving, surveillance, and medical diagnostics [3]–[9].

The threat is further amplified in *EdgeAI* deployments, where models operate on resource-constrained devices. These systems,

integral to autonomous vehicles, IoT surveillance, and mobile robots, must balance accuracy, latency, and power consumption under tight constraints [10]. Conventional defense strategies, including adversarial training, model ensembles, and certified robustness [11]–[13], impose heavy computational costs that are impractical in such environments. This motivates the need for *lightweight and portable* defenses that can provide practical robustness without retraining or specialized hardware.



Fig. 1: PatchBlock in action: detecting and mitigating adversarial patches in real time. Example shows object detection using YOLOv4 [14] under AdvYOLO [15] attack on INRIA dataset [16].

Our study of adversarial patch behavior reveals distinctive information-theoretic signatures. Using Mutual Information (MI) [17], we quantify localized dependencies across color channels in sliding-window chunks of an image. As illustrated in Fig. 1, regions overlapping with adversarial patches exhibit abnormally high MI values, clearly separating them from clean regions. This observation motivates our defense strategy, illustrated quantitatively in Fig. 2, where adversarial regions can be efficiently identified as statistical outliers.

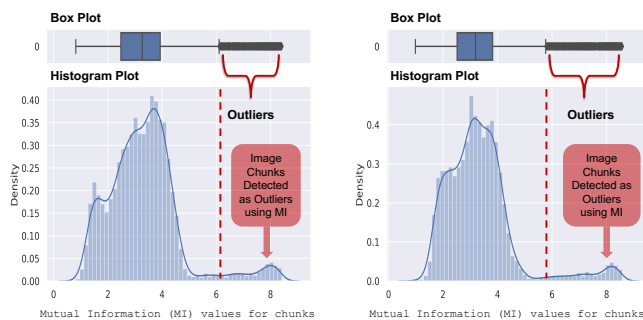


Fig. 2: Key insight behind PatchBlock: adversarial patches exhibit distinct statistical distributions captured via localized Mutual Information (MI) analysis. The plots show MI statistics computed over sliding-window chunks for a single image subjected to an AdvYOLO [15] adversarial patch attack, demonstrating that adversarial regions form clear statistical outliers compared to clean regions.

Building on this insight, we propose **PatchBlock**, a lightweight pre-processing framework that detects and neutral-

*These authors contributed equally to this work.

†Formally affiliated with Technology Innovation Institute (TII), UAE.

izes adversarial patches before images are fed into downstream models. PatchBlock combines: (i) *chunking*, where inputs are partitioned into local windows; (ii) *separating*, which applies a redesigned Isolation Forest with targeted cuts for faster convergence; and (iii) *mitigating*, where dimensionality reduction via Singular Value Decomposition (SVD) neutralizes anomalous regions. Unlike defenses that require model retraining or GPU-bound certified verification, PatchBlock runs efficiently on CPUs and overlaps with GPU inference, preserving throughput in embedded deployments, as illustrated in Fig. 3.

The novel contributions of this paper are:

- We propose **PatchBlock**, a lightweight and model-agnostic adversarial defense that detects and mitigates adversarial patches by localizing affected regions and applying dimensionality reduction to neutralize their influence. PatchBlock is designed as a pre-processing block that can be retrofitted into existing pipelines without retraining.
- PatchBlock leverages statistical methods, combining localized Mutual Information analysis with Isolation Forests and Singular Value Decomposition (SVD), to identify anomalous regions efficiently. By operating primarily on CPUs and requiring no additional GPU-based computation, PatchBlock is highly suitable for deployment in resource-constrained EdgeAI environments.
- To reduce computational overhead, we modify the Isolation Forest with a targeted-cut strategy, which accelerates convergence compared to standard implementations. In addition, we optimize MI computation method that exploits localized dependencies and parallelized processing, improving practicality for real-time defense.
- We conduct comprehensive evaluations of PatchBlock across multiple neural network models (ResNet-50 [18], VGG-19 [19], Vision Transformers [20], YOLOv4 [14]), datasets (ImageNet [21], INRIA [16], CASIA [22]), and adversarial patch attacks (Google Adversarial Patch (GAP) [6], AdvYOLO [15]). Results demonstrate that PatchBlock consistently improves robustness on diverse architectures and edge devices, achieving up to 77% accuracy recovery against strong patch attacks while maintaining minimal clean performance degradation.

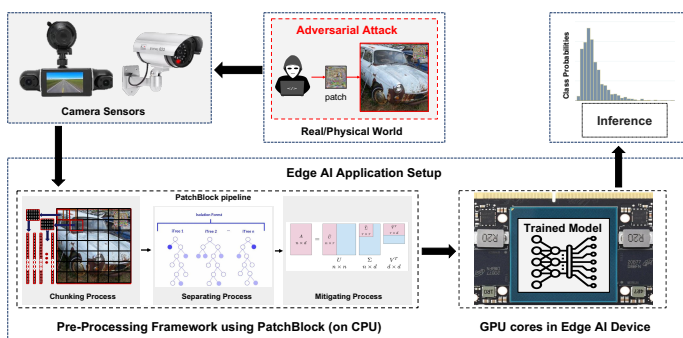


Fig. 3: Overview of the PatchBlock defense pipeline, deployed between sensor inputs and downstream inference engines in embedded EdgeAI devices.

