# SpotOn: Adversarially Robust Keyword Spotting on Resource-Constrained IoT Platforms

Mehreen Jabbeen
mehreen.jabbeen@cse.iitd.ac.in
Indian Institute of Technology Delhi
New Delhi, India

Vireshwar Kumar
viresh@cse.iitd.ac.in
Indian Institute of Technology Delhi
New Delhi, India

Rijurekha Sen
riju@cse.iitd.ac.in
Indian Institute of Technology Delhi
New Delhi, India

## ABSTRACT

IoT devices (e.g., voice assistants) that execute real-time speech commands are proliferating fast in our daily lives. In such a device, detecting the correct keyword spoken as a command triggers the supported function, and hence keyword spotting (KWS) using a machine learning (ML) model is the pivotal task in their functioning. However, KWS is vulnerable to adversarial machine learning (AML)-based attacks through which an adversary can craft an adversarial audio sample that sounds like a benign keyword to a human, but is detected as a different keyword by the KWS pipeline. In this paper, we propose SpotOn, a novel KWS pipeline that both recovers from AML attacks, as well as detects whether an attacker is using the device to generate AML noise. Using the Google speech command dataset, we demonstrate that SpotOn provides reasonable accuracy in correctly detecting keywords in the absence or presence of AML attacks. Through careful optimizations, we enable SpotOn to process streaming speech input on resource-constrained IoT devices. Overall, the design of SpotOn provides critical insights into making voice-controlled IoT devices suitable for safety-critical systems.

## CCS CONCEPTS

• **Security and privacy** → **Systems security**; • **Computer systems organization** → *Embedded systems*.

## KEYWORDS

Internet of Things (IoT), Adversarial Machine Learning (AML), Keyword Spotting (KWS), Neural Networks (NN)

**ACM Reference Format:**
Mehreen Jabbeen, Vireshwar Kumar, and Rijurekha Sen. 2018. SpotOn: Adversarially Robust Keyword Spotting on Resource-Constrained IoT Platforms. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 16 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 INTRODUCTION

Voice assistants have proliferated our daily lives accepting a myriad of spoken commands, from setting timers and alarms, to controlling our home appliances, to fetching directions as we drive. While cloud support still handles Automatic Speech Recognition (ASR) and other web-based APIs for these voice controlled services, a significant portion of audio processing is being increasingly pushed to the IoT or edge platform owned by the user [23, 27]. *Increase in responsiveness* as cloud communication latencies are replaced with on-device computations, as well as *privacy of voice data* of the users, are two main reasons of moving audio processing from cloud to IoT platforms. Embedded hardware platforms with multi-core processors and even neural network accelerators are being sold now to support on-device audio processing for voice assistants [9, 49, 50]. There is a simultaneous growth in efficient software libraries for audio processing at the edge [19, 54].

*Wake-word detection* i.e. detecting words like "Alexa", "OK Google", "Hello Siri" etc. that voice assistants constantly wait for, to know whether they are being addressed by the user, *keyword spotting* i.e. spotting a variety of words from a fixed vocabulary to trigger different functions or web APIs on the voice assistant like playing a song on Spotify or fetching directions from Google Maps, and *mapping word-to-intent* for controlling specific tasks on smart devices like changing colors of smart lights or altering cooking modes on microwave-ovens, are now primarily handled by IoT platforms. We term these tasks as Keyword Spotting or KWS in this paper. In a typical KWS pipeline, the IoT device first extracts the features, e.g., mel-frequency cepstral coefficients (MFCC), from the received audio sample, then feeds the features to a machine learning (ML) model to detect the keyword, and take appropriate actions based on the detected word.

Unfortunately, similar to other domains using ML models (e.g., those in the computer vision domain [12]), the models used in KWS are vulnerable to adversarial machine learning (AML)-based attacks [6]. An attacker can intelligently craft an *adversarial sample* which sounds like a *benign sample* to any human listening to the keyword, but when the adversarial sample is fed to the KWS pipeline during its operation phase, it is detected as a different keyword [8, 16, 29, 38, 44, 59]. This kind of attack can render a voice assistant difficult to use for a device owner in the best case, and cause breach of security (e.g. opening a smart lock instead of closing it), in the worst case. As more and more audio ML happens at the edge to enhance responsiveness and data privacy, user vulnerability can gradually increase, unless appropriate safe-guards are placed within the IoT platforms. This paper presents *SpotOn*, an end-to-end system for adversarially robust keyword spotting on resource constrained IoT platforms.

As ML based audio processing traditionally happened in the cloud using ASR models, a vast body of systems security papers target the problem of Adversarial ML attacks and possible defences for ASR models in the cloud [6, 42, 53, 55, 67]. These AML defences for ASR models on the cloud report two important metrics: (1) *benign accuracy*, i.e., the probability of correctly detecting the original sentence or phrase from the benign audio samples in the absence of any attack, and (2) *adversarial robustness*, i.e., the probability of correctly detecting the original sentence or phrase from the adversarial sample despite the perturbations added by the attacker. However, in SpotOn, we need a third critical metric to be evaluated, namely (3) *Runtime latency*, i.e., the time taken to execute the KWS pipeline with added AML defence for detecting the keywords. IoT platforms have limited computational resources to even run the neural models for keyword detection. However, the users should not detect any perceptible latency due to added security. Thus the choice of AML defence strategy in SpotOn needs to evaluated for computational efficiency on resource-constrained IoT platforms, while maintaining high benign accuracy and high adversarial robustness. This creates interesting trade-offs not analyzed in prior work on AML defences for cloud platforms.

Another research gap between AML defences for ASR models on cloud and AML defences for KWS models on IoT platforms, is the defence for ASR can rely on the significant length of the spoken sentences and phrases for the ASR task. The authors in [67], for example, use temporal dependencies in natural languages to detect and remove adversarial noise. Keywords, on the other hand, are short, and can be completely garbled by adversarial noise addition. The robustness a phrase gets from significant length, is absent for a keyword. Thus special care needs to be taken to preserve its robustness.

## 1.1 Main Contributions

We make four significant contributions in this paper, to make KWS pipelines more robust on real IoT hardware platforms, that are currently being used as voice assistants.

Our **first contribution** is *runtime attack recovery* from AML attacks. SpotOn uses a smart combination of *robust neural network model architecture*, with *audio input transformations*, for recovering from AML attacks at runtime, while the user is using the voice assistant. There is a wide variety of neural network (NN) architectures with dense layers, convolutions, memory cells, transformers etc. – which all show high accuracy in spotting keywords, with appropriate model training. However, does any of these NN architectures have any significant advantage in AML robustness? We empirically examine this for the first time using 8 different neural network architectures, and three categories of AML attacks (white box, gray box and black box). The self-attention mechanism of transformer architecture gives best adversarial accuracy, as well as least attack transferability. We therefore use transformer architecture as the backbone of SpotOn. Transformer alone, however, do not give the necessary adversarial robustness against all attacks, e.g. more powerful white box attacks. We therefore add input transformations to the audio, before MFCC features are extracted, to first decompose the audio waveform, discard some information and then reconstruct the audio by estimation. AML noise is significantly reduced through this process, which gives the desired adversarial robustness, when combined with the transformer model.

Our **second contribution** is *runtime attack detection* of black and gray box AML attacks, on the IoT platform itself. SpotOn detects whether the IoT platform is being used by an attacker to train adversarial noise. In black-box and gray-box attacks, the attacker buys the same IoT platform as the victim user. The attacker then iteratively queries the IoT platform, using a target keyword that he wishes to garble, fine-tuning the adversarial noise to be added to that keyword, based on the responses received from the IoT platform. SpotOn keeps a running fingerprint of audio keywords queried by the user, and if consecutive keywords exceed the similarity threshold with the running fingerprint, SpotOn decides that the device is being queried to train AML noise. It therefore stops responding to further audio inputs. The runtime attack detection works in parallel with the runtime attack recovery, as SpotOn cannot decide whether it is being used by a normal user or an attacker, without running the processing. We use the multi-core capability of voice assistant IoT platforms, for concurrent running of attack detection and recovery. If attack is detected, the recovered detected keyword is suppressed, while recovered detected keyword is returned or used in subsequent tasks, if no attack is detected.

Our **third contribution** lies in extensive optimizations to execute the runtime attack recovery and detection of SpotOn on resource constrained IoT platforms, in real time. We note that on low-end IoT devices, running the transformer model is not feasible due to their extremely limited computing capability and storage constraints. Hence, we optimize the transformer model for shortening the model size and reducing the computational overhead without affecting its accuracy. Reducing the number of layers in the transformer model, fusing the neural network layers, magnitude based weight pruning, integer quantization, mapping of resultant neural network to most efficient ARMNN kernels that use SIMD and vector instructions – are significant optimizations SpotOn uses to meet these strict latency requirements on resource constrained IoT platforms.

Our **fourth contribution** lies in a detailed analysis whether non-runtime AML defences are possible for KWS tasks. While SpotOn runs at high benign accuracy and adversarial robustness and processes streaming audio data at runtime, if we could reduce runtime computations for AML defences, it would be a huge plus for resource constrained IoT platforms. We examine traditional adversarial training as a non-runtime defence, where adversarial samples are added when the KWS model is trained. This, however, is limited by generating all possible adversarial samples from different attacks. We build a second novel non-runtime defence, based on a recent observation that attack noise overfits to the NN model the attacker uses [4]. We first extract the MFCC features from the received audio samples, at runtime. However, rather than feeding the features directly to the ML model, we perform a secret key-based feature permutation and feed the permuted features to the utilized transformer model. One specific device performs the same permutation during the training as well as the operation phase of its model. However, different devices perform different permutations based on their keys. Hence, during the training phase, the permutation results in a unique set of model parameters for the specific device. During the operation phase, this prevents the adaptive attacker from gaining any advantage after obtaining the knowledge that a permutation is being employed as the defence. This novel non-runtime defence works very well for gray and black box attacks, but unfortunately cannot defend against white box attack. Also, training millions of different KWS models with different feature shuffling, so that each voice assistant comes with a unique KWS model, has it own operational constraints.

In summary, this paper presents SpotOn, a runtime system to recover from AML attacks for voice assistants, and simultaneously detect, if the assistant is being used to generate AML noise by an attacker. Non-runtime defence alternatives are examined for comprehensiveness, but shown to have limitations. SpotOn has 98% benign accuracy and a decent adversarial robustness against black, gray and white box attacks, and has been carefully optimized to run in real-time on streaming data, on resource constrained IoT platforms.

