

SHEATH: Defending Horizontal Collaboration for Distributed CNNs against Adversarial Noise

Muneeba Asif^{*}, Mohammad Kumail Kazmi^{*}, Mohammad Ashiqur Rahman^{*}, Syed Rafay Hasan[†], Soamar Homs[‡]

^{*}Analytics for Cyber Defense (ACyD) Lab, Florida International University, USA

[†]Department of Electrical and Computer, Tennessee Technological University, USA

[‡]Information Warfare Division, AirForce Research Laboratory, USA

{masif004, mkazm004, marahman}@fiu.edu, shasan@tntech.edu, soamar.homs@us.af.mil

Abstract— As edge computing and the Internet of Things (IoT) expand, horizontal collaboration (HC) emerges as a distributed data processing solution for resource-constrained devices. In particular, a convolutional neural network (CNN) model can be deployed on multiple IoT devices, allowing distributed inference execution for image recognition while ensuring model and data privacy. Yet, this distributed architecture remains vulnerable to adversaries who want to make subtle alterations that impact the model, even if they lack access to the entire model. Such vulnerabilities can have severe implications for various sectors, including healthcare, military, and autonomous systems. However, security solutions for these vulnerabilities have not been explored. This paper presents a novel framework for Secure Horizontal Edge with Adversarial Threat Handling (SHEATH) to detect adversarial noise and eliminate its effect on CNN inference by recovering the original feature maps. Specifically, SHEATH aims to address vulnerabilities without requiring complete knowledge of the CNN model in HC edge architectures based on sequential partitioning. It ensures data and model integrity, offering security against adversarial noise in diverse HC environments. Our evaluations demonstrate SHEATH’s adaptability and effectiveness across diverse CNN configurations.

Index Terms—secure horizontal collaboration, adversarial noise detection, decentralized CNNs

I. INTRODUCTION

The internet of things (IoT) infrastructures can be integrated with artificial intelligence (AI) and machine learning (ML) to create artificial intelligence of things (AIoT). This advancement enhances urban living and sustainability in the form of developing smart cities, which address real-world problems like smart parking [1], fostering next-generation smart grid solutions [2], and enabling large-scale sensor deployments [3]. To enhance the efficiency of IoT operations using AIoT, there must be a focus on data management and analysis: collecting, delivering, and processing data, making decisions, and executing them accordingly [4]. These tasks require high computing power, leading IoT systems to leverage cloud platforms and shift computation to edge nodes for efficiency. However, IoT devices often have infrastructure limitations or failures that prevent cloud access. Moreover, communication with the cloud creates a single point of failure, and these channels are cyberattack-prone. Cloud reliance also raises privacy concerns, given the third-party access to information.

To address these security challenges, researchers have proposed distributing these tasks on edge devices or servers; this

comes with additional benefits, including latency reduction, privacy preservation, and energy efficiency [5], [6]. However, these edge IoT devices are often resource-constrained, such as low processing power, limited memory, and energy [7]. These limited processing capabilities highlight the need for external computational support. This support is offered by horizontal collaboration (HC), which enables cooperation among edge nodes in decentralized networks, thereby improving performance by sharing computational tasks [8], [9]. For example, the inference of trained convolutional neural networks (CNNs) can be partitioned onto multiple edge devices. This collaborative inference can be achieved by (i) sequentially dividing the model across multiple devices [10] or (ii) segmenting data for model parallelism followed by result amalgamation [11].

While this collaboration allows IoT devices/edge nodes to execute expensive ML inference operations, it provides several essential security benefits. For instance, each device possesses only parts of the data or model, limiting the exposure of sensitive information [12]. Such privacy is essential in areas like the military, healthcare, autonomous driving, and smart homes, where timely and accurate responses are essential. Misclassifications in these domains can lead to severe consequences, such as unintended casualties, wrong treatments, accidents, or security breaches. However, HC edge environments are still vulnerable to cyberattacks as nodes are often untrusted. Adversaries can compromise one or more nodes and inject noise into the feature maps at these nodes, thus compromising model integrity even without full model access, as shown in Fig. 1. Such a noise-based inference attack on capsule networks in an HC environment was demonstrated by Adeymo et al., causing an average accuracy drop of 62% [13].

To preserve the data and model integrity in the HC environment, the edge devices must be resilient against cyberattacks. Most existing solutions for detecting and defending against adversarial noise in the HC environment require full knowledge of the CNN [14] or involve retraining the model for increased robustness [15], [16]. However, these methods mainly apply to the segmented/amalgamated approach and not to sequential partitioning; hence, they fail to consider the dynamic aspects of HC. This is especially problematic for dynamic use cases, e.g., when the nodes are a fleet of drones. Therefore, we propose **Secure Horizontal Edge with Adversarial Threat Handling (SHEATH)**, a novel framework

Fig. 1. Horizontal CNN partitioning across IoT devices. Adversarial noise “Conv3” results in a “fox” misclassification of a “cat” image. The adversary can launch multi-node (i.e., multi-layer) attacks, as shown of “Conv1” and “Pool1” layers. Increased sparsity in deeper CNN layers intensifies the subtlety of these adversarial noises, which makes them harder to detect.

to address the aforementioned vulnerabilities in sequential partitioning-based HC edge architectures without requiring full knowledge of the CNN model.

As mentioned earlier, the different layers of the HC-based CNN model are deployed on different nodes that are often untrusted. Some of these nodes can be malicious or compromised where attacks occur, i.e., adversarial noise is added. SHEATH is a lightweight technique that defends the inference against such attacks on a selected untrusted node. SHEATH itself will be deployed on a trusted node to ensure the process. We consider a trusted node to be semi-honest, i.e., semi-honest nodes are assumed to execute the protocol correctly without malicious intent but may be curious about the data they process. This is a common assumption in secure multiparty computation and federated learning literature [17]–[19]. SHEATH comprises two modules: (i) Detect and (ii) Recover. The Detect module further consists of two parts: PseudoNet and the Comparator. PseudoNet computes the expected feature maps of the untrusted node to serve as a basis of the non-noisy (reference) feature maps for the Comparator. PseudoNets a less computational-intensive copy of the node that SHEATH targets to secure. For example, if the target node is a convolutional layer with 64 kernels, its corresponding PseudoNet will be a convolutional layer but with fewer kernels. The goal is to achieve a trade-off between accurate noise detection and heavy computational load. After this, the Comparator evaluates the difference between the outputs of both PseudoNet and the feature maps from the untrusted node. If this difference exceeds the predefined threshold, i.e., the adversarial noise is detected, the Recover module is activated. Recover infers the correct/expected feature maps and forwards them to the next node in the inference chain. The design of SHEATH ensures the integrity of the CNN model deployed on HC edge devices with an acceptable overhead and seamless integration.

However, SHEATH's deployment strategy can vary based on factors such as system architecture, computational capabilities, network bandwidth, security requirements, and specific use-case demands. We discuss multiple deployment scenarios and evaluate the framework on various CNN models and datasets. Overall, our contributions are threefold as follows:

- We conduct an in-depth analysis of the associated vulnerabilities and the implications of single and multi-layered

adversarial noise attacks on CNN models deployed on HC edge devices.

We propose SHEATH, a novel framework to detect adversarial noise and eliminate the effects of the aforementioned attacks on the HC-based CNNs by recovering the original feature maps. We also discuss various deployment strategies for SHEATH.

We comprehensively evaluate SHEATH's effectiveness across various CNN models and datasets, demonstrating its adaptability and robustness in diverse AIoT scenarios.

The rest of the paper is organized as follows: Section II elucidates edge computing in HC, potential attacks, and presents the research motivation. The threat model is discussed in Section III. Section IV presents the case studies, Section V presents the problem formulation, and Section VI discusses the proposed framework along with different deployment scenarios. Performance is empirically validated in Section VII. Pertinent literature is reviewed in Section VIII; finally, the paper is concluded in Section IX.

II. THEORETICAL FOUNDATIONS

This section overviews the distributed CNNs in an HC setup, their significance, their vulnerability to adversarial noise, and finally, highlights the motivation behind our research.

A. HC Edge Environment for Distributed CNNs

HC refers to the interaction between devices or nodes within a decentralized network. Contrary to vertical collaboration, where data is relayed from edge devices to centralized systems and back, HC utilizes multiple devices to share tasks and resources. This type of collaboration is substantial in edge computing and IoT settings where devices are resource-constrained. In traditional centralized models, a central node/server delegates the tasks to peripheral nodes. However, HC bypasses this central entity, allowing edge devices to interact directly with each other. By distributing tasks and leveraging the collective computational power of multiple devices, HC enhances system resilience and throughput, reduces latency, and optimizes resource utilization. Teixeira et

framework where devices collaboratively decide task distribution using a consensus method [21]. Additionally, HC principles can be applied to federated learning, as demonstrated by Yang et al., where devices work together for model training without sharing raw data, ensuring data privacy [22].

HC distributes CNN models across edge devices by partitioning and deploying them in ways such as layer-wise, feature-wise, or data-wise, facilitating collaborative inference despite the limited resources of each device [23].

Layer-wise partitioning has each device process specific layers of a CNN and then pass the results to the next device for further computation. Some of the advantages of HC are (i) efficiency and speed, (ii) data privacy and security, and (iii) reduced bandwidth consumption as there is minimized data transmission due to edge processing. However, distributed CNNs in HC remain vulnerable to adversarial noise.

