Journal of Hardware and Systems Security manuscript No.
(will be inserted by the editor)

Deep learning classification of motherboard components by leveraging EM side-channel signals

Erik J. Jorgensen · Frank T. Werner · Milos Prvulovic · Alenka Zajić

Received: date / Accepted: date

Abstract We show that a broad range of motherboard components can be classified based on their electromagnetic (EM) emanations using a lightweight convolutional neural network (CNN) architecture. Our random bootstrap sampling approach to cross-validation allows us to select robust models that achieve strong accuracy when classifying processor, memory, power, and ethernet integrated circuits (ICs) on internet of things (IoT) devices in two different excitation states. Our method also performs equal to or better than a k-Nearest Neighbors (k-NN) classifier baseline in all tested cases. We develop an anomaly detection procedure to flag the types of IC models not seen by the CNN in training. An analysis of signals passing through our end-to-end trained model allows us to generate insights about the discriminative features in the emanations of different components. We analyze how the convolution layers' outputs pass through linear transformation layers to understand which of the input features are most important. Our results demonstrate that a CNN architecture can be used to classify ICs accurately using a diverse set of EM emanations while providing insights about the relevant discriminative features between components.

Keywords EM side-channel · Component identification · Deep learning · Convolutional neural networks

1 Introduction

As companies opt to outsource electronic device manufacturing, the authenticity of the integrated circuits contained in those devices becomes of increasing concern. In applications of critical importance, like the monitoring or control of power distribution at utilities, spying or a potential service disruption through the use of counterfeit electrical components presents a critical security risk. The production pipeline of any electronic device involves multiple manufacturers and suppliers, each of which could introduce counterfeit components to a device. Counterfeits may even be placed without malicious intent, simply to reduce manufacturing costs with ignorance of functional differences or bugs present. It is estimated that counterfeit ICs represent about 1% of semiconductor sales [16], resulting in the loss of around $100 billion in sales [31]. The unknowing use of counterfeit components can hamper reliability, lead to inter-operability issues between software and hardware, and degrade manufacturer reputations. Consequently, it is essential to verify the authenticity of the individual components making up these devices.

To detect counterfeit ICs, it is common to analyze parts through physical or electrical inspection techniques [9]. Physical inspection can prove difficult, since counterfeit ICs are often superficially identical to the original. It can even be difficult to determine the authenticity of ICs by X-ray scanning [14]. Electrical inspection may involve testing components with burn-in tests, though code still may execute normally on counterfeit chips and make it difficult to find security flaws without more exhaustive and time-consuming testing [9]. For these reasons, it is essential to have a simple, time-efficient method for robustly identifying an electrical component to determine its authenticity.

It is well known that electronic components leak information through several different "side channels." When code executes on a device, these side channels could be variations in acoustic noise [5], temperature [15], power consumption [18], electromagnetic (EM) emanation [11], or any other information leak-

Erik J. Jorgensen · Frank T. Werner · Milos Prvulovic · Alenka Zajić
Georgia Institute of Technology

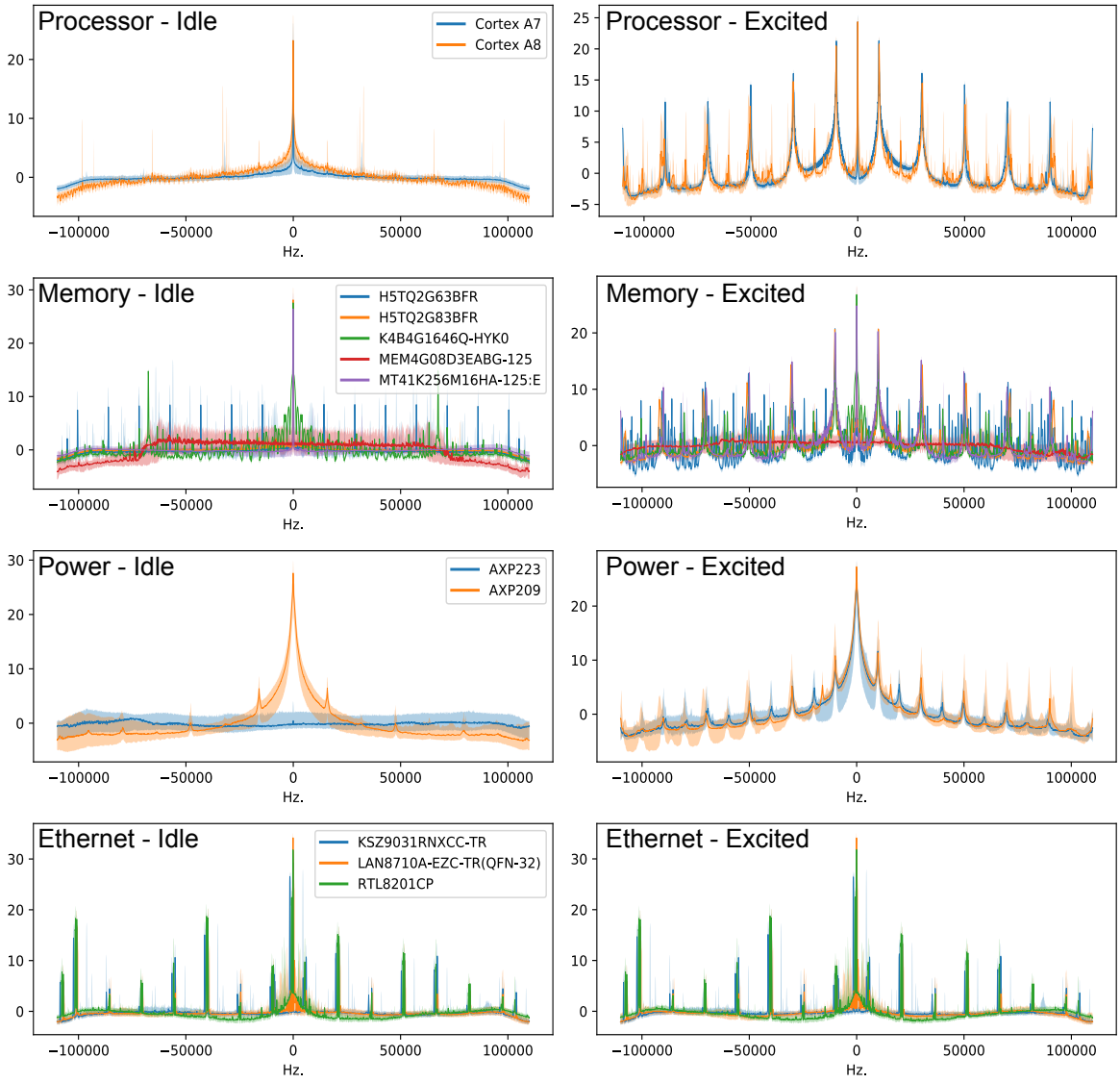

Fig. 1 EM emanation distribution of each component class and excitation state, horizontally centered by the component carrier frequency and with vertical axis converted to dB and mean subtracted. The median emanation measurement across all components of the same model is drawn with shading below and above corresponding to the 10% and 90% quantiles respectively. Components are measured in the idle (left) and excited (right) states. Both ethernet excitation states look identical but do differ, however imperceptibly.

age through empty space [6]. Any conductive material on a device acts as an antenna and can emanate significant information about device activity [2], [44]. With an external antenna, measuring these side channel emanations requires zero run-time overhead on the device. EM side-channels have traditionally been cause for security concern; being used to steal confidential information [2], [11], [4], [19]. At the same time, EM side-channels can be used to identifying devices like automobiles, desktops, phones, toys, and microcontrollers [10], [13], [37], [42], [20], [3]. Convolutional Neural Networks (CNNs) have even been combined with EM side-channels to classify cellphones and their sensor status [43]. While these EM side-channel techniques are successful in classifying whole devices, they have not been shown to identify individual ICs or components within those devices.