## 2 RELATED WORK

**Adversarial Machine Learning (AML) attacks:** Recent works have demonstrated numerous adversarial machine learning based attacks on different Voice processing systems (VPS). These attacks aim to generate adversarial examples that cause the target VPS to mis-classify the input. Such examples are skilfully crafted to trick the system into producing incorrect results. Optimization algorithms like Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), and Particle Swarm Optimization (PSO) have been extensively used in literature [14, 24] to produce such examples. Attacks like Kenansiville [5] and HVC [13] utilize signal processing based approaches to produce adversarial examples. Attacks like [3, 44] involve playing adversarial examples via compromised speakers. Metamorph [16] and Imperio [59] produce highly robust adversarial examples that remain successful even when played over-the-air. *We use three representative AML attacks [8, 14, 29], that use the above adversarial noise generation techniques, to evaluate SpotOn. Our three AML attacks are of black, gray and white box categories respectively, with the attacker having increasing information of the neural network model being targeted.*

**Defences against AML attacks:** A variety of defence mechanisms are being developed to at least detect AML attacks, and also remain robust against them if possible. Rajaratnam et al. [55] pre-process input audio using compression, band-pass filtering, audio panning and lengthening. The raw and pre-processed audios are then passed through a neural network. The two outputs match when the input is benign, and hugely vary when the input is adversarial. Kwon et al. [42] modify the input audio by adding a low level distortion using a low-pass filter. The additional distortion significantly alters the classification output of the adversarial input while having a negligible impact on the classification output of benign audio. Yang et al. [67] assess additional pre-processing techniques like quantization, down-sampling and local-smoothing. They also compare the transcription of a subset of utterance with the transcription for the entire utterance, and detect adversarial examples if the difference exceeds a threshold. These defences were, however, later shown to be incomplete by [63].

Liu and Ditzler [45] detect adversarial examples based on the DNN's quantization error. Däubener's [20] neural networks are trained to utilize the uncertainties in the DNN estimation to detect adversarial examples. Eisenhofer et al. [25] use psychoacoustic filtering and a band-pass filter, to remove frequencies above and below human perception. They train neural network model to augment it with psychoacoustic filtering and a band-pass filter. The resulting adversarial examples generated from such a model force adversarial perturbation to be in the human audible range. The legitimate user can hear the adversarial noise, and therefore detect the presence of an adversary.

All these defences detect the presence of an adversarial attack. But they do not allow the VPS system to operate to a reasonable extent in the presence of an attack. *Given an audio, SpotOn mitigates the effect of adversarial perturbation added, if any, and allows the model to produce correct classification result for the input keyword. It also runs on streaming audio on resource-constrained IoT platforms, while all previous defences were demonstrated on the cloud on ASR systems.*

**Attack prevention techniques:** Apart from detection of adversarial examples, there is another line of research focussing on prevention of adversarial ML attacks. In the image domain, some work has been done to evade query-based black box attacks. Stateful Detection [15] and Prada [39] focus on banning users who submit attack queries but are vulnerable against attackers who use multiple accounts to submit attack queries. On the other hand, Blacklight [43] is account-agnostic and prevents query-based black-box attacks by instantly identifying attack queries regardless of who issued them. *To the best of our knowledge, we are the first to propose a defence to mitigate query-based black-box attacks in audio domain.*

**Defence practicality on IoT:** To protect our smart IoT devices running KWS from adversarial ML attacks, the IoT device has to run the defence mechanism alongside KWS. This has its own challenges in terms of computational resources and latency. Successful software projects like Microsoft's open-source Edge Machine Learning library (EdgeML) [51] show the power of embedded ML on IoT devices. Compression of the deployed ML models using quantization, pruning of the ML model weights [32, 48, 66, 68], training small ML models with simpler architectures and fewer weights [22, 31, 40, 41, 56], specialized accelerators for matrix multiplication and other operations, all are collectively contributing to making IoT devices intelligent. *SpotOn carefully leverages these neural network optimization techniques, to run on streaming audio data on resource-constrained IoT platforms, without hampering user experience.*

## 3 THREAT MODEL, SYSTEM GOALS AND ARCHITECTURE

### 3.1 Conventional KWS Pipeline

Ideally, we would like to talk to our IoT devices as we verbally communicate with our fellow human beings. In practice, this task is facilitated by ASR, the holy grail for smart interactions, where long phrases and full sentences of free-form conversations are transcribed by IoT platforms. However, as current IoT platforms are constrained in compute, storage and power, the full-fledged ASR is executed through cloud interactions. When we ask Alexa to turn off the lights, Siri to start heart-rate monitoring, or Google Assistant to play a particular song, the IoT platform locally processes the wake-words "Alexa", "Hey Siri", or "Ok Google". On detecting the wake-word, the device then connects to the corresponding Amazon, Apple, or Google cloud service. The bigger phrases following the wake-word, e.g., "turn-off the lights", are deciphered in the cloud using more powerful ML models, and the results are sent back to the device to perform the necessary action.

However, to minimize the latency in processing speech commands, a small set of keywords, other than the cloud-triggering ones, are also programmed in the platforms to be detected locally using KWS. For example, in the command, "Alexa, stop", the keyword "stop" can be detected locally. Only if the device fails to detect any keyword in the utterance, it forwards the audio samples to the cloud for ASR. KWS is especially relevant for limited-function IoT platforms like smart watches and fitness bands, which can do only a few tasks like start GPS, stop step count, pick a phone call, etc. Each keyword (e.g., start, stop, play, or pause) corresponds to a particular task that the device can perform, and can be locally deciphered using KWS without the cloud interaction and internet connectivity. In addition to providing low latency in executing speech commands, running KWS locally mitigates the potential of losing privacy by avoiding the need of forwarding the user's speech samples to the cloud.

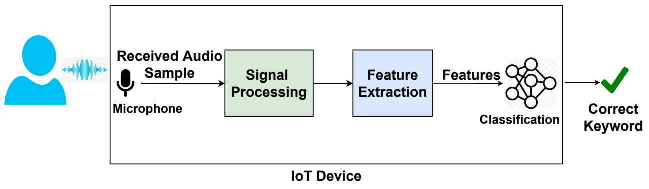


**Figure 1: Conventional keyword spotting pipeline.**

Figure 1 shows how KWS works on a IoT platform. When the user speaks a keyword, the received audio input is sampled by the platform's microphone. Then, the sampled audio is processed using conventional signal processing techniques to mitigate the effects of the external noise, and the processed sample is utilized to extract (MFCC) features. Finally, the extracted features are processed by the trained and stored ML-based classification model (henceforth referred to as the KWS model) to detect the keyword.

## 3.2 Representative AML Attacks

We use the following representative AML attacks to quantify the performance of different defence strategies.

**GA Attack:** It [8] is a black-box attack on Speech Commands classification model [57], where given a benign sample, attacker needs only their output labels from the target KWS model to craft adversarial samples. The GA attack aims for a particular target label, and hence it is a targeted attack. It needs the exact keyword audio it will manipulate using a gradient free genetic algorithm to train its adversarial noise. The authors assume that generated adversarial audio is being fed directly to the classification model.

**RL Attack:** It [29] is a gray-box attack that needs the probability distribution across all keywords the KWS model can detect. The RL attack tries not to give the correct output label, and hence it is an untargeted attack. Using a deep reinforcement learning architecture, it designs a real-time perturbation generator that approximates an optimal adversarial perturbation for future time points using observed data. The real-time adversarial perturbation generator is trained using a non-real-time adversarial perturbation generator by applying imitation learning and behavioural cloning technique.

**CW Attack:** It [14] is a white box targeted attack against Mozilla DeepSpeech ASR. To generate an adversarial perturbation, an attacker minimizes the CTC loss between the ASR output and the target transcription. It is a targeted attack which aims to change the input audio into a desired, pre-specified transcription. The attack operates directly on the raw samples used as input for the classifier. This attack method achieves a 100% success rate regardless of the desired transcription or the initial audio sample. By starting with arbitrary waveforms, such as music, it can effectively conceal speech within audio that would not typically be identified as speech. Moreover, by selecting silence as the target, it can successfully hide audio from speech-to-text systems.

The CW attack has been demonstrated on an ASR system [34]. How we port the attack for the KWS systems running on IoT platforms has been described in our experimental setup (Section 3.7).

## 3.3 Threat Model

The attacker's goal is to cause the IoT device of the victim to malfunction, without physically accessing it. As shown in Figure 2, the attacker plays an adversarial noise using a remote speaker such that the actual speech data of the user, corrupted with the noise, gets sampled by the IoT device's microphone. Due to this attack, the KWS model taking the corrupted input from the microphone, fails to detect the keyword correctly. This can cause annoyance and difficulty in the best case, for example, a smart door does not open as instructed with the "open" keyword by the user. The attacker can also violate the user safety in the worst case, for example, the smart door opens even if the user does not utter the "open" keyword and says something else.

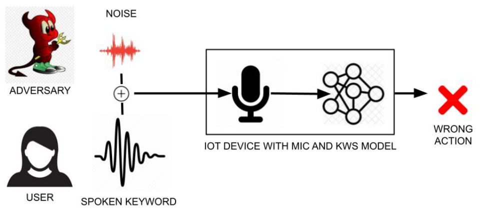


**Figure 2: Keyword spotting in an adversarial setting.**

**Adversarial Noise Generation** While a naive attacker can play a random noise on the speaker to garble the legitimate keyword spoken by the user, an advanced attacker can utilize the adversarial ML attacks to disrupt the correct functioning of ML models. That is, the attacker can first learn (or train) an appropriate adversarial noise for a specific keyword by interacting with the KWS model, and then play this learned noise. Figure 3 shows the workflow for the generation of the adversarial noise.

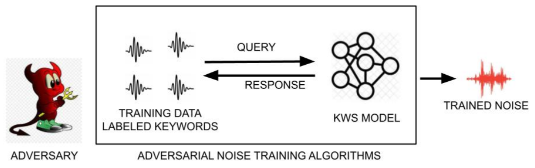

QUERY
RESPONSE
TRAINING DATA LABELED KEYWORDS
KWS MODEL
TRAINED NOISE
ADVERSARY
ADVERSARIAL NOISE TRAINING ALGORITHMS

**Figure 3: Adversarial noise generation.**

Typically, adversarial ML attacks consists of three phases:
❶ **Pre-processing Phase:**, where an adversary records the speech

commands spoken by the victim.
❷ **Design Phase:** The attacker then skilfully modifies the benign audio clip till he obtains an adversarial audio which satisfies his attack goal of outputting a false or attacker-controlled output.
❸ **Attack Phase:** The adversarial audio is fed to the target device.