B. Node Collaboration and Security in Edge Computing

To understand the problem practically, consider an image classification CNN model. It requires significant computational power and faces privacy issues with cloud computing. Edge computing offers a solution, but IoT devices at the edge have limited resources. Hence, HC, a method in which multiple edge nodes cooperate, is used to mitigate their resource constraints in data processing. In this collaboration, the CNN model is initially trained offline, and the inference is distributed among the multiple collaborating nodes. This means that the CNN model's various layers will be distributed on different nodes. Each node will process its share of the model and forward the output to the next node. This method is computationally efficient, but nodes are chosen dynamically based on availability, leading to node trust concerns, meaning there will be some trusted and untrusted nodes. Trusted nodes are semi-honest, i.e., they will perform their share of model computation correctly without being vulnerable to compromise. Contrarily, an untrusted node within an HC-based setup refers to a node whose security and trustworthiness have not been verified. While an untrusted node may not actively perform malicious actions, the possibility exists that it could be compromised or malfunction under certain conditions, such as exposure to adversarial noise. Hence, any attacks in untrusted nodes must be detected, and the original node outputs recovered thereupon. Thus, SHEATH, the proposed defense framework, will be deployed on a trusted node to protect (detect and recover) a designated untrusted node in HC-based edge networks, majority of the nodes are inherently untrustworthy, and establishing the trustworthiness of each node is a tedious process. Resorting to node removal as a means of ensuring security is impractical, as it disrupts the computation process and necessitates redeployment of the trained model. Additionally, removing a node requires halting the entire HC inference process, which undermines system continuity. In scenarios where multiple nodes are compromised, the process of replacing or removing affected nodes can experience significant delays, particularly when node availability is limited. During this time, halting the inference process may not be a feasible solution. SHEATH

addresses this challenge by focusing on the correction of detected noise rather than the removal of nodes, thereby maintaining a resilient and continuously operational system. The correction process is carried out by nodes equipped with PseudoNet, integrating seamlessly with the machine learning framework to ensure minimal disruption and maintain data flow continuity for real-time operations.

C. Node Trustworthiness and Dynamic Deployment

We assume that there is at least one trusted node within the architecture. These trusted nodes are leveraged for deploying and executing the SHEATH framework essential for detecting and mitigating adversarial noise introduced by untrusted nodes. Existing literature provides techniques for determining node trustworthiness, such as continuous monitoring of operational consistency, behavioral patterns, and historical performance [24], [25]. Additionally, trust estimation models, including entropy-based factors and adaptive schemes, are used to assess and update trust levels in real time [26]. The mechanisms that determine the trustworthiness of a node may also be applied to the untrusted ones. It is important to note that this paper does not delve further into the continuous evaluation of node trustworthiness.

SHEATH operates in a dynamic HC configuration where nodes can join or leave the network. This dynamic configuration allows SHEATH to adapt to network changes, by reassigning the framework to different trusted nodes as necessary. Furthermore, strategically placing trusted nodes is essential for maximizing SHEATH's effectiveness. By deploying SHEATH on nodes thoughtfully positioned within the network topology, the system ensures comprehensive coverage against adversarial noise attacks. For example, placing a trusted node immediately downstream of an untrusted node enables prompt detection and correction of malicious data alterations.

D. Adversarial Noise in HC-based Distributed CNNs

Distributing CNN models over HC-based edge devices is a decentralized approach to enhance computational efficiency and collaboration among multiple devices. While devices in the network might access only parts of the data or the model, it would be naive to assume that this setup is resilient against cyberattacks. Despite limited access, adversaries can introduce noise to the intermediate data, compromising the overall model integrity. These manipulations, albeit subtle, can have a cascading impact on the model, leading to significant errors or misinterpretations. The sequential nature of these attacks. This can have notable ramifications, especially in healthcare, defense, or autonomous navigation. To launch an attack on an intermediary node in the model, adversaries can target the statistical properties of that node's feature maps (FSMs). Introducing minimal but meaningful noise can degrade the model's efficacy without getting detected. An illustrative example is demonstrated in Fig. 1, where a CNN model, distributed across several IoT devices, processes an image of a cat. Each device processes different model layers, and the feature maps become more sparse as the data transitions

among devices. However, an attack on the “Conv3” layer, though initially unnoticeable, causes the model to mislabel the cat as a fox. Attackers can also target multiple layers, as shown in “Conv1” and “Pool1”, further amplifying the attack’s impact. As the FMs advance through the model, noise is embedded and increasingly concealed by decreasing feature densities, making its detection harder. Hence, a robust security framework is required for attack resilience in HC-based CNNs distributed on edge devices.

E. Research Motivation

The growing use of CNNs in decentralized systems like healthcare, military, and autonomous driving enhances efficiency but also introduces security risks. Despite limited model knowledge, adversaries can inject noise into one or more intermediary nodes, causing misinterpretations. For example, an erroneous image interpretation in military contexts could cause unintended harm; in healthcare, it might cause incorrect patient treatment; for autonomous vehicles, a simple error reading a traffic sign can be dangerous to road safety. While the distributed nature of CNNs in HC architectures offers advantages in terms of latency reduction and privacy, it does not guarantee immunity from adversarial interventions. Studies, including those by Adeymo et al., demonstrate how even partial model access can be exploited by adversaries [27]. Adversarial training is a widely recognized defense mechanism against such attacks [28]–[30]. However, in HC-based distributed CNN models, it is impractical due to the computational burden of retraining each node on adversarial feature maps and the mismatch between training-time perturbations and inference-time feature map manipulations. Consequently, there’s a critical need for innovative solutions that can preemptively identify and mitigate adversarial noise without the exhaustive requirement of model retraining, especially in environments where the specific segments under attack are not predefined. Motivated by these challenges, our research aims to develop a robust and efficient framework to detect potential adversarial intrusions and provide recovery strategies for horizontally collaborating edge nodes that are running parts of the CNN model. The goal is the integrity preservation of CNN models in HC environments while optimizing the tradeoff between security and computational performance.

III. THREAT MODEL

This section overviews our assumptions and the adversary’s knowledge and attack goals and summarises attack techniques.

A. Attack Assumptions

We consider the following attack assumptions for our work:

Communication channels integrity: Data transmission from one node to another is secure. Therefore, the attacker can occur only at the malicious node. It is worth noting that this paper focuses on data transfer challenges, assuming communication channels function normally with standard integrity approaches discussed in [31], [32]. These include combining RIPEMD-128 hash function with DES

encryption and node authentication using Diffie-Hellman and AES encryption for secure communication.
Node integrity: Some nodes in the HC architecture are trusted (can be semi-honest), while the rest are untrusted (can be malicious or compromised). Trusted nodes are essential for SHEATH to be deployed and executed.
Limited access: A malicious node can only add noise to the feature maps by accessing the input, output, and parameters of the CNN layers implemented at this node.
Attack types: One node deploys one layer of the HC-based CNN model. Thus, attackers may perform single-layer attacks by attacking a single node or a multi-layer attack by attacking multiple, non-consecutive nodes with varying levels of noise.

B. Attack Goal

The attacker’s primary goal is to discreetly disrupt the system functionality by injecting noise into one or more intermediate nodes. By adding noise to one of the layers deployed on one of the multiple edge devices, the attacker aims to weaken the model’s decision process and make it more vulnerable, leading to erroneous inferences.

C. Adversarial Capabilities

We assume the attacker’s capabilities include the following:
Partial model access: Attackers can access and potentially modify certain parts of the model, specifically those in the malicious node(s).
Data manipulation: Attackers can introduce noise or other perturbations into the data or intermediate feature maps that flow through the malicious node(s).
Stealthy intervention: Attackers can make their modifications subtle, i.e., hard to detect. They can leverage the statistical characteristics of the intermediary node’s feature maps to make the attack stealthy.

D. Attack Technique

In a distributed or collaborative learning environment, certain nodes or parties might only have access to specific segments of a model. However, this limited access does not preclude the potential for malicious activity. Leveraging this partial access, an attacker can initiate statistical attacks to compromise the model’s integrity. By subtly altering the statistical properties of the feature maps, the attacker can induce deviations in the model’s behavior. Such alterations might include modifying parameter values while ensuring that their aggregate statistical characteristics, like mean and variance, remain unchanged. This stealthy approach makes the perturbations challenging to detect, especially since the overall statistical footprint of the parameters appears untouched. The attacker’s goal, in these scenarios, often revolves around introducing subtle biases, inducing misclassifications, or degrading the overall performance of the model.

Given the complexity of modern deep learning models, even a minor alteration in one segment can trigger cascading effects across the model. This can fulfill the attacker’s objectives

while remaining undetected. Given a set of model parameters P with mean μ and variance σ^2 , an attacker can alter this set to produce a new set P^0 such that $(P^0) = (P)$ and $\sigma^2(P^0) = \sigma^2(P)$. The altered parameters P^0 are generated as:

$$P^0 = f(P; \epsilon) \quad (1)$$

Here, f is a transformation function representing the statistical attack. This function takes the original parameters and a perturbation vector to produce the altered parameters. This vector is designed so that it doesn't drastically change the mean and variance of the original parameters.