There is no established state-of-the-art solution to the problem of utilizing EM side channels to accurately identify individual components on an assembled motherboard. While there has been work on classifying single components or entire motherboards based on their EM signatures [7], [3], to our knowledge, there has been no work on classifying components already integrated onto a motherboard. This classification is important for device designers, who need ways of validating that the

components on their devices have not been replaced by a third-party assembler or supplier with counterfeits, such as a cheap alternative or inauthentic copy.

Here, we train a single Convolutional Neural Network (CNN) architecture to solve several component identification tasks using EM emanations. We show that this approach can distinguish individual IC models within four general classes (processors, memory ICs, power management ICs, and ethernet transceivers). We choose these four component types due to the security threat they pose if counterfeited given their handling of potentially secure data. Additionally, the relative expense of these four compared to other component types like I/O connectors, capacitors, op-amps, etc. makes them more likely targets of counterfeiting. To show the robustness of our methods we also test our method's capabilities while the IoT motherboards these ICs reside on are operating in "Idle" and "Excited" conditions. This idle state is the most realistic scenario for component validation on an already-assembled device. Then we test on an excited state to demonstrate the performance of our method in a similar setting as [40]. We train our network with a random bootstrap sampling method of cross-validation to achieve more consistently accurate performance over that of a k-Nearest Neighbors baseline. We compare our method against a common off-the-shelf classifier (k-NN) since the problem has no established state-of-the-art solution. It is important to emphasize that we test classification accuracy on held-out devices separate from a training set, rather than subsampling measurements from the same device for training and testing. Additionally, we implement an anomaly detection procedure to show our design can be used to detect components on which the network was never trained. Finally, by examining the signals passing through the neural network, we build hypotheses about the features of different classes of ICs that make them most distinguishable. In doing so, we present a generic approach for accurate and interpretable component-level classification and counterfeit detection on IoT devices.

The rest of this paper is organized as follows. Section 2 details the assumptions we make about counterfeit components. Section 3 presents the frequency domain structure of the two excitation conditions tested. Section 4 details the CNN architecture developed. Section 5 details the experiments performed to test the model's classification accuracy. Section 6 presents results and interpretations. Finally, we end with concluding thoughts.

2 Threat model

In 2010, the U.S. Immigration and Customs Enforcement convicted an individual for trafficking hundreds of thousands of counterfeit ICs. More than $15 million worth of counterfeit ICs were trafficked to be sold as "military grade" components to the U.S. Navy [41]. Many of the counterfeits had their original markings removed before being re-marked with a different brand, date code, or country of origin. Distributors unknowingly selling these parts found that at least 1500 of their flash memory chips eventually failed [38]. This prosecution culminated as the first federal sentencing for trafficking counterfeit ICs and was a, "warning to people who ... put health, safety, and our national security at risk" according to the U.S. Attorney Ronaldy C. Machen Jr.

Here we define an IC counterfeit as any semiconductor component that is misrepresented as an authentic model of the same type. Counterfeits could be reverse-engineered replicas, discontinued models, or out-of-spec components swapped into an embedded system with malicious intent, for cost-savings, or due to simple negligence.

We propose a classification system for components on IoT motherboards by leveraging their EM side-channel emissions. Our method is intended as a solution strategy for device designers to validate the authenticity of major components on their device after they are assembled on the motherboard. Upon receiving a fabricated device, the designer can validate the authenticity of components by measuring their EM side-channel emanations and feeding the signal through the model presented in this work. We make the following assumptions in this threat model:

1. Counterfeit components in our intended scenario have different underlying architecture. These architecture differences result in variations of the EM side-channel emanations compared to the authentic component. Even visually identical components with the same footprint and pinout may have transistor-level architecture differences that could result in security vulnerabilities or affect performance characteristics.
2. Each third-party supplier or assembler in the device manufacturing pipeline introduces a potential vulnerability to counterfeit components being swapped in for their authentic counterparts. Components must be tested after the device's complete assembly to be sure that no counterfeits were introduced anywhere in the manufacturing pipeline.
3. The device's program activity is in the same state as the devices on which the model is trained.

```
while(true){
  // Execute activity X
  for(i=0; i<x_count; i++){
    ptr1 = (ptr1 & ~mask1) | ((ptr1 + offset) & mask1);
    // X instruction, e.g. a load operation
    value = *ptr1;
  }
  // Execute activity Y
  for(i=0; i<y_count; i++){
    ptr2 = (ptr2 & ~mask2) | ((ptr2 + offset) & mask2);
    // Y instruction, e.g. a store operation
    *ptr2 = value
  }
}
```

Fig. 2 Pseudo-code generating alternating X/Y excitation [39].

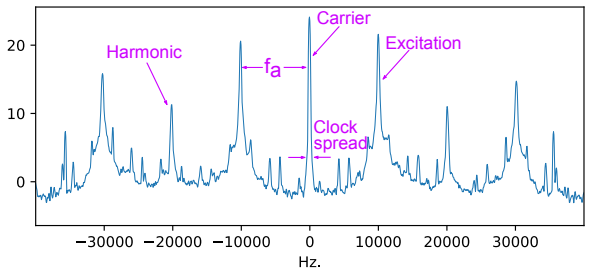

Fig. 3 Excited EM emanation frequency domain structure in decibel scale, set to zero-mean. The carrier frequency is down-converted to zero and shows a modest amount of spread due to clock frequency variation. Harmonics are present on both sides of the carrier, separated by the alternating frequency f_a of the executed program. Odd harmonics are stronger than even harmonics since the program execution acts similarly to a square wave in the time domain.

Our method validates the authenticity of hardware through side-channel emanations that change depending on software activity. Component validation is most simple and reliable for the device designer to perform when the device is powered on in an idle state without loaded program activity. We also test our method's performance on a device with controlled program activity to compare with prior work [40]. However, the validation that we present in this work is not possible when devices are running unknown programs.

4. The device designer has knowledge of at least one component in addition to the authentic component that could be a potential counterfeit swapped in for the original. IC fabricators regularly iterate on their designs, so it is reasonable to assume knowledge of older IC models that pose a counterfeit risk due to their lower cost but similar design. The neural network model we present here extracts discriminating features from the EM side-channel emanations of a known set of components and devices. The feature embeddings learned by the proposed CNN are used to differentiate between known component models that could be swapped in as counterfeits. Deviation from the CNN's activations for known component models is used as an indicator to detect never-before-seen counterfeits through anomaly detection.

3 Structured signals

When a component on a digital device is supplied power, EM emanations from the component follow consistent patterns based on a host of factors. Among any number of noisy variations in the signal, emanations change as a result of the component's signal activation. With repeated excitation, such as a device clock signal or repeating code pattern, we expect to see emanations that are especially noticeable in the frequency domain. In this work, we measure emanations when the device is in an idle state and also when it is activated by a controlled, repeating code execution.

3.1 Component excitations

When powering a device and simply remaining in an idle state, components exhibit minimal structure in their EM emanations besides a peak at the device clock frequency, as seen in the top-left panel of Figure 1. However, with controlled code execution, components can exhibit structured excitations. Here we execute a repeating code pattern to generate amplitude modulation with the clock frequency [6]. By repeating a pattern of two alternating code sequences (activity X and Y in Figure 2), we can control the modulating frequency and duty cycle of amplitude modulation on the device clock frequency. The program execution time spent in one loop versus the other controls duty cycle, while the execution time of the pattern T_a leads to an alternating rate $f_a = \frac{1}{T_a}$. That alternating rate shows up as amplitude modulated peaks (excitation) and harmonics centered at the carrier (clock) frequency, f_c. The result is a frequency domain pattern of spikes at frequencies $f_c \pm f_a$ and their harmonics $f_c \pm 2f_a, f_c \pm 3f_a, \ldots$ as detailed in Figure 3.