II. BACKGROUND

In this section, we outline the theoretical underpinnings and design choices motivating the PatchBlock defense framework.

A. Threat Model: Adversarial Patch Attacks

Adversarial patch attacks are localized perturbations where an attacker inserts conspicuous patterns (e.g., stickers) into

an image to mislead DNNs. These attacks have been shown to compromise classification, detection, and depth estimation pipelines, with serious implications for autonomous driving, surveillance, and medical diagnostics [2], [3], [5]–[9], [15], [23]. We consider a white-box threat model, consistent with prior works [24]–[26], where the attacker knows the model architecture and other details. Although adaptive attackers may reduce robustness margins, PatchBlock demonstrates resilience against strong patch-based attacks.

B. Mutual Information

Mutual Information (MI) quantifies the amount of information shared between two random variables. For X and Y , MI is defined as: $I(X; Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right)$ where $p(x, y)$ is the joint distribution and $p(x)$, $p(y)$ are the marginals. In images, MI captures statistical dependencies across local regions. Adversarial patches often introduce abnormal correlations, leading to MI values that deviate from those of clean regions. This provides a useful statistical signal for detection. Formally, for a patch $\mathbf{P}_{i,j}$ and its neighbors $\mathbf{N}_{i,j}$ within image \mathbf{I} : $I(\mathbf{P}_{i,j}; \mathbf{I}) = I(\mathbf{P}_{i,j}; \bigcup_{(k,l)} \mathbf{P}_{k,l})$ Although computing global MI is computationally expensive, PatchBlock employs an efficient localized approximation that captures these anomalies while remaining deployable in real time.

C. Outlier Detection

Isolation Forest (IF) is an ensemble-based anomaly detection algorithm that recursively partitions the feature space using randomly generated decision trees [27], [28]. The principle is that anomalies are rare and distinct, requiring fewer partitions to isolate. In PatchBlock, adversarial patches represent such anomalies: they occupy a small region of the image while differing statistically from clean content. IF is therefore a natural choice for patch localization, as it is unsupervised, does not rely on labeled data, and remains computationally lightweight. To further improve efficiency, we introduce a *targeted-cut* modification to IF. Instead of random splits, targeted cuts bias the partitioning process toward statistically informative regions, reducing tree depth and accelerating convergence. This design makes IF more practical for resource-constrained EdgeAI settings.

D. Dimensionality Reduction

Dimensionality reduction methods mitigate adversarial noise by projecting data into a lower-dimensional subspace while retaining essential information. PatchBlock uses Singular Value Decomposition (SVD) [29], [30], where the ratio of singular values is used as an estimate of information retention [31]. By attenuating anomalous components, SVD effectively suppresses patch influence while preserving clean features. While simpler operations such as zeroing out or blurring could also be applied, SVD provides a principled balance between robustness and fidelity [32], [33]. This choice helps PatchBlock neutralize patches without heavily degrading clean performance.

E. Edge Devices and Applications

EdgeAI platforms such as NVIDIA Jetson AGX Orin, AGX Xavier, and Orin Nano [34] are widely used for autonomous driving, robotics, and surveillance systems. These devices must process sensor data under tight computational, memory, and energy constraints. Representative applications include:

- **Autonomous Driving:** Real-time analysis of multimodal data (camera, LiDAR, radar) for environment perception, object detection, and navigation.
- **Surveillance:** On-device video analytics for anomaly detection, object tracking, and facial recognition, while reducing reliance on cloud resources.

For such use cases, a defense mechanism must improve robustness without introducing prohibitive latency or energy costs. PatchBlock is explicitly designed to meet this requirement: it executes efficiently on CPUs, runs in parallel with GPU inference, and introduces negligible system-level overhead.

III. PATCHBLOCK ALGORITHM

The implementation of the PatchBlock algorithm is carried out in three phases. The first process involves splitting up of the input image into chunks using a moving window, called the Chunking Process. This step is characterised by the size of the kernel and the stride length of the moving window. The second part deals with using Isolation Forest to detect anomalous chunks within the image that potentially contain the adversarial patch, explained in details in Algorithms 2, 3 and 4. The final step of the process is to use dimensionality reduction with Singular Value Decomposition to mitigate the regions which may have contained the patch. This three step pipeline is outlined in Algorithm 1, and the corresponding schematic diagram is presented in Fig. 4.