E. Attack Vector

In an HC-edge environment, we consider a set of edge nodes represented by $E = \{e_1, e_2, \dots, e_n\}$, where n is the total number of edge nodes present in the system. Furthermore, consider a CNN whose layers are denoted by the set $L = \{l_1, l_2, \dots, l_m\}$, with m being the number of layers in the CNN. The core part of this attack vector is the noise injection into select layers of the CNN distributed across the edge nodes. We represent this noise matrix N , wherein each element N_{ij} represents the noise introduced at layer l_i for edge node e_j . Mathematically, N is defined as:

$$N = \begin{bmatrix} N_{1,1} & N_{1,2} & \dots & N_{1,n} \\ N_{2,1} & N_{2,2} & \dots & N_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ N_{m,1} & N_{m,2} & \dots & N_{m,n} \end{bmatrix}$$

Given a data sample d , the genuine output of the model without noise interference at layer l_i in edge node e_j is denoted by $M(d; l_i; e_j)$. However, the output becomes perturbed under the influence of the attack vector. This perturbed output is represented by $M^0(d; l_i; e_j)$ and is given by:

$$M^0(d; l_i; e_j) = M(d; l_i; e_j) + N_{ij} \quad (2)$$

Upon successful noise injection, the malicious node subtly blends with the regular operations of the network while evading detection. This highlights the need for a robust defense framework for HC architectures with partitioned CNN models.

IV. CASE STUDY AND OBSERVATIONS

We devised a custom CNN model to understand the potential effect of an adversary operating on HC-based CNN.

A. System and Model Description

We consider an HC system that has partitioned the CNN model onto multiple edge devices/nodes, as shown in Fig. 1. We designed a reference CNN model named EdgeCNN to analyze and classify the MNIST dataset, a collection of handwritten digits commonly used for image classification tasks. This model has an architecture of two convolutional layers (with 32 filters), followed by a max-pooling layer, another convolutional layer (64 filters), and finally, two fully connected layers. The network is trained using a stochastic gradient descent (SGD) optimizer with a learning rate of 0.001. The

Fig. 2. Model Accuracy vs. Noise Parameters in Feature Vector when (a) noise is injected in the third convolutional layer's feature maps and (b) noise is injected in two non-consecutive convolutional layers "Conv1" and "Conv3".

Fig. 3. Impact of noise injection in different layers of (a) EdgeCNN and (b) LeNet CNN architectures in an HC setup.

The first node receives the input image and extracts the initial features. The subsequent nodes ("Conv1", "Conv2", "Pool1", and "Conv3", respectively, which are the convolutional and pooling layers. We assume that the node running the third convolutional layer is malicious. Hence, noise is introduced in the feature maps obtained from "Conv3". Finally, the following nodes process the fully connected (FC) layers, "FC1" and "FC2", to yield the model's output.

B. Noise Injection

We implemented a custom layer introducing Gaussian noise into the inputs to test the model's performance. This Gaussian noise is defined in Eq. 3. Here μ is the input, σ is the mean of the input, σ is the standard deviation of the input, B is the Bernoulli distribution, p is the noise percentage which is a binary mask dictating where the noise is to be added, and σ_p is the standard deviation percentage (noise magnitude) that was integrated into the EdgeCNN model. The model's performance was evaluated on the MNIST test set. This layer assessed the EdgeCNN model's robustness to various noise levels and impacted nodes.

$$I^0 = I + G(\mu; \sigma_p) \cdot B(np) \quad (3)$$

For evaluating our work, we also consider the Polarity Switch Attack as a representative adversarial threat within the HC-based CNN architecture. This attack involves inverting the sign of the feature map activations at the compromised node, effectively disrupting the learned representations and leading to model misclassification. Unlike input-level adversarial noise, this manipulation targets internal computations, making it harder to detect with conventional defenses. This aligns with the threat model in our HC-based CNN, where adversaries may access intermediate computations in distributed systems.

TABLE I
VARIATION IN THE STATISTICAL MEAN OF THE FEATURE MAP AGAINST VARYING PERCENTAGE OF NOISY NODES

CNN Model	Datasets	Mean Before Noise	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
EdgeCNN	MNIST	0.8668	0.9109	0.9511	0.9880	1.0535	1.0764	1.1200	1.1539	1.2194	1.2559	1.2954
LeNet-5	MNIST	1.0716	1.1198	1.1687	1.2236	1.2833	1.3342	1.3770	1.4453	1.4949	1.5472	1.6061
LeNet-5	Fashion	1.1062	1.1622	1.2217	1.2611	1.3537	1.3710	1.4208	1.4561	1.5646	1.5957	1.6559

TABLE II
VARIATION IN THE STATISTICAL STANDARD DEVIATION OF THE FEATURE MAP AGAINST VARYING PERCENTAGE OF NOISY NODES

CNN Model	Datasets	Stdev Before Noise	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
EdgeCNN	MNIST	1.4198	1.4284	1.4524	1.4508	1.4649	1.4746	1.4743	1.4690	1.4883	1.4808	1.4902
LeNet-5	MNIST	1.5921	1.6141	1.6128	1.6425	1.6590	1.6412	1.6551	1.6659	1.6785	1.6694	1.6714
LeNet-5	Fashion	1.2264	1.2381	1.2722	1.2847	1.3100	1.3070	1.3240	1.3371	1.3428	1.3681	1.3543

C. Observations

We appended a noise layer based on Eq. 3 after the convolution layer to assess the effects of varying the percentage of nodes impacted by noise (p) and the noise magnitude (ϵ). On the MNIST test set, the model's accuracy was determined for each p and ϵ combination. This test yielded accuracy variations with increasing noisy nodes across different magnitudes. Key observations include: (i) the EdgeCNN model initially attained a 98.87% accuracy without noise, (ii) the model's performance deteriorates with noise, contingent on the percentage of noisy nodes and its magnitude, and (iii) Fig. 2 demonstrates that even minor noise can considerably reduce accuracy, particularly when the magnitude is amplified. The impact of noise injection in different layers can be seen in Fig. 3.

D. Stealth Analysis

We analyzed the mean and standard deviation of the feature map across varying levels of noisy nodes and magnitude to assess the vulnerability of distributed CNNs in HC scenarios to adversarial noise. For EdgeCNN on the MNIST dataset, with a number of impacted feature vector elements escalated to 50%, the feature map's mean-variance and standard deviation-variance remained minimal, as shown in Tables I and II. The noise magnitude was kept at 50%. The LeNet-5 models on MNIST and Fashion datasets displayed an analogous pattern, with minimal fluctuations in their respective means and standard deviations, irrespective of the introduced noise magnitude. These results indicate that while major noise barely impacts feature statistics, it can drastically lower accuracy. This makes the attack stealthy while potentially evading the existing defense mechanisms, indicating the need for more robust countermeasures that consider such tactics.

V. PROBLEM FORMULATION

In response to challenges observed in Section IV and the detected threats in Section III, we outline the architecture of the proposed framework and its deployment. In HC, the CNN model is distributed across multiple edge nodes. Noise in any node can severely affect model performance. The objective is to develop a robust defense mechanism that maintains the system's output integrity across all data, layers, and nodes.

Noise Detection: The defense framework first examines integrity and robustness on HC edge devices with minimal node outputs to detect noise in intermediate feature maps.

which is essential for the recovery process. A noise detection method D , compares the perturbed output $M^0(d; l_i; e_j)$ with the actual output $M(d; l_i; e_j)$. If the difference exceeds a predefined threshold, the output is flagged as noisy. The threshold must be carefully set to balance false positives and false negatives in noise detection.

Model Recovery: Once the adversarial noise is detected, the next step is to recover the original feature maps to replace the adversarially perturbed ones bringing them as close as possible to the genuine output, hence ensuring the integrity of the system's overall performance. Let $\hat{M}(d; l_i; e_j)$ represent the recovered output after applying the defense mechanism, which is a function of the perturbed output $M^0(d; l_i; e_j)$ and the detected discrepancy δ_{ij} .

Objective Function: The primary goal is to minimize the overall discrepancy between the recovered output and the genuine output across all data samples, layers, and edge nodes:

$$\min_{d, l_i, e_j} \sum_{l_i \in L, e_j \in E} \|M(d; l_i; e_j) - \hat{M}(d; l_i; e_j)\| \quad (4)$$

Subject to the satisfaction of the following two constraints:

Detection Constraint:

$$\delta_{ij} = D(M^0(d; l_i; e_j); M(d; l_i; e_j)); \quad \delta_{ij} \geq \epsilon \quad \forall d; l_i \in L; e_j \in E \quad (5)$$

This ensures that the discrepancy between the actual and perturbed output is calculated for each data sample, layer, and edge node. It acts as a prerequisite for the recovery step, identifying the instances where the adversarial noise has affected the output (intermediate feature maps).

Recovery Constraint:

$$\hat{M}(d; l_i; e_j) = \text{RecoveryMethod}(M^0(d; l_i; e_j); \delta_{ij}) \quad (6)$$

Eq. 6 defines how to correct the detected noise. It specifies how the perturbed output is adjusted to minimize the discrepancy with the actual output, thus restoring the model's accuracy compromised by the adversarial noise.

VII. PROPOSED FRAMEWORK FOR SECURE HORIZONTAL EDGE WITH ADVERSARIAL THREAT HANDLING

We propose SHEATH, a novel framework to ensure CNN performance on HC edge devices with minimal overhead and latency, addressing adversarial noise in partially

trusted HC architectures. The adversary can access the parameters of the model segment deployed on a compromised node including the input and output feature maps. SHEATH detects noise in a node's output and corrects it to the expected output for the next node. SHEATH's Detect and Recover modules are responsible for noise detection and feature map correction, respectively. The SHEATH's Detect consists of PseudoNet and Comparator. This is illustrated in Fig. 4, and the necessary technical details are also discussed in this section. By adding noise to these elements, the adversary can compromise the model's integrity.