3.2 Features for classification

As seen in each row of Figure 1, the distinguishing features of emanations differ greatly between the idle and excited states. In the idle state, a hand-crafted classifier might extract features related to the clock frequency spread or the noise floor trail-off at the edges of the

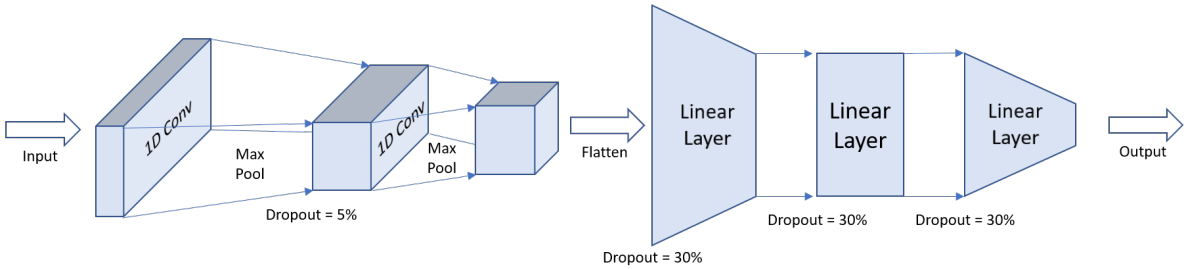

Fig. 4 High-level schematic of the CNN architecture depicting the convolution blocks (left) and fully-connected blocks (right).

measurement bandwidth. Distinguishing different component models in the excited state, the same features may be used in addition to the alternating frequency, relative excitation and harmonic magnitudes, and modulations on other unknown sources.

4 Preprocessing and model architecture

We develop a preprocessing pipeline that is implemented uniformly across eight datasets (four component types with two excitation patterns each) to decrease biases and analyze differences between different datasets. Here we detail the main preprocessing procedure and our CNN model architecture for classification.

4.1 Preprocessing

To reduce noise and generate several samples per measurement, we employ the procedure outlined in Algorithm 1. EM side-channel signals exhibit significant phase noise, resulting in jittery measurements and a visible spread in the frequencies of the carrier and excitation signals [34]. We designed the preprocessing Algorithm 1 to amplify the prominence of frequency spikes and carrier shape above the noise floor, mainly by averaging frequency-domain subsamples captured over time and converting them to decibel scale. This feature selection allows us to generate robust input features to our model from the set of experimental devices available to us. Each EM emanation measurement can contain millions of samples due to the large bandwidth needed to capture the alternating frequency and harmonics of excited components. Consequently, we subsample each measurement to increase the number of training samples available, reduce the dimensionality of the inputs, and capture potential random time-variability of the signal. From each measurement x with total length N, we subsample non-overlapping windows $z^{(i)}$ of length L. This results in a total of $M = floor(\frac{N}{L})$ segments each corresponding to $T_\mathrm{S} = \frac{L}{f_\mathrm{s}}$ seconds of the measurement, where f_s is the sampling frequency of the measurement device. The resulting measurement matrix $Z \in \mathcal{R}^{M\times L}$ holds each subsample as a row. To reduce the effect of discontinuities at each end of subsamples, a Hamming window $\mathbf{w}_\mathrm{h}$ of length L is applied to each of those subsamples. When measuring EM emanations while executing repeating code patterns, we expect signal structure to be revealed best in the frequency domain, so each subsample is converted to the frequency domain with a Short-Time Fourier Transform (STFT). We transform each complex subsample to its frequency magnitude and ignore phase information. We then perform a small circular shift of the frequency magnitude to center its maximum value which we assume to be the device's clock frequency. Next we convert the signal to decibels to increase the influence of weak signal components like the excitations and their harmonics. Non-overlapping groups of length N_G of the M subsamples are averaged to reduce random noise power, resulting in $M_G = ceiling(\frac{M}{N_G})$ subsamples per measurement arranged into a matrix $G \in \mathcal{R}^{M_G\times L}$. Finally, we scale each subsample to have a mean of zero and variance of one to ensure that the model is fed samples in a consistent range.

Algorithm 1 Measurement preprocessing algorithm

Inputs: Measurement $\mathbf{x} \in \mathcal{R}^N$
Allocate: Subsamples $Z \in \mathcal{R}^{M\times L}$
Processed samples $G \in \mathcal{R}^{M_G\times L}$
for $i = 1$ **to** M **do**
  $Z_{i,*} \leftarrow [\mathbf{x}_{(i\cdot L)}, \ldots, \mathbf{x}_{((i+1)\cdot L)}]$
  $Z_{i,*} \leftarrow Z_{i,*} \odot \mathbf{w}_\mathrm{h}$
  $Z_{i,*} \leftarrow STFT(Z_{i,*})$
  $Z_{i,*} \leftarrow \sqrt{\mathrm{Re}(Z_{i,*})^2 + \mathrm{Im}(Z_{i,*})^2}$
  $Z_{i,*} \leftarrow circshift(Z_{i,*})$
  $Z_{i,*} \leftarrow 20\log_{10}(Z_{i,*})$
end for
for $j = 1$ **to** M_G **do**
  $G_{j,*} \leftarrow \frac{1}{N_G}\sum_{n=1}^{N_G} Z_{((j-1)\cdot M_G+n),*}$
  $G_{j,*} \leftarrow G_{j,*} - mean(G_{j,*})$
  $G_{j,*} \leftarrow \frac{G_{j,*}}{std(G_{j,*})}$
end for

4.2 CNN Architecture

While other machine learning models may prove effective at distinguishing EM emanations, CNNs and their variety of trainable and deterministic layers can learn features on-the-fly and allow nearly unlimited model flexibility, all while retaining a level of interpretability. The lightweight convolutional neural network architecture we employ here is developed to strike a balance of fast training and inference times while maintaining interpretability and performance on a varied set of component classes and excitations. The architecture is made up of two convolution blocks and three fully-connected blocks as depicted in Figure 4 and described next.

Each convolution block uses a convolution with kernel size $K = 3$, stride length $S = 3$, and padding $P = 1$. The output of each convolution is fed to a ReLU activation [12] and then a max pooling layer with both kernel size and stride length of four. The result of the strided convolution and pooling is a downsampling of twelve times in the length of samples fed through the layer. The first convolution block takes as input a univariate subsample of length 4096 and learns three single-channel filters. This multivariate output is passed to the second convolution block with the same basic architecture. This block differs with the addition of a dropout layer [35] randomly zeroing out five percent of its filter weights, and by learning five three-channel filters. Between strided convolutions, pooling, and increasing numbers of filters, the convolution layers perform an effective downsampling of the input feature space of approximately 144-to-5 while only requiring $p_{conv} = 62$ parameters to be learned. The five-channel output of the convolution layers is flattened to a single vector of length 140 to be fed to the linear layers.

The first two fully-connected blocks follow the same structure as each other, differing only in the number of input features per layer. Both blocks learn linear transformations of their input with $M = 50$ neurons, before a dropout layer of 30% and ReLU activation layer. The final linear layer has O neurons, where O is the number of output classes for the specific component classification problem (given in Table 1). The linear layers require much more effort to train than the convolutions; learning $p_{\text{linear}} = 9600 + 51 \cdot O$ parameters. The entire network thus learns $p = p_{\text{conv}} + p_{\text{linear}} = 9662 + 51 \cdot O$ parameters to classify inputs between O classes.

5 Experimental design

To measure the ability of a CNN to classify different components on an IoT device, we test an array of different component classes all cross-validated with the same CNN architecture. In this section, we detail the hierarchy of different components tested, the measurements capturing component EM emanations, and the training and cross-validation parameters used to obtain the results in Section 6.