## 3.4 SpotOn System Architecture

In this section, we focus on the problem of safeguarding our IoT devices from malicious users. To solve this problem, there are two issues that need to be resolved: (1) protection from being mis-used by an adversary, and (2) getting the keywords correctly classified by the IoT device in presence of AML attacks.

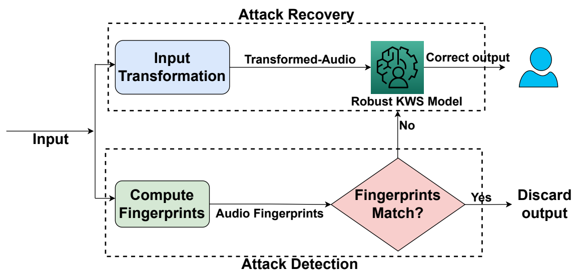


**Figure 4: Proposed KWS pipeline called SpotOn.**

We propose an end-to-end pipeline, SpotOn, that prevents an adversary from misusing an IoT device and provides correct model output even when the input is compromised by an adversary. For robust KWS, SpotOn comprises of two parallel-running modules: a detection module and a recovery module. Upon receiving an input, the detection module executes an algorithm to determine whether the input is an attack query. If so, the output is blocked, so the attacker can't learn what the model's output is, which ultimately hinders his attack process. The recovery module parallelly applies input transformations based on signal processing to process the input. The transformations based pre-processing makes sure that any adversarially crafted noise present in the input gets filtered out before being fed to the model for classification. We elaborate on the design choices undertaken in the making of SpotOn in section 4 and 5.

## 3.5 SpotOn System Goals

SpotOn aims to optimize the following evaluation metrics.
❶ **Benign Accuracy:** It is the accuracy of the target KWS model on the original or benign audio sample in the absence of any AML attack. This should be high.
❷ **Adversarial Robustness:** It is the accuracy of the KWS model against adversarial samples, i.e., benign audios perturbed by the adversarial noise. This should be high.
❸ **Attack Detection Accuracy:** It is the percentage of query-based attacks detected before the attack completes. It should be high.
❹ **SpotOn Runtime:** It is desirable to be able to process streaming audio input without missing any input samples. Hence, low run-time latency is necessary.

## 3.6 IoT Platforms

Table 1 presents the specifications of the two hardware platforms we use for our implementation and performance evaluations. The Raspberry Pi platform is analogous to a smart home system, like Alexa and Siri. Sensortile is analogous to wearables like smart watches. Both platforms use keyword spotting extensively to trigger different actions they are programmed for and should be robust against AML attacks.

**Table 1: Specifications of the platforms used for evaluation.**

| **Device** | Processor | RAM | Frequency |
|---|---|---|---|
| **Raspberry Pi** | Quad-core Cortex-A72 | 4 GB | 1.5 GHz |
| **Sensortile [61]** | Cortex-M4 | 128 KB | 80 MHz |

## 3.7 Experimental Dataset:

We use a subset of the Google speech command dataset [65], as shown in Appendix A. We implement a variety of neural network architecture-based KWS models as shown in Table 2. More details about these KWS models are presented in Appendix B. These implementations allow us to assess the contribution of model architecture towards adversarial robustness.

**Table 2: KWS model architectures and their notations.**

| **Model Architecture** | **Notation** | **Benign Accuracy (%)** |
|---|---|---|
| Convolutional Neural Network | cnn | 95.14 |
| Deep Neural Network | dnn | 87.58 |
| Depthwise Separable Convolutional Neural Network | dscnn | 95.55 |
| Convolutional Recurrent Neural Network | crnn | 97.09 |
| Gated Recurrent Units | gru | 96.52 |
| Long Short Term Memory Network | lstm | 82.42 |
| Basic Long Short Term Memory Network | b_lstm | 96.32 |
| Transformer | trans | 94.69 |

For GA attack, we randomly choose 50 audio clips from each of the ten classes. We produce adversarial samples wherein for each audio clip, we obtain nine audio clips corresponding to nine other target classes, generating a total of 4,500 (=500*9) adversarial samples. We have such a set of 4,500 adversarial samples for each KWS model. To implement RL attack, we train the RL algorithm-based *adversarial model* corresponding to a given KWS model using 30,000 audio samples from the dataset. The remaining 8,500 audio samples are used as the test data which is neither seen by the victim KWS model nor the adversarial model during its training. Then, the adversarial model takes each unseen test data sample as the input and generates a suitable adversarial noise. The 8,500 benign samples perturbed by the adversarial noise become our adversarial samples for a given KWS model. We repeat the procedure for each KWS model, generating 8,500 adversarial samples for each. Similarly, to adapt CW attack for KWS, we replaced DeepSpeech with a transformer-based KWS model. For each class $c$ of the ten classes, 90 benign audios are selected randomly from the rest of the 9 classes. The target class for these 90 audios is set as $c$.The attack pipeline is then run for these 900 audios to perturb them using CW attack to generate adversarial audios.

## 4 SPOTON: RUNTIME RECOVERY FROM AML ATTACKS

In this section, we present the recovery module of SpotOn. We discuss the design choices made in SpotOn to counter an attack if the IoT device receives an adversarial input. This essentially means the KWS model outputs correct classification keyword even when an adversarial input is given.

### 4.1 Searching for a robust model architecture

In this section, we empirically measure the performance of various KWS models in terms of both benign accuracy and adversarial robustness. The objective is to determine whether there exists a model that is robust enough to adversarial attacks.

Table 2 shows the benign accuracy and Table 3 shows the adversarial robustness values for different KWS models. We observe that the benign accuracy of all models is $> 85\%$. As such, any of these models can be used for KWS task. However, in Table 3, against the GA attack, we find the adversarial robustness drops drastically to $1 - 20\%$, for all models except the transformer. The transformer model has a less drastic drop, with an adversarial accuracy of around 58%. Moreover, with the RL attack, the adversarial accuracy of all models other than the transformer, drops to $30 - 65\%$. The transformer model is very robust against the RL attack, maintaining the robustness of more than 90%. CW attack, a white box attack is the strongest attack among the three attacks being considered in this work. Even the transformer model shows only 17% adversarial robustness towards CW attack. Evaluating other smaller models for CW attack does not make sense as it is clear they will perform much worse.

**Table 3: Adversarial robustness of different KWS models.**

| Model | Adversarial Robustness (%) | |
|---|---|---|
| | GA Attack | RL Attack |
| cnn | 11.22 | 61.13 |
| dnn | 1.29 | 64.66 |
| dscnn | 7.29 | 55.49 |
| crnn | 17.18 | 59.74 |
| gru | 9.38 | 49.34 |
| lstm | 9.44 | 34.29 |
| basic-lstm | 16.80 | 56.61 |
| **trans** | **57.99** | **90.01** |

Therefore, despite the fact that all models' benign accuracy is sufficient for practical application in IoT devices, they fail significantly in terms of their resistance against AML attacks. *The adversarial robustness data in Table 3 show that some neural network models are impacted more than others, and the transformer model is least impacted.*

### 4.2 Is ensemble of non-transformer models more robust?

As shown above in Table 3, the KWS models (except for the transformer model) do not perform well in the face of the AML attacks. To increase the adversarial robustness, one potential approach is to employ an ensemble of non-transformer KWS models [2]. The ensemble can be visualised as a giant neural network architecture where each of the models provides the probabilities for each keyword. To output the final keyword, different aggregation methods can be used to combine the output probabilities, e.g. average of the probabilities obtained from all models or the average across a random subset of models. As we consider an adaptive adversary, the adversarial sample training can adapt itself accordingly, i.e., the average probability value can be used to generate the adversarial samples. We find that the robustness against the GA attack is still only 13% with the ensemble approach. Robustness against RL attack (Table 4) is also poor.

**Table 4: Benign and adversarial accuracy for the RL attack with the ensemble of seven non-transformer KWS models.**

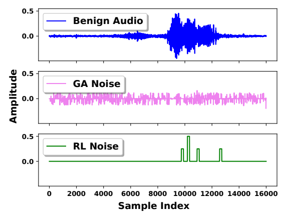
| Aggregation Method | Benign Accuracy (Mean/SD) | Adversarial Robustness (Mean/SD) |
|---|---|---|
| Average | 97.73 | 66.52 |
| Randomly pick 4 models | 97.36/0.09 | 63.72/0.26 |
| Randomly pick 1 model | 92.98/0.21 | 55.25/0.26 |

Essentially, the probability distributions across keywords is so similar across different KWS models that an ensemble also gives the same distribution. Hence, the ensemble is as useful as an individual KWS model for the attacker, making the ensemble approach less effective as a defence.

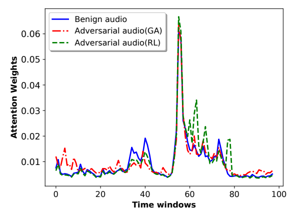
### 4.3 Why are transformers more robust against AML?

Transformer models are currently employed as the de-facto neural network architecture in the computer vision domain because of their high accuracy and excellent adversarial robustness. We also learn from the evaluations in Section 4.1 that the Transformer is the most robust model.

**Self-attention mechanism in transformer** To understand where the transformer architecture's adversarial robustness stems from, we analyze the effectiveness of its self-attention mechanism.



**(a) Audio waveforms.**



**(b) Attention masks.**

**Figure 5: (a) Benign audio sample corresponding to the keyword "Stop" and the corresponding adversarial noise in the GA and RL attacks. (b) Attention masks generated by the transformer model on the benign and adversarial audio samples show minimal changes.**

Figure 5a shows the benign audio sample for an instance of the "Stop" keyword, and the noise generated for this particular audio sample in the GA and RL attacks, respectively. Figure 5b shows the attention masks generated in the transformer model on the benign sample and the two adversarial samples. It is interesting to see in

Figure 5b that even in the presence of adversarial noise, the attention mask remains almost intact. This robustness of the attention mask has also been observed in the vision domain, and potentially increases the robustness against adversarial perturbations in KWS.

## 4.4 Attack transferability across models

We next evaluate attack transferability for the different neural networks. An AML attack is said to be transferable across models, if it can successfully reduce accuracy of (target) KWS models other than the (source) one it was trained with. This is relevant if the target IoT device cannot be physically accessed by the adversary, then he will not be able to query that model for the output label in the GA attack or output the probability distribution across labels in the RL attack. In that case, it has to produce the adversarial noise by querying other KWS models (source model) it can access, and use that noise to attack the model on the target device (target model).