A. SHEATH-Detect

This module is responsible for detecting the presence of noise in an untrusted node to ensure that it sends the correct output to the subsequent node. Given the node's untrusted nature, the Detect module requires a reference set of noise-free feature maps generated by PseudoNet. These feature maps are then sent to the Comparator for verifying whether the difference between the noisy and non-noisy feature maps exceeds the predefined threshold.

1) PseudoNet: The key idea for PseudoNet is to have a secondary mechanism to recompute the expected feature maps from the untrusted node using fewer computational resources. The primary mechanism is the reference of comparison (non-noisy) in the Comparator. Generation of reference feature maps can be achieved in two ways: (a) redundancy, which refers to the duplication of the whole targeted node, or (b) constructing a similar subset of the untrusted node with reduced hyperparameters (PseudoNet approach). The latter is a relatively lightweight mechanism. Let U be the untrusted node's layer(s) P the corresponding PseudoNet's layer(s), U the parameters in U (e.g., lters, neurons, kernel size), and P the parameters in P . Then the formation of PseudoNet's layer(s) can be modeled as:

$$P = \alpha U \quad (7)$$

Where $0 < \alpha < 1$ is the reduction factor, and $P \subset U$, making P a computationally lighter counterpart of U . This technique can be applied and tailored PseudoNet for any CNN layer L . For example, if this reduction technique applied to a convolutional layer, then is denoted as conv. The set of parameters, P , consists of (W_{conv}, b_{conv}) . Here, W_{conv} is a subset of lters from U , and b_{conv} are the biases for those lters. The output is defined in Eq. 8.

$$\text{Output}_P = f(W_{conv} \text{Input} + b_{conv}) \quad (8)$$

where f is the activation function and \otimes is the convolution.

If the desired layer is the pooling Layer, is represented as pool. The set of parameters P (e.g., max, average), specify pooling operation and size. The output is defined in Eq. 9.

$$\text{Output}_P = \text{pool}(\text{Input}) \quad (9)$$

If the target layer is the dense layer, is represented as dense. Set of parameters, P , are (W_{dense}, b_{dense}) . W_{dense} is

Algorithm 1: Alpha Determination

```

Input: Base model's target layer with N lters
Output: Detection of sweet spot value of alpha
Initialize alpha = 0;
Initialize P_best = 1;
for k = 1 to N do
    alpha = k/N;
    Construct PseudoNet with lters;
    Evaluate detection accuracy A(alpha);
    Measure computational overhead C(alpha);
    Compute performance metric
    P(alpha) = A(alpha) * C(alpha);
    if P(alpha) > P_best then
        P_best = P(alpha);
        alpha = alpha;
end
return alpha;

```

a subset of weights from U and b_{dense} is a subset of biases from U . In this case, the output will be

$$\text{Output}_P = f(W_{dense} \text{Input} + b_{dense})(\text{Input}) \quad (10)$$

where f is the activation function and \cdot is the dot product.

For each layer type, the subset of parameters P should reflect the lightweight aspect of PseudoNet compared to U .

An appropriate α can be chosen for each layer type based on the computational and security requirements.

For brevity, we focus our explanation of the SHEATH framework on securing a convolutional layer within an untrusted node(s), as convolutional layers are fundamental to CNN architectures and crucial in defining the feature representation of input data. Their complexity, in terms of the number and size of lters and strides, makes them the primary target for attackers. We obtain this layer's PseudoNet by reducing the number of lters/kernels. To illustrate, F_{LC3} is the full set of lters in the original layer, the F_P represents the lters in our corresponding PseudoNet being a subset of F_{LC3} .

Parameter reduction strategy in PseudoNet. Determining the sweet spot of parameter reduction in models is a delicate process. A higher reduction of parameters can affect the accuracy, while a lower reduction can lead to computational inefficiencies. However, as a rule of thumb, the desired value of α , which lies between 0 and 1, can be empirically determined by iteratively testing the reduced model's performance and comparing it to the baseline. The parameter α serves as the reduction factor, defining the proportion of lters in PseudoNet relative to the base model to improve the computational efficiency. For EdgeCNN, where the target layer of the base model contains 64 lters, we conducted an iterative analysis by incrementing the number of lters in PseudoNet by one per iteration (e.g., one lter in the first iteration, two in the second, and so on). At each step, we evaluated SHEATH's detection accuracy and the corresponding computational overhead to identify a good tradeoff point (sweet spot) between accuracy and efficiency. The value of α , which provides this sweet spot,

is detailed in Algorithm 1 and validated experimentally, as shown in Fig. 11.

Fig. 4. Overview of the SHEATH Framework. SHEATH has two modules: Detect and Recover. It is deployed on a trusted node in HC-based edge devices to defend against adversarial noise from propagating to the rest of the CNN model. Integrated within drone-based trusted nodes, SHEATH secures the untrusted node (IoT dev2) by taking input from the preceding trusted node (IoT dev3). In the case of noise detection, SHEATH forwards the recovered output to the subsequent layer (IoT dev4).

2) Comparator: After the PseudoNet computes the reference feature maps for the Comparator, the next step is to compare these maps with the noisy feature maps produced by the target layer in the untrusted node. The primary method of this comparison is computing the mean squared error (MSE) between the noisy and non-noisy feature maps. As PseudoNets a reduced subset of the untrusted node, only the corresponding feature maps from both are compared, which makes noise detection computationally efficient. Given two sets of lters, one from the PseudoNet represented as F_p and the other, a subset from the target layer represented as F_{LC3} (which corresponds to F_p), the MSE is defined as:

$$MSE(F_p; F_{LC3}) = \frac{1}{k} \sum_{i=1}^k (p_i - c_i^0)^2 \quad (11)$$

where p_i represents the lter values from the PseudoNet, c_i^0 represents the corresponding lter values from the subset of the target layer, F_{LC3} , and k is the total number of lters being compared. If the computed MSE surpasses a predefined threshold, it indicates that there is noise in the system. Subsequently, measures to correct this noisy output are activated to ensure the overall system integrity.

Threshold Selection Strategy The threshold for MSE is selected based on the comparison performed during the sanity check conducted in a noise-free environment. To elaborate the process of the sanity check, we computed and compared the MSE between the FMs generated by the target layer “Conv3” of the noise-free base model EdgeCNN against the FMs generated by the PseudoNet. Since one lter represents one FM, the MSE between each FM of the base model EdgeCNN and each FM of PseudoNet is 0. An MSE value of 0 indicates that there is no difference between the FMs of the noise-free base model and PseudoNet. Any MSE value greater than 0 indicates the presence of noise. The results of

the sanity check is illustrated in Fig. 5. Therefore, in an ideal, noise-free situation, the MSE between the PseudoNet and the base model's target layer is 0. For our case, the sanity check yielded an MSE of 0, which we used as the threshold.

B. SHEATH-Recover

The primary role of this module is to ensure that accurate and noise-free output is delivered to the subsequent nodes in the system. If the Detect module determines that there is no noise in the feature maps, the output of the untrusted node is forwarded as-is to the subsequent node. However, if noise is detected, the Recover module detects and corrects noise, forwarding the output to the next node. It trains convolutional layers by using feature maps from PseudoNet. Feature maps are merged based on lter index, e.g., if the untrusted node generates 64 feature maps and PseudoNet yields v_e , Recover predicts the remaining 59 feature maps. The merged output, consisting of the predicted and PseudoNet feature maps, is forwarded to the subsequent node, ensuring a noise-free output. Mathematically, F_{LC3} is the complete set of lters in the target layer of the untrusted node, F_p is a subset of lters in F_{LC3} that matches PseudoNet, n is the total number of feature maps in the target layer of the untrusted node, and v_e is the number of feature maps produced by PseudoNet where $v_e + n = n$. The objective, then, is to generate the missing $(n - v_e)$ feature maps using the convolutional layers in Recover module and merge them with the output from PseudoNet.

Generating Missing Feature Maps

$$FM_{\text{missing}} = \text{Conv}_{i,f}(\text{Input}_{\text{Recover}}) \quad (12)$$

Where FM_{missing} represents the set of missing feature maps, $F_{LC3} - F_p$. $\text{Conv}_{i,f}$ is the convolution operation using the optimal l and f obtained from the grid search, and $\text{Input}_{\text{Recover}}$ is the input to the Recover module.

Fig. 6. Single-layer attack in HC-based edge devices.

Fig. 5. Sanity Check: No difference between the FM EdgeCNN and PseudoNet. Hence, '0' is selected as SHEATH's detection threshold.