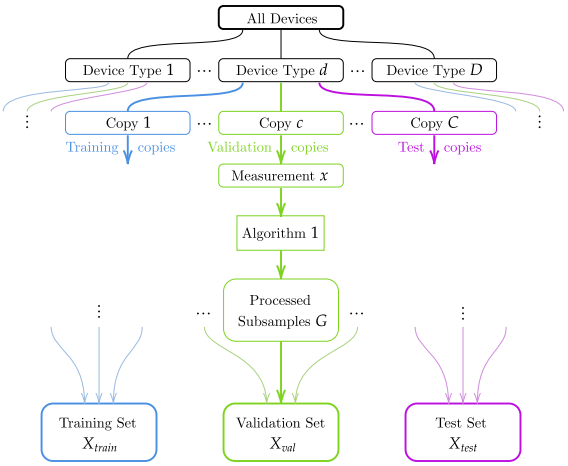

Fig. 5 Data breakdown and preprocessing pipeline. Device copies are broken into training (blue), validation (green), and test (magenta) groups for preprocessing and finally aggregated together with all other copies of their respective sets.

5.1 Device hierarchy

Here we use a large set of open source hardware IoT devices manufactured by Olimex to test a variety of component classification problems. This set of devices spans several copies of each of the $D = 7$ device types: A10-OLinuXino-LIME, A13-OLinuXino, A13-OLinuXino-MICRO, A20-OLinuXino-LIME, A20-OLinuXino-LIME2, A20-OLinuXino-MICRO, and A33-OLinuXino. We classify the component models within each of four classes of components: processors, memory chips, power management ICs, and ethernet transceivers. Though there are seven device types, some of them share the same components, and not all device types have components from each class. The component models on each device type are listed in Table 1. For each of the four component classes we measure and classify between $O = 2$ and 5 component models.

5.2 Cross-validation

An essential piece of developing any machine learning model is a rigorous separation of training, validation, and test samples. As depicted in Figure 5, we set aside

Device Type	Processor	Memory	Power	Ethernet
A10-OLinuXino-LIME [23]	Cortex A8	K4B4G1646Q-HYK0	AXP209	RTL8201CP
A13-OLinuXino [24]	Cortex A8	H5TQ2G83BFR	AXP209	—
A13-OLinuXino-MICRO [25]	Cortex A8	H5TQ2G63BFR	—	—
A20-OLinuXino-LIME [27]	Cortex A7	MT41K256M16HA-125:E	AXP209	LAN8710A-EZC-TR
A20-OLinuXino-LIME2 [28]	Cortex A7	MT41K256M16HA-125:E	AXP209	KSZ9031RNXCC-TR
A20-OLinuXino-MICRO [29]	Cortex A7	MT41K256M16HA-125:E	AXP209	LAN8710A-EZC-TR
A33-OLinuXino [26]	Cortex A7	MEM4G08D3EABG-125	AXP223	—
Number of Classes (O)	2	5	2	3

Table 1 Component models and counts across different device types and component classes. Blank boxes indicate that the device does not have that component.

a group of devices called the training/validation set that are used to select models with the best ability to generalize to unseen data. The remaining devices, called the test set, are used for evaluating performance against a baseline classifier. In our problem scenario, we have several copies each of a variety of different types of IoT devices available for measuring. Although the copies each have the same components, measuring multiple copies allows us to train a model that is more robust to manufacturing variabilities. To make sure each type of IoT device is represented evenly across the training/validation and test sets, we partition the copies of each device type into groups of $C_{\text{train/val}}$, and C_{test} devices. With D different types of devices, assuming each with C copies, this yields training/validation and test sets of size $D \cdot C_{\text{train/val}}$, and $D \cdot C_{\text{test}}$ respectively.

With just the training/validation set, we use a random bootstrap sampling method to settle on the best-generalized model and the training devices that yielded it. The following procedure is repeated R times to cross-validate the model training. For each device type, we randomly separate the $C_{\text{train/val}}$ device copies into C_{train} training and C_{val} validation devices. Some devices have multiple of the same components (specifically, multiple of the same memory chips), which we group together on each device to maintain consistent set split sizes. The measurements of each device's components are preprocessed into subsamples as detailed in Algorithm 1. Then, the $D \cdot C_{\text{train}} \cdot M_G$ subsampled measurement matrices across all training devices are aggregated together into the matrix $X_{\text{train}} \in \mathcal{R}^{(D \cdot C_{\text{train}} \cdot M_G) \times L}$. We then train the supervised model on the data in X_{train} with its corresponding component labels $\mathbf{y}_{\text{train}} \in \mathcal{R}^{(D \cdot C_{\text{train}} \cdot M_G)}$, and evaluate the model performance on a similarly aggregated validation set $X_{\text{val}} \in \mathcal{R}^{(D \cdot C_{\text{val}} \cdot M_G) \times L}$ and its labels. For the purpose of comparing model performance, we use accuracy defined as the proportion of correctly classified samples (True Positives + True Negatives) out of all samples. The model that achieves the best validation accuracy across the R different random training/validation combinations is saved and evaluated on the test set to measure its overall performance.

The value of R can be chosen to strike a balance between computation time and finding the model's best possible validation performance. Given $D \cdot C_{\text{train/val}}$ devices partitioned in this manner, the number of possible combinations of training and validation devices is given in Equation 1.

$$\text{combinations} = \binom{C_{\text{train/val}}}{C_{\text{train}}}^{D} \tag{1}$$

5.3 Measurements

To capture the EM emanations from each of the components, we measure with the probe setup depicted in Figure 6. A handmade circular coil probe with 1 mm radius [1] is held in place on top of the component with the motorized EM Probe Station from Riscure [33]. The probe connects to the Keysight M9391A PXIe Vector Signal Analyzer to record the signal in an In-Phase/Quadrature representation [36]. We isolate signals coming from the processor, memory, and other components by taking advantage of the fact that the different classes of components have different clock frequencies. Knowing the general clock frequency of the component type allows us to isolate them from others by measuring the carrier and any signals modulated onto them in the expected frequency range.

Number of Measurements	Processor	Memory	Power	Ethernet
Training Set	7	11	6	4
Validation Set	21	33	18	12
Test Set	40	63	35	23
Total	68	107	59	39

Table 2 Number of component measurements across all device types in the training, validation, and test sets for each component class. These constitute an approximate 10%/30%/60% split for the training/validation/testing sets with a total of 273 distinct components.

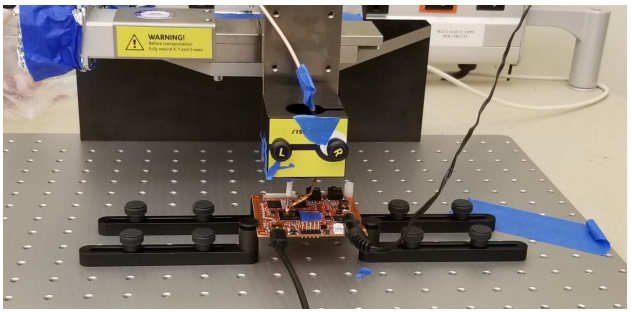

Fig. 6 Measurement setup for capturing EM emanations off of IoT boards.

Each component is measured separately while the IoT device is in an idle or excited state as detailed in Section 3.1. The excited state consists of a code alternating between addition and load operations at 10 kHz and 50% duty cycle. Each measurement is captured for approximately ten seconds at a sampling rate of 281,600 samples per second for a total measurement length of 2,816,027 samples. Measurements are band limited to a 220 kHz range surrounding the device's carrier frequency as determined by [32] or manually. Each measurement is preprocessed according to Algorithm 1 using subsequences of length $L = 4096$ and groupings of $N_G = 50$ subsamples. This results in each ten-second measurement being converted into fourteen subsamples that each represent up to 0.73 seconds of the measurement.

