EMShepherd: Detecting Adversarial Samples via Side-channel Leakage

Ruyi Ding
Northeastern University
ding.ruy@northeastern.edu

Cheng Gongye
Northeastern University
gongye.c@northeastern.edu

Siyue Wang
Northeastern University
wang.siy@northeastern.edu

Aidong Adam Ding
Northeastern University
a.ding@northeastern.edu

Yunsi Fei
Northeastern University
y.fei@northeastern.edu

ABSTRACT

Deep Neural Networks (DNN) are vulnerable to adversarial perturbations — small changes crafted deliberately on the input to mislead the model for wrong predictions. Adversarial attacks have disastrous consequences for deep learning empowered critical applications. Existing defense and detection techniques both require extensive knowledge of the model, testing inputs and even execution details. They are not viable for general deep learning implementations where the model internal is unknown, a common 'black-box' scenario for model users. Inspired by the fact that electromagnetic (EM) emanations of a model inference are dependent on both operations and data and may contain footprints of different input classes, we propose a framework, EMShepherd, to capture EM traces of model execution, perform processing on traces and exploit them for adversarial detection. Only benign samples and their EM traces are used to train the adversarial detector: a set of EM classifiers and class-specific unsupervised anomaly detectors. When the victim model system is under attack by an adversarial example, the model execution will be different from executions for the known classes, and the EM trace will be different. We demonstrate that our air-gapped EMShepherd can effectively detect different adversarial attacks on a commonly used FPGA deep learning accelerator for both Fashion MNIST and CIFAR-10 datasets. It achieves a 100% detection rate on most types of adversarial samples, which is comparable to the state-of-the-art 'white-box' software-based detectors.

CCS CONCEPTS

• **Security and privacy** → **Side-channel analysis and countermeasures**; **Embedded systems security**.

KEYWORDS

Side-channel attacks; Adversarial machine learning; Neural network hardware

ASIA CCS '23, July 10–14, 2023, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0098-9/23/07…$15.00
https://doi.org/10.1145/3579856.3582827

ACM Reference Format:
Ruyi Ding, Cheng Gongye, Siyue Wang, Aidong Adam Ding, and Yunsi Fei. 2023. EMShepherd: Detecting Adversarial Samples via Side-channel Leakage. In *ACM ASIA Conference on Computer and Communications Security (ASIA CCS '23), July 10–14, 2023, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3579856.3582827

1 INTRODUCTION

Recent advances in deep learning have revolutionized many application domains, including computer vision [22] and natural language processing [41]. Society has benefited significantly from technological developments in AI-empowered authentication and access control [44], medical diagnosis [23, 47], autonomous driving [14], etc. However, deep-learning models in critical applications face serious security threats, including the most common adversarial attacks [25]. An adversary can manipulate the input to DNN models for inference with carefully selected perturbation to result in misclassification or misdetection. The disastrous consequences of adversarial examples include access right escalation (e.g., to critical industrial control systems or nuclear plants), fraudulent medical claims, and driving accidents.

A large body of work has been developed for both defense and detection against adversarial attacks [12, 21, 35, 52, 54]. Defense techniques harden the DNN models through adversarial training [24] or stochastic methods [37, 58, 59]. However, the adversarial examples are provided by certain adversarial generation methods, and the model retrained may not be resilient to other unknown adversarial attacks, potentially stronger with different feature characteristics. Model retraining also has privacy implications as it requires to be iterative to keep up with new attacks [60]. Furthermore, these protection methods have been circumvented recently by the most sophisticated attack [16]. Another line of work is to detect the adversarial examples during model execution, which can be external to the model, therefore, more agile, general, and robust. Existing detection methods rely on observing intermediate execution features or model behavior [38, 61] or input statistics [26, 39], and leveraging them for adversarial detection, a 'white-box' scenario with the model internal and run-time execution details known.

However, there are plenty of cases where the model users have limited access to the model intermediate parameters or testing images, which we also called it a 'black-box' system. For example, machine learning models in the healthcare system may be kept confidential due to their values and privacy concerns, where the model suppliers tend to offer model users only limited interfaces

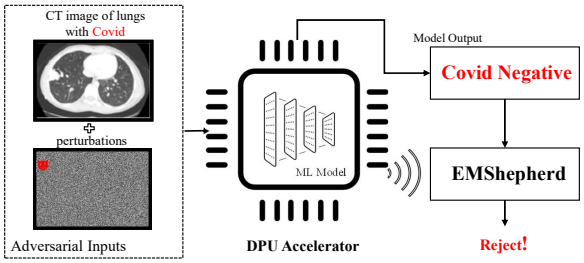


Fig. 1: EMShepherd detection framework: the medical machine learning model misdiagnoses an attacked CT lung image with Covid, captured by EMShepherd

so as to prevent reverse engineering or membership inference attacks [49]. Current detection strategies do not suit such "black-box" systems, and they require direct access to the model structure and parameters, execution details such as activations, testing images, or model intermediate outputs, including model logits. With privacy concerns, we target building an adversarial detector without access to both testing inputs and model execution details (i.e., feature maps).

Inspired by the simple Electromagnetic analysis (SEMA), which associates the EM emanation patterns of a computer platform with the operations it is running and data processing [10], we leverage side-channel EM leakage for detecting adversarial examples. The intuition is that different classes of images will activate the network model differently and yield different patterns in the EM traces. For adversarial examples that impose perturbations on a source class and fool the victim model into misclassifying it as a target class, the semantic information leaked from the EM trace may present a discrepancy from that of the target class (learned with prior training), i.e., revealing an anomaly.

In this paper, we propose EMShepherd, a framework of adversarial sample detection via EM side-channel leakage, which treats the victim model as a 'black-box' without probing it for execution details. The victim model is deployed on a physical device that the user can access, e.g., an IoT edge device or a local inspection station. Note remote access to DNN models in the cloud, a so-called Machine-learning-as-a-Service (MLaaS) scenario, is out of scope for our work. The DNN implementation on the device can be software running on a CPU or GPU or a hardware accelerator running on FPGA, NPU, or TPU. Fig. 1 shows an example setup of EMShepherd, where the deep learning inference system is attacked by samples with malicious perturbations, and the additional air-gapped EMShepherd fends off the adversarial sample at runtime with correct detection. For running the adversarial detector, neither the victim model (both static model internals and dynamic execution details) nor the inputs are needed. Note our work uses EM side-channel as an example, while the framework is generally applicable to power side-channel as well, collected either by equipment like an oscilloscope or through on-chip power sensors [2, 36] where the measuring resolution has to be commensurate with the detection goal.

This work makes several contributions as follows:

- We leverage side-channel EM leakage, for the first time, for detecting adversarial attacks under a "black-box" scenario. We propose EMShepherd adversarial detector, which requires no prior knowledge of the victim models, adversarial attacks, intermediate execution details, and the model inputs.
- Our novel EMShepherd framework consists of scripts for EM measurements, a novel data processing method to tame the EM traces for follow-on class-specific feature extraction and learning, and an unsupervised anomaly detector.
- We evaluate the framework on 5 different adversarial attacks on the Fashion MNIST dataset running on a common FPGA deep learning accelerator. Our results show that the EM-based detector can effectively detect all attacks with over 90% detection accuracy and an acceptable false positive rate (less than 10%). EMShepherd is also applied on a large VGG neural network accelerator with the CIFAR-10 dataset.
- We further evaluate its performance on a robust retrained model with adversarial examples, and EMShepherd demonstrates high accuracy and low false positive rate as well.
- We compare our method with the state-of-the-art white-box software-based detection methods. The results show that the performance of our adversarial detector is comparable to the prior methods.

2 BACKGROUND

This section presents relevant background on adversarial attacks, protection, and EM side-channel.

2.1 Adversarial Attacks on DNNs

DNN is an artificial neural network with multiple layers to represent a function, $F : X \rightarrow Y$, with parameters ω_F such as weights, kernels, and biases, where X denotes the input space and Y the output space. In the training phase, a DNN is trained with a dataset of input-output pairs to arrive at optimal values of ω_F, to minimize the loss function J_F, which is a distance measurement between the model predicted result $F(x)$ and the ground truth $y*$. The widely-used optimizers include stochastic gradient descent (SGD) [20] and Adam [31]. Taking image classification as an example, the DNN model runs inference on the unknown input, x_t, and predicts a class, out of m classes, with the largest probability.

$$y = F(x_t) = \text{softmax}(Z(x_t)) \quad (1)$$

where the vector $Z(x_t)$ is known as logits. Our detection method assumes that the defender can only query the model and know the output y while the logits $Z(x)$ are unavailable.

DNN model is vulnerable to adversarial attacks. An adversarial sample (x') is a carefully crafted sample, which has a human-imperceivable difference from the original benign sample (x), but causes the DNN to misclassify it to a different class $F(x') \neq y$. If $F(x')$ is an arbitrary class except for y, x' is an untargeted adversarial sample. A more restrictive and harmful case is the targeted adversarial sample, where $F(x') = l \neq y$, a specific target class. In this paper, we consider both untargeted and targeted attacks. The difference between the adversarial example and benign example can be measured by L_p, defined as $\Delta_p = \sum_{i=0}^{n}(|x_i - x'_i|^p)^{\frac{1}{p}}$. Common choices of L_p include: L_0, the number of pixels changed; L_1, the

Manhattan norm; L_2, the Euclidean distance norm; L_{inf} the largest absolute change of any pixels. The adversarial attack can be viewed as an optimization problem:

$$\textbf{minimize} \quad \Delta_p(x, x') \quad \textbf{s.t.} \quad F(x') \neq y \tag{2}$$

GoodFellow et al. [25] first proposed the concept of the adversarial sample and introduced Fast Gradient Sign Method (FGSM) to generate adversarial samples. Madry et al. introduced the Projected Gradient Descent (PGD) [40] attack to improve the attack efficiency of the Basic Iterative Method (BIM) [33]. Other learning-based attacks include DeepFool [43] and Carlini and Wagner Attack (CW) [16], where CW attack is proven to be one of the strongest attacks [15, 16] at the cost of generation speed.

