BlueSeer: AI-Driven Environment Detection via BLE Scans

Valentin Poirot [1,2], Laura Harms [1,2], Hendric Martens [1], Olaf Landsiedel [1,2]
[1] Kiel University, Germany
[2] Chalmers University of Technology, Sweden
{vpo,lah,ol}@informatik.uni-kiel.de,stu217810@mail.uni-kiel.de

ABSTRACT

IoT devices rely on environment detection to trigger specific actions, e.g., for headphones to adapt noise cancellation to the surroundings. While phones feature many sensors, from GNSS to cameras, small wearables must rely on the few energy-efficient components they already incorporate. In this paper, we demonstrate that a Bluetooth radio is the only component required to accurately classify environments and present BlueSeer, an environment-detection system that solely relies on received BLE packets and an embedded neural network. BlueSeer achieves an accuracy of up to 84% differentiating between 7 environments on resource-constrained devices, and requires only ~12 ms for inference on a 64 MHz microcontroller-unit.

KEYWORDS

environment detection, environment classification, embedded neural network, Bluetooth Low Energy, BLE

ACM Reference Format:
Valentin Poirot [1,2], Laura Harms [1,2], Hendric Martens [1], Olaf Landsiedel [1,2]. 2022. BlueSeer: AI-Driven Environment Detection via BLE Scans. In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC '22), July 10–14, 2022, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3489517.3530519

1 INTRODUCTION

Smartphones, wireless peripherals, and small wearables constantly accompany us in our daily life: at home, in transit, while shopping, or even at work. The ability to detect our surrounding environments, known as Environment Detection or Classification [15], is a desirable trait in mobile systems: a smartphone can automatically enter into silent mode when it detects that we enter a theater or airplane mode once we board a plane; wireless headphones can adapt their degree of noise cancellation to match the current environment: total cancellation in offices, but limited cancellation in streets to ensure that the user can still hear oncoming traffic and emergency vehicles; fitness trackers can distinguish between indoor and outdoor activities by checking the user's surroundings; and smart speakers can control their initial volume if they detect they are in public spaces. Environment detection also enables informed decisions for co-located systems: the performance of distance estimation and localization algorithms relying on radio signal propagation can suffer from multi-path fading indoors [2, 8], environment detection allows such algorithms to select the best propagation model based on the current environment. While environment detection can aid localization systems [13], it does not aim at pinpointing a device position within a given area; instead, it classifies the surrounding environment into general categories such as shopping center, office, home, or street.

Modern smartphone systems feature many sensors that we can use to infer surroundings, such as Global Navigation Satellite Systems (GNSS, e.g., GPS), cameras, microphones, or ultra-wideband radios. In contrast, small wearables such as fitness trackers or in-ear earphones often lack this luxury. While GNSS chips offer precise localization and, with access to maps, provide environment detection with high accuracy, they remain the most energy-hungry on-device sensors [1], and would drain an unacceptable share of the energy budget of a small IoT platform. Microphones are a more energy-friendly alternative to perform environment detection [15], but the presence of microphones on fitness trackers for the sake of environment detection raises concerns on privacy.

Yet, most embedded wearables incorporate one common component: a Bluetooth radio. With 4 billion new devices shipped in 2020 alone [4], Bluetooth and Bluetooth Low Energy (BLE) are the go-to solutions for wireless communication on the most modest IoT wearables. Although it seems incongruous at first that we can use a BLE radio to infer its surroundings, this paper demonstrates that we can classify environments with high accuracy solely from received BLE packets. Specifically, we show that the density, diversity, and dynamics of mobile devices, which we can infer from listening to their periodic packet broadcasts, form a wireless fingerprint that can be relied on to categorize surroundings. With this insight, we are able to provide a new approach to accurate and energy-efficient environment detection; any device, from the smallest BLE-enabled sensors up to smartphones, can accurately infers the environment by simply turning on its BLE radio periodically. If a device already scans for BLE transmissions, no additional radio-on time is required; the received packets are used as the basis for the classification.