5.4 Anomaly Detection

We test the ability of our model to flag counterfeit IC models on which the network was not trained through an anomaly detection procedure. We perform these tests on memory components since we have the greatest diversity of memory components across device types and training our CNN requires at least two classes to build class-discriminating features. For each of our five tests, we train the CNN with one memory component model held out of the training and validation sets. This component serves as the "unknown" component. The CNN is then trained with only four output neurons now representing the "known" four memory component models. After using the same cross-validation setup as presented in Section 5.2, we reintroduce the unknown component model to the testing set along with examples of the four known models. We pass the side channel emanations of the test set consisting of emanations from all five models through our network and use the output activations as the input to an anomaly detection procedure similar to those presented in [21] and [17]. We then perform the same test holding out each of the remaining four memory components in turn from training.

Intuitively, a known component's emanations will have strong correlation with one of the trained components and consequently produce a strong activation at its corresponding output neuron. A component on which the model was not trained will produce weaker activations across the output neurons due to its low correlation with the emanations of the trained components. We measure the strength of the maximum activation relative to the other output neurons as a metric for the CNN's classification confidence. We expect this difference to be large for known components that the CNN classifies confidently, and small for unknown components that have EM emanations unlike any of the known components on which the CNN was trained.

We train a simple generative classifier for anomaly detection with the statistics $(\mu_{conf}, \sigma_{conf})$ of the CNN's classification confidence for the training samples. We define the classification confidence for a particular sample as the average difference between the maximally activated output neuron and the remaining output neuron activations. Test samples with classification confidence below $\mu_{conf} - \sigma_{conf} F^{-1}(p)$ are classified as unknown, where $F^{-1}(p)$ is the quantile function of a standard gaussian distribution and $p \in [0, 1]$ is the chosen quantile. The choice of p can be made as a trade-off between the rate of components being flagged as potential counterfeits versus being classified as the most similar

component model on which the CNN was trained. We choose $p = 0.99$ in our experiments.

5.5 Training parameters

We train the CNN in PyTorch [30] using the adapted ADAM optimizer with adjusted weight decay from [22] with default learning rate and weight decay parameters. Weights are randomly initialized by default in PyTorch from a uniform distribution scaled based on the layer type and size. We train with random batches of 50 subsamples without replacement and a maximum of 500 epochs. We implement an early-stopping procedure to reduce training time for iterations that don't converge or show poor generalization performance. To do so, we compute the validation loss every 10 batches and halt training if the loss does not improve after 500 batches. This training procedure yields training times for a single component, excitation type, and cross-validation fold of between 3 and 20 seconds, or more than 18 hours of total training and cross-validation time on our machine with an Intel i7-9800X CPU with 64GB memory and NVIDIA Quadro P400 GPU with 2GB memory.

For each classification problem, we perform the random bootstrapping method outlined in Section 5.2 with $R = 50$ training sequences and random training and validation sets for each device type. We have nine or ten copies of each device type, resulting in a total of 273 distinct components measured across the processor, memory, power, and ethernet models with which we employ a 10%/30%/60% training/validation/testing set split. We use groups of $C_{\text{train/val}} = 4$ device copies for each type. We train on $C_{\text{train}} = 1$ randomly selected device then validate the model's performance on the remaining $C_{\text{val}} = 3$ devices. The restricted set of training devices and much larger set of testing devices allows us to demonstrate the robustness of our method, while noting that it could be improved with larger sets of devices. From Equation 1, we compute that there are between 256 and 16,384 possible sampling combinations depending on the number of device types with that component class. With this many possible combinations, we achieve significant computation savings in the training process by choosing the best validated model across only $R = 50$ random combinations. Then, for each of the eight component classification problems (four component classes with two types of excitation each) we repeat the entire classification pipeline ten times to determine how consistently the CNN performs with so many random factors.

Finally, we train a k-NN classifier with the same preprocessing and cross-validation pipeline to use as a baseline for accuracy comparison. To select the best value of k, for each of the R random bootstrap samples, we train and choose the best performing model across $k = \{1, \ldots, 10\}$.

	Idle	Excited
Processor	0.000	0.025
Memory	0.175	0.000
Power	0.086	0.029
Ethernet	0.043	0.087

Table 3 Absolute difference of median CNN prediction accuracy over k-NN accuracy across all component classes and excitation states. Positive values mean the CNN outperformed the k-NN baseline.

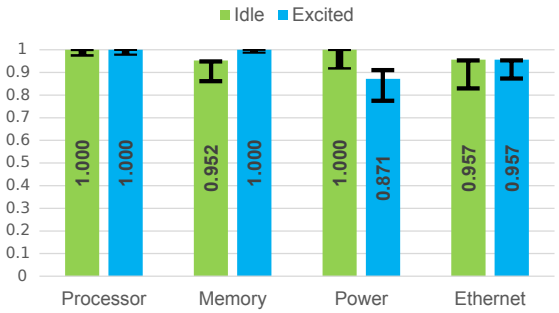

Fig. 7 Classification accuracy distributions across all four component classes in two excitation states each. Bar heights and inlaid number represent the median prediction accuracy across 10 trials, with bottom and top black markings representing the minimum and maximum accuracies respectively.

6 Results and Interpretability

We present the results of our classification experiments based on ten trials each of four component classes in two excitation modes, for a total of eighty classification trials or 4000 rounds of training for cross-validation.

6.1 Classification results

We compute the prediction accuracy on the five or six test set copies across all device types. As stated in Section 4.1, each measurement is preprocessed into a set of fourteen subsamples which are used to generate a majority-rule prediction for each component measurement. Prediction accuracies here correspond to the exact fraction of components predicted correctly in the test set for each component class. The size of these test sets in each scenario are shown in Table 2. This number of components varies for each component class due to some device types not having a certain component

class (namely, power and ethernet) and other device types having multiples of individual components (memory chips) on a single board. As an example, a memory classification accuracy of 0.952 corresponds to 60 of the 63 test set components being predicted correctly with three predicted incorrectly. A very similar accuracy of 0.957 for ethernet transceivers corresponds to 22 of 23 components predicted correctly, meaning that only one was predicted incorrectly.

Table 3 shows that our CNN achieves equal or better median accuracy over the baseline k-NN classifier in every test scenario. As shown in Figure 7, our CNN architecture achieves a median classification accuracy over 95% in all cases except when classifying power management ICs in the excited mode. In all but that same case, the excited mode produces equal or better results respectively for each of the minimum, median, and maximum accuracies obtained with the idle state measurements.

As shown in Figure 7, the processor model is most consistently predicted correctly, followed by the memory, ethernet, then power ICs. These results follow intuition, since EM emanations related to program activity should be most present near the CPU. Even without these excitations, processor classification results in idle state are still nearly perfect due to the subtle spread of sideband activity, seen in the top left of Figure 1.

The memory chips nearby the processor are spuriously active in the idle state, but consistently active for most classes when the processor is actively transferring data in the excited state. This discrepancy could explain the difference between the two excitation modes' prediction accuracy. We note that the MEM4G08D3EABG-125 memory chip seems to use spread frequency clocking, which results in no visible carrier frequency or modulated excitation in the excited state.

Since both the idle and excited state emanations of ethernet components look nearly identical as we see in Figure 1, we expect the network to perform similarly on both. Indeed, we find small differences in the minimum prediction accuracies across the ten trials which we may attribute to the random cross-validation sampling; though overall they share the same median accuracy.

We hypothesize that prediction performance suffers when predicting power management ICs in the excited state due to the virtually identical emanations. We also believe that since we only had one device type representing the AXP223 power management IC, versus five types with the AXP209, the large class imbalance proved difficult to overcome for the model. There exist many techniques to deal with class imbalances, such as class-weighted sampling or loss functions [8], but we did not test them here so that all processing schema remain the same. The distinct emanation signatures of the different power management ICs in the idle state yield much better prediction accuracy.