Our evaluation results are shown in Table 5 and Table 6. We do not evaluate CW attack for transferability as being a white-box attack, it inherently assumes access to target model. In Table 5, we observe that the RL attack is transferable across the models as the robustness values in the non-diagonal cells are in the same range as in the diagonal cells. Table 6 shows that the GA attack, however, is not transferable across the models as the robustness values in the non-diagonal cells are significantly higher than in the diagonal cells.

We finally focus on the transferability results for the transformer model for both attacks. Independent of whichever model the adversary learns with, the target device can be robust to both attacks if it uses the transformer model (the last columns in Tables 5 and 6). In other words, the transformer model not only provides robustness when the adversarial sample is generated using the transformer model but also when the adversarial samples are generated using other KWS models. This further encourages us in using the transformer model in SpotOn for KWS tasks to improve adversarial robustness.

## 4.5 Enhancing robustness with input transformations

Transformer model alone cannot defend against powerful attacks like GA and CW, with an adversarial robustness of 58% and 17% respectively. Therefore, there is still some necessity of improvement in the adversarial robustness of transformers. In this section, we combine the notions of input transformation functions along with the transformer model, to boost the robustness numbers above 80% usability threshold.

Prior works have already applied Input-transformations including audio compression, quantization-dequantization, down- and up-sampling, and band-pass filtering. Even though the majority of these transformations are capable of detecting adversarial samples, they are useless against an adaptive attacker, who is well aware of the defence mechanism. However, there exist certain transformations that are more robust than others, even while facing an adaptable adversary.

Waveguard [37] employs audio input transformations, Linear Predictive Coding (LPC), and Mel Spectrogram Extraction and Inversion (Mel) to remove the effects of the adversarial noise in ASR. These transformations compress audio and produce perceptually informed representations.

Mel transformation comprises of extraction and inversion step. Extraction step decomposes the input audio into time and frequency components using the short-time Fourier transform (STFT). From the complex STFT coefficients, the phase information is discarded leaving behind only the magnitude spectrogram. This magnitude spectrogram is then compressed to obtain Mel Spectrogram. Thereafter, the inverse step begins wherein the Mel spectrogram is utilized to estimate the magnitude spectrogram which in turn is utilized to estimate the phase information. The estimated magnitude spectrogram and phase information are then used by inverse STFT to reconstruct the audio.

LPC transformation divides the input audio into overlapping windows. LPC coefficients for each window are estimated using a linear regression method. The number of LPC coefficients determine the compression level of the original audio. To reconstruct the audio from the estimated LPC coefficients, a random-noise excitation signal is used.

It is the reconstruction phase in both the transformations that potentially removes the adversarial perturbation, therefore yielding a clean audio. To get around these transformations, an adversary has to add a perturbation that could be retained in the Mel spectrogram for mel transformation and in LPC coefficients for LPC transform. This essentially means the perturbation no longer remains imperceptible.

Legitimate audio inputs are long in ASR unlike a single word in KWS. Whether the same filters can defend KWS against adversarial attacks, as effectively as they defend ASR, need to be empirically examined.

We use the KWS models with the different neural network architectures and present the benign accuracy and adversarial robustness values corresponding to the Mel and LPC filter-based input transformations in Figure 6. As seen from Figure 6a, unfiltered audios have around 90% benign accuracy and 80% adversarial robustness for all non-transformer models. Adding the transformations on the input audios reduces these benign accuracy values a little, but for most of the models it still remains around 80%. For the transformer model, the benign accuracy remains above 95% even after applying the transformations. The effect of transformations in the absence of attacks is, therefore, bit detrimental for the non-transformer models.

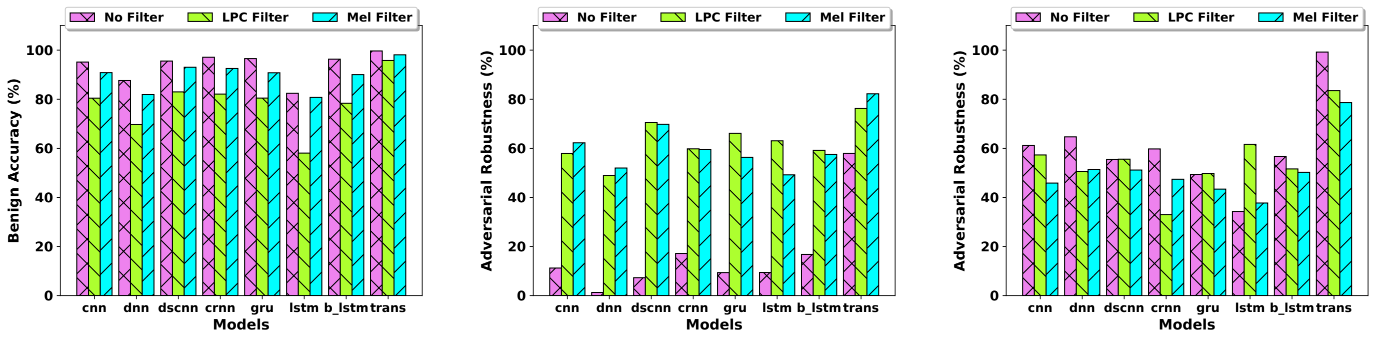
The GA attack drastically affects the adversarial robustness of all KWS models, as seen from the pink bars in Figure 6b. The robustness of all models, other than the transformer, with the unfiltered audios, is between only 0-20% for the GA attack. The two filtering techniques boost the adversarial robustness, as seen from the cyan and green bars in Figure 6b, but the maximum observed adversarial robustness lies below 80% for the non-transformer models. While as for the transformer model, Mel filter boosts the adversarial accuracy from 58% to 82%. As seen from the adversarial robustness values shown in Figure 6c, none of the two filtering mechanisms is effective against the RL attack for most of the models. While as for CW attack, the Mel filter greatly boosts the adversarial robustness of transformer model from 10% to 91% as shown in Table 7. Thus, the adversarial robustness of transformer model towards the GA and CW attacks, can improve with the input-transformations like the Mel filter.

**Table 5: Transferability of the RL attack across KWS models.**

| | **Target Model** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Source Model** | **cnn** | **dnn** | **dscnn** | **crnn** | **gru** | **lstm** | **b_lstm** | **trans** |
| **cnn** | 61.13 | 66.99 | 57.80 | 64.91 | 50.98 | 37.62 | 62.05 | 89.73 |
| **dnn** | 60.69 | 64.66 | 59.08 | 63.93 | 51.04 | 36.32 | 61.18 | 91.22 |
| **dscnn** | 59.61 | 65.21 | 55.49 | 64.06 | 50.47 | 37.33 | 60.87 | 90.51 |
| **crnn** | 58.92 | 62.39 | 53.45 | 59.74 | 49.84 | 34.14 | 56.55 | 89.91 |
| **gru** | 60.53 | 64.69 | 55.93 | 63.06 | 49.34 | 33.18 | 57.31 | 89.77 |
| **lstm** | 60.84 | 66.12 | 57.92 | 64.88 | 51.72 | 34.29 | 60.38 | 90.33 |
| **b_lstm** | 61.52 | 63.25 | 58.89 | 63.22 | 51.89 | 33.89 | 56.61 | 90.27 |
| **trans** | 67.30 | 65.37 | 65.50 | 65.76 | 57.43 | 65.79 | 71.76 | 90.01 |

**Table 6: Transferability of the GA attack across KWS models.**

| | **Target Model** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Source Model** | **cnn** | **dnn** | **dscnn** | **crnn** | **gru** | **lstm** | **b_lstm** | **trans** |
| **cnn** | 11.22 | 60.64 | 77.29 | 79.16 | 78.67 | 77.98 | 80.49 | 89.37 |
| **dnn** | 87.99 | 1.29 | 90.87 | 90.82 | 86.65 | 86.42 | 91.52 | 93.56 |
| **dscnn** | 81.62 | 69.67 | 7.29 | 82.49 | 84.73 | 83.64 | 87.27 | 92.11 |
| **crnn** | 74.93 | 63.96 | 70.22 | 17.18 | 75.27 | 75.13 | 78.13 | 89.37 |
| **gru** | 81.64 | 55.80 | 84.02 | 85.44 | 9.38 | 63.71 | 85.31 | 91.98 |
| **lstm** | 82.09 | 53.29 | 84.60 | 84.00 | 61.33 | 9.44 | 83.69 | 90.75 |
| **b_lstm** | 71.98 | 56.07 | 71.73 | 70.62 | 68.47 | 63.36 | 16.80 | 87.28 |
| **trans** | 82.02 | 73.04 | 87.95 | 86.48 | 84.62 | 84.28 | 87.68 | 57.99 |



(a) Benign audio samples. (b) Adversarial audios generated with GA. (c) Adversarial audios generated with RL.

**Figure 6: KWS accuracy of the conventional model (without any additional filter), and models with Mel and LPC filters.**

**Table 7: Adversarial Robustness of transformer model with LPC and Mel Filter towards Carlini-Wagner attack.**

| No Filter | LPC Filter | Mel Filter |
|---|---|---|
| 10.33 | 87.67 | **91.22** |

From the above evaluation, it is evident that applying input transformations to non-transformer models does not yield good adversarial robustness. Transformer model alone also does not help against GA and CW attack. Hence, combination of both input transformation and choice of robust model is critical for SpotOn.

## 5 SPOTON: RUNTIME DETECTION OF ATTACKER QUERIES

In this section, we discuss the attack detection module of SpotOn. This module is responsible for ensuring the IoT device can not be used by an attacker to generate the attack samples.

For query-based attacks, an attacker repeatedly queries a target model. The attacker starts with a benign audio that he wants to make adversarial by perturbing it iteratively, checking the output of each intermediate audio by sending it to the target model. The output of the query determines the next perturbation to be added. This process continues till he obtains a perturbed audio which is being mis-classified by the model. In this process, attacker sends hundreds to thousands of queries to the model. To prevent such type of attacks, there should be a mechanism to differentiate between legitimate audios and attack queries. The user should receive the model output for the legitimate audio while as an attacker should not receive any output for the attack query. We can essentially stop the attack from occurring if we prevent the attack queries from reaching the model.

In a white-box setting, where an attacker has perfect knowledge of the model architecture and its parameters, an attacker generates an adversarial audio by directly interacting with the victim model. Such type of attacks can be prevented by concealing the model inside secure enclaves. Schemes like [10, 36, 46, 52, 58] protect the model by keeping either the whole model or its critical layers inside the secure enclaves. These secure enclaves prevent unauthorized access from untrusted parties besides ensuring confidentiality of the intermediate results.