2.2 Existing Software Detection Methods

The existing adversarial detection methods, all software-based, can be classified into three categories.

Distributional Detection: These detectors perform some statistical analysis on the inputs or intermediate values of model execution (e.g., activation values) to find adversarial samples. Grosse [26] used Maximum Mean Discrepancy test (MMD) to determine whether the benign and adversarial inputs have the same underlying distribution or not. Feinman [21] use Kernel Density Estimation to measure the distance of distributions. However, these detection methods are ineffective on more complex datasets [15]. Ma [39] introduced Local Intrinsic Dimensionality (LID) to characterize adversarial regions of the model. However, LID is proven to perform poorly on a number of attacks [38].

Latent Space Detection: The second type of detector employs a pre-processing step to reduce variation. Grosse [26] found that adversarial examples tend to place a higher weight on larger principal components, narrowing down the targets for detection. Some approaches train denoisers to reconstruct the inputs by removing the adversarial noise added by the attacker, such as auto-encoders used in MagNet [42] and the mean blur method used in [35]. Most of them work for simple attack methods such as FGSM, but cannot resist the state-of-the-art CW attack.

Inconsistency Detection: This approach focuses on the model misbehavior during inference of adversarial examples. Feinman et al. [21] proposed Bayesian neural network uncertainty to measure the uncertainty of a DNN under a given input. By introducing some randomness (e.g., Dropout [52]) during the inference, the DNN model tends to give the same outputs for benign inputs but different outputs for adversarial ones. The Feature Squeezing approach [61] reduces the color depth and observe that adversarial samples are likely to induce different classification results while benign inputs are not. Tao et al. [54] introduced the Attacks meet Interpretability structure (AmI), which measures the inconsistency of the victim DNN with another neural network enhanced with human perceptible attributes under adversarial examples. In the Network Invariant Checking (NIC) work proposed by Ma [38], the key idea is during model execution, there are class-dependent provenance channels (the distribution of activated neurons in the network) and activation value channels (value distributions of activated neurons). It employs a one-class SVM to determine outliers. NIC shows promising results against a broader range of attacks, including the CW attack. Our approach generally falls into the type of inconsistency detection, in a black-box victim system scenario. In Appendix B, we present a motivation example for our design EMShepherd .

2.3 Electromagnetic and Power Side-Channel

Both EM emanations and power consumption of a computer system depend on the circuit operations and data [9]. Such side-channels have been extensively analyzed to retrieve the secret key of cryptographic algorithms [17, 19, 32], and recently have been utilized to infer deep neural network model information. Yu [63] proposed a SEMA to retrieve the topology of the victim model. Batina [11] applied differential EM analysis to recover simple MLP model parameters of microcontroller implementations. Zhang [64] successfully extracted the structure of a network running on an FPGA via power side-channel. Chmielewski [18] targeted GPU DNN implementations and recovered the model structure with EM side-channel information. All the prior work focuses on reverse engineering partial *model* information, while our work associates the EM emanation patterns with input sample classes.

There are multiple strategies to analyze EM/power side channel information, such as traditional statistical way and modern learning-based methods. Statistical analysis requires alignment of the EM/power measurements with the computation processes and relies on certain power models, such as Hamming Weight and Hamming Distance model, or mutual information between distributions to discern the secret. When leveraging EM side-channel leakage of DNN model execution for classification and adversarial detection, the model structure is complex, the execution is computation-intensive, and the hardware platform supports highly parallel operations, and therefore learning-based methods are more suitable for coping with the misalignment, noise, and feature extraction, etc.

2.4 Target Platform

We choose Xilinx® Deep Learning Processing Unit (DPU) as our platform. DPU is a popular configurable hardware neural network accelerator on FPGA and achieves the best throughput for DNN inference [66]. DPU supports common CNNs such as VGG[51], ResNet[27], GoogLeNet[53], YOLO[46], and MobileNet[28]. Xilinx provides Vitis AI [8], a development stack to compile neural networks software trained with generic DNN platforms such as [4], onto a DPU accelerator.

3 THREAT MODEL AND ADVANTAGES OF OUR HARDWARE-BASED ADVERSARIAL DETECTOR

3.1 Threat Model

The victim is a DNN classifier, which is pre-trained with a public dataset. The testing dataset may be kept private. We assume the strongest 'white-box' attack model, where the attacker has full knowledge of the victim model and training dataset in order to generate adversarial samples with minimum perturbations. On the contrary, the detection system assumes the most limited scenario, under a 'black-box' view of the victim, without access to the victim's inputs, parameters, and intermediate outputs or execution

details. The only information available to the detector to distinguish adversarial samples is the EM side-channel measurement and the victim model's prediction class. For training the adversarial detector with EM traces, a public benign dataset is used.

3.2 Advantages

Compared to software-based adversarial detection methods, our hardware-based detector, EMShepherd, has three distinct advantages: privacy-preserving, portability, and robustness.

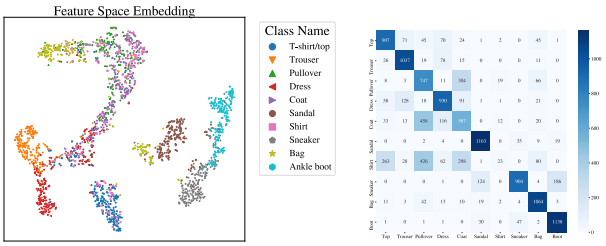
- EMShepherd protects the DNN model user's data privacy as it is agnostic to the model's inputs, which instead are always required by prior reconstruction-based detection methods [42, 62]. The sensitive inputs should not be shared with *third-party detectors*. Our design only requires the output class labels and the EM signals, which are passively leaked to common acquisition equipment.
- EMShepherd also protects the model confidentiality. No model information, including hyper-parameters, parameters, and logits, is needed, in stark contrast to the previous software-based detection methods [21, 38]. The EM data processing and the adversarial detector training process are both victim model-agnostic. Therefore, our method has more general usage, applicable to closed-source DNN applications, which are pervasive in edge devices where the user only queries the models for the final prediction output.
- Owing to the model-agnostic feature, EMShepherd can be easily ported for wide-range hardware devices with different DNN implementations for diverse applications. It can be used as a 'plug and play' (PnP) device, aside from the target system, to work automatically without user intervention or contact with the victim system.
- Adaptive attack [55] is a threat to most software defense methods where the attacker adjusts the adversarial perturbations to mislead both the victim models and defense systems. However, due to the high complexity and non-explicit dependency of the EM signals on computations and data, it is extremely hard to have an adaptive attack on our detection method, i.e., adversarial examples whose EM signals are deliberately controlled to evade the EM-based detector.

4 EMSHEPHERD DESIGN

We next present the design rationales for the EMShepherd framework, the composition of the adversarial detector and salient functions.

4.1 EM Emanations of DNN

As mentioned in Section B, semantic information can be used in adversarial detection, but how can we get it under a 'black-box' setting? We leverage EM side-channel leakage to characterize the semantic computational information for benign inputs. Model inference is a highly computation-intensive task, involving multiple stages of parallel computation, making the EM signals complex and their dependency on computations hard to model. Learning-based methods can tackle these noisy EM signals well to extract class-specific features. We build convolutional neural network (CNN) classifiers based on an EM dataset collected from a target system running on Fashion MNIST (with 10 classes). Figure 2(a) presents



(a) CNN Feature Space (b) Confusion Matrix

Fig. 2: Feature Space Representation and Confusion Matrix of EM signals

the feature space embedding of a CNN classification model on the testing EM dataset using a commonly-used visualization tool, T-distributed stochastic neighbor embedding (TSNE) [56]. Figure 2(b) shows the confusion matrix of the CNN model prediction results. We notice that,

- The CNN classifier can extract class-related features from the EM signals. Some classes' features are distinct from others, while some overlap with others.
- The embeddings of the classes with similar semantic information are located near each other, such as the cluster of (Sandal, Sneaker, Ankle Boot) (all shoes) and the cluster of (Shirt, Coat, Pullover) (all tops).

Based on these findings, we will discuss our design of EMShepherd , which leverages this semantic information in the EM signals with the model outputs to detect adversarial samples. The design will also overcome the low prediction accuracy for some classes by further exploiting the feature space with anomaly detectors.

4.2 Overview of the Detection System

Figure 3 shows an overview of the EMShepherd detection framework. The victim model is an image classifier running on a Xilinx DPU, and an EM trace is collected for each model execution. During the training phase, the model is queried with a benign training dataset and corresponding a training EM trace dataset is collected. These traces will be used to train EM classifiers whose outputs are utilized to fit a set of class-specific anomaly detectors. After this phase, all the trainable components are fitted and the parameters are fixed, an EMShepherd detector is generated. In the detection phase, the pre-trained EMShepherd takes in the EM trace collected during an image inference, processes it, and feeds it to the follow-on EM classifiers and anomaly detector, to accurately detect adversarial examples guided by the output label from the victim classifier.