Challenges. By default, many BLE devices announce their services and thereby their presence so that other devices can connect to them via packets called advertisements. BLE Advertisements contain the different services offered by each equipment, for example volume control for smart speakers or temperature for thermometers. The presence of specific services influences the classification: for example, more keyboards are found in offices than restaurants, more smart assistants are located in homes than in transport. The number of devices in proximity also discriminates between environments: we receive many more BLE signals in a busy street than deep in a forest. However, this wireless fingerprint highly evolves with the time of day: crowded transport at rush-hour shares more similarities with a concert venue than riding an empty bus. As the

DAC '22, July 10–14, 2022, San Francisco, CA, USA

ACM ISBN 978-1-4503-9142-9/22/07…$15.00
https://doi.org/10.1145/3489517.3530519

number of advertisements and the number of services within packets highly vary with time and place, feature engineering is required. We need to extract meaningful features over all received advertisements to be able to pass it as input to, e.g., a neural network classifier.

Approach. We present *BlueSeer*, an environment-detection system specifically tailored for resource-constrained IoT platforms. To work, BlueSeer only requires a BLE radio on the device: periodically, BlueSeer scans for BLE advertisements from nearby devices, performs feature extraction from the raw data, and employs an embedded neural network to predict the current surrounding environment. We show that the knowledge extracted from BLE packets is sufficient to infer the environment accurately.

Contributions. This paper makes the following contributions:

(1) We show that it is possible to categorize environments exclusively from received BLE advertisements;
(2) We present BlueSeer, an Environment Detection system able to classify environments solely using received BLE packets. BlueSeer distinguishes between 7 categories: home, office, shopping, transport, nature, street, and restaurant;
(3) We carry out extensive feature engineering and identify 23 features from BLE advertisements, ranging from number of devices in proximity and RSS measurements, to the diversity of offered services;
(4) We devise a neural network and show that its quantized, embedded version classifies its environment with up to 84% accuracy on a low-power platform with a 64 MHz MCU, and uses 65 KB of memory. We make BlueSeer's implementation and its dataset open-source[1].

Outline. The paper is structured as follows: §2 provides background on Bluetooth Low Energy advertisements, §3 dives into the design of BlueSeer, §4 provides an in-depth evaluation of BlueSeer and its embedded neural network, §5 summarizes the related literature and §6 concludes this paper.

2 BACKGROUND: BLUETOOTH LE

BLE. Introduced as part of Bluetooth 4.0 in 2010 [3], Bluetooth Low Energy (BLE) is a short-range wireless technology in the 2.4 GHz ISM band. BLE targets direct, one-hop communication and provides datarates of up to 2 Mbit/s within 10-50 meters, typically. It uses 40 2-MHz wide frequency channels, 37 reserved for connection exchanges, and 3 for advertisements and broadcasts.

Advertising. BLE has two operation modes: connected and non-connected mode. The non-connected mode is used to disclose the presence of connectable devices (for example after turning on headphones) or broadcast data to nearby devices (such as COVID contact tracing keys). The advertiser, such as a small wearable, operates by broadcasting an advertisement packet on all advertising channels (cf. Fig. 1) and repeats the packet pseudo-periodically at a fixed interval, plus a random delay to avoid collisions. On the receiving end, a scanner, such as a smartphone, scans the medium for advertisements by listening to a specific channel during a scan window. The scanner iterates over all advertising channels by switching channels after a period called scan interval. A device initiates a connection by responding to an advertisement: data exchange then proceed on the remaining 37 channels.

[1]Available at: github.com/ds-kiel/blueseer

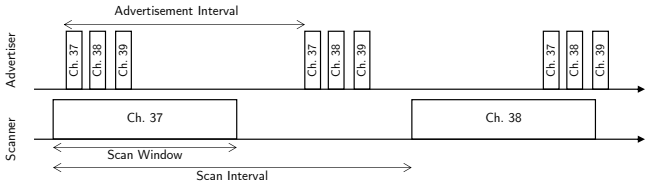


Figure 1: BLE Advertisements. The advertiser pseudo-periodically sends advertisement packets on all three advertising channels. The scanner periodically listens for advertisements.