In contrast, black-box attacks allow the attacker to conduct query-based attacks by querying the model to retrieve either the whole classification probability vector or just the classification label. Thus, although the attacker has no knowledge of the model, he still succeeds in performing the attack by receiving the model output to his attack queries. We therefore propose SpotOn, a detection mechanism to detect attack queries against DNN models. SpotOn not only stops the attack from happening, but also aids in recovery from the attack. We next discuss the steps involved in the detection of attack queries. The detection procedure consists of two steps, ❶ computing fingerprints of incoming audio and ❷ comparing and matching fingerprints.

**Audio Fingerprints:** Similar to how fingerprints are used to identify people, audio fingerprinting is a technique for matching or identifying an audio based on its distinctive features. It entails establishing a fingerprint of the audio data that can be compared to a database of already-existing fingerprints to identify matches. Numerous works [7, 26, 33, 64] on audio fingerprinting exist in literature. We choose Olaf [60] as it is specifically designed for embedded platforms. Explaining the working of Olaf is out of scope of this paper.

**Computing and Matching Fingerprints:** Spoton computes fingerprint of each incoming audio before feeding it to the model. The fingerprint comprises of a number of fingerprint objects. These objecs are secure-one way hashes and therefore not easy to reverse. Since the attacker builds his attack by making changes to successive attack queries, most parts of the consecutive attack queries remain the same. We aim to capture this similarity among attack queries.

After extracting the fingerprints from an audio, we compare them with the existing fingerprints. A match of certain fingerprint objects will detect an audio as an attack query. The amount of match is determined by a threshold value. We experimented with a number of threshold settings and discovered that 0.1 worked best. As the attack queries discovered using this approach won't be sent to the model for inference, the attacker's ability to launch the attack will be hindered. For legitimate audios, we find all the fingerprint objects are distinct. Hence they are successfully forwarded to the model for inference.

We evaluate the above approach on GA attack. GA attack with an attack success rate of 87% needs 240 queries on an average to generate an adversarial example. We take 10 benign audios from each keyword class and generate adversarial examples from each of them. Table 8 shows the results. We find that 96% of the attack queries are getting detected within initial 1-2 queries.

**Table 8: SpotOn's Detection results**

| | Without Prevention | | With Prevention | |
|---|---|---|---|---|
| Attack | Attack Success Rate | Average #attack queries | Attack Detection rate | Average queries to detect |
| *GA* | 87% | 240 | 99% | 5 |

By blocking the model output for an adversary, SpotOn effectively prevents an adversary from crafting adversarial samples. Thus, SpotOn not only helps in producing correct classification output for adversarial examples, but also prevents device mis-use.

## 6 SPOTON: OPTIMIZATIONS FOR IOT PLATFORMS

SpotOn strengthens the KWS pipeline in two ways: (1) transforming the input audio with signal-processing based filters, and (2) by bolstering the model with the transformer architecture. Existing transformer-based models are large in size and computationally heavy. For example, the Keyword Transformer (KWT) model [11], inspired by the Vision Transformer (ViT) from the image classification task, contains 5.5 million model parameters, requiring 28 MB of space to be stored on the disk. It fits the Raspberry PI 4 IoT platform (hardware specifications given in Table 1), with 4 GB RAM and support for SD card of any arbitrary size. On wearable IoT platforms like Sensortile with only 1 MB of flash memory for model storage and 128 KB of RAM, the KWT model will not even fit on the device, let alone run! As SpotOn strengthened KWS pipeline is designed to run on resource-constrained IoT platforms, we next describe several optimizations we implement in SpotOn for practical deployment on resource-constrained IoT platforms.

### 6.1 Model Optimization for IoT Deployment

We apply several optimizations to the original KWT model with the aim to fit the optimized model within the limited RAM and flash size of a low-end IoT device and improve the KWS runtime latency. SpotOn equipped with the optimized model should be able to process streaming audio data on a device with a slow CPU processor, and with hundreds of KBs of RAM and flash.

❶ **Smaller model architecture:** The KWT model inspired by the ViT model, has three self-attention heads per layer. Each attention head consists of a MLP layer with a significantly high dimension. The output of the three heads is aggregated and passed to the next layer. However, we observe that the voice data has a much lower variability than image data, i.e., the images of the same object can differ significantly while the audio samples of the same keyword have similar spectrograms. This makes aggregating outputs from three heads redundant, as shown in the spectrograms in Figure 7, where the three spectrograms for the three self-attention heads of the first layer of the KWT model show a very similar pattern. We, therefore, reduce the number of attention heads in SpotOn.

❷ **Layer fusion:** Once the model has been trained, dead operations are eliminated and successive expensive computations are fused to

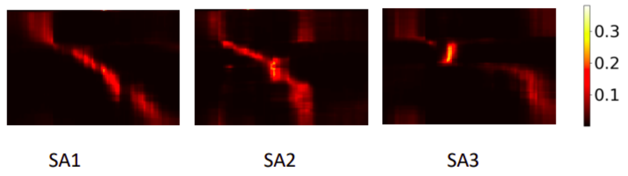


**Figure 7: Spectrogram outputs corresponding to "Stop" observed for the first three attention heads in the KWT model.**

one faster combined operation. This reduces runtime latency without affecting accuracy.

❸ **Weight pruning:** This technique has been inspired by the human brain. Once the human brain has developed, it cuts connections that have insignificant weights, in order to save on computation. Similarly, once the model has been trained, we prune down weights of magnitude below a threshold, which have an insignificant impact on the model output.

❹ **Integer quantization:** The KWT model is initially trained using input data in "float32" dtype, with the model weights also in "float32" dtype. Post-training, we quantize the model weights to "int8" dtype, which reduces the storage requirement of each parameter of the model by 4x. Quantizing both weights and inputs also reduces computational requirements. As model weights are fixed post-training, the range of weights is known. The range of inputs is estimated using a representative dataset, which samples about 100 examples from the dataset. This range estimation helps in maintaining model accuracy post-quantization, as the quantized values are still able to express the significant range of both weights and inputs.

❺ **Kernel mapping:** Hardware vendors release libraries of kernels that are optimized for faster execution on their processors. For example, ARM has released ARMNN and CMSIS-NN libraries for Cortex-A application processors and Cortex-M micro-controllers respectively, for faster execution of linear algebra and other operators needed for neural networks. We map the functions of our optimized model to these library functions, to take full advantage of the hardware resources.

Specifically, we use the TensorFlow Lite Micro (TFLiteMicro) library that uses the ARMNN or CMSIS-NN library under the hood, depending on the deployment platform. We implement several new operators for the KWT model using existing library functions. For example, the operator `tf.image.extract_patches` is implemented using `space_to_batch_nd`, `reshape`, `split`, `stack`, and `squeeze`. `space_to_batch_nd` is implemented using `transpose` and `reshape`. `transpose` is implemented using `strided_slices` and `reshape`.

Table 9 shows the model size and inference latencies with and without optimizations. Trans is the original KWT model which is 28 MB in size and takes 584 msecs to run on Raspberry PI 4. However, this model cannot fit Sensortile, so the table has missing values there. The optimized transformer model (Trans Opt.) takes 3.3 MB for PI 4 and only 500 KB for Sensortile. The difference between the sizes of the Trans Opt. model between the two platforms comes from the differences in their underlying software frameworks.

The Trans Opt. model processes 1 second of input audio, in 115 msecs on the PI 4 which has 1.5 GHz processor clock and quad-core Cortex A-72 processors, and in 5 seconds on Sensortile which has

**Table 9: Model Size and Latency.**

| | Size (MB) | | Latency (ms) | |
|---|---|---|---|---|
| | Rpi | Sensortile | Rpi | Sensortile |
| Trans | 28 | - | 584 | - |
| Trans Opt. | 3.30 | 0.5 | 115 | 5226 |

80 MHz processor clock and single core Cortex-M4 processor. As these models have to cope with streaming voice data without losing any samples, these inference times dictate how much buffer will be needed to hold incoming audio samples while ML processing is going on. For PI 4, ping pong buffers of size two is enough, where each buffer holds 1 second of incoming audio. While processing one buffer with Trans Opt., another buffer stores incoming samples. For Sensortile, six such buffers will be needed. As one second of audio data takes only 32 KB to be stored in int8 format, both PI 4 and Sensortile has enough storage to not miss any samples.

SpotOn equipped with the Trans Opt. model can therefore process streaming data on both large and tiny wearable IoT platforms after our careful optimizations. Raspberry PI 4 sees negligible latency to get classification output, while Sensortile perceives a slight delay. We will further improve this latency perceptibility of SpotOn on wearable platforms in the future, with an additional focus on heating and energy. Moreover, as shown in Appendix C, these optimizations marginally reduces the adversarial robustness.

## 6.2 SpotOn Runtime

Given SpotOn strengthened KWS has to run on resource-constrained IoT platforms, we need to analyze the defence's processing time for streaming audio data. In addition to the computations performed at runtime by the transformer model, SpotOn's detection module and the component responsible for applying the Mel filter to the input audio also add to the runtime computations, resulting in an increase in runtime latency. We next measure the end-to-end runtime latency of SpotOn on IoT platforms.

We implement and evaluate SpotOn on Raspberry Pi 4, which is is analogous to a smart home system. The run-time breakup between the detection module (550ms), Mel filter (423 msecs) and the transformer model (115 msecs) is given in Table 10. However, as detection and recovery modules operate concurrently in SpotOn, the full operation is completed in 800 ms. Thus, one second of audio input takes 800ms on average, to be processed by SpotOn. The processing time is significantly less than the buffering time, and therefore guarantees no sample will be missed even with some standard deviation in processing time (40 msecs over 100 runs, as shown in Table 10).

**Table 10: SpotOn runtimes**

| | Latency(ms) | Std. Deviation(ms) |
|---|---|---|
| Mel Filter | 423 | 64 |
| Trans Model | 115 | 21 |
| Detection Module | 550 | 12 |
| SpotOn | 800 | 40 |

SpotOn thus detects keywords on Raspberry Pi platform with a reasonable accuracy in the absence of any attack, as well as in presence of AML attacks, with a 800 msecs processing time for

1 second of incoming audio. It thus perfectly meets the desired values of the metrics we defined for our system, *benign accuracy*, *adversarial accuracy* and *runtime latency*.

Devices like Alexa and Siri can therefore be effectively protected against adversarial ML attacks using our defence pipeline which comprises of input audio filtering and transformer-based KWS model besides an attack detection module.