4.3 Notation and Definition

We next define notations used along with our system design. The victim model $\mathcal{M}_v$ is a pre-trained N-class classification model running on a device, facing adversarial attacks. One input image to the victim model is denoted $Im_i \in \mathcal{I}_v$, and the corresponding output label is y_i. The corresponding EM trace for the execution collected is $\mathbf{T}_i \in \mathcal{T}_v$, each with P number of points. Every $\mathbf{T}_i$ can be partitioned into multiple computation segments $\mathbf{T}_i = concat(\{\mathbf{B}_1, \mathbf{B}_2, ..., \mathbf{B}_M\}_i)$, where the number of segments, M, depends on both the structure of

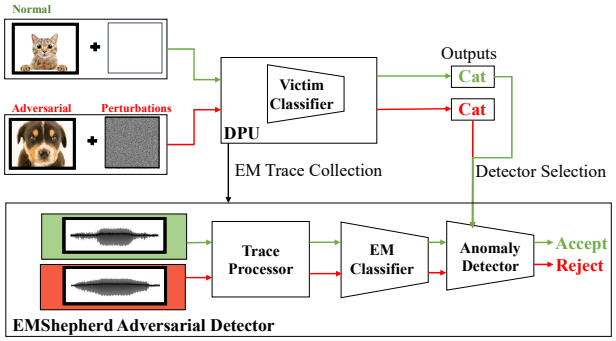


Fig. 3: Overview of the EMShepherd detection flow

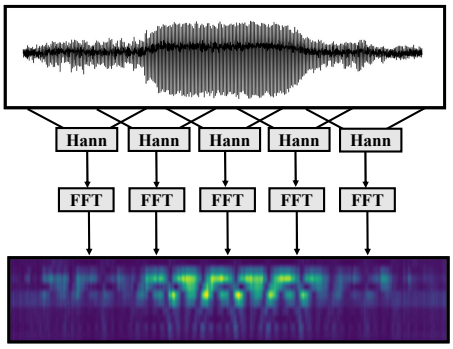


Fig. 4: Short-Time Fourier Transform

$\mathcal{M}_v$ and its implementation on the victim device. Note that M may be larger than the number of layers, as off-chip communications can happen within a layer due to the on-chip resource constraints.

EM segments are 1-D time series, and are preprocessed by Short-Time-Fourier-Transform (STFT) to generate 2-D EM spectrograms (details will be given in Section 4.4), $\mathbf{B}_m \rightarrow \mathbf{S}_m(t, f)$ in both time and frequency dimensions, where $m = 1, 2, \ldots, M$ denotes the segment index. Correspondingly one EM classifier is built on one EM segment, denoted C_m. Given an input image Im_i, the logits of EM classifier m is denoted as $\{\mathbf{L}_m\}_i$. Then logits of all M EM classifiers are concatenated into one vector $\boldsymbol{l}_i = concat(\{\mathbf{L}_m\}_i)$, providing a holistic view of the victim model's internal processing for input image Im_i.

The class-specific anomaly detectors are built for all N classes, denoted $\{\mathcal{D}_n\}$, where $n = 1, 2, ..., N$, one for each class. $\mathcal{D}_n$ is trained with $\boldsymbol{l}_i$ of the benign training samples with $y_i = n$, and a threshold is selected for each class. For a testing input Im_j with the prediction label n, the loss of the anomaly detector, denoted as $\ell(Im_i)$, is compared with the corresponding class threshold, L_T, to detect whether the input is adversarial. More details will be illustrated in Section 4.6.

4.4 Data Collection and Preprocessing

Fig. 3 shows that the raw EM traces, $\mathbf{T}_i \in \mathcal{T}_v$, will first go through a trace processor for denoising and transformation. The trace processor performs two tasks: extracting high-energy segments from raw traces and converting each EM segment into a spectrogram. We analyze the trace profile and find that different input images will result in different amplitudes on the EM traces, but they all have the same number of segments due to the hardware accelerator structure. We pick local maximum leakage points and split each trace into multiple segments. Note that we do not need to align the segments in the time domain as we will use Short-Time Fourier Transformation to do time-frequency domain conversion. Fig. 4 depicts the processing method - Short-Time Fourier Transform. A sliding Hanning window (e.g., 256 points) with a stride of half of the window is used to transform the raw signals progressively. Between two windows, the overlapping time points make sure that no information is lost by preserving the signals on the windows' boundaries. In each window, we apply FFT to convert the signal from the time domain to the frequency domain to generate a spectrum.

$$\mathbf{S}_m(t, f) = \int_{(t-1)w}^{(t+1)w} \mathbf{B}_m(\tau) e^{-j2\pi f \tau} d\tau \tag{3}$$

where $m = 1, 2, \ldots, M$ denotes the segment index, w denotes half of the STFT window size, $t = 1, 2, \cdots, P_m/w$ denotes the index of time windows in a segment, and f is the frequency selected.
There are three benefits of using STFT.

- Noise-Filtering. As shown in Fig. 4 where the Y-axis of the spectrogram is the frequency and the brighter the color the higher the amplitude, the main energy (the brighter part) is focused on the operating frequency of the victim model, which is $150MHz$ in our case. The rest frequencies have relatively lower energy. Thus, we can select 15 bands around the operating frequency out of 256 bands and filter out other lower-energy components, which increases the signal-noise ratio (SNR) of the remaining EM frequencies.
- Dimension-Reduction. The number of points in the spectrogram is reduced by w times in the time domain compared to the raw EM segment, which makes the follow-on model learning capture the temporal patterns easier.
- 2-D image. Compared to 1-D raw EM traces, the spectrogram naturally fits CNN classifiers and kernels, which not only provides time and frequency information but also the change of the spectrums along the time. More details will be presented in Section 5.2.2 that the EM classifiers indeed exploit both time and frequency information in the spectrograms for classification.

4.5 EM Classifiers

As aforementioned, the EM trace can leak class-specific computation and activation information during the inference process for an input sample Im_i. We train a DNN classifier on each segment of EM spectrogram, with all benign samples in the EM training dataset, as depicted in Fig. 5.

After that, we will concatenate all the EM classifiers' outputs into one logit vector for all the benign samples that belong to one class. This is based on the observation that for one input image, although all the constituent EM classifiers give out the same class prediction, their logit vectors may differ significantly, which may carry finer-grained feature/semantic information. We use the experimental results on the Fashion MNIST dataset as an example. As shown in Fig. 5, for an input image *Sneaker*, the classifier on the first spectrogram gives out the correct prediction of *Sneaker*, with a 0.88 confidence score. However, Classifier M on the M^{th} spectrogram,

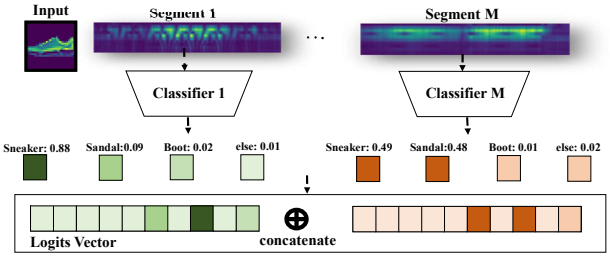


Fig. 5: EM-based classifiers and logits output

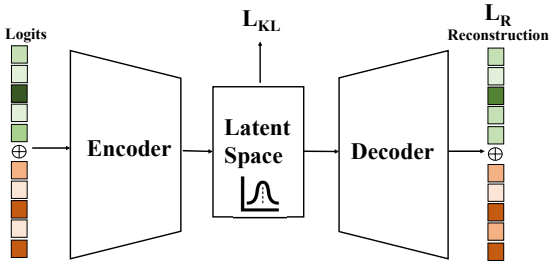


Fig. 6: The structure of anomaly detector and VAE

although also gives out the correct prediction, has much lower confidence as 0.49. One explanation is that the victim model processes the semantic information layer by layer to get a final discriminative output, where various EM segments correspond to the computation on different layers and different features. Further, according to our experimental results, it is not necessary for every EM classifier to correctly identify the image class to be useful for the later anomaly detector. One segment of the model's internal operations can focus on some features that are not informative enough to identify the image class, and the corresponding EM classifier may not result in a high score/logit for the class. However, such information (this particular segment is not informative enough to distinguish image classes) can still be helpful when adversarial samples cause a different pattern, such as resulting in a high confidence score for a class instead of among the outputs of this segment EM classifier. It is the *deviation* of the logits from benign ones that contribute to anomaly detection. The concatenated logits vector, $\boldsymbol{l} = concat(\mathbf{L}_1, ..., \mathbf{L}_M)$, provides a pattern of how segments of internal model operations correlate to the class identification, regardless of high or low, which resembles an entire inference process across layers of the victim DNN model.

4.6 Anomaly Detection Models

The concatenated logits vector $\boldsymbol{l}$ of benign inputs from the same class are likely to be similar. Therefore, we can build OOD detector to distinguish adversarial samples which very likely do not fall into the target class (claimed by the victim model). We select a reconstruct-based detector using Variational AutoEncoder (VAE). The structure of VAE is shown in Fig. 6, with an encoder followed by a decoder, where the middle layer is the latent-space representation. The encoder and decoder of our VAE each contain four fully-connected layers. The loss of the VAE includes a latent-space regularizer loss ℓ_{KL} in addition to the encoder-decoder's reconstruction loss ℓ_{recon}. The ℓ_{KL} measures the Kullback-Leibler divergence of the latent space when fitted to some distribution assumption, which is a Gaussian distribution in our case. The ℓ_{recon} measures the difference between the output and the input of the autoencoder. When fitting the VAE, we utilize ADAM optimizer to minimize the total loss $\ell_{total} = \ell_{recon} + \lambda\ell_{KL}$, where λ is a constant.