As displayed in Table 4, we find the highest False Positive or False Negative rates for component models with the fewest overall measurements taken, especially when they exhibit virtually identical spectra to another model. We note that these error rates are most pronounced when we were only able to train on a single IC of a certain model while other models were common enough across multiple device types to allow several measurements. This was most pronounced for the H5TQ2G63BFR memory IC, AXP223 power IC, and KSZ9031RNXCC-TR and RTL8201CP Ethernet ICs which each had only one training example in our cross-validation setup.

Our anomaly detection results presented in Table 5 show that our method is able to detect unknown ICs on which the CNN was never trained in many cases. Each group of five rows depicts the results of a test with one memory component held out from training and treated as unknown. Our method yields the lowest accuracy when detecting an anomalous memory IC model which has EM emanations similar to a model with which the CNN was trained. When the CNN is trained on all models directly, it learns feature embeddings that adequately distinguish each IC model. Without training on all IC models, the network relies on feature embeddings whose activations may not differentiate all unknown ICs from known ICs. Datasets generated and analyzed herein are available from the corresponding author on reasonable request.

6.2 Interpreting results

Although we did not extract hand-crafted features, a careful inspection of the inputs as they pass through the network can give an indication of what our CNN deemed important for component classification. Figure 8 shows how the distribution of training subsamples for the processors in idle state look as they enter the network (top) and exit the convolution layers (bottom). Although the the inputs for each class look similar, we note subtle differences in the height of the carrier peak, presence of small modulated excitations, slope of the sidebands, and sharpness of the trail-off at the outer edges of the measured bandwidth. After passing through the convolution layers of the network, we see that the five filter channels extracted these distinct features from the input. To the naked eye, these outputs can be hypothesized to represent the sideband slopes

			Idle				Excited			
	Model	# ICs	TPR	TNR	FPR	FNR	TPR	TNR	FPR	FNR
Processor	A7	23	1.000	1.000	0.000	0.000	1.000	0.965	0.035	0.000
	A8	17	1.000	1.000	0.000	0.000	0.965	1.000	0.000	0.035
Memory	H5TQ2G63BFR	5	0.820	0.988	0.012	0.180	0.980	1.000	0.000	0.020
	H5TQ2G83BFR	12	0.933	0.967	0.033	0.067	1.000	1.000	0.000	0.000
	K4B4G1646Q-HYK0	6	1.000	0.993	0.007	0.000	1.000	1.000	0.000	0.000
	MEM4G08D3EABG-125	12	0.933	0.994	0.006	0.067	1.000	1.000	0.000	0.000
	MT41K256M16HA-125:E	28	0.921	0.954	0.046	0.079	1.000	0.997	0.003	0.000
Power	AXP223	6	0.767	1.000	0.000	0.233	0.400	0.959	0.041	0.600
	AXP209	29	1.000	0.767	0.233	0.000	0.959	0.400	0.600	0.041
Ethernet	KSZ9031RNXCC-TR	5	0.800	1.000	0.000	0.200	0.800	0.994	0.006	0.200
	LAN8710A-EZC-TR(QFN-32)	12	0.992	0.845	0.155	0.008	0.958	0.855	0.145	0.042
	RTL8201CP	6	0.883	0.994	0.006	0.117	0.900	0.976	0.024	0.100

Table 4 Classification metrics averaged across 10 trials for each component model and excitation type. Shown is the number of ICs of each component model in the test set, True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), and False Negative Rate (FNR) for both excitation states.

Models	TPR	TNR	FPR	FNR	Accuracy
Unknown	0.980	0.974	0.026	0.020	0.975
H5TQ2G83BFR	1.000	1.000	0.000	0.000	1.000
K4B4G1646Q-HYK0	0.967	1.000	0.000	0.033	0.997
MEM4G08D3EABG-125	0.908	0.998	0.002	0.092	0.981
MT41K256M16HA-125:E	0.993	1.000	0.000	0.007	0.997
H5TQ2G63BFR	0.780	1.000	0.000	0.220	0.983
Unknown	0.492	0.951	0.049	0.508	0.863
K4B4G1646Q-HYK0	1.000	1.000	0.000	0.000	1.000
MEM4G08D3EABG-125	0.883	1.000	0.000	0.117	0.978
MT41K256M16HA-125:E	1.000	0.826	0.174	0.000	0.903
H5TQ2G63BFR	0.840	1.000	0.000	0.160	0.987
H5TQ2G83BFR	1.000	1.000	0.000	0.000	1.000
Unknown	0.350	0.958	0.042	0.650	0.900
MEM4G08D3EABG-125	0.867	1.000	0.000	0.133	0.975
MT41K256M16HA-125:E	1.000	0.889	0.111	0.000	0.938
H5TQ2G63BFR	0.820	1.000	0.000	0.180	0.986
H5TQ2G83BFR	0.983	1.000	0.000	0.017	0.997
K4B4G1646Q-HYK0	1.000	1.000	0.000	0.000	1.000
Unknown	1.000	0.973	0.027	0.000	0.978
MT41K256M16HA-125:E	0.989	1.000	0.000	0.011	0.995
H5TQ2G63BFR	0.920	1.000	0.000	0.080	0.994
H5TQ2G83BFR	1.000	0.876	0.124	0.000	0.900
K4B4G1646Q-HYK0	1.000	1.000	0.000	0.000	1.000
MEM4G08D3EABG-125	0.925	1.000	0.000	0.075	0.986
Unknown	0.775	0.963	0.037	0.225	0.879

Table 5 Classification metrics for anomaly detection averaged across 10 trials for memory ICs in an excited state

. Each group of five rows represents an individual memory IC model being removed from training and treated as unknown. Shown are the True/False Positive/Negative rates and per-model classification accuracy.

(blue), carrier strength and sideband slopes (orange), carrier versus excitation peak differences (green), overall sideband decay shape (red), and sideband trail-off (purple). The network learns to extract and emphasize many of the subtle differences between the inputs that we see with the naked eye.

To add evidence to our interpretation hypotheses, we pass the inputs one layer further in the network to analyze how the convolution layer outputs shown in Figure 8 are modified by the first linear layer. As seen in the top of Figure 9, each of the 50 neurons' weights (each row of $W \in \mathcal{R}^{50\times 140}$) show distinct differences in weight magnitude corresponding to the channels of the convolution layers' output. To understand how these weights affect their inputs, we analyze how the weights of an "average neuron" $\mathbf{w}_{\text{ave}} = \sum_{i=1}^{50} W_{i,*}$ align with the input. We compute the element-wise weighting of the convolution layer outputs with the average neuron and overlay it with the same convolution layer outputs displayed in Figure 8. We note where $\mathbf{w}_{\text{ave}}$ has strong correlation with the input of each class. Without looking at the individual tendencies of each neuron in this layer, we argue that this average neuron highlights which part of its input that *most* of the 50 neurons emphasize to discriminate between classes. These peaks correspond to the carrier frequency peak magnitudes and the outer edges of the sideband trail-off. Nuanced features like sideband slope or carrier versus excitation peak differences may be differentiated by individual neurons, but the average neuron seems to look at these highlighted peaks as the most important general features.

Although this analysis strays from theoretical bases and likely suffers from some level of hindsight bias, it is worth noting that a similar analysis is entirely impossible with many common classifiers such as the k-NN classifier. When we have a decent idea of important discriminative features from prior work or visual inspection of the input signals, it is useful to compare our intuitions with the features extracted by such a general model. In cases where features are not visually discriminative, this model-guided analysis can be useful to yield insights.