## 7 CAN RUN-TIME DEFENCES BE AVOIDED?

This section discusses additional defence strategies that are employed during model training phase. By training the model in a specific way, these defences aim to harden the model and increase its adversarial robustness. Such defences have the advantage of not requiring IoT devices to execute a separate defence module, which otherwise would increase runtime latency.

### 7.1 Adversarial Training

Adversarial training is one of the most effective defence strategies against AML attacks in the computer vision literature [47, 62]. The goal is to expose the victim ML model to adversarial examples during training itself, so that the detrimental effects of attacks at runtime are reduced. This is achieved by training the model with adversarial examples, assuming the noise generation model used by the attacker can be reproduced by the defenders. By learning the features from the adversarial training data, the ML model is expected to behave more robustly in an attack scenario.

**Limitations of Adversarial training:** A drawback of adversarial training is that it necessitates having enough adversarial audios for training as well as knowledge of the specifics of the attack strategy. It is also unable to defend against new adversarial attacks.

### 7.2 Key-Based Feature Permutation

This defence is motivated by the work of Abdullah et al. [4]. In their work, they claim that attack samples generated from optimization attacks do not exhibit transferability even across models trained with similar configurations. Thus, attack audios generated from one model cannot fool another model trained with identical setup (same architecture, hyperparameters, training data, random seed). We leverage this limitation of optimization attacks to build a potential defence.

In this defence, the features extracted from the received audio sample are permuted before they are used for classification. This permutation is done using a secret key which is known only to the device and possibly the vendor supplying the device. The device performs the same permutation during the training as well as operation phase of the ML model, i.e., the model is trained with features permuted using the unique key, and the same key is used for permuting the features at run-time for inference. Training using the unique key results in a unique set of model parameters. Thus, each device has a unique model and a unique way of permuting features. This mitigates the issue of the adaptive attacker gaining any advantage after obtaining knowledge about the defence mechanism. The workflow of this approach is shown in Figure 8

The number of extracted MFCC features is equal to the product of two parameters viz. the number of sliding filters and the number of Discrete Cosine Transform (DCT) coefficients computed for each filter. In this work, we retain the sliding filters in their original form while permuting the coefficients computed by each filter. We highlight that the same permutation mapping must be utilized in each filter to maintain the spectral correlation. Hence, given the number of coefficients per filter as $n$, the number of possible permutations is $n!$. With the typical value of $n = 40$, the feature permutation practically prevents the attacker from guessing the model parameters of the target device.

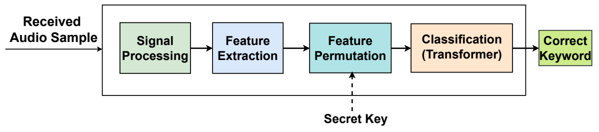


**Figure 8: KWS pipeline with model strengthened with feature permutation.**

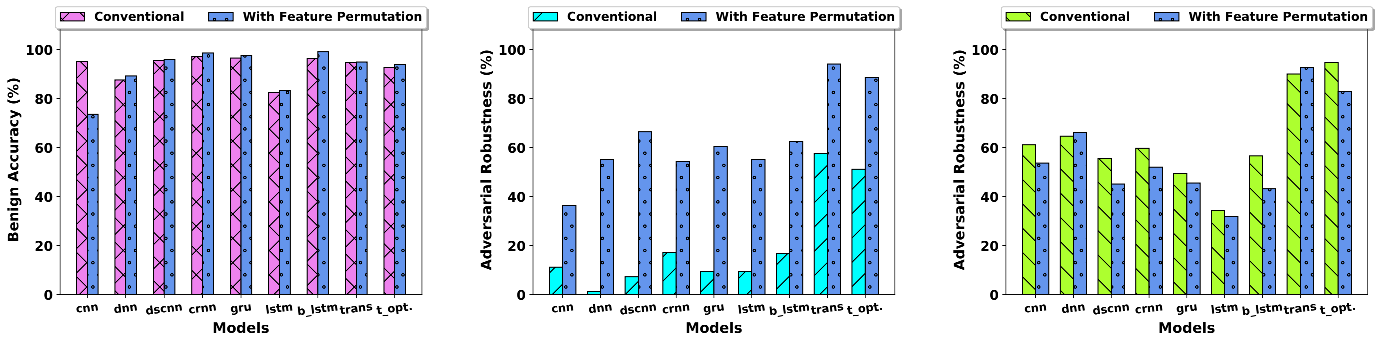
**How to permute MFCC features?** There could be multiple ways to permute the extracted MFCC features, but not all of them provide the desired values of benign accuracy and adversarial robustness. One possible way is to permute the entire MFCC feature vector. With this approach, for $n$ features, $n!$ permutations are possible. By permuting the features in this way, we may produce a vast number of unique models. The spectral correlation in the speech features is, however, destroyed as features from nearby frames become distant. We have observed poor benign accuracy as well as poor adversarial robustness with such permutation (results omitted for space constraints). Thus, although a large number of unique permutations are possible with this approach, there exists an undesirable trade-off.

We keep the temporal windows of the MFCC features constant, and shuffles the features within each window. This maintains the larger temporal ordering within the audio frame, with shuffling done at smaller time-scales. This approach thereby balances the requirement for a significant number of available unique permutations, while maintaining high benign accuracy and adversarial robustness.

**Importance of Feature Permutation** We finally analyze the key underlying hypothesis of this defence framework: that the adversarial samples crafted by querying a model trained with features permuted by a secret key $k$ will not be very effective against a model trained with features permuted with a secret key other than $k$. Figure 9 shows benign accuracy and adversarial robustness with conventional KWS models, where input MFCC features are not permuted, vs. models with MFCC features permuted. Benign accuracies and RL adversarial robustness remain intact with feature permutation. GA adversarial robustness greatly improves with feature permutation, for all KWS models.

We specifically evaluate the effectiveness of the feature permutation mechanism on transformer-based KWS models. We train five transformer-based KWS models, each using features permuted by a unique key, $k_i$, where $i \in 1, 5$. These trained models are further used to generate the adversarial samples using the two attacks. This way, we obtain five sets of adversarial samples for each of the two attacks. Finally, the adversarial samples generated with a particular model are fed to the other four models to evaluate the adversarial accuracy.

The benign accuracy of the five models along with the conventional model without any feature permutation is listed in Table 11. We observe that the feature permutation does not adversely affect the benign accuracy of the transformer model.



(a) Benign audio samples. (b) GA adversarial audio samples. (c) RL adversarial audio samples.

**Figure 9: KWS accuracy with and without feature permutation.**

**Table 11: Benign accuracy of Transformers using different keys.**

| Conventional Model | $k_1$ | $k_2$ | $k_3$ | $k_4$ | $k_5$ |
|---|---|---|---|---|---|
| 94.7 | 94.69 | 94.16 | 94.9 | 94.5 | 93.06 |

**Table 12: Adversarial robustness of Transformers using different keys against the GA attack.**

| | Target Model | | | | |
|---|---|---|---|---|---|
| Source Model | $k_1$ | $k_2$ | $k_3$ | $k_4$ | $k_5$ |
| $k_1$ | 73.13 | 85.82 | 91.28 | 86.04 | 92.00 |
| $k_2$ | 89.98 | 54.95 | 93.44 | 88.66 | 94.08 |
| $k_3$ | 90.08 | 85.77 | 73.13 | 84.84 | 91.33 |
| $k_4$ | 92.66 | 88.00 | 92.51 | 60.60 | 93.2 |
| $k_5$ | 89.00 | 85.04 | 89.77 | 84.73 | 76.33 |

**Table 13: Adversarial robustness of Transformers using different keys against the RL attack.**

| | Target Model | | | | |
|---|---|---|---|---|---|
| Source Model | $k_1$ | $k_2$ | $k_3$ | $k_4$ | $k_5$ |
| $k_1$ | 88.68 | 82.97 | 92.25 | 84.56 | 85.11 |
| $k_2$ | 89.32 | 84.77 | 92.64 | 85.12 | 90.08 |
| $k_3$ | 89.24 | 85.18 | 92.5 | 85.08 | 90.43 |
| $k_4$ | 88.72 | 83.56 | 92.41 | 84.88 | 85.98 |
| $k_5$ | 89.43 | 84.1 | 92.74 | 85.32 | 85.67 |

**Table 14: Adversarial robustness of Transformers using different keys against the CW attack.**

| | Target Model | | | |
|---|---|---|---|---|
| Source Model | *k1* | *k2* | *k3* | *k4* |
| *k1* | 16.66 | 18 | 19.77 | 30.66 |
| *k2* | 15.33 | 14.77 | 17.33 | 27.77 |
| *k3* | 16.44 | 14.77 | 20.22 | 29.33 |

The results corresponding to the GA, RL and CW attacks are shown in Table 12, Table 13, and Table 14 respectively. It is evident from these tables that in case of GA and RL attack, adversarial examples generated using a particular permutation of the MFCC features fed to the transformer model, are not able to fool the other models using different feature permutations. Thus, having a unique model in each device makes it difficult for the attacker to attack a specific device even if it gets access to a similar device. However, in case of CW attack, this defence does not seem promising as adversarial audios are transferable to models other than their source models as well.

**Challenges:** We next discuss the challenges of this defence scheme:
**❶ Device specific model training:** The proposed idea demands every device to run a unique version of the KWS model. This requires the vendor to train every single model separately with a unique key. Upon training a transformer model with a sample random shuffle of MFCC features, 90% accuracy is reached by the model within 500 epochs and 26 minutes of training time. Thus the per-device model training overhead is reasonable. We try to overcome this overhead by using SMT solvers like Gurobi [30] and Z3 [21]. Based upon the recent advances in DNN verification [28], we use these solvers to generate new models without actual re-training. The goal is to obtain a new model by specifying a verification query on a model which is satisfiable only when the original model can be modified in a desired way. By specifying different constraints each time, the solvers alter the original model and produce a new model within few seconds. However, training a neural network with an SMT solver has its own challenges [1]. We will therefore optimize further with fine-tuning with pre-trained models, as part of our future work.
**❷ Not applicable to all attacks:** This defence does not seem applicable to all attacks as is evident from Table 14. Therefore, for white-box attacks, more insights are needed.

## 8 CONCLUSION

Although the voice-based KWS is becoming increasing prevalent, the existing literature fails to provide practical mechanisms for enhancing the robustness of KWS against adversarial ML attacks on resource-constrained IoT devices. In this paper, we propose a system SpotOn which provides reasonable benign accuracy and adversarial robustness on IoT devices. SpotOn uses only a 3MB ML model and runs within 800ms on an IoT platform. Our study provides the intuition and paves the path for the development of novel signal processing techniques and ML models not only from the perspective to provide robustness against adversarial ML attacks, but also to facilitate realtime speech processing on IoT devices.