The VAE total loss ell_{total} during inference can be used to detect OOD samples. When inferring benign samples, the EM classifiers' logits will match the prediction output, which fits the pre-trained VAE model with a lower loss. However, when the victim model is attacked by an adversarial sample that misleads the prediction to be a different target class, the EM signals will be Out-of-Distribution. Therefore, the EM classifiers' logits will also be Out-of-Distribution leading to a higher VAE loss. By selecting a VAE loss threshold based on the validation of benign samples, one can detect adversarial samples with a controllable false positive rate.

5 EXPERIMENTS AND EVALUATIONS

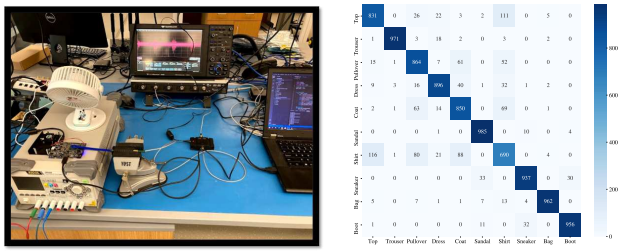
In this section, we present the experimental setup and evaluation results.

5.1 Experiment Setup

EM Trace Collection: The device under test (DUT) is a Xilinx DPU [3] running on an Ultra96-V2 board [7], a multi-processor System-on-Chip with ARM cores and Xilinx Zynq UltraScale+ FPGA. The board runs the official PYNQ image v2.5 from the vendor AVNET [1] [5]. The EM probe is PBS set 2 with a pre-amplifier [6]. We use a Lecroy WR640Zi oscilloscope [34] to collect EM traces. Fig. 7(a) shows a picture of our trace collection setup. The control and monitoring workstation first sends a command via SSH to the DUT with the pre-trained DNN model (bitstream) deployed, and the DUT loads an input image and starts executing inference. Meanwhile, the oscilloscope is triggered to capture the EM trace until the DUT finishes execution. Then the trace is streamed to the workstation for storage and processing. The collected dataset of EM traces is used to train the EM classifiers and the anomaly detectors. The training is performed on a server, with an AMD Ryzen 9 3900X 12-Core processor, 32 GB of RAM, and one Nvidia GTX TITAN GPU card.

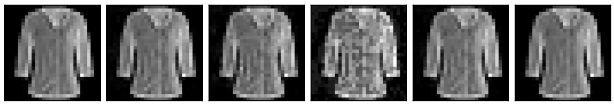
Datasets and Victim Models: We start from a LeNet-5 convolutional neural network on Fashion MNIST to evaluate our EMShepherd framework. We also experiment with a robust LeNet-5 retrained with adversarial examples. Furthermore, we evaluate our framework on a larger VGG model over the colored CIFAR-10 dataset (See Appendix C).The Fashion MNIST dataset is representative of computer vision tasks suitable for edge devices such as FPGA accelerators and mobile systems. It consists of a training set of 60,000 examples and a test set of 10,000 samples, which are 28×28 grayscale images, labeled into 10 classes. The LeNet-5 CNN achieves a 91.2% prediction accuracy on the dataset [13].The confusion matrix of the LeNet-5 on Fashion MNIST is given in Fig. 7(b). Note that among the ten classes, the model is more likely to misclassify Class *Shirt* (the 6^{th} row in the confusion matrix) to other three classes, *T-shirt*, *Pullover*, and *Coat*, due to similar features. This lower classification rate for these classes will affect the performance of our adversarial detector accordingly, analyzed

[1] B1600, which supports up to 1600 multiplications and additions per clock cycle.



(a) Trace Collection Setup (b) Confusion Matrix

Fig. 7: Collection Setup and Victim's Confusion Matrix



Origin **PGD L_1** **PGD L_2** **PGD L_{inf}** **CW L_2** **DF L_2**

Fig. 8: Adversarial images (*T-shirt* to *trouser*)

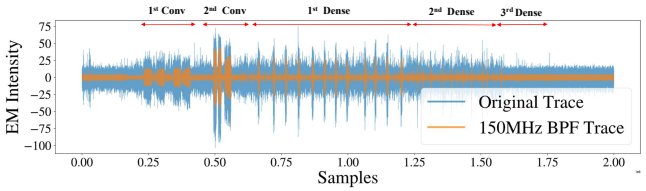


Fig. 9: EM trace (blue) and BPF-filtered trace (yellow)

in detail in Section 5.4. CIFAR-10 is more complex, consisting of 60,000 32×32 color images in 10 classes. For the CIFAR-10 experiment in Appendix C, we adopt a larger VGG-like model and obtain a 90.5% testing accuracy. We randomly divide these datasets into a training subset (60%), a validation subset (20%), and a testing subset (20%).

Adversarial Attacks: Our adversarial detector can detect a wide range of adversarial examples with EM emanations. We employ three state-of-the-art adversarial attack methods discussed before in Section 2.1: CW (targeted), PGD (targeted), and DeepFool (untargeted). For PGD attacks, we test different distance measurements: L_1, L_2, and L_{inf} to evaluate the model robustness against various distance losses. For CW and DeepFool attacks, we test the commonly used L_2 measurements. All the attack implementations are from the Foolbox library with commonly-used parameters [45]. For targeted attacks, we consider a general attack model where the targeted label (misclassification) can be any of the incorrect classes. When evaluating the adversarial detector performance, we sample the examples to different adversarial classes for both targeted and untargeted attacks. For each class, we select 9,000 adversarial samples equally distributed among the rest 9 source classes. Fig. 8 shows one image from the source class of *shirt* and corresponding adversarial images generated by various attack methods to a target class of *trouser*. When evaluating CIFAR-10, we present the results of PGD L1 attacks due to the page limit.

Detection Evaluation Metrics: We evaluate two main components of the detector, EM classifiers and anomaly detectors, with two metrics: testing accuracy and F1-score. Note that our training is only on benign examples while the detection (inference) is on unknown benign or adversarial samples. In practice, the number of adversarial samples is far less than the benign ones. Due to this imbalance, we use F1-score [1], F_{em}, to measure the classification performance.

$$F_{em} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{4}$$

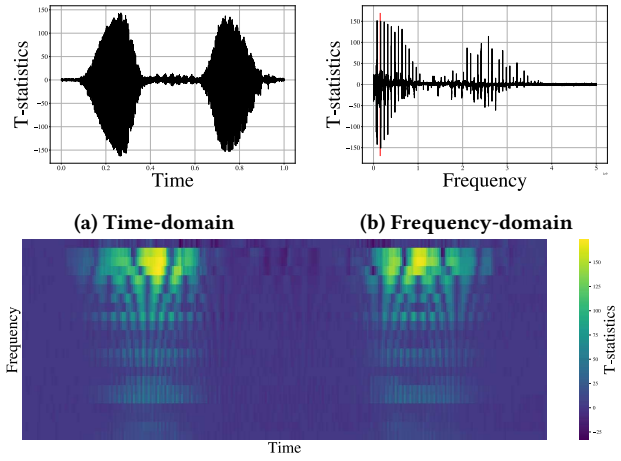
where TP, FP, FN are the true positive (adversarial detection rate), false positive and false negative ratio of the prediction results, respectively. We plot Precision-Recall Curve (PR curve) to show the trade-off between detection precision and recall. Just like ROC AUC [29], the Precision-Recall Area Under Curve (PR AUC) Score can be used for comparison between different detection settings.

Baseline Comparison: To the best of our knowledge, EMShepherd is the first hardware-based adversarial detector. It captures the contradiction between semantic EM signals and the victim output under a 'black-box' setting. We compare our method with four prior software-based detection methods, Kernel Density Estimation (KDE) [21], Network Invariant Checking (NIC) [38], Feature Squeezing (FS) [61] and MagNet [42], against PGD and CW L_2 adversarial samples. However, these baseline methods aren't all 'black-box': KDE and NIC requires the model's intermediate outputs; FS and MagNet requires testing inputs. The metric we use for comparison is Detection Rate (DR) when the FP rate is controlled at 10%.

5.2 EM Trace Processor

5.2.1 EM traces and segmentation. Fig. 9 shows an example EM trace for one benign image inference, where the blue curve is the original trace with sample point as the x-axis and EM leakage intensity as the y-axis. We apply a bandpass filter (BPF) with a center frequency 150MHz to reduce the noise and obtain a clearer signal (yellow trace). After BPF, the high-intensity segments will be clean enough and can be easily partitioned. Among the segments in Fig. 9, the first 6 are long segments (more than 30,000 sample points) and the following ones are shorter (less than 10,000 sample points). We infer that the longer segments come from the first two convolutional layers, which utilize more Processing Element (PE) for parallel computation. The rest shorter segments come from the dense layers, which run faster and turn out to be less informative. In real applications, the detector has a black-box view of the model and has no information about which layer the segment comes from, but can automatically process the trace with BPF and partitioning.

5.2.2 EM Spectrograms. Class-related features/signals in the EM traces have to be preserved to build a highly accurate EM classifier. We show that the spectrograms generated by our STFT data processing method outperform both the time-domain traces and the simple frequency-domain spectrum (after applying the fast-fourier-transformation (FFT) on the entire time-domain EM trace). To localize and detect class-related signals, we run the victim model on two classes of input images and collect EM traces. We applied the student T-test across the two class-datasets, on three kinds of data representations of EM traces: spectrograms, the original time-domain traces, and the frequency-domain spectrums. The T-test statistically tests the average difference between two groups of data. A large absolute value of T-statistics on a point indicates that the



(a) Time-domain (b) Frequency-domain

(c) Spectrogram T-scores

Fig. 10: (a) T-statistics of time-domain traces; (b) T-statistics of frequency-domain spectrums, the red line marks the operating frequency 150MHz; (c)T-statistics of spectrograms, the intensity of each point represents the T value.