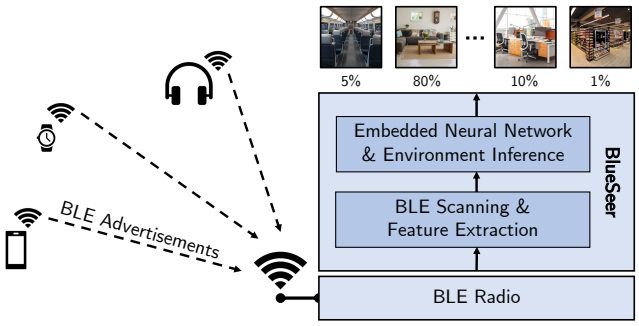


Figure 2: BlueSeer: System architecture. BlueSeer scans the wireless medium for BLE packets and extract features from them. An embedded neural network classifies the environment between 7 categories.

Advertising data. The payload of BLE advertisements consists of a list of optional fields called Advertising Data (AD). Advertisers include AD to define, e.g., their device name, address, the list of offered services (such as sound or heart rate sensor), or data related to its specific manufacturer. Each AD is identified by its universally unique identifier (UUID), defined in the standard [3]. For example, the COVID Exposure Notification AD has the UUID 0xFD6F [9]. The presence of specific AD fields (e.g., announcing keyboard capabilities), the variety of advertised services, as well as the variety of advertisements received within one scan form a wireless fingerprint that we use to infer the current environment.

3 DESIGN: BLUESEER

With BlueSeer, we demonstrate that a BLE radio is the only component required for a device to detect its environment. All BLE advertisements received by a device compose a wireless fingerprint of the surroundings. This fingerprint builds the basis for inferring the environment's category. We present the system architecture of BlueSeer in §3.1, dive deep into the data collection and feature extraction processes in §3.2, describe the embedded neural network used by BlueSeer in §3.3, and precise implementation-specific details in §3.4.

3.1 Overview

Wireless Fingerprints. Different environments often exhibit different characteristics: multiple wireless keyboards are often found nearby in offices, smart home assistants at home; buses driving around town pass by many people and devices for a short duration; and the occasional wireless speaker can be found in city parks when the weather allows. We show in this paper that the devices found within an environment, their density, variety, the motion of those devices as well as the mobility of the receiver, all form a unique composition specific to each environment. Further, since many of these devices voluntarily disclose their proximity and services via BLE advertisements, each environment has its own wireless fingerprint that we exploit for classifying the environment. Thus, the sole presence of a BLE radio is the only requirement for IoT platforms to classify their surroundings.

BlueSeer. BlueSeer consists of two main components: (1) the scanning and feature extraction block, which produces data for a classifier, and (2) the classification process that relies on an embedded neural network, as we depict in Fig. 2. Periodically, BlueSeer scans the wireless medium for BLE advertisements. From the raw packets' data, BlueSeer extracts features related to the device density, variety, as well as the dynamics of the environment (see §3.2). We then feed these features into an embedded neural network, whose weights are quantized to fit in the memory of small, constrained IoT platforms (see §3.3). Since many wearables do not feature constant internet connectivity, we cannot rely on a complex classifier on the cloud or nearby edge-device to infer the category and must rely on a memory-efficient model. With its neural network, BlueSeer discriminates between 7 different environments:

- Home (house, apartment)
- Office
- Shopping (such as supermarkets, malls)
- Transport (e.g., car, bus, train)
- Nature (city parks, forests, countryside)
- Street (both walking and standing)
- Restaurant

3.2 Feature Extraction

Raw data. Rather than pass the raw data to a classifier, we perform feature extraction on the received packets to produce 23 informative features that we pass onto the embedded neural network. As BLE scans return anything from a few results up to a hundred packets in crowded spaces such as restaurants, and the packet length evolves with the number of services advertised, raw packets are bad candidates as input features. In the following, we present a selection of the most notable features produced by BlueSeer.

Dynamic environments. Since the device's density and dynamics are markers of an environment, we extract the number of unique devices we encounter within a scan as our first feature. The number of devices that left our vicinity since the previous scan, which we call lost devices, and the number of new devices that we hear for the first time serve as our second and third features. These metrics represent both the density and dynamics of the environment.