Merging Feature Maps: The merged feature maps, taking the union of those from the PseudoNet and Recover modules are:

$$FM_{merged} = F_p \cup (F_{LC3} \cap F_p) \quad (13)$$

Thus, the final FM_{merged} is a union of the feature maps from PseudoNet and the missing ones produced by the Recover module to match the target F_{LC3} . The training for these convolutional layers is executed offline, ensuring no real-time delays or performance hitches. The architecture of the Recover module, including the number of layers and filters within these layers, is determined optimally through a grid search algorithm. The primary objective is a tradeoff between accuracy and overhead. The grid search function is given by

$$G(\text{grid}) = \min_{l \in [1, n], f \in [1, m]} \text{Loss}(\text{Model}_{l,f}) \quad (14)$$

Where G is the grid search function over the design grid, comprising all possible combinations of the number of convolutional layers l and number of filters f in each layer. Eq. 14 represents the grid search function where l varies from 1 to n (number of layers) and f varies from 1 to m (number of filters per layer), and the goal is to minimize the loss function for the given model configuration. $\text{Loss}(\text{Model}_{l,f})$ represents the loss function of the model configuration trying to match $F_{LC3} \cap F_p$ as accurately as possible.

C. Training Process

In SHEATH, all training, including the CNN model, PseudoNet and recovery module is performed offline prior to deployment to avoid burdening resource-constrained edge devices with computationally intensive tasks. HC-based CNNs distribute inference workloads across multiple devices, leveraging collective computational power for efficient real-time processing while addressing individual resource limitations. Offline training reuses data and intermediate FMs from the base CNN's training, requiring no additional resources beyond standard CNN development. The sweet spot value of reduction factor for PseudoNet is determined offline through iterative testing to balance detection accuracy and efficiency, as described in Algorithm 1. Similarly, the recovery module's architecture is optimized using grid search to minimize loss and ensure deployment efficiency. This approach ensures a lightweight, scalable design suitable for resource-constrained environments.

Once trained, the PseudoNet and recovery module are deployed to the edge devices as part of the overall CNN model.

Fig. 7. Non-consecutive multi-node attack in HC edge devices.

During deployment, edge devices perform inference only using these pre-trained components. Edge devices do not require access to training data or intermediate feature maps during deployment, ensuring the approach aligns with the resource constraints of HC-based CNNs and ensures efficient operation.

D. Deployment Limitations

Consider a sequence of edge nodes $E = \{e_1; e_2; \dots; e_x\}$. For the sake of brevity, we consider each node has one layer of the CNN model to discuss the feasibility of SHEATH deployment in three scenarios: (i) single-layer attack, (ii) non-consecutive multi-layer attacks, and (iii) consecutive multi-layer attacks.

Single Layer Attack: If e_j is untrusted, SHEATH is deployed on e_{j+1} to verify the computations performed by e_j as shown in Fig. 6. For instance, in a setup of nodes $E = \{e_1; e_2; e_3; e_4; e_5\}$, if e_2 becomes compromised, then SHEATH is on e_3 , ensuring e_3 validates and possibly rectifies computations from e_2 before progressing with its tasks.

Non-consecutive Multi-node Attacks: For several non-consecutive compromised nodes, say e_j (where $|j_i - j_j| > 1$), SHEATH is deployed on e_{j+1} and e_{j+1} respectively as can be seen in Fig. 7. Rectifying on a system with nodes $E = \{e_1; e_2; e_3; e_4; e_5; e_6\}$, if nodes e_2 and e_4 are both untrusted, SHEATH is deployed on e_3 and e_5 . These nodes will then be responsible for individually confirming and potentially appending the computations of e_2 and e_4 .

Consecutive Multi-node Attacks: As seen in Fig. 8, when two consecutive nodes, e_j and e_{j+1} , are under threat, SHEATH applied on e_{j+2} may receive compromised input, reducing its efficacy. Taking $E = \{e_1; e_2; e_3; e_4\}$, if e_2 and e_3 are both untrusted, SHEATH on e_4 may be ineffective, as it gets potentially flawed data from e_3 .

Deploying SHEATH, i.e., PseudoNet and recovery module on untrusted nodes may introduce additional computational overhead. However, our framework is designed to minimize this overhead through the following. (1) **Lightweight Design:** As detailed in Algorithm 1, we determine a sweet spot for the reduction factor that balances detection accuracy and computational efficiency. This ensures a lightweight design by using only a subset of the original model's parameters. Moreover, our experimental results (see Section VII, RQ8) show that the computational overhead introduced by SHEATH's

overhead, then the peak complexity of SHEATH is likely less than redundancy as in Eq. 15.

$$O(t_d + t_m) < O(t_r) \quad (15)$$

Fig. 8. Consecutive multi-node attack in HC edge devices.

Detect and Recover modules is minimal. (2) Selective Deployment: Instead of deploying SHEATH for every untrusted node, we can focus on protecting the most critical nodes or layers that are highly susceptible to adversarial noise. (3) Scalability with Model Complexity: Our experiments indicate that the overhead does not increase linearly with model complexity. For larger models like VGG10, the number of Iters required in the recovery module does not proportionally increase, as shown in Fig. 16. This suggests that the defense remains resource-efficient even as models scale.

Limitations: The deployment of SHEATH provides a mechanism to secure the system, but it isn't without challenges.

One significant limitation is the susceptibility to consecutive multi-node attacks, as seen in Fig. 8. In these cases, even if SHEATH is in place, it may operate on flawed data. This results in a cascading effect of the noise compromising the rest of the model. For SHEATH to be effective, it operates assuming it receives trustworthy input. In situations where e_i is compromised, SHEATH on e_{i+1} is designed to handle and rectify the potential noise or alterations. However, when both e_i and e_{i+1} are compromised, the SHEATH on e_{i+2} faces a dilemma. The data it receives has passed through two nodes of potential malicious alterations. This hampers its ability to distinguish actual data from the compromised one. It is important to distinguish between multi-node and multi-layer scenarios. SHEATH can effectively defend against adversarial noise when multiple layers are deployed on the same node, as it can monitor and correct within that single node. However, the framework faces limitations when multiple layers are distributed across two consecutive untrusted nodes, e.g., if the last layer of 'Node1' and the first layer of 'Node2' are both compromised, SHEATH on 'Node3' may receive data that has been altered by two prior untrusted nodes. Thus, in such cases, we can reconsider the distribution of nodes to avoid model deployment on two consecutive untrusted nodes.

E. Theoretical Complexity Analysis for SHEATH

The other alternative for security in HC-based CNN architectures would be to do a redundancy, i.e., duplicating the targeted untrusted node(s) (one node redundancy for single-layer and multiple node redundancy under a multi-layer attack(s)). The theoretical computational complexity for SHEATH and doing a complete redundancy can be represented as follows. Let t_d , t_m , and t_r be the times and m_d , m_m , and m_r be the memories for SHEATH-Detect, SHEATH-Recover, and redundancy, respectively. The complexity for SHEATH is typically $O(t_d)$, occasionally peaking at $O(t_d + t_m)$ upon noise detection. Contrarily, redundancy consistently operates at $O(t_r)$. If t_d is minimal, t_m is infrequent, and t_r has significant

overhead, then the peak complexity of SHEATH is likely less than redundancy as in Eq. 15. Considering resource efficiency, SHEATH primarily utilizes resources at $O(t_d)$, with occasional rises, whereas redundancy demands $O(t_r)$ continuously. This implies higher long-term costs for redundancy. In latency, SHEATH incurs $O(t_d)$, only rising upon noise detection, while redundancy is consistently $O(t_r)$, potentially higher due to synchronization or switching. For overhead, SHEATH mainly correlates with $O(t_d)$, with occasional costs of $O(t_m)$. Redundancy, however, consistently incurs $O(t_r)$ due to backup system maintenance. In conclusion, SHEATH offers a more efficient paradigm than redundancy, especially with infrequent noise detections. We experimentally validate the complexity for both SHEATH and redundancy in RQ8 of Section VII.

VII. PERFORMANCE EVALUATION

To evaluate the performance of SHEATH across diverse CNN configurations, we experiment with three CNN models: LeNet, MiniVGGNet, and EdgeCNN, our custom test-set CNN. We evaluate both SHEATH-Detect and SHEATH-Recover to determine whether noise is being accurately detected and the FMs are being corrected. We use several metrics for this: (1) Accuracy which measures its ability to identify FMs with adversarial noise correctly; (2) Precision, representing the proportion of correctly identified noisy FMs out of all classified as noisy; (3) Recall, indicating the success in identifying actual noisy FMs out of all present; and (4) F1-score which balances noise detection and reduces false positives as the harmonic mean of recall and precision. All evaluations in this study were conducted under a simulation on edge devices rather than on a physical hardware-accelerated edge architecture. All experiments were executed on a system equipped with a 12th Gen Intel(R) Core (TM) i5-1235U processor operating at a frequency of 1.30 GHz, supported by 16.0 GB RAM running a 64-bit operating system. We investigated and conducted experiments for the following research questions (RQs).

- RQ1: How effectively does SHEATH detect adversarial noise across different layers of varied CNN architectures?
- RQ2: What is the performance across different datasets and in safety-critical scenarios?
- RQ3: How does the number of Iters and other hyperparameters influence SHEATH's detection performance?
- RQ4: What are the false positives and false negatives?
- RQ5: How does the noise detection accuracy of SHEATH vary with the strength of adversarial noise?
- RQ6: What is the performance with model complexity?
- RQ7: Does the location of noise injection in the model impact SHEATH's performance?
- RQ8: What computational overhead is added by SHEATH-Detect and SHEATH-Recover, and how does the complexity compare with that of a complete redundancy?
- RQ9: What is SHEATH's efficacy when multiple non-consecutive nodes are attacked?

Fig. 9. SHEATH-detect's Efficiency across Different Layers in CNN Architectures (a) MiniVGGNet and (b) EdgeCNN.

Fig. 10. SHEATH's Performance with varied datasets.