## REFERENCES

[1] 2023. [Online]. Available: https://www.cs.utexas.edu/~bornholt/post/nnsmt.html. [Accessed: 25 Jan 2023]..

[2] Mahdieh Abbasi and Christian Gagné. 2017. Robustness to Adversarial Examples through an Ensemble of Specialists. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=S1cYxlSFx

[3] Hadi Abdullah, Washington Garcia, Christian Peeters, Patrick Traynor, Kevin RB Butler, and Joseph Wilson. 2019. Practical hidden voice attacks against speech and speaker recognition systems. In *Network and Distributed Systems Security (NDSS) Symposium*.

[4] Hadi Abdullah, Aditya Karlekar, Vincent Bindschaedler, and Patrick Traynor. 2021. Demystifying limited adversarial transferability in automatic speech recognition systems. In *International Conference on Learning Representations (ICLR)*.

[5] Hadi Abdullah, Muhammad Sajidur Rahman, Washington Garcia, Kevin Warren, Anurag Swarnim Yadav, Tom Shrimpton, and Patrick Traynor. 2021. Hear" no evil", see" kenansville"*: Efficient and transferable black-box attacks on speech recognition and voice identification systems. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 712–729.

[6] Hadi Abdullah, Kevin Warren, Vincent Bindschaedler, Nicolas Papernot, and Patrick Traynor. 2021. Sok: The faults in our asrs: An overview of attacks against automatic speech recognition and speaker identification systems. In *2021 IEEE symposium on security and privacy (SP)*. IEEE, 730–747.

[7] Eric Allamanche, Jürgen Herre, Oliver Hellmuth, Bernhard Fröba, Throsten Kastner, and Markus Cremer. 2001. Content-based Identification of Audio Material Using MPEG-7 Low Level Description.. In *ISMIR*. Citeseer.

[8] Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava. 2018. Did you hear that? adversarial examples against automatic speech recognition. *arXiv preprint arXiv:1801.00554* (2018).

[9] Amazon. 2023. *Qualified AVS Solutions*. Retrieved December 9, 2023 from https://developer.amazon.com/en-US/alexa/solution-providers/dev-kits

[10] Sebastian P Bayerl, Tommaso Frassetto, Patrick Jauernig, Korbinian Riedhammer, Ahmad-Reza Sadeghi, Thomas Schneider, Emmanuel Stapf, and Christian Weinert. 2020. Offline model guard: Secure and private ML on mobile devices. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 460–465.

[11] Axel Berg, Mark O'Connor, and Miguel Tairum Cruz. 2021. Keyword Transformer: A Self-Attention Model for Keyword Spotting. In *Interspeech*. ISCA, 4249–4253.

[12] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition* 84 (2018), 317–331.

[13] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden voice commands In 25th USENIX Security Symposium (USENIX Security 16). *Austin, TX* (2016), 1–20.

[14] Nicholas Carlini and David Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 1–7.

[15] Steven Chen, Nicholas Carlini, and David Wagner. 2020. Stateful detection of black-box adversarial attacks. In *Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence*. 30–39.

[16] Tao Chen, Longfei Shangguan, Zhenjiang Li, and Kyle Jamieson. 2020. Metamorph: Injecting inaudible commands into over-the-air voice controlled systems. In *Network and Distributed Systems Security (NDSS) Symposium*.

[17] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1724–1734. https://doi.org/10.3115/v1/D14-1179

[18] François Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1251–1258.

[19] ARM CMSIS-DSP. 2023. A software library, a suite of common signal processing functions for use on Cortex-M and Cortex-A processor based devices. [Online]. Available: https://www.keil.com/pack/doc/CMSIS/DSP/html/index.html.[Accessed: 26 June 2023]..

[20] Sina Däubener, Lea Schönherr, Asja Fischer, and Dorothea Kolossa. 2020. Detecting Adversarial Examples for Speech Recognition via Uncertainty Quantification. (2020).

[21] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems: 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings 14*. Springer, 337–340.

[22] Don Dennis, Chirag Pabbaraju, Harsha Vardhan Simhadri, and Prateek Jain. 2018. Multiple instance learning for efficient sequential data classification on resource-constrained devices. In *Advances in Neural Information Processing Systems*. 10953–10964.

[23] Ben Dickson. 2019. 4 reasons Google's on-device AI is very important. [Online]. Available: https://bdtechtalks.com/2019/05/13/google-assistant-on-device-machine-learning/.

[24] Tianyu Du, Shouling Ji, Jinfeng Li, Qinchen Gu, Ting Wang, and Raheem Beyah. 2020. Sirenattack: Generating adversarial audio for end-to-end acoustic systems. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. 357–369.

[25] Thorsten Eisenhofer, Lea Schönherr, Joel Frank, Lars Speckemeier, Dorothea Kolossa, and Thorsten Holz. 2021. Dompteur: Taming audio adversarial examples. In *30th USENIX Security Symposium (USENIX Security 21)*. 2309–2326.

[26] Daniel PW Ellis, Brian Whitman, and Alastair Porter. 2011. Echoprint: An open music identification service. (2011).

[27] Brian Feldman. 2019. Google's Most Interesting I/O Announcement Was a Totally Un-Google Move. [Online]. Available: https://nymag.com/intelligencer/2019/05/googles-assistant-can-now-fit-on-your-phone.html.

[28] Ben Goldberger, Guy Katz, Yossi Adi, and Joseph Keshet. 2020. Minimal Modifications of Deep Neural Networks using Verification.. In *LPAR*, Vol. 2020. 23rd.

[29] Yuan Gong, Boyang Li, Christian Poellabauer, and Yiyu Shi. 2019. Real-Time Adversarial Attacks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 4672–4680. https://doi.org/10.24963/ijcai.2019/649

[30] Zonghao Gu, Edward Rothberg, and Robert Bixby. 2012. Gurobi optimizer reference manual, version 5.0. *Gurobi Optimization Inc., Houston, USA* (2012).

[31] Chirag Gupta, Arun Sai Suggala, Ankit Goyal, Harsha Vardhan Simhadri, Bhargavi Paranjape, Ashish Kumar, Saurabh Goyal, Raghavendra Udupa, Manik Varma, and Prateek Jain. 2017. Protonn: Compressed and accurate knn for resource-scarce devices. In *International conference on machine learning*. PMLR, 1331–1340.

[32] Philipp Gysel, Jon Pimentel, Mohammad Motamedi, and Soheil Ghiasi. 2018. Ristretto: A Framework for Empirical Study of Resource-Efficient Inference in Convolutional Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* (2018). https://doi.org/10.1109/TNNLS.2018.2808319

[33] Jaap Haitsma and Ton Kalker. 2002. A highly robust audio fingerprinting system.. In *Ismir*, Vol. 2002. 107–115.

[34] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567* (2014).

[35] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[36] Jiahui Hou, Huiqi Liu, Yunxin Liu, Yu Wang, Peng-Jun Wan, and Xiang-Yang Li. 2021. Model Protection: Real-time privacy-preserving inference service for model privacy at the edge. *IEEE Transactions on Dependable and Secure Computing* 19, 6 (2021), 4270–4284.

[37] Shehzeen Hussain, Paarth Neekhara, Shlomo Dubnov, Julian McAuley, and Farinaz Koushanfar. 2021. WaveGuard: Understanding and Mitigating Audio Adversarial Examples. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*.

[38] Dan Iter, Jade Huang, and Mike Jermann. 2017. Generating adversarial examples for speech recognition. *Stanford Technical Report* (2017).

[39] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. 2019. PRADA: protecting against DNN model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 512–527.

[40] Ashish Kumar, Saurabh Goyal, and Manik Varma. 2017. Resource-efficient machine learning in 2 KB RAM for the internet of things. In *International Conference on Machine Learning*. 1935–1944.

[41] Aditya Kusupati, Manish Singh, Kush Bhatia, Ashish Kumar, Prateek Jain, and Manik Varma. 2018. FastGRNN: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. In *Advances in Neural Information Processing Systems*. 9017–9028.

[42] Hyun Kwon, Hyunsoo Yoon, and Ki-Woong Park. 2019. POSTER: Detecting audio adversarial example through audio modification. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 2521–2523.

[43] Huiying Li, Shawn Shan, Emily Wenger, Jiayun Zhang, Haitao Zheng, and Ben Y Zhao. 2020. Blacklight: Defending black-box adversarial attacks on deep neural networks. *arXiv preprint arXiv:2006.14042* (2020).

[44] Zhuohang Li, Yi Wu, Jian Liu, Yingying Chen, and Bo Yuan. 2020. Advpulse: Universal, synchronization-free, and targeted audio adversarial attacks via subsecond perturbations. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1121–1134.

[45] Heng Liu and Gregory Ditzler. 2020. Detecting adversarial audio via activation quantization error. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–7.

[46] Renju Liu, Luis Garcia, Zaoxing Liu, Botong Ou, and Mani Srivastava. 2021. Secdeep: Secure and performant on-device deep learning inference framework for mobile and iot devices. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*. 67–79.

[47] 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).

[48] Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, and William J Dally. 2017. Exploring the regularity of sparse structure in convolutional neural networks. *arXiv preprint arXiv:1705.08922* (2017).

[49] Mediatek. 2023. *MediaTek MT8512 featuring Amazon AZ1 Neural Edge processor*. Retrieved December 9, 2023 from https://www.mediatek.com/blog/amazon-az1-neural-edge-processor-powered-by-mediatek

[50] Mediatek. 2023. *MT8163V/A Highly integrated 64-bit quad-core tablet platform*. Retrieved December 9, 2023 from https://www.mediatek.com/products/tablets/mt8163

[51] Microsoft. 2022. Edge Machine Learning library. [Online]. Available: https://github.com/Microsoft/Ell. [Accessed: 1 July 2022]..

[52] Fan Mo, Ali Shahin Shamsabadi, Kleomenis Katevas, Soteris Demetriou, Ilias Leontiadis, Andrea Cavallaro, and Hamed Haddadi. 2020. Darknetz: towards model privacy at the edge using trusted execution environments. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. 161–174.

[53] Raphael Olivier, Bhiksha Raj, and Muhammad Shah. 2021. High-Frequency Adversarial Defense for Speech and Audio. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2995–2999.