During a single scan, a receiver might receive more than one packet from the same source since every advertiser chooses its own advertisement interval (see §2). This fact gives two kinds of information about our surroundings: (1) the type of devices in proximity, as low-power platforms tend to select long intervals to save energy; and (2) how long devices stay nearby: we should receive many packets from devices with short intervals unless they quickly leave our vicinity. Therefore, we measure the average interval between two transmissions from the same source, and how long devices stay in our vicinity on average, and use both as additional features.

Signal strength. Whenever the BLE radio receives a new advertisement, it records the Received Signal Strength (RSS). We use the RSS as a rough approximation of the distance between a sender and receiver [7]. Successive measures also produce a simple estimation of a device's motion. When combined, the RSS datapoints represent the estimated device density of the current environment. We collect the following statistics from raw RSS values and use them as features: the highest and lowest RSS measures received as well as the average RSS over all devices.

Device diversity. Along with the device density and environment dynamics, the device variety is an important factor in distinguishing between environments. While BLE lacks the device class categories used by Bluetooth Classic, we deduce approximate categories from the list of services advertised. For example, the TX Power AD exposes the transmit power used by the sender; larger devices such as laptops tend to transmit at high power, while battery-powered sensors rely on low levels to operate longer. We collect the following metrics as features: the lowest, highest, average, and count of TX Powers advertised by nearby devices to infer information on their types.

We can further take advantage of the diversity of AD options advertised by BLE packets. By counting how many unique services are offered, we can infer the diversity of devices. By counting the total number of services, we can infer the homogeneity. The presence of manufacturer data is also a marker of specific environments; devices from different manufacturers send specific AD, and different devices by the same manufacturer also send unique data. We extract the average manufacturer data length to use it as an additional marker for environment detection.

3.3 Embedded Neural Network

The feature extraction process produces a total of 23 unique features that encompass the device density, variety, the radio conditions, as well as the dynamics of the environment. As each scan generates the same number of features, we directly feed them as input to an embedded neural network.

Sliding-window. Yet, using the features from a single scan may lead to inaccuracies, especially in highly dynamic environments. For example, a car driving along a shopping street sees many advertisements, but receives almost no packets a few seconds later as it enters a residential district. We keep track of the last scans within a sliding-window and concatenate the result of several BLE scans together. We feed the resulting vector as input to BlueSeer's embedded neural network. Because we use several consecutive scans, the neural network's decisions are less subject to fluctuations in highly dynamic environments. Importantly, BlueSeer performs inference after each individual scan: BlueSeer does not require to refill the window with fresh scans between two inferences entirely; instead,

earlier scans remain part of the sliding window for a number of scans equal to its capacity. We set the sliding-window's capacity to the last 5 scans, see §4.2.

Neural architecture. With BlueSeer, we target constrained devices, such as fitness trackers or in-ear earphones that feature a <100 MHz MCU and ~200 KB memory. Thus, we devise a two-layer dense neural network: the input features are fed into (1) a 500-neuron dense layer with Rectified Linear Unit (ReLU); that feeds (2) the dense output layer with softmax function and 7 output classes (see §4.1). To ensure that the model's weights do not monopolize all memory, we employ quantization, fixing weights to integer representations [6]. Our neural network takes 65 KB of memory and requires ~12 ms to infer the environment on a 64 MHz MCU.

Collection and training. We collect 70 600 datapoints from 7 environments, using 62 000 for training and 8 600 for the test dataset for the final evaluation. For each environment, we collect data from different physical locations at different time, within the same city and in nearby locations. For example, for Home, the training set contains samples from 6 locations: three individual houses and three apartments. For Shopping, we feature 5 environments: three different grocery stores, one clothing store, and a shopping mall. The test dataset consists of physically different recordings: we do not separate one supermarket recording into training and test data, but collect two recordings each at a different time, one for training, one for testing. We train the network for 20 epochs, with an exponentially decreasing learning rate initially set to 0.01. Once the network is trained, we quantize its weights and retrain it for 10 additional epochs to mitigate the accuracy drop due to quantization.

3.4 Implementation

We implement BlueSeer for the Zephyr RTOS and use its open-source BLE stack implementation [19]. We use Tensorflow for training and employ quantization-aware retraining for the second training step to ensure that the quantized model is on-par with the original performance. We rely on Tensorflow Lite for Microcontrollers for on-device inference [6].