RQ10: What is SHEATH's efficiency in recovering original feature maps with different levels of noise injection?

RQ11: What is SHEATH's performance when its feature map recovery process is scaled to larger CNN architectures?

A. Evaluation Results

This subsection entails the answers to the aforementioned RQs with experimental results.

RQ1 - Analysis of SHEATH's Detection Efficiency across Different Layers in CNN Architectures: Analyzing CNN layers' resilience to noise reveals vulnerabilities. Deeper LeNet layers suffered more accuracy drop, while MiniVGGNet's Conv2A had a significant dip. SHEATH's evaluation on MiniVGGNet indicated high accuracy in early layers. Notably, Conv2B accuracy jumped from 89.83% with one layer to 99.89% with two, emphasizing layer analysis in noise detection. Simultaneous attacks on multiple layers of EdgeCNN in "Conv2" and "Conv3", with each layer having its detection mechanism, showed considerable detection performance with accuracy increasing from 95% with one layer to 99.01% with two for "Conv3". SHEATH's consistent high detection, represented in Fig. 9(a) and 9(b), underscores its defense capability against noise, regardless of the injection point.

RQ2 - SHEATH's Performance with Varied Datasets and in Safety Critical Scenarios: Assessing the adaptability of an adversarial noise detection mechanism across varied datasets is vital for understanding its robustness and real-world relevance. We evaluated SHEATH across representative datasets using a single layer. For MiniVGGNet on Fashion, accuracy and F1 score stood at 91.72% and 87.35%. For LeNet on the Fashion dataset, the values were 97.86% and 90.55%, while on MNIST, they reached 98.62% and 92.67%. For VGG10 on Fruits-300, accuracy and F1 score were 96.62% and 89.21%, respectively. Additionally, evaluating the detection performance of SHEATH in safety-critical scenarios is imperative to establish its resilience in environments where human safety is at stake. We assessed SHEATH's performance using the German Traffic Sign Recognition Benchmark (GTSRB) dataset. The GTSRB dataset comprises of images captured in real-world road settings, which provides a reliable testbed for autonomous driving systems. For MiniVGGNet trained on the GTSRB dataset, the detection accuracy and F1 score were 95.34% and 87.98%, respectively. This underscores SHEATH's proficiency in detecting adversarial noise in safety-critical environments.

Fig. 11. Influence of Hyperparameters on SHEATH: (a) Accuracy and (b) F1-Score when no. of layers are varied in PseudoNet

TABLE III
FALSE POSITIVES AND FALSE NEGATIVES IN PSEUDONET.

CNN Model	Dataset	False Positive	False Negative	ROC
EdgeCNN	MNIST	0	150	0.925
LeNet	MNIST	0	42	0.979
MiniVGGNet	CIFAR-10	0	370	0.962

environments. As illustrated in Fig. 10, SHEATH performs well across datasets. The variance in performance suggests that dataset characteristics can influence SHEATH's efficiency. While SHEATH showcases broad adaptability, customizing it for specific datasets might further optimize its capabilities.

RQ3 - Influence of Hyperparameter Variations on SHEATH-Detect's Performance across CNN Architectures: In deploying CNN-based systems like SHEATH, hyperparameter choices like layer count, significantly impact performance. We hypothesized that SHEATH can detect adversarial noise even with a single affected layer, but balancing layer count for improved detection while considering computational overhead is crucial. Experimental results showed EdgeCNN's detection accuracy starting at 95.13% for one layer, reaching 100% by seven layers, with a similar trend in the F1-score. For LeNet, accuracy began at 98.62%, hitting 100% by three layers, maintaining this trend. MiniVGGNet started at 89.83%, achieving 100% accuracy by seven layers. These findings emphasize SHEATH's effectiveness with few layers and the importance of optimizing for computational efficiency, as seen in Fig. 11.

RQ4 - Evaluation of SHEATH's performance in false positives and false negatives across CNN models and datasets: Evaluating SHEATH's adversarial noise detection also involves assessing false positives and negatives. Table III outlines these metrics for SHEATH across varied CNN models and datasets with a single layer in the PseudoNet. Specifically, EdgeCNN showed zero false positives and negatives, while LeNet on the same dataset, had 150 false negatives.

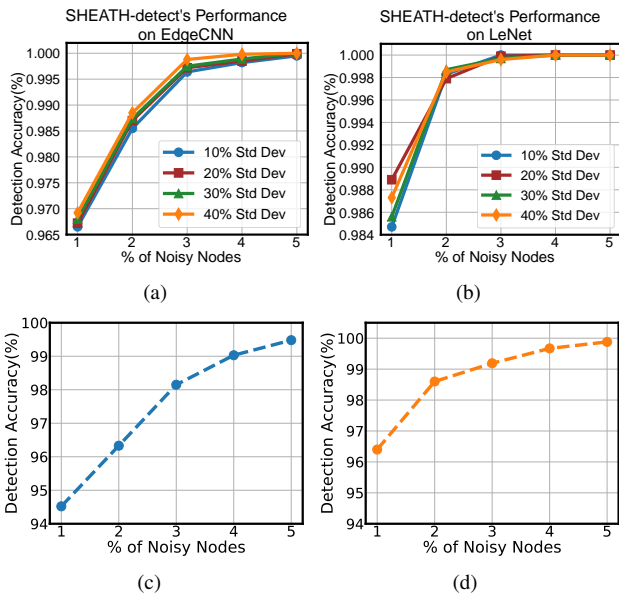


Fig. 12. SHEATH-*Detect*'s performance against varied adversarial strengths in (a) EdgeCNN and (b) LeNet for Gaussian Noise and (c) EdgeCNN and (d) LeNet for Polarity Switch Attack

recorded zero false positives and only 42 false negatives. For the CIFAR-10 dataset, *MiniVGGNet* had zero false positives but a larger 370 false negatives. To address the challenge of false negatives, the number of filters in the *PseudoNet* was progressively increased, enabling the model to capture more complex noise patterns that might otherwise go undetected. The baseline, established with 1 filter, resulted in the highest number of false negatives. Our findings demonstrate that employing 3 filters in the *PseudoNet* reduced the false negatives to 41% of the baseline, with 4 filters false negatives were reduced to 19% of the baseline, and with 5 filters false negatives were significantly reduced to 3.3% of the baseline. It's vital to optimize SHEATH's capabilities for enhanced noise detection across diverse CNNs.

RQ5 - Robustness Against Varied Noise Characteristics:

To assess SHEATH's detection capabilities against different adversarial noise attacks, we evaluated it under Gaussian noise and polarity switch attacks. The polarity switch attack involves inverting the sign values within the feature vector of the CNN model, changing positive values to negative and vice versa. This manipulation affects only the signs, not the magnitudes of the values. The severity of this perturbation is measured by the percentage of noisy nodes, indicating the portion of the feature vector impacted by this adversarial technique. Such attacks typically have a more pronounced effect on the CNN architecture's memory-centric or fully connected layers. For *EdgeCNN*, accuracy also improved as Gaussian noise levels augmented. Specifically, with a noise level of 1% and a standard deviation of 10%, the accuracy stood at 0.9660. However, with a 5% noise increase, the accuracy impressively surged to 0.9998. This pattern was consistent across all standard deviation values. Additionally, when subjected to the polarity switching, *EdgeCNN* exhibited robustness. At a noise level of 1%, the detection accuracy was 94.52%, showcasing the model's resilience to this specific adversarial perturbation.

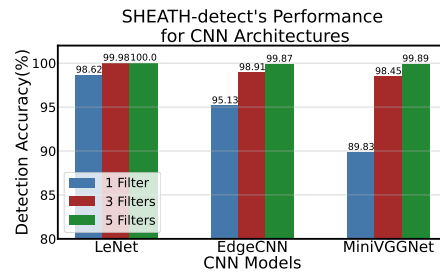


Fig. 13. SHEATH's performance across different CNN architectures

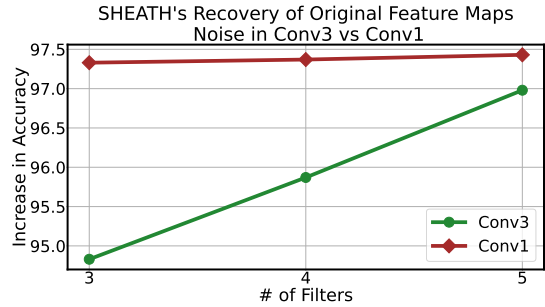


Fig. 14. Impact of location of noise injection on SHEATH.

Moreover, at a noise level of 5%, the detection accuracy further increased to 99.48%. In the evaluation of LeNet, the model initially achieved an accuracy of 98.47% with 1% noise and a 10% standard deviation. The accuracy improved to 100% with Gaussian noise levels of 3% and 4%. In tests involving polarity switching, the model's accuracy was 96.40% at a noise level of 1%, and increased to 99.88% at 5% noise. These results demonstrate that the SHEATH framework effectively maintains high detection accuracy across varying levels of adversarial noise, as illustrated in Fig. 12(a) - 12(d).

RQ6 - SHEATH's Performance with Model Complexity (Scalability):

We explored how diverse CNN architectures influence performance to determine if SHEATH maintains efficacy with escalating model intricacy. The study included *EdgeCNN*, LeNet, and *MiniVGGNet*. For *EdgeCNN*, *PseudoNet*'s accuracy increased from 95.13% with one filter to 99.87% with three. LeNet exhibited 98.62% accuracy for one filter, achieving 100% by three filters. *MiniVGGNet* began at 89.83% with one filter, escalating to 99.89% by the third. As illustrated in Fig. 13, these findings highlight SHEATH's adaptable and robust performance across various complexities.