Algorithm 1: PatchBlock Algorithm

IN: I : Unprocessed image sample, T : trees ensemble size in Isolation Forest, c : outlier score, $info$: SVD parameter, k : size of kernel, str : stride size
OUT: \hat{I} : Processed image sample
*/*ChunkingProcess*/*
 Generate n image chunks $X \leftarrow (x_1, \dots, x_n)$ from image I , using size of kernel = k and stride size = str
*/*SeparatingProcess*/*
 Set $s = 0.3 \times size(X)$
 $iForest = FastIsolationForest(X, T, s)$
 initialize $Y = (y_1, \dots, y_n)$
 for $i = 1$ to n do:
 $y_i = AnomalyScore(x_i)$, where $x_i \in X$ [From Algorithm 2]
 end for
 Get Y sorted in descending order
 Pick anomalies $(z_1, \dots, z_r) \rightarrow Z$ as $(1 - c) * n$ top y_i s
*/*MitigatingProcess*/*
 for $j = 1$ to r do:
 $M \leftarrow \{m_1, \dots, m_r\}$
 such that $m_j \leftarrow x_i$ for matching pairs of (i, j)
 Perform Dimension Reduction using SVD as:
 for $k = 1$ to r do:
 $\hat{m}_k \leftarrow SVD(m_k, info)$
 Superimpose \hat{m}_k -s in place of m_k -s in image I to get \hat{I}
 return \hat{I}

For the purpose of detecting the anomalies, PatchBlock uses an ensemble of Fast Isolation Trees, which is a Fast Isolation Forest and is described in detail in Algorithm 2. Specifically, we have proposed a significantly efficient version of the Isolation Tree, which we call Fast Isolation Tree, that uses targeted cuts on the attributes of the dataset, instead of random cuts, for quicker convergence and therefore faster detection of potential anomalies [35]. The Fast Isolation Tree algorithm is described in Algorithm 3, wherein it uses a function called *GradientSplit*, which is used to make the targeted cuts.

A. Targeted Cut in Isolation Trees

The distributions of attribute values can be used as a foundation for examining the differences between two types of instances. The underlying concept is that the cumulative discrepancy in value distributions across various attributes can

Algorithm 2: FastIsolationForest($X, T, size$)

IN: $X = (x_1, \dots, x_n)$: input data, T : number of trees in the forest, s : sample size
OUT: *FastIsolationForest* (*FastIsolationTrees* ensemble)
 initialize : *FastIsolationForest*
 set $height_{max} = ceiling(\log_2 s)$
 for $i = 1$ to T do:
 $X' \leftarrow Sample(X, s)$
 $FastIsolationForest \leftarrow FastIsolationForest \cup FastIsolationTree(X', 0, height_{max})$
 end for
 return *FastIsolationForest*

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

IN: $X = (x_1, \dots, x_n)$: input data, $height$: height of tree, $height_{max}$: maximum height of tree, Q : attributes of X
OUT: a *FastIsolationTree*
 if $height \geq height_{max}$ or $|X| \leq 1$, then:
 return *externalNode*{ $Size \leftarrow |X|$ }
 else
 Randomly select $k(k \leq q)$ distinct attributes from Q
 Pick out attribute \hat{q} which has highest value of separability index using Algorithm 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

indicate the extent of the difference between anomalous and normal instances. The separability of an attribute is influenced by two factors: the distance between the peaks and the spread of values within each category of instances. In this case, the values for each type of instance are averaged (denoted by $E()$) to obtain a mean value, representing the center of the values. The variance function ($Var()$) is once again used to measure the degree of dispersion in the values. These two metrics are used to generate the function to calculate the separability index $Separation(X^q, v)$, which is given as:

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

An approximately optimal method is proposed for searching the splitting point for a given attribute. By calculating the gradients of separability index values between adjacent attribute values, the method can help bypass attribute values unlikely to be selected as split points, thus accelerating the search process. This gradient calculating function G is given as:

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

The function used to update the *step* parameter is defined as:

$$step = \begin{cases} \frac{3}{1+e^{G * \log_{10} |X^q|}} * \frac{|X^q|}{100} & \text{if } G < 0 \\ 0.7 - \frac{1.3}{1+e^{G * \log_{10} |X^q|}} * \frac{|X^q|}{100} & \text{if } G \geq 0 \end{cases} \quad (2)$$

These two functions are used in the following gradient splitting algorithm for finding out the attribute with highest separability index.