BlueSeer is platform-independent; we use the nRF52840 SoC (nRF52) to collect data and evaluate the performance of the embedded neural network. The nRF52 features a 64 MHz Cortex-M4 CPU with FPU, 256 KB of RAM, and a Bluetooth 5.2 radio supporting BLE. The nRF52 is a well-representative platform and depicts a performance comparable to commercial fitness trackers.

4 EVALUATION

In this section, we experimentally demonstrate the effectiveness of BlueSeer. We first evaluate the impact of the neural network architecture in terms of accuracy and memory usage. Then, we assess the importance of the features produced by the feature extraction process, as well as the impact of the number of scans used as input. Finally, we evaluate the overall performance of BlueSeer on unseen data and its on-device computation, memory, and energy footprint.

Metrics. We evaluate the following set of metrics: (1) Accuracy: Top-1 accuracy obtained from the quantized neural network; (2) Memory: Memory usage to store all weights and temporary computations, both in RAM and ROM (flash); (3) Energy: Energy consumption of running BlueSeer, on the embedded device, including neural network computation and BLE scanning; and (4) Compute-time: Execution time of the neural network on the resource-constrained device.

4.1 Neural Architecture

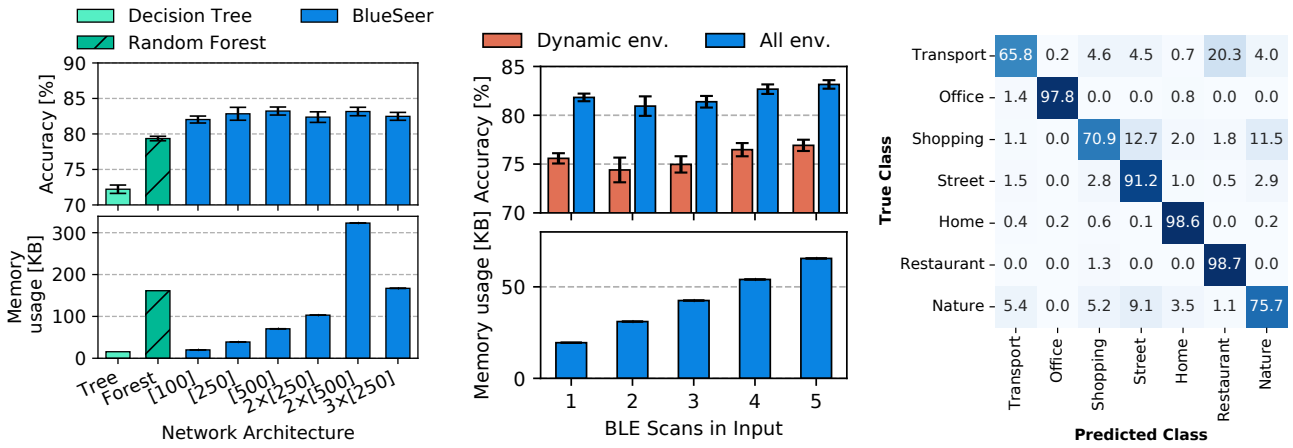
Scenario. We investigate the impact of the number of layers and neurons on the accuracy of the system and the memory consumed by the model. We evaluate different neural network architectures and compare them to decision trees (DT), a memory and compute-efficient machine learning alternative, as well as against random forests, that are known to improve accuracy compared to DTs while remaining compute-efficient. We limit the DT to a maximum depth of 40 and the random forest to contain a maximum of 10 concurrent DTs. We average results over 10 models using k-fold cross-validation.

Results. Fig. 3a depicts the accuracy and memory usage of different neural network models after quantization, as well as the performance of decisions trees and random forests. The baseline decision tree achieves 72.2% accuracy and consumes 15 KB of memory, while the random forest composed of ten DTs achieves 79.4% accuracy with 161 KB memory usage, roughly 10× the size of a single DT. An embedded neural network with a single hidden layer achieves 82%, 82.8%, and 83.2% for 100, 250, and 500 neurons, respectively. Memory-wise, these models consume 20 KB, 39 KB, and 70 KB, respectively, where 5 KB are assumed for intermediary results storage. Adding more layers does not improve accuracy further but induces a significant memory overhead: 82.4% accuracy and 103 KB of memory when using two layers of 250 neurons, 83.2% and 318 KB with 2 layers of 500 neurons, and 82.5% and 167 KB for three layers of 250 neurons. We select a neural network comprising one hidden layer of 500 neurons for BlueSeer. For the most memory-constrained hardware, the model with 100 neurons is the best trade-off between accuracy and memory.