RQ7 - Impact of location of noise injection on SHEATH:

Fig. 14 shows the SHEATH-*Recover* module's performance when the noise is injected in (a) earlier layer and (b) last layer of the model, "Conv1" and "Conv3", respectively with an increasing number of filters. As the number of filters increases, SHEATH's accuracy when "Conv1" is under attack is better than when "Conv3" is under attack, emphasizing that SHEATH effectively detects adversarial noise and eliminates the noise impacts regardless of the location.

RQ8 - Computational Overhead of SHEATH in real-time CNN environments:

When deploying verification tools like SHEATH in real-time CNN environments, assessing the computational overhead is essential to understand the extra processing time and resources. For optimal system performance,

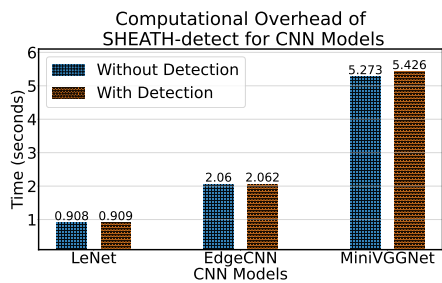


Fig. 15. SHEATH-detect’s computational overhead.

this should be minimal, even as it improves model robustness. We measured this overhead as the increase in processing time for models using SHEATH compared to their baseline computation time without it. It is worth noting that the following computation times were measured in the inference mode, which does not leverage the GPU acceleration. LeNet had an increase of 1.7ms (0.908s to 0.9097s), *EdgeCNN* an increase of 1.87ms (2.06s to 2.06187s), and MiniVGGNet, a more noticeable overhead of 153ms (5.273s to 5.426s). In summary, SHEATH-*Detect* introduces a minimal computational overhead, as shown in Fig. 15, ensuring its potential suitability for real-time deployments. Furthermore, using empirical data, we evaluated the performance of SHEATH against the redundancy approach regarding time complexity and memory overhead. The t_d , t_m , and t_r are 0.2033s, 0.4632s, and 0.2784s, respectively. The worst-case time complexity for SHEATH (when both detection and correction/recovery are applied) is $t_d + t_m = 0.6665$ seconds. However, during infrequent attacks or noise scenarios, SHEATH’s average performance time is t_d , making it competitive against t_r . Similarly, m_d , m_m , and m_r are 0.0Mb, 0.015625Mb, and 0.0Mb, respectively. The memory overhead for the SHEATH-*Recover* module is minimal, with an increased overhead of m_m MiB compared to redundancy. While the worst-case time complexity of SHEATH is slightly higher than doing redundancy, it offers competitive average performance, particularly under infrequent attack scenarios. Its adaptability, threat-handling flexibility, and low memory overhead make it an efficient alternative. In environments with frequent attacks, redundancy may be preferable, though such conditions are generally unfavorable for HC systems.

RQ9 - SHEATH’s efficacy in case of multiple nodes: We implemented the deployment scenario as depicted in Fig. 14. In the *EdgeCNN* model, the noise was added to the nodes having “*Conv1*” and “*Conv3*”. If the impact of noise in “*Conv1*” is not mitigated, it will lead to a cascading effect, which will further decrease the model accuracy when “*Conv3*” is also noisy. Hence, we deployed SHEATH for both layers/nodes and compared the model accuracy with and without SHEATH. As shown in Table. IV, SHEATH successfully recovers the original feature maps for both “*Conv1*” and “*Conv3*”. Hence, the model maintains a high accuracy consistently, regardless of the noise levels. It is worth mentioning that the accuracy is 97.55% for all noise levels, as it is measured on the same dataset using the same recovery modules.

RQ10 - SHEATH’s efficacy in recovering original feature maps: Table IV provides a comprehensive evaluation of

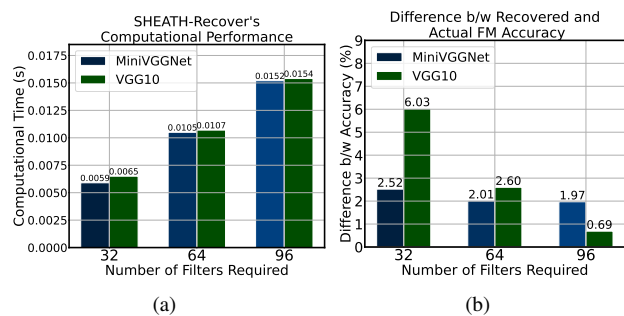


Fig. 16. Scalability Analysis of SHEATH-Recover’s Performance for MiniVGGNet and VGG10 models: (a) Computational Performance and (b) Difference between recovered and actual FM Accuracy

the SHEATH-Recover module’s effectiveness in identifying and rectifying noise disturbances, particularly under Gaussian Noise Attack and Polarity Switch Attack scenarios. Utilizing the Pseudonet framework within the *EdgeCNN* architecture, the model generated five feature maps, while the SHEATH-Recover module effectively recovered the remaining 59 feature maps. Specifically, amidst Gaussian Noise perturbations, the model’s accuracy improved significantly with SHEATH. At a standard deviation of 50% and a noise level of 65%, the model’s accuracy improved from 18.43% without SHEATH to 97.55% with SHEATH. Notably, even under a higher noise intensity, such as at a standard deviation of 90%, the model’s accuracy showed improvement from 12.51% to 97.55%. Similarly, for polarity switch attacks, and at a noise level of 85% the model’s accuracy improved from 6.26% without SHEATH to 97.36% with SHEATH. Furthermore, at higher noise levels, such as at a noise level of 95%, the accuracy improved from 4.23% to 97.36%. Hence, SHEATH can efficiently recover original feature maps amidst varying levels of noise injection.

RQ11 - Performance for large CNN architectures:

We evaluated the performance of our mitigator in the SHEATH-*Recover* module, examining both accuracy and computational overhead, particularly as it scales to larger CNN architectures. Using the grid search function (Eq. 14), we found that the mitigator for both *MiniVGGNet* and *VGG10* requires two convolutional layers to predict the remaining FMs for subsequent nodes. The second layer consistently uses the filter count that matches the number of FMs to be predicted, while the first layer’s filter count varies, enabling an analysis of computational overhead and the difference in classification accuracy between the original and recovered models.

For *MiniVGGNet*, increasing the filter count in the first convolutional layer from 32 to 96 resulted in an increase in computational overhead from 0.0059 seconds to 0.0152 seconds. Concurrently, the accuracy difference between the recovered and the original model decreased from 2.52% to 1.97%. Similarly, for *VGG10*, increasing the filters in the first layer from 32 to 96 led to an increase in computational overhead from 0.0065 seconds to 0.0154 seconds, while the accuracy difference narrowed significantly from 6.03% to 0.69% as illustrated in Fig. 16(a) and Fig. 16(b). Notably, when SHEATH’s FM recovery process is scaled from a smaller CNN architecture to a larger model, the computational overhead of the mitigator remains minimal, showing its scalability.

B. Discussion and Future Work

SHEATH enhances the security of HC-based CNNs by detecting and mitigating adversarial noise from untrusted nodes. It ensures integrity and robustness with minimal overhead, making it suitable for real-time, resource-constrained environments. Its adaptability across CNN architectures and datasets highlights its versatility and effectiveness. However, SHEATH relies on the presence of at least one trusted node within the network. In highly dynamic or hostile environments, ensuring the availability and trustworthiness of such nodes can be challenging. Lastly, another limitation is its reliance on threshold-based noise detection, which can be affected by floating-point precision limitations. This could be mitigated by integrating adaptive thresholding techniques that adjust dynamically based on statistical patterns in the feature maps or by implementing sensitivity adjustments to improve the *Comparator*'s ability to detect subtle deviations.

Adversarial Attacks and Learning: Traditional adversarial attacks like FGSM, PGD, and JSMA are ineffective in HC-based CNNs due to their reliance on full model access, gradients, and iterative feedback. In distributed HC architectures, each node only processes a segment of the model, preventing adversaries from computing gradients or loss functions. These attacks target input data, but in HC setups, adversaries control only intermediate nodes, restricting their influence to feature maps, not inputs. Manipulating feature maps is the primary feasible attack since FGSM and JSMA depend on comprehensive model knowledge, including architecture, parameters, and Jacobians, which adversaries in HC architectures lack. Thus, traditional adversarial attacks cannot function effectively in this context, as also noted by Adeyemo et al. and Goldblum et al., respectively [33], [34].

Adversarial training is a widely-used defense mechanism that enhances robustness by incorporating adversarially perturbed inputs during training. However, in HC-based CNN architectures, adversarial training presents two significant disadvantages. First, the training process becomes extremely cumbersome, as each node would need to retrain on adversarial feature maps specific to its assigned layer, which contradicts the resource-efficient design of edge-based architectures. Second, the feature maps learned during adversarial training may become irrelevant during inference. In HC scenarios, adversaries introduce noise at intermediate nodes, resulting in FMs that differ significantly from those encountered during training. This mismatch limits the effectiveness of adversarial training in defending against intermediate feature map manipulations. Therefore, while adversarial training can be applied, it is not readily usable in our HC scenario.