B. Efficient MI Computation for Image Chunks

For the task of defending against patch-based adversarial attacks, our defense mechanism involves identifying and re-constructing image chunks that exhibit anomalous behavior. A

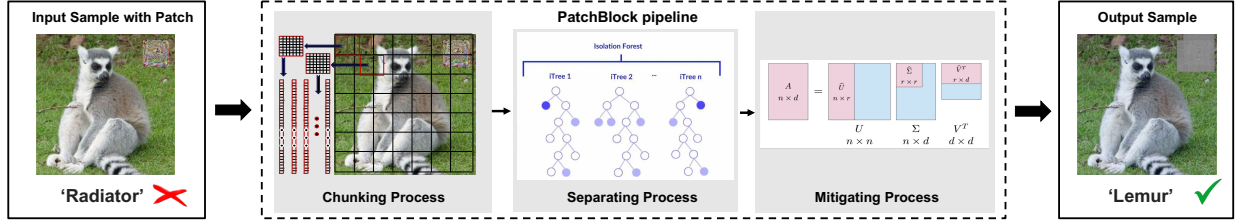


Fig. 4: PatchBlock Pipeline: The three processes of Chunking (using moving window to obtain kernels, converted to vectors), Separating (using Mutual Information and Fast Isolation Forests with targeted cuts by gradient splitting) and Mitigating (using Singular Value Decomposition).

Algorithm 4: GradientSplit(X)

IN: $X = (x_1, \dots, x_n)$: input data, X^q : sorted values of q -th attribute
OUT: $bestX$: attribute with highest separability index,
highestSeparation: largest separability index value
initialize: $step$ as $\lceil |X^q| * 0.001 \rceil$
Let $highestSeparation = Separation(X^q, x_1)$
Let $bestX = (x_1 + x_2)/2$ and $i = 0$
while $i < |X^q|$ **do**:
 Set $i = i + step$
 Set $currentSeparation = Separation(X^q, x_i)$
if $currentSeparation > highestSeparation$ **then**:
 Set $highestSeparation = currentSeparation$
 Set $bestX = (x_i + x_{i+1})/2$
end if
 Update $step$ using equation 2, which uses equation 1
end while
return ($bestX, highestSeparation$)

critical step in this process is the calculation of the mutual information (MI) between image chunks and the overall image, which helps in detecting regions that may have been altered by an adversarial patch. However, computing the MI between each chunk and the entire image is computationally intensive and becomes a bottleneck for real-time applications.

To address this issue, we propose an efficient method for MI computation that significantly reduces the computational load while maintaining the effectiveness of the defense mechanism. Instead of calculating the MI between each chunk and the entire image, we compute the MI between each chunk and its neighboring chunks. This approach is based on the observation that local dependencies are more relevant for detecting anomalies introduced by adversarial patches, as these patches are often localized and disrupt the local statistical properties of the image.

1) *Localized Mutual Information Computation:* We propose to compute the MI between each image chunk $\mathbf{P}_{i,j}$ and its immediate neighbors $\mathbf{N}_{i,j}$:

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

where \mathcal{P} and \mathcal{N} are the sets of possible pixel intensity values in the chunk and its neighbors, respectively.

This localized MI computation leverages the fact that natural images exhibit strong local dependencies due to spatial continuity. Anomalies introduced by adversarial patches disrupt these local dependencies, leading to significant deviations in the MI between a chunk and its neighbors.

2) *Justification for Localised MI:* The use of localized MI is theoretically sound based on the property of Markov Random Fields (MRFs) in image modeling. In an MRF, the probability of a pixel (or chunk) intensity depends primarily on its local neighborhood. Therefore, the statistical dependencies captured by the MI between a chunk and its neighbours are sufficient for detecting anomalies.

Moreover, the mutual information satisfies the property of diminishing returns with increasing neighborhood size due to the saturation of shared information. This means that adding more distant patches to the computation contributes marginally to the MI value. Mathematically, for larger neighborhoods $\mathbf{N}'_{i,j} \supseteq \mathbf{N}_{i,j}$: $I(\mathbf{P}_{i,j}; \mathbf{N}'_{i,j}) \approx I(\mathbf{P}_{i,j}; \mathbf{N}_{i,j})$ for sufficiently large images where the mutual information between distant chunks is minimal.

3) *Computational Efficiency:* By restricting MI computation to neighboring chunks, we reduce the complexity from $O(N^2)$ to $O(N)$, where N is the number of chunks in the image. We also employ parallel processing to compute MI scores for all chunks simultaneously. Let K be the number of neighbors for each chunk. The total number of MI computations is $O(N \cdot K)$, which is far less than the original $O(N^2)$ computations required when comparing each chunk to the entire image.

IV. EXPERIMENTAL RESULTS

We evaluate the effectiveness of the proposed PatchBlock defense across diverse models, datasets, and adversarial patch attacks. The goal is to demonstrate robustness, adaptability, and portability in both image classification and object detection tasks on embedded EdgeAI devices. All chosen benchmarks and attack strategies are widely used in prior work, ensuring fair comparison and reproducibility.

A. Experimental Setup

PatchBlock is deployed as a pre-processing module on CPU cores, while the underlying deep learning models run on GPU cores. This setup reflects realistic deployment scenarios where lightweight defenses operate alongside high-throughput inference engines.

Image Classification: We evaluate classification robustness using the ImageNet dataset [21], which provides diverse classes and challenging variations. The backbone models include ResNet-152 and ResNet-50 [18], VGG-19 [19], and Vision Transformers (ViT) [20], covering both convolutional and transformer-based architectures.