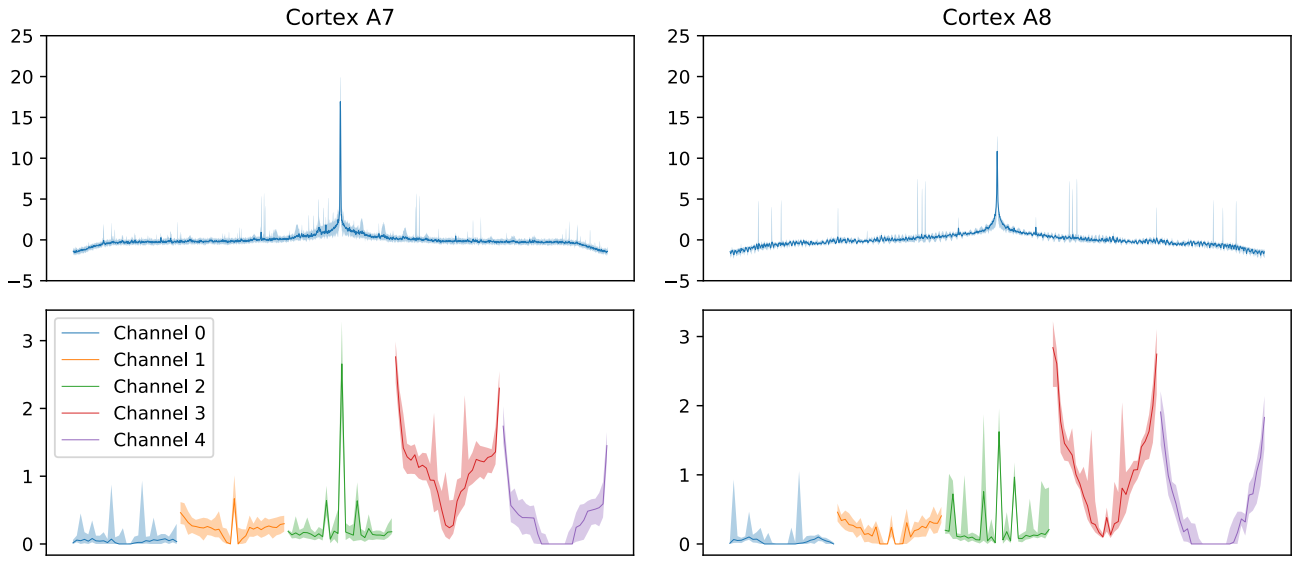

Fig. 8 Median of training set subsamples for processors in the idle state, scaled to dB and subtracted to zero-mean for each component model with shading below and above corresponding to the 10% and 90% quantiles respectively (top). The flattened output of the same median and shaded quantiles after passing through all convolution layers, just before entering linear layers (bottom).

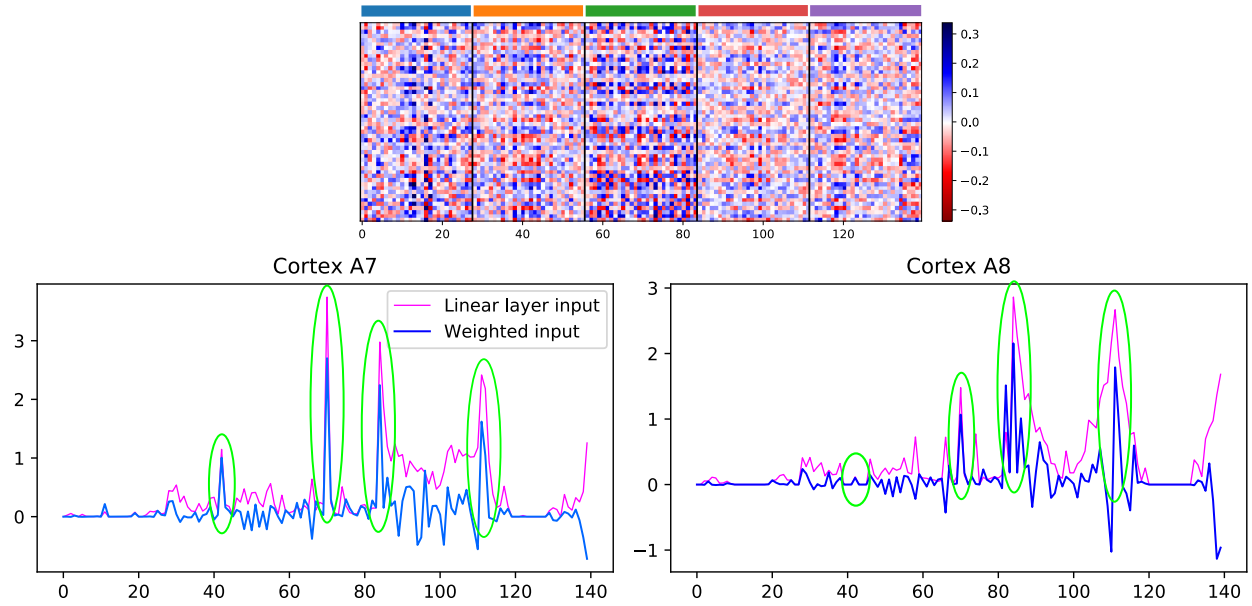

Fig. 9 First linear layer weights W with black vertical lines separating the weights that correspond to the convolution output channels as color-coded in Figure 8 (top). Comparison of the inputs of each component model weighted by the sum of neuron weights (blue), overlaying the inputs (magenta) to the first linear layer (bottom). Weighted inputs with strong correlation (or anti-correlation) to inputs circled in green.

7 Conclusion

Our work develops a lightweight CNN to classify a broad range of motherboard components in idle and excited states based on their EM emanations. Our robust preprocessing and cross-validation pipeline allows us to train models with strong classification accuracy across a range of component classes. We outperform a k-Nearest Neighbors classifier baseline in all tested cases. We show that a simple anomaly detection procedure can be used with our CNN to detect counterfeit ICs on which the CNN was not trained. While our network performs well across a variety of component classification problems, there is room for improvement with future work. We use a simple anomaly detection procedure based solely off of the CNN's output activa-

tions, whereas the state-of-the-art in that domain takes cues from all layers in the CNN. We do not claim to have made a comprehensive architecture search and it is likely that architecture tweaks could improve performance; especially with individual models tuned to each of the eight individual component identification problems. To emphasize the utility of this end-to-end trained method, we provide an in-depth analysis of signals passing through our model. This process allows us to understand the discriminative features of different components when they may not be visually apparent. Our results demonstrate that a lightweight CNN can classify diverse EM emanations accurately while generating insights about their discriminative features.