VIII. RELATED WORKS

We explore related works in terms of mechanisms of collaborative inference and security challenges in HC.

A. Collaboration Inference

Collaborative inference in IoT and edge devices has garnered significant interest, driving advancements in optimizing the deployment of deep learning models across distributed

TABLE IV
COMPARISON OF MODEL ACCURACIES WITH AND WITHOUT SHEATH WHEN SUBJECTED TO DIFFERENT NOISE TYPES

Std Dev (%)	Noise Level	Model Accuracy with Noise	Model Accuracy with SHEATH	Noise Type
10	65	0.8549	0.9755	Gaussian Noise
50	65	0.1843	0.9755	Gaussian Noise
90	65	0.1251	0.9755	Gaussian Noise
-	75	0.1039	0.9736	Polarity Switch
-	85	0.0626	0.9736	Polarity Switch
-	95	0.0423	0.9736	Polarity Switch

edge systems. In [35], Huang et al. proposed DeColla, a decentralized deep learning inference system tailored for IoT devices, optimizing DNN computations for enhanced efficiency and robustness. Similarly, Shlezinger et al. [36] introduced an edge ensemble framework leveraging multiple edge devices to improve prediction accuracy under constrained computational resources. Disabato et al. [37] presented a methodology for distributing the computation of CNNs onto IoT units. Their approach focused on enabling real-time recall and autonomous decision-making, addressing the challenges of implementing deep learning solutions on resource-constrained IoT devices. Naveen et al. [38] developed a model to optimize the allocation of deep learning tasks across edge nodes. Their primary goal was to minimize inference latency, ensuring faster and more efficient deep learning inference in distributed IoT edge clusters. Du et al. [39] proposed a distributed in-situ CNN inference system tailored for IoT applications. Their system utilized a loosely coupled structure, synchronization-oriented partitioning, and decentralized asynchronous communication to address the resource constraints of individual IoT devices.

Furthermore, Hemmat et al. [40] introduced the EdgeAI framework, which focuses on distributed inference with local edge devices. Their approach aimed to achieve minimal latency using partitioning, pruning, and parallel inference techniques. Zhang et al. [41] proposed DeepSlicing, a collaborative CNN inference system. By combining data and model partitioning, they aimed to enable efficient parallel inference with reduced latency, addressing the challenge of low-latency collaborative inference for CNNs on edge devices. Mao et al. [5] presented MoDNN, a locally distributed mobile computing system. Their system was designed to accelerate DNN computations on resource-constrained mobile devices by partitioning techniques and reducing non-parallel data delivery time. Zhao et al. [42] introduced the DeepThings framework, which focuses on the deployment of CNNs on resource-constrained IoT edge clusters. Their approach utilized scalable partitioning, distributed work stealing, and optimized work scheduling to ensure efficient CNN-based inference applications. Zeng et al. [11] proposed CoEdge, a distributed DNN computing system. Their system orchestrates cooperative DNN inference over heterogeneous edge devices with the primary goal of reducing overall latency by using adaptive workload partitioning and ensuring energy-efficient execution.

B. Security in HC Inference

The security of collaborative inference systems in decentralized networks is crucial, particularly with the rise of edge

devices and IoT. Odetola et al. [43] introduced FeSHI, a stealthy hardware Trojan attack that targets CNN layer feature maps. By exploiting the statistical attributes of these feature maps, they were able to design stealthy triggers and payloads, achieving misclassification at the CNN output without full knowledge of the architecture. Mohammed et al. [44] highlighted the potential security risks in distributed CNNs. They proposed Trojan attacks on CNNs implemented across multiple nodes, demonstrating the potential for misclassification in distributed CNN inference. Adeyemo et al. [27] explored the vulnerabilities of DNN models to adversarial noises, such as the Fast Gradient Signed Method (FGSM) and Gaussian Noise Perturbation (GNP). Their research aimed to determine attack detection accuracy and devise strategies to bolster system security against adversarial noise. Furthermore, Odetola et al. proposed the first hardware accelerator for adversarial noise attacks based on memristor crossbar arrays, aiming to significantly improve the throughput of a visual adversarial perturbation system, which can further enhance the robustness and security of future deep learning systems [45]. Baccour et al. [46] introduced the DistPrivacy methodology to secure sensitive data on DNNs. Their approach balanced latency, privacy level, and limited resources, ensuring that sensitive data remains confidential even in a distributed inference environment. Despite these works, there remains a gap in detecting adversarial noise-based attacks on HC-based CNNs and developing countermeasures. Our work with SHEATH aims to bridge this gap by both detecting the noise and eliminating its impacts by recovering the original feature maps in case of detection.

IX. CONCLUSION

In conclusion, we proposed a novel framework, SHEATH, to secure HC-based CNN architectures against adversarial noise. SHEATH, first, detects the adversarial noise and, then, eliminates its effect on CNN inference by recovering the original feature maps in case of a detected attack. It operates without requiring complete knowledge of the CNN model, ensuring robust data and model protection. We tested and validated SHEATH's performance under diverse attack scenarios. Our experiments reiterate SHEATH's adaptability and effectiveness across different CNNs to maintain a high inference accuracy even under attacks. For instance, CNN's overall model accuracy was increased from 18.43% without SHEATH to 97.55% with SHEATH despite a noise injection attack while incurring minimal overhead. In the future, we will extend the proposed concept for collaborative infrastructures without trusted nodes and test SHEATH on an FPGA-based hardware HC implementation to further reduce the overhead.

ACKNOWLEDGEMENT

This research was supported in part by the Air Force Office of Scientific Research (AFOSR)/ Air Force Research Laboratory (AFRL) Summer Faculty Fellowship Program (SFFP) for summer 2023 and the Department of Energy (DOE) under Award DE-NA0004016. The views and conclusions contained herein are those of the authors and should not be interpreted as

necessarily representing the official policies or endorsements, either expressed or implied, of the AFRL, DOE, or the U.S. Government.

REFERENCES

- [1] Reuters. Government firms should spend more on ai safety. <https://www.reuters.com/technology/governments-firms-should-spend-more-ai-safety-top-researchers-say>.
- [2] E. Esenogho, K. Djouani, and A. M. Kurien. Integrating artificial intelligence internet of things and 5g for next-generation smartgrid: A survey of trends challenges and prospect. *IEEE Access*, 10:4794–4831.
- [3] K. P. Seng, L. M. Ang, and E. Ngharamike. Artificial intelligence internet of things: A new paradigm of distributed sensor networks. *International Journal of Distributed Sensor Networks*, 18(3):15501477211062835, 2022.
- [4] Dataconomy. Aiot: The convergence of ai and iot. <https://dataconomy.com/2022/06/02/aiot/>, 2022.
- [5] J. Mao et al. Modnn: Local distributed mobile computing system for deep neural network. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2017, pages 1396–1401, 2017.
- [6] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45:615–629, 04 2017.
- [7] F. Pereira et al. Challenges in resource-constrained iot devices: Energy and communication as critical success factors for future iot deployment. *Sensors*, 20(22):6420.
- [8] W. Shi et al. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.
- [9] X. Wang et al. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys and Tutorials*, 22(2):869–904.
- [10] E. Li, L. Zeng, Z. Zhou, and X. Chen. Edge ai: On-demand accelerating deep neural network inference via edge computing. *2019 IEEE Transactions on Wireless Communications*, 19(1):447–457.
- [11] L. K. Zeng et al. Coedge: Cooperative dnn inference with adaptive workload partitioning over heterogeneous edge devices. *IEEE/ACM Transactions on Networking*, 29(2):595–608, 2021.
- [12] Adam Zewe. Collaborative machine learning that preserves privacy. <https://news.mit.edu/2022/collaborative-machine-learning-privacy-0907>, 2022.
- [13] A. Adeyemo, F. Khalid, T. Odetola, and S. R. Hasan. Security analysis of capsule network inference using horizontal collaboration. In *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1074–1077.
- [14] F. Khalid et al. Fademl: Understanding the impact of pre-processing noise filtering on adversarial machine learning. In *2019 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 902–907.
- [15] 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)*, pages 582–597.
- [16] H. Lee, H. Bae, and S. Yoon. Gradient masking of label smoothing in adversarial robustness. *IEEE Access*, 9:6453–6464.
- [17] X. Liu et al. Secure and lightweight feature selection for horizontal federated learning. *2024 IEEE Transactions on Information Forensics and Security*.
- [18] X. Mu, Y. Tian, Z. Zhou, and J. Xiong. Lightweight federated learning secure aggregation protocols. In *2024 International Conference on Networking and Network Applications (NaNA)*, pages 438–443.
- [19] Y. Wang et al. A platform-free proof of federated learning consensus mechanism for sustainable blockchains. *IEEE Journal on Selected Areas in Communications*, 40(12):3305–3324, 2022.
- [20] A. Teixeira et al. Horizontal collaboration in iot: Benefits and challenges. *Journal of IoT*, 5:45–58, 2017.
- [21] R. Kumar et al. A framework for collaborative computing using horizontal collaboration in iot. In *Proceedings of the International Conference on IoT Design and Implementation*, pages 115–126, 2019.
- [22] Q. Yang et al. Federated learning: Concepts and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):12.
- [23] S. Wang, Z. Zhou, and J. Liu. Layer-wise adaptive distributed mobile deep learning. In *IEEE International Conference on Mobile Ad Hoc and Smart Systems*.