Person Detection: For detection, we focus on the Person Detection task, a subdomain where adversarial patches are especially effective. We use the INRIA dataset [16], which captures diverse real-world conditions, and the CASIA dataset [22], which includes multiple patch instances per frame. YOLOv4 [14] serves as the detection backbone due to its widespread adoption and high efficiency in embedded settings.

B. Adversarial Patch Attacks

We evaluate PatchBlock against state-of-the-art adversarial patch attacks.

- **Classification:** Google Adversarial Patch (GAP) [6] and LAVAN [7], which represent strong white-box attack strategies.

- **Detection:** AdvYOLO patch [15], specifically designed to compromise YOLO-based detectors.

These attacks cover a range of patch designs and objectives, providing a rigorous test of the defense mechanism.

C. EdgeAI Devices

Experiments are conducted on two widely used embedded EdgeAI platforms:

- **NVIDIA Jetson Orin:** 12-core ARM Cortex-A78AE CPU and NVIDIA Ampere GPU with up to 2048 CUDA cores.
- **NVIDIA Jetson Orin Nano:** 6-core ARM Cortex-A78AE CPU and NVIDIA Ampere GPU with 1024 CUDA cores.
- **Lambda Tensorbook:** Intel Core i7-11800H (8 cores, 2.3–4.6 GHz), NVIDIA 3080 Ti GPU with 16 GB VRAM.

PatchBlock runs exclusively on the CPU cores, while the classification and detection models are executed on the GPU. This separation demonstrates that PatchBlock introduces minimal overhead, enabling real-time deployment in resource-constrained environments.

D. PatchBlock Experimental Results

This section presents a detailed evaluation of the PatchBlock defense mechanism across diverse tasks, models, and devices, demonstrating its robustness, adaptability, and efficiency.

TABLE I: Performance Analysis of PatchBlock with the Google Adversarial Patch for Image Classification Task (224x224)

Lambda Tensorbook	Clean Accuracy (%)	Adversarial Accuracy (%)	Robust Accuracy (%)	PatchBlock Runtime (sec)
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	Clean Accuracy (%)	Adversarial Accuracy (%)	Robust Accuracy (%)	PatchBlock Runtime (sec)
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	Clean Accuracy (%)	Adversarial Accuracy (%)	Robust Accuracy (%)	PatchBlock Runtime (sec)
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

We evaluate PatchBlock on the ImageNet dataset using three neural network models, VGG-19, ResNet-50, and Vision Transformers (ViT-16), under the Google Adversarial Patch (GAP) attack. Table I reports clean accuracy, adversarial accuracy (with GAP applied), robust accuracy (after applying PatchBlock), and **CPU runtime** across three devices: the Lambda Tensorbook, Jetson Orin, and Jetson Orin Nano. PatchBlock substantially improved model robustness, achieving up to 77.90% robust accuracy on ViT-16 with the Tensorbook, recovering from 0% adversarial accuracy. Robust accuracy remained consistent across devices despite hardware differences; for example, VGG-19 achieved 64.80% on Jetson Orin and 65.45% on Jetson Orin Nano. Moreover, PatchBlock’s CPU runtime was well suited for resource-constrained environments, ranging from 0.0716 seconds on the Tensorbook to 0.3174 seconds on the Jetson Orin Nano for VGG-19. Minor accuracy variations across devices are not hardware-induced. Accuracy is computed on a randomly selected subset of 1,000 images from the ImageNet validation set, leading to small fluctuations (~1%) due to sampling variability, while model parameters and defense configurations remain identical across devices.

In Table II, we report the performance of PatchBlock with the LAVAN Patch for Image Classification Task on the ImageNet

TABLE II: Performance Analysis of PatchBlock with the LAVAN Patch for Image Classification Task (224x224)

Lambda Tensorbook	Clean Accuracy (%)	Adversarial Accuracy (%)	Robust Accuracy (%)	PatchBlock Runtime (sec)
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	Clean Accuracy (%)	Adversarial Accuracy (%)	Robust Accuracy (%)	PatchBlock Runtime (sec)
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	Clean Accuracy (%)	Adversarial Accuracy (%)	Robust Accuracy (%)	PatchBlock Runtime (sec)
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

dataset. The trends are consistent with those of the Google Adversarial Patch.