References

1. Adibelli, S., Juyal, P., Nguyen, L., Prvulovic, M., Zajić, A.: Near-field backscattering based sensing for hardware trojan detection. IEEE Transactions on Antennas and Propogation p. to appear (2020)
2. Agrawal, D., Archambeult, B.: The em side-channel(s). Proc. Crypto. HW and Emb. Sys. (CHES) pp. 29–45 (2002)
3. Ahmed, M.M., Hely, D., Perret, E., Barbot, N., Siragusa, R., Bernier, M., Garet, F.: Authentication of microcontroller board using non-invasive em emission technique. In: 2018 IEEE 3rd International Verification and Security Workshop (IVSW), pp. 25–30 (2018)
4. Alam, M., Khan, H.A., Dey, M., Sinha, N., Callan, R., Zajić, A., Prvulovic, M.: One&done: A single-decryption em-based attack on openssl's constant-time blinded rsa. In: Proceedings of the 27th USENIX Security Symposium (2018)
5. Backes, M., Dürmuth, M., Gerling, S., Pinkal, M., Sporleder, C.: Acoustic side-channel attacks on printers. In: Proceedings of the 19th USENIX Conference on Security, USENIX Security '10, p. 20. USENIX Association, USA (2010)
6. Callan, R., Zajić, A., Prvulovic, M.: A practical methodology for measure the side-channel signal available to the attacker for instruction level events. IEEE MICRO **14**, 1–12 (2013)
7. Cobb, W.E.: Exploitation of unintentional information leakage from integrated circuits. Ph.D. thesis, Air Force Institute of Technology (2017)
8. Cui, Y., Jia, M., Lin, T.Y., Song, Y., Belongie, S.: Class-balanced loss based on effective number of samples (2019)
9. Dogan, H., Forte, D., Tehranipoor, M.M.: Aging analysis for recycled fpga detection. In: 2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 171–176 (2014)
10. Dong, X., Weng, H., Beetner, D.G., Hubing, T.H., Wunsch, D.C., Noll, M., Göksu, H., Moss, B.: Detection and identification of vehicles based on their unintended electromagnetic emissions. IEEE Transactions on Electromagnetic Compatibility **48**(4), 752–759 (2006)
11. van Eck, W.: Electromagnetic radiation from video display units: An eavesdropping risk? Computers & Security **4**(4), 269 – 286 (1985). DOI https://doi.org/10.1016/0167-4048(85)90046-X
12. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: G. Gordon, D. Dunson, M. Dudík (eds.) Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, *Proceedings of Machine Learning Research*, vol. 15, pp. 315–323. PMLR, Fort Lauderdale, FL, USA (2011)
13. Göksu, H., Wunsch, D., Dong, X., Kökce, A., Beetner, D.: Detection and identification of vehicles based on their spark-free unintended electromagnetic emissions. IEEE Transactions on Electromagnetic Compatibility **60**, 1594–1597 (2018). DOI 10.1109/TEMC.2017.2773706
14. Guin, U., Huang, K., DiMase, D., Carulli, J.M., Tehranipoor, M., Makris, Y.: Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain. Proceedings of the IEEE **102**(8), 1207–1228 (2014)
15. Hutter, M., Schmidt, J.M.: The temperature side-channel and heating fault attacks. In: Smart Card Research and Advanced Applications: 12th International Conference (CARDIS), vol. 8419 (2013). DOI doi:10.1007/978-3-319-08302-5_15
16. Kae-Nune, N., Pesseguier, S.: Qualification and testing process to implement anti-counterfeiting technologies into ic packages. In: 2013 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 1131–1136 (2013)
17. Khan, H.A., Sehatbakhsh, N., Nguyen, L.N., Prvulovic, M., Zajić, A.: Malware detection in embedded systems using neural network model for electromagnetic side-channel signals. Journal of Hardware and Systems Security **3**(4), 305 – 318 (2019). DOI 10.1007/s41635-019-00074-w
18. Kocher, P., Jaffe, J., Jun, B.: Tdifferential power analysis. In: Advances in Cryptology (CRYPTO), vol. 1666 (1999)
19. Kuhn, M.: Compromising emanations: Eavesdropping risks of computer displays. Tech. rep., Computer Laboratory, University of Cambridge (2004)
20. Laput, G., Yang, C., Xiao, R., Sample, A., Harrison, C.: Em-sense: Touch recognition of uninstrumented, electrical and electromechanical objects. In: Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, UIST '15, p. 157–166. Association for Computing Machinery, New York, NY, USA (2015). DOI 10.1145/2807442.2807481. URL https://doi.org/10.1145/2807442.2807481
21. Lee, K., Lee, K., Lee, H., Shin, J.: A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In: Advances in Neural Information Processing Systems, vol. 31, pp. 7167–7177. Curran Associates, Inc. (2018)
22. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019)
23. Olimex: A10-olinuxino-lime. Available at https://www.olimex.com/Products/OLinuXino/A10/A10-OLinuXino-LIME-n4GB/open-source-hardware (2020/05/08)
24. Olimex: A13-olinuxino. Available at https://www.olimex.com/Products/OLinuXino/A13/A13-OLinuXino/open-source-hardware (2020/05/08)
25. Olimex: A13-olinuxino-micro. Available at https://www.olimex.com/Products/OLinuXino/A13/A13-OLinuXino-MICRO/open-source-hardware (2020/05/08)
26. Olimex: A20-olinuxino. Available at https://www.olimex.com/Products/OLinuXino/A33/A33-OLinuXino/open-source-hardware (2020/05/08)
27. Olimex: A20-olinuxino-lime. Available at https://www.olimex.com/Products/OLinuXino/A20/A20-OLinuXino-LIME/open-source-hardware (2020/05/08)

28. Olimex: A20-olinuxino-lime2. Available at https://www.olimex.com/Products/OLinuXino/A20/A20-OLinuXino-LIME2/open-source-hardware (2020/05/08)
29. Olimex: A20-olinuxino-micro. Available at https://www.olimex.com/Products/OLinuXino/A20/A20-OLinuXino-MICRO/open-source-hardware (2020/05/08)
30. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019). URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
31. Pecht, M., Tiku, S.: Bogus: electronic manufacturing and consumers confront a rising tide of counterfeit electronics. IEEE Spectrum **43**(5), 37–46 (2006)
32. Prvulovic, M., Zajić, A., Callan, R.L., Wang, C.J.: A method for finding frequency-modulated and amplitude-modulated electromagnetic emanations in computer systems. IEEE Transactions on Electromagnetic Compatibility **59**(1), 34–42 (2017)
33. Riscure: Em probe station. Available at https://getquote.riscure.com/en/quote/2101064/em-probe-station.htm (2020/05/08)
34. Rubino, R.: Wireless device identification from a phase noise prospective. Master's thesis, University of Padova, Italy (2010)
35. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(56), 1929–1958 (2014). URL http://jmlr.org/papers/v15/srivastava14a.html
36. Technologies, K.: M9391a pxie vector signal analyzer: 6 ghz. Available at https://www.keysight.com/en/pd-2317933-pn-M9391A/pxie-vector-signal-analyzer-1-mhz-to-3-ghz-or-6-ghz?&cc=US&lc=eng (2020/05/08)
37. Vann, J.M., Karnowski, T.P., Kerekes, R., Cooke, C.D., Anderson, A.L.: A dimensionally aligned signal projection for classification of unintended radiated emissions. IEEE Transactions on Electromagnetic Compatibility **60**(1), 122–131 (2018)
38. venturebeat.com: Feds close huge chip counterfeiting case (2011). URL https://venturebeat.com/2011/09/25/feds-close-the-books-on-a-huge-chip-counterfeiting-scheme/. [Online; posted 25-September-2011]
39. Werner, F., Chu, D.A., Djordjević, A.R., Olćan, D.I., Prvulovic, M., Zajić, A.: A method for efficient localization of magnetic field sources excited by execution of instructions in a processor. IEEE Transactions on Electromagnetic Compatibility **60**(3), 613–622 (2018)
40. Werner, F.T., Yilmaz, B.B., Prvulovic, M., Zajić, A.: Leveraging em side-channels for recognizing components on a motherboard. IEEE Transactions on Electromagnetic Compatibility pp. 1–14 (2020). DOI 10.1109/TEMC.2020.3016892
41. www.ice.gov: Visiontech administrator sentenced to prison for role in sales of counterfeit circuits destined to us military (2011). URL https://www.ice.gov/news/releases/visiontech-administrator-sentenced-prison-role-sales-counterfeit-circuits-destined-us. [Online; posted 25-October-2011]
42. Yang, C., Sample, A.P.: Em-id: Tag-less identification of electrical devices via electromagnetic emissions. In: 2016 IEEE International Conference on RFID (RFID), pp. 1–8 (2016)
43. Yilmaz, B.B., Mert Ugurlu, E., Zajić, A., Prvulovic, M.: Cell-phone classification: A convolutional neural network approach exploiting electromagnetic emanations. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2862–2866 (2020). DOI 10.1109/ICASSP40776.2020.9054006
44. Yilmaz, B.B., Zajić, A., Prvulovic, M.: Capacity of em side channel created by instruction executions in a processor. Processings of IEEE IEMCON pp. 1–5 (2019)