[54] Picovoice. 2022. *Design, develop, and ship useful voice features. The end-to-end platform for embedding private voice AI into any software in a few lines of code*. Retrieved December 9, 2023 from https://picovoice.ai/

[55] Krishan Rajaratnam, Kunal Shah, and Jugal Kalita. 2018. Isolated and Ensemble Audio Preprocessing Methods for Detecting Adversarial Examples against Automatic Speech Recognition. In *Proceedings of the 30th Conference on Computational Linguistics and Speech Processing (ROCLING 2018)*. 16–30.

[56] Oindrila Saha, Aditya Kusupati, Harsha Vardhan Simhadri, Manik Varma, and Prateek Jain. 2020. RNNPool: Efficient Non-linear Pooling for RAM Constrained Inference. *arXiv preprint arXiv:2002.11921* (2020).

[57] Tara Sainath and Carolina Parada. 2015. Convolutional neural networks for small-footprint keyword spotting. (2015).

[58] Alexander Schlögl and Rainer Böhme. 2020. eNNclave: offline inference with model confidentiality. In *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*. 93–104.

[59] Lea Schönherr, Thorsten Eisenhofer, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. 2020. Imperio: Robust over-the-air adversarial examples for automatic speech recognition systems. In *Annual Computer Security Applications Conference*. 843–855.

[60] Joren Six. 2020. OLAF: Overly lightweight acoustic fingerprinting. In *21st International Society for Music Information Retrieval Conference (ISMIR 2020)*.

[61] STMicroelectronics. [n. d.]. SensorTile development kit. https://www.st.com/en/evaluation-tools/steval-stlkt01v1.html. [Online; accessed 02-Feb-2022].

[62] Florian Tramer and Dan Boneh. 2019. Adversarial training and robustness for multiple perturbations. *Advances in Neural Information Processing Systems* 32 (2019).

[63] 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.

[64] Avery Wang et al. 2003. An industrial strength audio search algorithm.. In *Ismir*, Vol. 2003. Washington, DC, 7–13.

[65] P. Warden. 2018. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *ArXiv e-prints* (April 2018). arXiv:1804.03209 [cs.CL] https://arxiv.org/abs/1804.03209

[66] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. 2017. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5687–5695.

[67] Zhuolin Yang, Bo Li, Pin-Yu Chen, and Dawn Song. 2018. Characterizing audio adversarial examples using temporal dependency. *arXiv preprint arXiv:1809.10875* (2018).

[68] Shuochao Yao, Yiran Zhao, Aston Zhang, Lu Su, and Tarek Abdelzaher. 2017. DeepIoT: Compressing deep neural network structures for sensing systems with a compressor-critic framework. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. 1–14.

[69] Shun-Zheng Yu. 2021. Explicit Duration Recurrent Networks. *IEEE Transactions on Neural Networks and Learning Systems* (2021), 1–11. https://doi.org/10.1109/TNNLS.2021.3051019

## A GOOGLE SPEECH DATASET

Table 15 presents the details of the dataset used for experiments in this paper.

**Table 15: Utilized keywords from the Google speech dataset.**

| Keyword | Total Samples | Samples for Training | Samples for Testing |
|---|---|---|---|
| Yes | 4,044 | 3,000 | 1044 |
| No | 3,941 | 3,000 | 941 |
| Up | 3,723 | 3,000 | 723 |
| Down | 3,917 | 3,000 | 917 |
| Left | 3,801 | 3,000 | 801 |
| Right | 3,778 | 3,000 | 778 |
| On | 3,845 | 3,000 | 845 |
| Off | 3,745 | 3,000 | 745 |
| Stop | 3,872 | 3,000 | 872 |
| Go | 3,880 | 3,000 | 880 |
| **Total** | 38,546 | 30,000 | 8,546 |

## B MACHINE LEARNING MODELS

Table 17 presents the architecture details of the different neural networks used for experiments in this paper.

❶ **Deep Neural Network (DNN):** It is made up of Fully Connected (FC) layers. Each layer is followed by a rectified linear unit (RELU) based activation function. The output consists of a softmax layer that outputs probabilities.

❷ **Convolutional Neural Network (CNN):** It comprises of convolutional layers followed by RELU and a max-pooling layer, to reduce feature dimensionality. To exploit the local spatial and temporal features, it treats the speech input as an image and performs 2D convolutions over it.

❸ **Depth-wise Separable Convolutional Neural Network (DSCNN):** It convolves the input feature map with a 2D filter which is followed by a point-wise convolution [18]. This reduces the parameters and the number of computations. The output is obtained through pooling and FC layers.

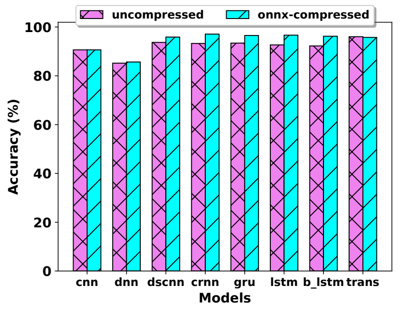
❹ **Recurrent Neural Network (RNN):** It feeds the output of the current processing step as the input to the next processing step. If the input sequence is long enough, RNNs fail to carry information from earlier steps to later ones [69]. To mitigate this limitation, LSTM and GRU cells have been developed [17, 35]. They have a gating mechanism using which they regulate the flow of information and capture long-term dependencies.

❺ **Convolutional Recurrent Neural Network (CRNN):** It comprises of a convolutional layer followed by an RNN layer and an FC layer. The RNN layer is responsible for encoding the signal and is made up of GRU cells. Using a mixture of CNN and RNN, CRNNs are able to exploit both local and global temporal dependencies in audio inputs.

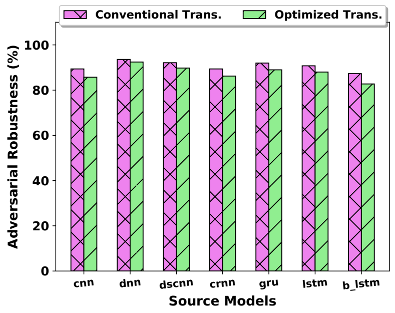
❻ **Transformer (TRANS):** It comprises of encoder and decoder units to compute self-attention (significance weights) across different parts of the audio inputs. It also utilizes a multi-layer perceptron (MLP) at the end of its pipeline.

**Table 16: SpotOn's Recovery results**

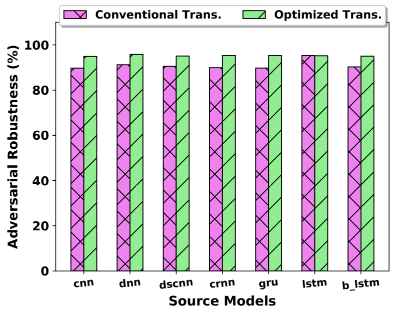
| Attack | GA | | RL | | CW | |
|---|---|---|---|---|---|---|
| Model | Transformer | Trans Opt. | Transformer | Trans Opt. | Transformer | Trans Opt. |
| Only Model | 57.99 | 51.17 | 99.21 | 94.73 | 10.33 | 8.0 |
| LPC+Model | 76.2 | 68.57 | 83.47 | 83.47 | 87.67 | 82.33 |
| Mel+Model | 82.2 | 73.88 | 78.58 | 73.49 | 91.22 | 86.67 |



(a) Benign audio samples.



(b) GA adversarial audio samples.



(c) RL adversarial audio samples.

**Figure 10: Benign accuracy of uncompressed and ONNX-compressed version of models in (a). Adversarial robustness with and without transformer model optimizations in (b) and (c).**

**Table 17: KWS model architectures and sizes. C represents the convolution layer and the numbers in parentheses correspond to the number of convolution features, convolution filter height, width, and stride along y-axis and x-axis directions, respectively. L denotes the low-rank linear layer. For LSTM and GRU models, the number in parentheses correspond to the number of memory elements. DSC denotes depthwise separable convolution layer and the number in the parentheses correspond to the number of features, kernel size and stride in both time and frequency axes.**

| Model | Architecture | Size |
|---|---|---|
| cnn | C(64,20,8,1,1)-C(64,10,4,1,1)-L(32)-FC(128) | 3.8MB |
| dnn | FC(144)-FC(144)-FC(144) | 314KB |
| dscnn | C(64,10,4,2,2)-DSC(64,3,1)-DSC(64,3,1)-DSC(64,3,1)-DSC(64,3,1)-AvgPool | 118KB |
| crnn | C(48,10,4,2,2)-GRU(60)-GRU(60)-FC(84) | 314KB |
| gru | GRU(154) | 317KB |
| lstm | LSTM(144), Projection(98) | 322KB |
| basic-lstm | LSTM(118) | 257KB |
| trans | heads(3),encoders(12),dim(192),mlp-dim(768) | 21.8MB |

## C PRACTICALITY VS. ADVERSARIAL ROBUSTNESS TRADE-OFF

Figure 10a shows the benign accuracies of KWS models converted to ONNX format. Trained models when converted to ONNX format for deployment on the PI 4 platform, undergo some optimizations discussed in Section 6.1 like layer fusion, dead layer removal, and integer quantizations. As seen from the plot, these optimizations do not seem to affect the benign accuracies of any model.

More interesting than benign accuracy, is the analysis of the effect of model optimizations on adversarial robustness. On one hand, we have highlighted the adversarial robustness of transformer architecture in Section 4.3, arising from self-attention heads. On the other hand, we have reduced self-attention heads in the transformer model in Section 6.1 for practical deployment on IoT platforms. We, therefore, need to evaluate *how much self-attention is necessary for adversarial robustness?* Is there a trade-off between adversarial robustness and practical deployment on IoT platforms?

Figure 10b and Figure 10c compare the original vs. optimized transformer models, for adversarial robustness against attacks generated with all non-transformer KWS models listed along the x-axis. Optimized transformer model (Trans Opt.) has the same adversarial robustness against RL attack samples generated with all non-transformer KWS models, as the conventional un-optimized Transformer model (Figure 10c). Trans Opt. possesses nearly the same adversarial robustness as the un-optimized transformer when it comes to GA attack samples produced by non-transformer models. However, as shown in Table 16, the adversarial robustness of transformer model post optimizations slightly drops from 99% to 95% for RL attack, 58 to 51% for GA attack and from 10% to 8% against CW attack. Using input-transformations in combination with optimized transformer model, can boost the robustness to 80%.

Our optimizations consider the characteristics of audio signals, that one attention head per transformer layer is enough for audio (Figure 7), while three have been used in the image classification domain. With such careful optimizations, optimized Transformer model marginally reduces the adversarial robustness. This is an extremely promising result, as in comparison to the original, un-optimized transformer model, these optimizations allow us to significantly reduce model size and runtime latency for practical IoT deployment while maintaining benign accuracy and adversarial robustness.