TABLE III: Performance Analysis of PatchBlock with the Adv-YOLO Patch for Object (Person) Detection Task (416x416)

Lambda Tensorbook	Clean Accuracy (%)	Adversarial Accuracy (%)	Robust Accuracy (%)	PatchBlock Runtime (sec)
INRIA	100	14.28	92.86	0.326
CASIA	100	17.07	100	0.315

Jetson Orin	Clean Accuracy (%)	Adversarial Accuracy (%)	Robust Accuracy (%)	PatchBlock Runtime (sec)
INRIA	100	14.28	92.86	0.782
CASIA	100	17.07	100	0.764

Jetson Orin Nano	Clean Accuracy (%)	Adversarial Accuracy (%)	Robust Accuracy (%)	PatchBlock Runtime (sec)
INRIA	100	14.28	92.86	1.384
CASIA	100	17.07	100	1.352

PatchBlock was further evaluated for person detection on the INRIA and CASIA datasets under Adv-YOLO patch attack using the YOLO-v4 model. Table III highlights clean accuracy, adversarial accuracy, robust average precision (AP), and runtime. PatchBlock restored average precision to 92.86% on INRIA and 100% on CASIA, recovering from adversarial AP values as low as 14.28%. Runtime remained within feasible limits for real-time applications, ranging from 0.326 seconds on Tensorbook to 1.384 seconds on Jetson Orin Nano for INRIA.

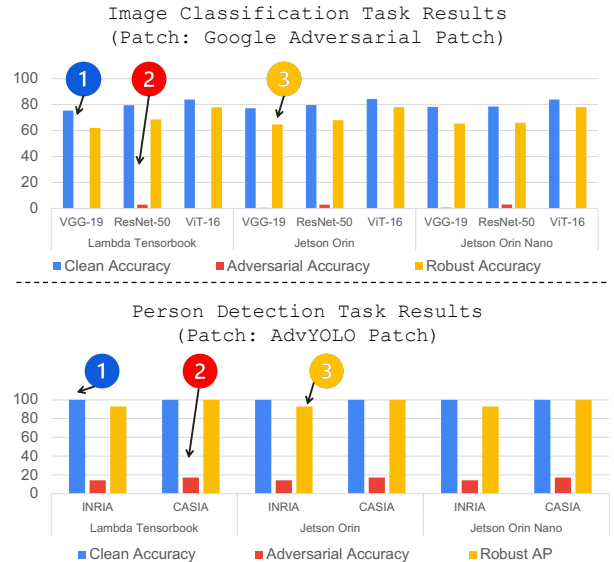


Fig. 5: PatchBlock performance across Image Classification and Person Detection tasks under adversarial patch attacks. Results show clean accuracy, adversarial accuracy (with GAP or AdvYOLO patches), and robust accuracy obtained after applying PatchBlock as a pre-processing defense. PatchBlock consistently restores accuracy across models, datasets, and devices, demonstrating its portability and effectiveness.

E. Key Findings

We present the key findings of all the experiments in Fig. 5. Consistent across machine learning tasks, neural network models, adversarial patches, and EdgeAI devices, PatchBlock provides strong robustness, achieving high Robust Accuracy **3**, which is significantly higher than the Adversarial Accuracy **2**, almost closing the gap with the Clean Baseline Accuracy **1**.

F. PatchBlock vs. State of the art

To compare our work with the state-of-the-art, we review adversarial defense techniques for adversarial patches, which can be broadly categorized into two types: certified defenses and empirical defenses. Certified defenses include *De-randomized Smoothing (DS)* [24] and *PatchGuard* [25]. Empirical defenses include methods like *Dimensionality Reduction* [31], *Jujutsu* [36], *ODDR* [37], *Anomaly Detection* [38], *FNC* [39] and *Localized Gradient Smoothing (LGS)* [26].

TABLE IV: Performance of our proposed defense compared to state-of-the-art defenses against GAP [6] attack with the ResNet-50 model on the ImageNet dataset.

Defense Technique	Robust Accuracy (%)	Defense Technique	Robust Accuracy (%)
Localised Gradient Smoothing [26]	53.86%	Jujutsu [36]	60%
De-Randomised Smoothing [24]	35.02%	FNC [39]	59.6%
PatchGuard [25]	30.96%	Anomaly Unveiled [38]	67.10%
Dimensionality Reduction [40]	66.2%	PatchBlock (Ours)	68.65%

PatchBlock achieved 68.65% robust accuracy, outperforming defenses such as Jujutsu (60%) and Feature Norm Clipping (59.6%). In addition to accuracy gains, PatchBlock has lower computational requirements, making it more suitable for deployment on resource-constrained EdgeAI devices than more complex defenses such as ODDR. The runtime results in Table V further highlight PatchBlock’s efficiency advantage.

Efficiency Comparison. PatchBlock (PB) achieves the lowest runtime and energy cost among defenses. Execution time per sample: PB = 0.317 s, ODDR = 0.431 s, JEDI [41] = 1.133 s. Energy consumption per batch: PB = 15.83 J, ODDR = 20.57 J, JEDI = 48.87 J. The energy values are estimated from measured device power (via onboard sensors on Tensorbook) integrated over time. Thus, PatchBlock outperforms ODDR and JEDI by a significant margin in both efficiency metrics.

V. DISCUSSIONS

In this section, we explore the tuning of hyper-parameters for PatchBlock and analyze the impact of its optimizations on runtime efficiency.