4.2 Feature Analysis

Scenario: Features importance. The feature extraction process produces 23 distinct features representing device density, variety, and environment dynamics. We evaluate the importance of the produced features and how they affect the performance of BlueSeer. We withhold a subset of features from the training dataset and compare the achieved accuracy with the model using all features. For all models, we use a single BLE scan as input. We average results over 10 models using k-fold cross-validation.

Results. With all features present and one BLE scan, the baseline model achieves 81.5% accuracy. Removing features related to the environment's variety, such as the number of services, decreases the accuracy down to 77.6%, while removing RSS-related features drops the accuracy to 77.1%, and removing the TX Power information drops to 79.2%. Some redundancy is present in the feature set: removing the number of devices only decreases the accuracy to 80.7%, while removing the number of advertisements received achieves 80.8% accuracy. Device variety, represented by the available services and TX Power, as well as the environment dynamics, with the RSS measures, are important factors to distinguish between environments. The combination of all features provided by the feature extraction process enables the high accuracy of BlueSeer.



(a) Accuracy and memory costs of different models.

(b) Number of BLE scans as input.

(c) Confusion matrix.

Figure 3: Evaluating BlueSeer. (a) A single hidden layer with 500 neurons is sufficient to classify environments accurately and easily fits into memory-constrained devices. (b) By including multiple scans as input to the neural network, BlueSeer reduces the risks of fluctuation in highly-dynamic environments. (c) BlueSeer is able to accurately classify restaurant, home and office samples, but environments with high-mobility are harder to classify. The best out of 10 models achieves 85.5% accuracy.

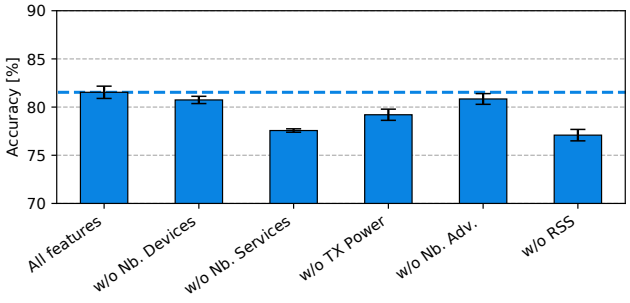


Figure 4: Feature analysis. The number of services and RSS measures play an important role in distinguishing environments. Some features provide redundancy in the input. The dotted line represents the accuracy when all features are present.

Scenario: Number of scans. We investigate how the number of BLE scans used as input to the model affects the overall accuracy, see §3.3. More scans should avoid fluctuations due to highly dynamic environments, but induce a larger input and slightly larger memory footprint. We average results over 10 models using k-fold cross-validation.

Results. Fig. 3b depicts the classification accuracy based on the number of BLE scans used as input. The model achieves 82%, 80.7%, 81.2%, 82.5%, and 83.2% accuracy for the input ranging from 1 up to 5 scans, respectively. As the number of scans increases, the model more accurately distinguishes between environments with high dynamics such as transport. Interestingly, the street and shopping categories perform slightly better with one scan than with two scans (street drops from 86% with one scan to 82% with two) before increasing again with three or more scans. Other environments (nature, home, restaurant) always benefit from more scans. Over all environments, it is more beneficial to include more scans; we therefore always include 5 scans for BlueSeer.

4.3 Overall Performance

Scenario: Per-class accuracy. We dissect the performance of BlueSeer's quantized neural network and look at the per-class accuracy on the test dataset. Each class in the test set comprises 1200 elements. We take the model that produces the best accuracy out of 10 trained models using k-fold cross-validation.