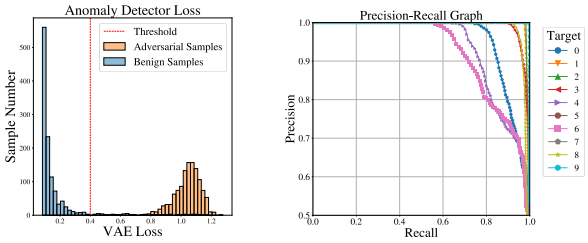
traces of two different classes differ significantly here, thus this location contains a strong class-specific signal.

Fig. 10 presents the T-statistics results for Class 0 and Class 1. The T value is shown on the y-axis, while the X axis shows the time for the time-domain trace in Fig. 10 (a) and frequency for the frequency-domain spectrum in Fig. 10 (b). For 2-D spectrogram T scores shown in Fig. 10 (c), the X axis is the time, Y axis is the frequency while the T value is represented by the intensity on the heatmap. Intuitively the spectrogram depicts the time-varying spectrum, while the frequency-domain spectrum just presents average frequency components. When comparing Fig. 10 (a) and (c), we can view each row of Fig. 10 (c) as a constituent component of Fig. 10 (a). By filtering the bottom rows and only keeping the rows with high intensity (near the top), we are filtering irrelevant noise with low T-values. When comparing Fig. 10 (b) and (c), we can view each column of Fig. 10 (c) as a spectrum for a short time window, and the spectrum is varying along the time. The energy (high intensity) focuses on the frequency band near the top of Fig. 10 (c) (i.e., the beginning frequencies of Fig. 10 (b)), which is around the operating frequency of the DUT.

Fig. 10 also shows the peak absolute value of T-statistics of the spectrogram is 173.14 compared to 162.79 and 151.64 for the time-domain traces and frequency-domain spectrums, respectively. We can conclude that the spectrogram contains more signals for classification than the other two forms of data. In our experiment, we only select 15 frequency bands of the spectrogram around the device operating frequency and discard other bands. This bandpass filtering is effective de-noising. As spectrogram preserves both the frequency and time information, its 2-D form resembles an image and suits CNN classification naturally.

5.3 Evaluation of EM Classifiers

Table 1 gives the performance of EM classifiers (we use VGG-11 models) for LeNet-5 on MNIST. For the original raw trace in Fig. 9,



(a) Target 1 VAE loss (b) PR curves for classes

Fig. 11: VAE Loss of Target 1 and PRcurves for ten targets

we analyze the first 18 segments corresponding to computations for the two convolutional layers and the first dense layer of the victim model, while the last two layers leak less information (with lower intensity and shorter time). We process each of the segments separately with STFT and build a classifier, locating the most prominent class-specific information.

For most of the segments, the EM classifier does extract some model execution information of that segment, where the class prediction accuracies based only on EM traces range from 23% to 70%. Some segments (the 12th-16th) from dense layers reveal little information, resulting in only 10% accuracy (the same as a random guess among 10 classes). We choose to only use the strong signals from convolutional layers for adversarial detection in the next part without much degradation.

For different classes, the amount of information carried in each segment also varies. As an example, we separate the first three-segment EM classification performances by the output class labels and report them in Table 2. For instance, for the 8^{th} class, Segment 0 contains most information; for Class 5, Segment 1 contains most information; for Class 1, Segment 2. Different segments of DNN execution focus on different semantic features. Since the most informative feature varies from one output class to another, the most informative EM segment also varies.

To visualize the utilization of varying parts of EM segments, we use the GradCAM [48] on our EM classifiers. The results are shown in Appendix A. For different classes, the benign inputs generally activate different neurons within a segment, and our EM classifiers capture the patterns. When an adversarial example does not activate these neurons, it results in different output logits for the EM classifier.

5.4 Evaluation of Anomaly Detector

We concatenate the logits from EM classifiers for all segments to get a logits vector reflecting the execution flow of the victim model. The follow-on VAE extracts compressed latent features from the benign logits vectors so that the benign vectors can be reconstructed from the compressed latent features with small loss. The adversarial examples cause different execution flows and their logits vectors can not be well reconstructed from compressed latent features. Fig. 11(a) presents the testing reconstruction loss of the pre-trained VAE for both benign and adversarial samples, where the blue bars are for benign samples and the yellow bars are for adversarial examples from PGD L2 attack. The red dash vertical line is an

Table 1: The EM classifiers' performance for each segment

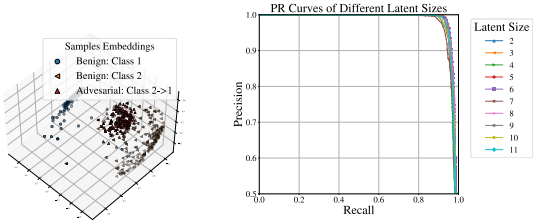
Layer Type	Convolutional Layers						Fully-connected Layers							
Index	1st Layer				2nd Layer		3rd Layer							
Segment	0	1	2	3	4	5	6	7	8	9	10	11	12-16	17
Accuracy	0.68	0.70	0.64	0.56	0.68	0.68	0.23	0.38	0.37	0.10	0.33	0.33	0.10	0.59
F1-score	0.65	0.67	0.61	0.52	0.66	0.66	0.18	0.35	0.32	0.02	0.26	0.26	0.02	0.57

Table 2: The EM classifiers' classification report for Segment 0, 1, 2

Class		0	1	2	3	4	5	6	7	8	9
Segment 0	Accuracy	0.58	0.92	0.55	0.55	0.49	0.67	0.35	0.64	**0.89**	0.83
	F1-score	0.56	0.81	0.38	0.65	0.50	0.63	0.37	0.72	**0.91**	0.77
Segment 1	Accuracy	0.70	0.81	0.71	0.69	0.40	**0.85**	0.37	0.91	0.81	0.85
	F1-score	0.72	0.86	0.53	0.71	0.42	**0.90**	0.04	0.82	0.84	0.88
Segment 2	Accuracy	0.59	**0.95**	0.47	0.65	0.35	0.82	0.47	0.65	0.62	0.80
	F1-score	0.68	**0.92**	0.53	0.68	0.40	0.70	0.08	0.74	0.61	0.82

empirically selected threshold to determine whether the input is benign or adversarial. It shows that two distributions are disjoint and adversarial examples can be easily distinguished from benign examples. Fig. 11(b) shows the precision-recall curves of the VAE for all 10 classes. Similar to the receiver operating characteristic curve, the PR curve shows the model performance trade-off between precisions and recalls. The classification algorithm is desired to have both high precision and high recall. Therefore, a larger Area Under Curve (AUC) indicates a better classifier. Three classes, Class 0, 4, 6, have relatively worse performance, due to the original classification inaccuracy of the victim model among these classes.

In Fig. 12, we visualize the features of a selected adversarial sample (generated by PGD L_2 attack) and two benign samples (one of the source class and the other of the target class) with a 3-D embedding of their logits vectors. The embedding demonstrates that the logits of the adversarial sample are different from both those of the source class and the target class, while relatively closer to the former (more different from the target). Combined with the victim model output (misclassified to a target class), the anomaly detector finds that essentially the adversarial example bears more similarity to another class (the source class) than the predicted one, presenting a conflicted result and therefore capturing the discrepancy.



(a) Embeddings in 3D **(b) PR curves for latent space**

Fig. 12: Embedding and PR-curve for different latent space

5.5 Overhead and Delay

We measure the overhead and delay of the EMShepherd detection framework. As the detector is outside of the victim model and the system, it will not affect the victim model execution at all. It detects an adversarial example within 169 milliseconds after the victim model finishes execution on the experimental platform. The delay is composed of the average processing time of EM traces (10 ms), EM classifiers inference (128 ms) and anomaly detector execution (31 ms). The processing time can be reduced by running the EM classifiers for different trace segments in parallel. It can be further reduced by running the detection along with the measurements in a pipelined fashion - starting processing a segment as soon as it is measured while the victim model is still executing the next segment.

5.6 Impact of the Detector Parameters

In this section, we evaluate the impact of different experimental settings on the performance of our adversarial detector. For instance, the trace sampling frequency and sliding window size of STFT will impact the classification accuracy of EM classifiers, and the structure of the anomaly detector (such as the latent space size) may affect the generalizability of VAE. Different attack methods or the distance measures used in attacks will also lead to different detection results.

EM Trace Sampling Frequency: The sampling frequency of EM traces has an effect on the information density for the classifiers. We adjust the oscillator's sampling frequency from 500MHz to 20GHz and report the EM classification accuracy of the first segment in Table 3. Note that according to the Nyquist-Shannon sampling theorem, the minimum sampling frequency should be larger than 300 MHz (two times of operating frequency). The results show that a sampling frequency above 2GHz is sufficient to achieve good classification accuracy.

Table 3: Classification accuracy versus sampling frequency

Frequency (GHz)	0.5	1	2	4	10	20
Accuracy	0.32	0.32	0.65	0.67	0.67	**0.68**

Sliding Window Size: We further compare the classifiers' accuracy with different STFT window sizes. When the Hanning window size changes, the number of bands with most signals also changes. Table 4 presents the results for different STFT configurations. As the window size changes from 64 to 1024, the classifiers' average accuracy goes up first and then goes down, with the maximum accuracy of 67% at the window size of 128 with the top 15 frequency bands (above the red dash line in Fig. 14 (d)) in the Appendix are kept for the follow-on classifier and anomaly detector).