A. Hyper-Parameter Tuning for PatchBlock

The following are the primary hyper-parameters used in PatchBlock:

Kernel Size k: The kernel size determines the dimensions of the chunks created during the *Chunking Process*. Larger kernel sizes reduce the number of chunks, speeding up processing but potentially reducing detection accuracy. Conversely, smaller kernel sizes provide finer granularity but increase computational overhead. Through experimentation, a kernel size of 50 pixels was found to balance accuracy and runtime effectively.

Information Retention info: During the *Mitigation Process*, the dimensionality reduction via SVD is applied to identified anomalous chunks. The *info* parameter specifies the proportion of variability to retain in the reduce dimension. Retaining 85% – 90% of the original information was optimal, ensuring sufficient mitigation of adversarial patches while preserving the integrity of clean data.

Outlier Score Threshold c: The outlier score determines the threshold for identifying chunks as anomalous in the *Separating Process* using the Isolation Forest algorithm. An optimal threshold of 1% was selected, ensuring reliable detection of adversarial regions without excessive noise.

B. Impact of PatchBlock Optimizations on Runtime

Our experimental results demonstrate that the proposed use of Localized Mutual Information reduces MI computation time from approximately 250 ms to 60 ms per image on a Lambda Tensorbook, without compromising defense effectiveness. This improvement enables real-time processing and makes the defense method more practical for deployment. Table V reports the absolute runtime of the PatchBlock defense and the corresponding neural network model in use, across different devices, for the Image Classification task. The numbers mentioned here are for one sample. The same strategy of using batches was also used for the other task of Object Detection, since the proportions of runtime were very similar.

TABLE V: Analysis of Runtime of PatchBlock and Models across Devices

Time (sec)	Lambda Tensorbook		Jetson Orin		Jetson Orin Nano	
	PatchBlock	Model	PatchBlock	Model	PatchBlock	Model
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

It is noteworthy that PatchBlock runs on the CPU cores, while the model itself runs on the GPU cores. To streamline this process and optimize resource utilization by minimizing CPU/GPU idle time, we use batches of 2–4 samples to process PatchBlock on the CPU before feeding the defended samples to the model running on the GPU. This is motivated by the fact that the absolute runtime of the PatchBlock algorithm lies within 2–4 times the absolute runtime of the neural network models. This trend of Runtime for the PatchBlock defense and the corresponding neural network models is consistent across all machine learning tasks and devices, and the same strategy of using batches is equally useful.

VI. CONCLUSIONS

In this paper, we introduced PatchBlock, a lightweight pre-processing defense designed to counter patch-based adversarial attacks while practical for deployment on embedded EdgeAI devices. PatchBlock operates through a three-phase pipeline—chunking, separating, and mitigating—that enables accurate detection and suppression of adversarial patches while preserving task-relevant features. The approach leverages a targeted-cut strategy within isolation trees, aggregated into an isolation forest, to localize suspicious regions, followed by dimensionality reduction to effectively mitigate the adversarial influence. Unlike adversarial patches, which require extensive training for each task, model, and dataset, PatchBlock is agnostic to such factors and demonstrates strong generalizability across tasks, architectures, and diverse patch attack strategies.

ACKNOWLEDGMENT

This research was partially funded by the NYUAD Center for Cyber Security (CCS), funded by Tamkeen under the NYUAD Research Institute Award G1104 and the Technology Innovation Institute (TII) under the “CASTLE: Cross-Layer Security for Machine Learning Systems IoT” project.