Results. Fig. 3c depicts the confusion matrix of the quantized model's inference over the test set. BlueSeer is able to accurately classify environments with low dynamics such as home, restaurant, and office with 98% or more accuracy. However, BlueSeer is less accurate when it comes to dynamic environments with devices in motion. Shopping shares many similarities with a busy street, where many passersby come and go. As transport covers trains, buses and cars, the class might be too general and could be split into sub-categories. Similarly, nature contains both city parks, where many people visit on sunny days, with forests, where meeting a passerby is less likely. Higher granularity in the categories could improve accuracy but would require more training data.

Scenario: On-device execution. We now measure BlueSeer's footprint on resource-constrained devices. We use the Tensorflow Lite for Microcontrollers for on-device inference. We measure the ROM and RAM used by the neural network and the library, the feature extraction and inference time, as well as the overall energy cost for BlueSeer.

Results. Table 1 summarizes all resources used up when performing BlueSeer's inference on the nRF52 SoC (cf. §3.4). BlueSeer

Table 1: On-device requirements for BlueSeer.

	Feat. Extr.	Model	Others	Total
Flash	1.8 KB	65 KB	46.2 KB (TFLite)	113 KB
RAM	11.9 KB	2 KB	1 KB	15 KB
CPU	<1 ms	~12 ms	3 sec (1 scan)	13 ms
Energy	6.4 µJ	111.3 µJ	56.8 mJ (1 scan)	57.9 mJ

uses 113 KB of storage: 65 KB to save the weights, 46 KB for Tensorflow Lite's library. 2 KB of RAM are reserved for dynamic allocation for the input, temporary results, and output, 1 KB for the library, and 12 KB for the packet parsing. It takes less than 1 ms to extract features and ~12 ms to run the inference step on the 64 MHz MCU. Energy-wise, BlueSeer requires 57.9 mJ to perform one scan and inference. Assuming BlueSeer executes one scan and inference every 10 sec and uses an AAA battery with a capacity of 800 mAh, we can run over 62 000 BlueSeer inferences. This represents over a week of constant operation on a single AAA battery.

5 RELATED WORK

Audio-based detection. Several works establish acoustic sensing as an accurate enabler for environment detection. Ma et al. rely on microphones and Hidden Markov Model classifiers to distinguish between 12 environments such as bus, car, street, and office, from 3-second long audio recordings and achieve up to 93% accuracy [15]. However, the authors do not discuss the problems of privacy arising from relying on microphones to infer surroundings. Heittola et al. represent audio fingerprints as histograms and compare new recordings with previous histograms to distinguish between ten environments [10]. Choi et al. combine microphone and camera inputs to detect the user position and activity [5]. Acoustic-based systems are also proven to improve Human Activity Recognition (HAR) [20].

Sensor-based detection. Accelerometers play a notable role in transportation mode detection, as well as HAR. Liang and Wang introduce a CNN to parse a smartphone's accelerometer data and distinguish which transport (such as bus, car, bike) the user is using [14]. Kern et al. distribute accelerometers over the body to perform HAR [12]. Sankaran et al. show that a barometer can also help distinguish between idle, walking, and in-vehicle users [17]. Yang et al. rely on channel state information to detect building occupancy with only a wifi radio [18]. However, they only detect if people are present in a room and how many, but do not detect the users' activities.

Wireless-enabled localization. Several works investigate the use of Bluetooth and BLE packets as a driver for localization and distance estimation [2, 21]. Bertuletti et al. demonstrate that RSS measures are noisy and lead to a 30% distance estimation error [2]. Zhuang et al. achieve <3m localization using BLE beacons [21]. Ultra-wideband radios are much more accurate and typically achieve <10 cm localization error, sometimes down to the centimeter [11, 16]. BlueSeer does not aim at solving localization but rather classifies the general environment of the device.

6 CONCLUSION

Environment detection allows devices to react to their environments: smartphones can automatically switch to silent mode when entering a theater, while headphones can adapt their noise cancellation to their surroundings. Although modern phones can rely on many sensors to infer their environments, such as GNSS chips and cameras, small IoT wearables lack this diversity and must rely on the few, energy-efficient components they already incorporate. This paper shows that a Bluetooth radio is the unique component required to classify the current environment with high accuracy: the BLE packets received by a device form a wireless fingerprint, unique enough to infer the correct environment. We present BlueSeer, an AI-driven environment-detection system specifically tailored to resource-constrained wearables: from BLE packets, BlueSeer extract 23 unique features that are fed to a quantized, embedded neural network. Periodically, BlueSeer scans the wireless medium for BLE advertisements and executes an embedded neural network to classify between 7 categories: home, office, shopping, transport, nature, street, and restaurant. We show that BlueSeer achieves up to 84% accuracy on resource-constrained devices, while requiring only 65 KB of memory, and takes ~12 ms to execute on a 64 MHz microcontroller-unit.