Table 4: Classification accuracy vs. the sliding window size

Window size	64	128	**256**	512	1024
Band number	60	30	**15**	8	4
Accuracy	0.53	0.56	**0.67**	0.59	0.34

Latent Space Size: The latent space of VAE is a variable that may affect the performance of anomaly detection. We vary the size of the latent space between 2 and 9, which reflects the model's capability to express the input data features. The results are presented in Fig. 12. Overall the latent space size has no significant effect on anomaly detection. The space size of 6 is slightly better than others.
Attack Methods: Our anomaly detector can detect adversarial samples for a wide range of existing attacks, PGD, CW, and DeepFool, with different attack distance metrics. We tested five attack methods, $PGD(L_1)$, $PGD(L_2)$, $PGD(L_{inf})$, $CW(L_2)$, and $DeepFool(L_2)$, and the adversarial detector all has a reasonable detection performance, shown in Table 5.

Comparing the different distances used in attacks, we observe that using an EM-based detector has better performance for the L_1 attack, then L_{inf}, and the worst is L_2 for PGD attacks. For L_1 attack, only a few pixels are modified to make an adversarial example from the original source-class sample, as shown in Fig. 8 (b). Although the victim model is misled to predict it to be the target class, the EM trace of the inference bears more similarity to the original class sample, distinctly different from the EM traces of the target class samples. Therefore, it is easier to detect the adversarial. While for L_{inf} attack, many pixels are changed, randomly distributed, easily visualized in Fig. 8 (d). The EM trace will differ both from that of the source class and that of the target class, still caught easily by the adversarial detector. For L_2 attack, there are more pixels changed than the L_1 attack, but around the object in the image rather than randomly distributed like in L_{inf}, making the EM trace somewhat between those of the source class and target class and causing the anomaly detector low confidence in making the prediction.
Target Classes: Fig. 11(b) shows that the prediction F1-scores of the victim model on the three classes, 0, 4, and 6, are 0.84, 0.83, and 0.70, respectively, lower than other classes with scores above 0.9. Such inaccuracy affects the performance of our adversarial detectors. The samples from these classes, therefore, include similar computation along a large part of the victim model execution flow, making the detection hard from the EM emanations of the execution. Table 5 also presents the detector performance on various adversarial target classes (columns) by different attack methods (rows) using F1-scores. Particularly Class 0(T-shirt) and 6(Shirt) do not perform as well as other classes. For other classes, our detection framework can detect close to 94% of adversarial samples with less than a 10% false positive rate.

5.7 Comparison with Other Methods

Table 6 shows our comparison between the hardware-based EMShepherd with state-of-the-art software detection methods. Note that our detector is under a stricter 'black-box' scenario where only the EM traces of model execution along with the model prediction output are available. We control the False Positive (FP) rate under 10% and evaluate the detection rates under targeted PGD L_2 attacks and CW L_2 attacks on all the 10 classes. We draw three major conclusions.

- EMShepherd outperforms all baseline methods in the detection of PGD and CW attacks, with a 94% detection rate on average. This demonstrates that our EMShepherd successfully captures the different computations of the model inference for benign and adversarial samples.
- The detection performance varies in different target classes. PGD attacks on Class 0, 4, and 6 cannot be easily detected, due to the relatively lower EM classification accuracies for these three classes (See Table 2).
- Our detector performs consistently across the two different attacks, while the performance of other methods varies significantly for the two attacks. PGD attacks can be effectively detected by MagNet, which utilizes only testing inputs (semantic information) from the victim inputs. On the other hand, NIC, which focuses on the execution flow, has a better performance on CW adversarial samples. Our method is more general as it obtains both of such information from EM traces.

6 ADVERSARIAL DETECTION FOR ROBUST MODELS

The EMShepherd framework is model-agnostic and should also work for robust models enhanced with adversarial defense mechanisms, such as adversarial training. The robust model is only resilient to adversarial examples similar to the ones used in retraining, and may be circumvented by other unknown adversarial examples or maliciously-designed adaptive attacks (stronger adversarial examples). We evaluate the effectiveness of our detector on a robust model under a different adversarial attack. We train a robust LeNet-5 CNN model with benign samples and adversarial examples (with the correct labels) generated by the FGSM method. Such a robust model is weak against the CW attack, while EMShepherd succeeds in detecting the CW adversarial examples. We evaluate the detection performance on a testing dataset with benign, FGSM (regarded as noisy benign), and targeted CW examples, and the VAE loss distributions are presented in Fig. 13. Our findings are as follows:

- EMShepherd can detect the stronger adversarial samples (yellow bars in Fig. 13) and most of the benign examples correctly (blue and magenta bars). Using the threshold of 0.7, the DR of the targeted CW attack is 100% when the FPR on unattacked samples (benign and FGSM) is 2.6%.
- It is noticeable that the FPR of FGSM samples is 5.1%. Note that although EMShepherd for the robust model is trained with only benign samples, it still correctly classifies FGSM samples, consistent with the robust model.

Table 5: F1-scores of the VAE Detector

Attack	0	1	2	3	4	5	6	7	8	9
PGD(L_2)	0.820	0.999	0.957	0.957	0.899	0.999	0.758	0.999	0.968	0.999
PGD(L_{inf})	0.946	0.999	0.956	0.981	0.918	0.999	0.884	0.999	0.963	0.999
PGD(L_1)	0.948	0.999	0.957	0.982	0.924	0.999	0.910	0.999	0.968	0.999
CW(L_2)	0.797	0.999	0.957	0.982	0.924	0.999	0.756	0.999	0.958	0.999
DF(L_2)	0.918	0.999	0.811	0.963	0.804	0.999	0.756	0.999	0.968	0.999

Table 6: Detection Rate(%) when $FPR = 10\%$

Method	PGD L_2 Targeted Class										CW L_2 Targeted Class									
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
EM	74.9	**100**	**100**	**99.5**	67.4	**100**	68.0	**100**	**99.9**	**100**	72.4	**100**	**100**	**100**	**100**	**100**	65.6	**100**	**100**	**100**
KDE	57.5	53.8	51.9	42.4	50.6	57.3	50.7	56.3	52.8	65.2	49.1	59.8	48.0	45.9	54.5	69.6	44.7	68.8	60.7	74.6
NIC	55.0	55.8	52.8	43.0	41.4	57.2	50.0	83.0	47.0	56.9	**82.0**	81.7	77.3	86.5	79.6	81.0	**74.0**	82.5	87.8	85.6
FS	64.4	51.2	72.1	60.8	69.9	40.1	66.8	39.6	65.2	50.1	69.5	62.0	68.0	62.8	79.0	66.3	66.8	73.9	67.4	69.5
MagNet	**88.0**	87.2	81.2	88.2	**82.0**	84.5	**87.4**	87.2	88.1	85.9	67.2	62.3	63.2	62.7	65.3	80.1	65.2	74.3	68.4	68.8

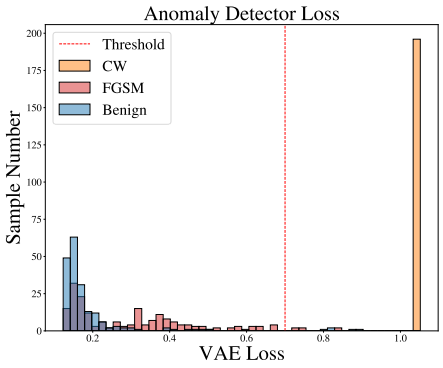


Fig. 13: Robust VAE Loss

7 CONCLUSION AND FUTURE WORK

In this work, we propose a novel adversarial detection framework, EMShepherd, leveraging the EM side-channel of model execution. EM traces embody rich input (class)-dependent inference information, well suited for classification and anomaly detection. Our framework extracts EM feature invariants for different classes and use them for unsupervised anomaly detection. The adversarial detector can be deployed as an air-gapped, third-party, PnP system in the proximity of the victim system in operation. It is totally passive and noninvasive without probing the model execution or retraining the model. The performance of our black-box adversarial detector is comparable to the state-of-the-art software-based white-box detection method, but has a much broader and more general application to diverse DNN implementations and applications.

Our future work will adapt the framework for the detection of more attacks, such as Trojan attacks, backdoor attacks, and data poisoning attacks. The EM side-channel leakage of deep learning engines during execution can be further leveraged for more applications, e.g., membership inference attacks where the input categories are reverse engineered.