REFERENCES

- [1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *CoRR*, vol. abs/1412.6572, 2015.
- [2] A. Guesmi, M. A. Hanif, B. Ouni, and M. Shafique, "Physical adversarial attacks for camera-based smart systems: Current trends, categorization, applications, research challenges, and future outlook," *IEEE Access*, 2023.
- [3] A. Guesmi, M. A. Hanif, and M. Shafique, "Advrain: Adversarial raindrops to attack camera-based smart vision systems," *Information*, vol. 14, no. 12, p. 634, 2023.
- [4] Y.-C.-T. Hu, J.-C. Chen, B.-H. Kung, K.-L. Hua, and D. S. Tan, "Naturalistic physical adversarial patch for object detectors," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 7828–7837.
- [5] A. Guesmi, R. Ding, M. A. Hanif, I. Alouani, and M. Shafique, "Dap: A dynamic adversarial patch for evading person detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 24 595–24 604.
- [6] T. Brown, "Adversarial patch," 2017. [Online]. Available: <https://arxiv.org/pdf/1712.09665.pdf>
- [7] D. Karmon, "Lavan: Localized and visible adversarial noise," in *International Conference on Machine Learning*, 2018.
- [8] A. Guesmi, M. A. Hanif, I. Alouani, B. Ouni, and M. Shafique, "Ssap: A shape-sensitive adversarial patch for comprehensive disruption of monocular depth estimation in autonomous navigation applications," *arXiv preprint arXiv:2403.11515*, 2024.
- [9] X. Li and S. Ji, "Generative dynamic patch attack," *arXiv preprint arXiv:2111.04266*, 2021.
- [10] M. Rohith, A. Sunil *et al.*, "Comparative analysis of edge computing and edge devices: key technology in iot and computer vision applications," in *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*. IEEE, 2021, pp. 722–727.
- [11] H. Zhang and J. Wang, "Defense against adversarial attacks using feature scattering-based adversarial training," *Advances in neural information processing systems*, vol. 32, 2019.
- [12] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 582–597.
- [13] Z. Yan, Y. Guo, and C. Zhang, "Deep defense: Training dnns with improved adversarial robustness," *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [14] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.
- [15] S. Thys, W. V. Ranst, and T. Goedemé, "Fooling automated surveillance cameras: adversarial patches to attack person detection," *CoRR*, vol. abs/1904.08653, 2019. [Online]. Available: <http://arxiv.org/abs/1904.08653>
- [16] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 886–893 vol. 1.
- [17] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, vol. 69, no. 6, p. 066138, 2004.
- [18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [20] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 558–567.
- [21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.
- [22] S. Yu, D. Tan, and T. Tan, "A framework for evaluating the effect of view angle, clothing and carrying condition on gait recognition," in *18th international conference on pattern recognition (ICPR'06)*, vol. 4. IEEE, 2006, pp. 441–444.
- [23] L. Jing, R. Wang, W. Ren, X. Dong, and C. Zou, "Pad: Patch-agnostic defense against adversarial patch attacks," 2024. [Online]. Available: <https://arxiv.org/abs/2404.16452>
- [24] A. Levine and S. Feizi, "(de) randomized smoothing for certifiable defense against patch attacks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6465–6475, 2020.
- [25] C. Xiang, A. N. Bhagoji, V. Sehwal, and P. Mittal, "{PatchGuard}: A provably robust defense against adversarial patches via small receptive fields and masking," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2237–2254.
- [26] M. Naseer, S. Khan, and F. Porikli, "Local gradients smoothing: Defense against localized adversarial attacks," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1300–1307.
- [27] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth IEEE international conference on data mining*. IEEE, 2008, pp. 413–422.
- [28] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended isolation forest," *IEEE transactions on knowledge and data engineering*, vol. 33, no. 4, pp. 1479–1489, 2019.
- [29] S. Garg, N. Chattopadhyay, and A. Chattopadhyay, "Robust perception for autonomous vehicles using dimensionality reduction," in *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2022, pp. 1516–1521.
- [30] D. Freedman, R. Pisani, and R. Purves, "Statistics. 2007," ISBN: 0-393970-833, 1978.
- [31] N. Chattopadhyay, A. Guesmi, M. A. Hanif, B. Ouni, and M. Shafique, "Defending against adversarial patches using dimensionality reduction," in *DAC*. ACM, 2024, pp. 222:1–222:6.
- [32] N. Chattopadhyay, A. Chattopadhyay, S. S. Gupta, and M. Kasper, "Curse of dimensionality in adversarial examples," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.
- [33] N. Chattopadhyay, S. Chatterjee, and A. Chattopadhyay, "Robustness against adversarial attacks using dimensionality," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2021, pp. 226–241.
- [34] M. Ditty, "Nvidia orin system-on-chip," in *2022 IEEE Hot Chips 34 Symposium (HCS)*. IEEE Computer Society, 2022, pp. 1–17.
- [35] Z. Liu, X. Liu, J. Ma, and H. Gao, "An optimized computational framework for isolation forest," *Mathematical problems in engineering*, vol. 2018, no. 1, p. 2318763, 2018.
- [36] Z. Chen, P. Dash, and K. Pattabiraman, "Jujutsu: A two-stage defense against adversarial patch attacks on deep neural networks," in *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 689–703.
- [37] N. Chattopadhyay, A. Guesmi, M. A. Hanif, B. Ouni, and M. Shafique, "Oddr: Outlier detection & dimension reduction based defense against adversarial patches," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025, pp. 22 999–23 008.
- [38] N. Chattopadhyay, A. Guesmi, and M. Shafique, "Anomaly unveiled: Securing image classification against adversarial patch attacks," *arXiv preprint arXiv:2402.06249*, 2024.
- [39] C. Yu, J. Chen, Y. Xue, Y. Liu, W. Wan, J. Bao, and H. Ma, "Defending against universal adversarial patches by clipping feature norms," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 434–16 442.
- [40] N. Chattopadhyay, A. Guesmi, M. A. Hanif, B. Ouni, and M. Shafique, "Defending against adversarial patches using dimensionality reduction," in *DAC*. ACM, 2024, pp. 222:1–222:6.
- [41] B. Tarchoun, A. Ben Khalifa, M. A. Mahjoub, N. Abu-Ghazaleh, and I. Alouani, "Jedi: Entropy-based localization and removal of adversarial patches," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4087–4095.