REFERENCES

[1] Fehmi Ben Abdesslem, Andrew Phillips, et al. 2009. Less is More: Energy-Efficient Mobile Sensing with Senseless. In *ACM MobiHeld*. 61–62.
[2] S. Bertuletti, A. Cereatti, et al. 2016. Indoor distance estimated from Bluetooth Low Energy signal strength: Comparison of regression models. In *IEEE SAS*. 1–5.
[3] Bluetooth SIG. 2019. Bluetooth Core Specification v5.2.
[4] Bluetooth SIG. 2021. Bluetooth 2021 Market Update.
[5] Woo-Hyun Choi, Seung-Il Kim, et al. 2011. Acoustic and visual signal based context awareness system for mobile application. In *IEEE ICCE*. 627–628.
[6] Robert David, Jared Duke, et al. 2021. TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems. In *MLSys*, Vol. 3. 800–811.
[7] Bernhard Etzlinger, Barbara Nußbaummüller, et al. 2021. Distance Estimation for BLE-based Contact Tracing – A Measurement Study. In *Wireless Days (WD)*.
[8] Chiara Falsi, Davide Dardari, et al. 2006. Time of arrival estimation for UWB localizers in realistic environments. *EURASIP J. on Advances in Signal Processing* 2006 (2006), 1–13.
[9] Google and Apple. 2020. Exposure Notification - Bluetooth Specification v1.2.
[10] Toni Heittola, Annamaria Mesaros, and Tuomas Virtanen. 2010. Audio context recognition using audio event histograms. In *European Signal Processing Conference*. 1272–1276.
[11] Benjamin Kempke, Pat Pannuto, et al. 2016. SurePoint: Exploiting Ultra Wideband Flooding and Diversity to Provide Robust, Scalable, High-Fidelity Indoor Localization. In *ACM SenSys*. 137–149.
[12] Nicky Kern, Bernt Schiele, et al. 2003. Multi-sensor Activity Context Detection for Wearable Computing. In *Ambient Intelligence*. Springer, 220–232.
[13] Patrick Lazik, Niranjini Rajagopal, et al. 2015. ALPS: A Bluetooth and Ultrasound Platform for Mapping and Localization. In *ACM SenSys*. 73–84.
[14] Xiaoyuan Liang and Guiling Wang. 2017. A Convolutional Neural Network for Transportation Mode Detection Based on Smartphone Platform. In *IEEE MASS*.
[15] Ling Ma, Ben Milner, et al. 2006. Acoustic Environment Classification. *ACM Trans. Speech Lang. Process.* 3, 2 (July 2006), 1–22.
[16] Yunfei Ma, Nicholas Selby, et al. 2017. Minding the Billions: Ultra-Wideband Localization for Deployed RFID Tags. In *ACM MobiCom*. 248–260.
[17] Kartik Sankaran, Minhui Zhu, et al. 2014. Using Mobile Phone Barometer for Low-Power Transportation Context Detection. In *ACM SenSys*. 191–205.
[18] Jianfei Yang, Han Zou, et al. 2018. Device-Free Occupant Activity Sensing Using WiFi-Enabled IoT Devices for Smart Homes. *IEEE Internet of Things J.* 5, 5 (2018).
[19] Zephyr Project. 2016. Zephyr Real-Time Operating System.
[20] Yi Zhan and Tadahiro Kuroda. 2014. Wearable sensor-based human activity recognition from environmental background sounds. *J. of Ambient Intelligence and Humanized Computing* 5, 1 (2014), 77–89.
[21] Yuan Zhuang, Jun Yang, et al. 2016. Smartphone-Based Indoor Localization with Bluetooth Low Energy Beacons. *Sensors* 16, 5 (2016).