REFERENCES

[1] 2019. F-Score Definition | DeepAI. https://deepai.org/machine-learning-glossary-and-terms/f-score. (Accessed on 01/23/2022).
[2] 2020. Running Average Power Limit Energy Reporting. https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html.
[3] 2020. Zynq DPU v3.2 IP Product Guide. https://www.xilinx.com/support/documentation/ip_documentation/dpu/v3_2/pg338-dpu.pdf. (Accessed on 01/23/2022).
[4] 2021. TensorFlow. https://www.tensorflow.org [Online; accessed 24. Jan. 2022].
[5] 2021. Welcome to Ultra96-PYNQ's documentation! — Ultra96-PYNQ v2.6 documentation. https://ultra96-pynq.readthedocs.io/en/latest [Online; accessed 23. Jan. 2022].
[6] 2022. Probe Set PBS 2 (incl. Preamplifier). https://aaronia-shop.com/products/probe-set-pbs-2-incl-preamplifier [Online; accessed 23. Jan. 2022].
[7] 2022. Ultra96-V2 | Avnet Boards. https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/ultra96-v2 [Online; accessed 23. Jan. 2022].
[8] 2022. Vitis AI. https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html [Online; accessed 24. Jan. 2022].
[9] Dakshi Agrawal, Bruce Archambeault, Josyula R Rao, and Pankaj Rohatgi. 2002. The EM side-channel (s). In *Int. Workshop on Cryptographic Hardware & Embedded Systems*. 29–45.
[10] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. 2002. The EM side-channels. In *Int. WkShp on Cryptographic Hardware & Embedded Systems*.
[11] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2019. {CSI}{NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *USENIX Security Symp.* 515–532.
[12] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. 2017. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv preprint arXiv:1704.02654* 2 (2017), 1.
[13] Shobhit Bhatnagar, Deepanway Ghosal, and Maheshkumar H Kolekar. 2017. Classification of fashion article images using convolutional neural networks. In *Int. Conf. on Image Information Processing (ICIIP)*. 1–6.
[14] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).
[15] Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence & Security*. 3–14.
[16] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symp. on Security & Privacy*. IEEE, 39–57.
[17] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. 2002. Template attacks. In *Int. Workshop on Cryptographic Hardware & Embedded Systems*. Springer, 13–28.
[18] Łukasz Chmielewski and Léo Weissbart. 2021. On reverse engineering neural network implementation on gpu. In *International Conference on Applied Cryptography and Network Security*. Springer, 96–113.

[19] Debayan Das, Anupam Golder, Josef Danial, Santosh Ghosh, Arijit Raychowdhury, and Shreyas Sen. 2019. X-DeepSCA: Cross-device deep learning side channel attack. In *Proc. Design Automation Conf.* 1–6.

[20] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research* 12, 7 (2011).

[21] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410* (2017).

[22] David Forsyth and Jean Ponce. 2011. *Computer vision: A modern approach.* Prentice hall.

[23] Kenneth R Foster, Robert Koprowski, and Joseph D Skufca. 2014. Machine learning, medical diagnosis, and biomedical engineering research-commentary. *Biomedical engineering online* 13, 1 (2014), 1–9.

[24] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *J. Machine Learning Research* 17, 1 (2016), 2096–2030.

[25] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[26] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280* (2017).

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs.CV]

[28] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *aXiv:1704.04861* (2017).

[29] Jin Huang and Charles X Ling. 2005. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering* 17, 3 (2005), 299–310.

[30] Peng-Tao Jiang, Chang-Bin Zhang, Qibin Hou, Ming-Ming Cheng, and Yunchao Wei. 2021. Layercam: Exploring hierarchical class activation maps for localization. *IEEE Transactions on Image Processing* 30 (2021), 5875–5888.

[31] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[32] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *Annual Int. Cryptology Conf.* Springer, 388–397.

[33] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. 2016. Adversarial examples in the physical world.

[34] Teledyne LeCroy. 2022. http://cdn.teledynelecroy.com/files/pdf/waverunner_6_zi_datasheet.pdf [Online; accessed 23. Jan. 2022].

[35] Xin Li and Fuxin Li. 2017. Adversarial examples detection in deep networks with convolutional filter statistics. In *Proc. IEEE Int. Conf. on Computer Vision*. 5764–5772.

[36] Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. 2021. PLATYPUS: Software-based Power Side-Channel Attacks on x86. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE.

[37] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. 2018. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 369–385.

[38] Shiqing Ma and Yingqi Liu. 2019. Nic: Detecting adversarial samples with neural network invariant checking. In *Proc. Network & Distributed System Security Symposium (NDSS 2019)*.

[39] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. 2018. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613* (2018).

[40] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[41] Christopher Manning and Hinrich Schutze. 1999. *Foundations of statistical natural language processing.* MIT press.

[42] Dongyu Meng and Hao Chen. 2017. Magnet: a two-pronged defense against adversarial examples. In *ACM SIGSAC Conf. on Computer & Communications Security*. 135–147.

[43] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proc. IEEE Conf. Computer Vision & Pattern Recognition*. 2574–2582.

[44] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. *Deep face recognition.* British Machine Vision Association.

[45] Jonas Rauber, Wieland Brendel, and Matthias Bethge. 2017. Foolbox: A Python toolbox to benchmark the robustness of machine learning models. In *Workshop on Reliable Machine Learning in the Wild.* http://arxiv.org/abs/1707.04131

[46] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. *arXiv preprint arXiv:1506.02640* (2016).

[47] Jonathan G Richens, Ciarn M Lee, and Saurabh Johri. 2020. Improving the accuracy of medical diagnosis with causal machine learning. *Nature communications* 11, 1 (2020), 1–9.

[48] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. 2016. Grad-CAM: Why did you say that? *arXiv preprint arXiv:1611.07450* (2016).

[49] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[50] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[51] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs.CV]

[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[53] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. Going Deeper with Convolutions. *arXiv preprint arXiv:1409.4842* (2014).

[54] Guanhong Tao, Shiqing Ma, Yingqi Liu, and Xiangyu Zhang. 2018. Attacks meet interpretability: Attribute-steered detection of adversarial samples. *arXiv preprint arXiv:1810.11580* (2018).

[55] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. 2020. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems* 33 (2020), 1633–1645.

[56] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[57] Kira Vinogradova, Alexandr Dibrov, and Gene Myers. 2020. Towards interpretable semantic segmentation via gradient-weighted class activation mapping (student abstract). In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 13943–13944.

[58] Siyue Wang, Xiao Wang, Pu Zhao, Wujie Wen, David Kaeli, Peter Chin, and Xue Lin. 2018. Defensive dropout for hardening deep neural networks under adversarial attacks. In *Proceedings of the International Conference on Computer-Aided Design*. 1–8.

[59] Xiao Wang, Siyue Wang, Pin-Yu Chen, Yanzhi Wang, Brian Kulis, Xue Lin, and Sang Peter Chin. 2019. Protecting Neural Networks with Hierarchical Random Switching: Towards Better Robustness-Accuracy Trade-off for Stochastic Defenses. In *IJCAI*.

[60] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE Conf. on Computer Communications*. IEEE, 2512–2520.

[61] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155* (2017).

[62] Yijun Yang, Ruiyuan Gao, Yu Li, Qiuxia Lai, and Qiang Xu. 2022. What You See is Not What the Network Infers: Detecting Adversarial Examples Based on Semantic Contradiction. *arXiv preprint arXiv:2201.09650* (2022).

[63] Honggang Yu, Haocheng Ma, Kaichen Yang, Yiqiang Zhao, and Yier Jin. 2020. DeepEM: Deep Neural Networks Model Recovery through EM Side-Channel Information Leakage. In *IEEE Int. Symp. on Hardware Oriented Security & Trust (HOST)*. IEEE, 209–218.

[64] Yicheng Zhang, Rozhin Yasaei, Hao Chen, Zhou Li, and Mohammad Abdullah Al Faruque. 2021. Stealing neural network structure through remote fpga side-channel analysis. *IEEE Trans. on Information Forensics & Security* 16 (2021), 4377–4388.

[65] Dawei Zhou, Nannan Wang, Chunlei Peng, Xinbo Gao, Xiaoyu Wang, Jun Yu, and Tongliang Liu. 2021. Removing adversarial noise in class activation feature space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7878–7887.

[66] Jiang Zhu, Lizan Wang, Haolin Liu, Shujuan Tian, Qingyong Deng, and Jianqi Li. 2020. An Efficient Task Assignment Framework to Accelerate DPU-Based Convolutional Neural Network Inference on FPGAs. *IEEE Access* 8 (2020), 83224–83237. https://doi.org/10.1109/ACCESS.2020.2988311

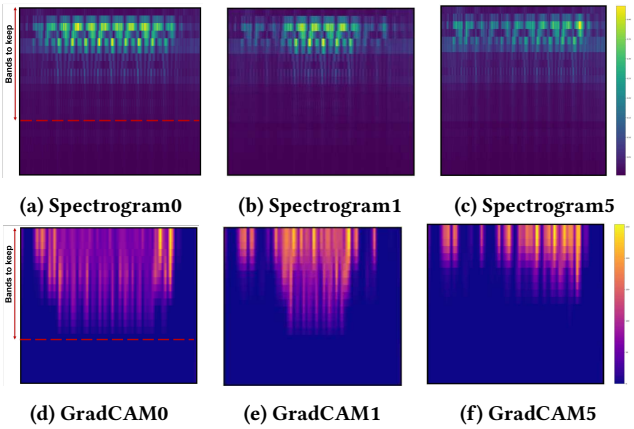


Fig. 14: The average spectrogram and GradCAM results of the first EM segment from Class 0, 1, 5

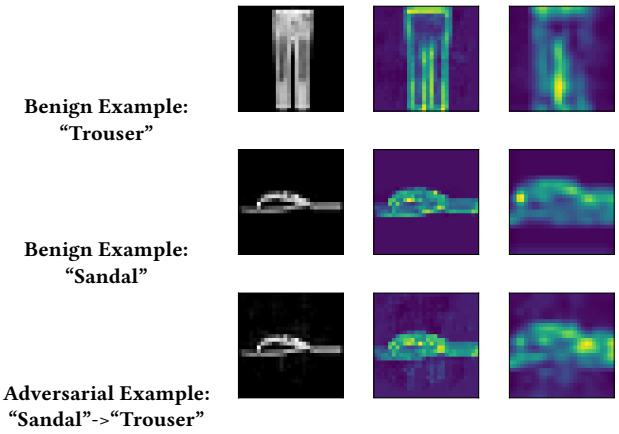


Fig. 15: GradCAM illustration of adversarial attacks. The malicious noise in adversarial sample (the third row) increases with the model depth, which finally causes it's misclassification to the target.

A EMSHEPHERD GRADCAM

The GradCAM uses the gradient information flowing into the last convolutional layer of the classifier to assign importance values to each neuron for a particular decision of interest. Fig. 14 (a)-(c) present the average spectrogram of the first EM segment for Class 0, 1, and 5, respectively, and Figure 14 (d)-(f) show the corresponding coarse GradCAM localization (red heatmaps). The GradCAM heatmaps illustrate the sensitivity (magnitude of gradient) of the EM classification on the input neurons, which are the pixels on the spectrogram here. The most sensitive (bright) parts are where the EM classifier focuses on when making a decision. For example, the primary signal of Class 1 is from the middle part of the spectrogram as shown in Fig. 14 (b), and correspondingly GradCAM in Fig. 14 (e) shows that our classifier also emphasizes the central part. The GradCAM heatmaps also guide the defender to improve the EM segments' pre-processing by selecting the significant frequency bands.

B CLASS ACTIVATION MAP IN ADVERSARIAL DETECTION

Class Activation Map (CAM) [30, 48] is commonly used to explain the behavior of deep neural networks, showing how the network progressively (with more layers) identifies the important region of the input (features) that leads to the class prediction. For benign samples, CAMs can represent the images' semantic information [57, 65]. However, the adversarial perturbations can impact the focus of neural networks, which leads to wrong predictions. For example, in Fig. 15, we present an example originally comes from Class "Sandal" and is classified as "Trouser" via CW attack, together with samples from source and target class, followed by their class activation maps of the first two convolutional layers, respectively. We conclude that:

- For benign samples, the class activation maps (semantic information) visually show features that lead to the classification result.
- The CAMs of adversarial samples do not resemble those of benign examples that represent the target class. And because of adversarial noise, their CAMs diverge from the source class to some anomalies gradually by the model depth.
- CAMs of different layers vary, and the impact of adversarial perturbations will be amplified by the network depth.

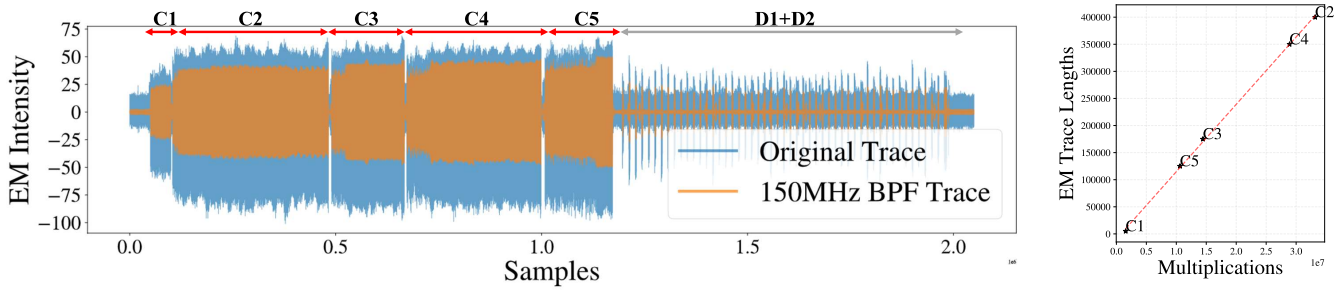
Therefore, one way for adversarial detection is to train an out-of-distribution detector to figure out the mismatch between CAMs from benign samples and ones from adversarial samples.

C ADVERSARIAL DETECTION FOR VGG MODEL

To show the scalability of the EMShepherd framework, we further apply it to a VGG-like model on CIFAR-10 dataset and evaluate the adversarial detection performance. The results show that our framework can cope with large victim model execution on more complex datasets.

EM Traces of VGG Model Execution on CIFAR-10: Compared with the grayscale Fashion MNIST, the CIFAR-10 dataset includes colored images used for objection detection. The victim model and our EM trace collector both have to change accordingly.

- **Larger victim model:** The size of CIFAR-10 images is $32 \times 32 \times 3$, requiring more sophisticated models. Due to the limited resources on DPU, we choose a VGG-like model for implementation. The model includes 7 layers: 5 consecutive convolutional layers followed by 2 dense layers, which achieves testing accuracy 90.5% on CIFAR-10 (93.6% by the benchmark VGG-16 [50]). It uses Tensorflow2 building of Ultra96 with a working frequency of 150MHz.
- **Lower EM sampling frequency:** Due to the increase of execution time, the length of CIFAR-10 EM traces are longer than the Fashion MNIST one. Fig. 16(a) shows an example CIFAR-10 EM trace under a sampling frequency 1 GHz. The blue part is the raw EM signal and the orange part stands for the signals after a bandpass filter at the DPU operating frequency.
- **Layer-wise separation:** As annotated on Fig. 16(a), the EM trace can be partitioned into 5 convolutional layer segments (C1-C5) and 2 dense layer segments (D1 and D2). We use C1-C5 for building the EM classifiers as they are for computations and easily



(a) An Example of CIFAR-10 EM traces **(b) Multiplications**

Fig. 16: CIFAR-10 EM trace and layer operations

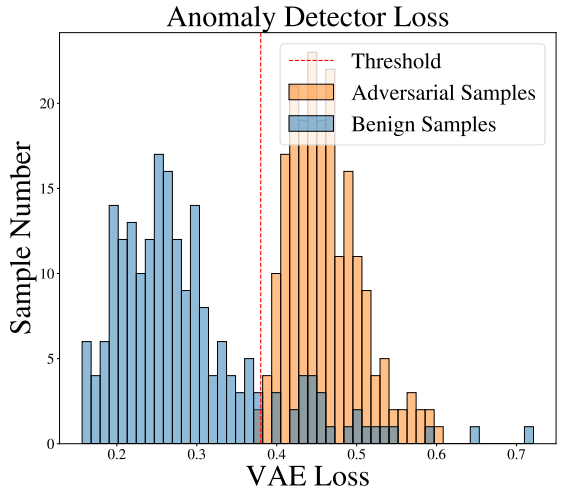


Fig. 17: CIFAR-10 VAE Loss

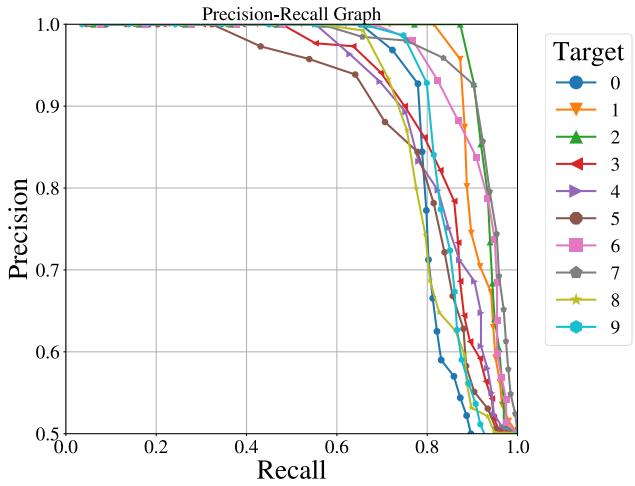


Fig. 18: CIFAR PRcurves

Table 7: CIFAR-10 EM Classifiers Performance

Convolutional layer	C1	C2	C3	C4	C5
Accuracy	0.42	0.46	0.43	0.49	0.27

distinguished by the data transmission segments between two computation ones. We find that the length of each EM segment (layer execution) is directly proportional to the number of multiplications in that layer, and the results are presented in Fig. 16(b), where C1-C5 are marked accordingly.

CIFAR-10 Layer-wise EM Classifiers: Table 7 shows the classification performance of layer-wise EM classifiers on CIFAR-10. It shows that around half of EM traces can be correctly classified. The reason for CIFAR-10 EM classifiers with lower prediction accuracies than Fashion MNIST ones is because of inherent characteristics of the datasets. CIFAR-10 image is 3-channel RGB while Fashion MNIST image is 1-channel Grayscale. The VGG model used for CIFAR-10 is deeper and larger than LeNet-5. The resolution of the EM traces is much lower due to the lower sampling frequency and limited storage/processing capabilities. Comparing the different layer segments, we find that C_5, the last convolutional layer has the lowest prediction accuracy. Because C_5 has a larger receptive field and a large number of kernels running in parallel, many neurons and activation are concurrent and time points on the EM traces bear low signal-to-noise ratios. Note although the classifiers achieve lower accuracy than the previous Fashion MNIST cases, these classifiers are sufficient for the follow-on anomaly detectors to catch adversarial examples, as we have analyzed it is the deviation of classifiers' logits that is the characteristics of adversarial examples, i.e., a relative value instead of absolute accuracy.

CIFAR-10 Anomaly Detector: We evaluate the performance of our anomaly detector on CIFAR-10 EM classifiers. The experimental results show that our detection framework still achieves fairly good performance on the colored CIFAR-10 dataset. The logits from all 5 segments (convolutional layers) are utilized as inputs for the VAE anomaly detector, against targeted PGD attacks on CIFAR-10. Fig. 17 shows the VAE loss of the vectors of logits for both benign samples and adversarial examples. With an optimal threshold selected, the detection accuracy for the adversarial examples is close to 100% with some false positives on the benign examples. Fig. 18 shows the precision-recall curves for different classes: the best detection result is from target class 4 (deer) with the F1-score of 0.906 and the worst one is class 7 (horse) with the F1-score of 0.821.