Received May 14, 2020, accepted May 18, 2020, date of publication May 22, 2020, date of current version June 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2996576

An Embedded System for Collection and Real-Time Classification of a Tactile Dataset

OLCAY KURSUN AND AHMAD PATOOGHY, (Member, IEEE)
Department of Computer Science, University of Central Arkansas, Conway, AR 72035, USA

Corresponding author: Ahmad Patooghy (apatooghy@uca.edu)

This work was supported by the Arkansas INBRE Program with a grant from the National Institute of General Medical Sciences (NIGMS) and the National Institutes of Health under Grant P20 GM103429.

ABSTRACT Tactile perception of the material properties in real-time using tiny embedded systems is a challenging task and of grave importance for dexterous object manipulation such as robotics, prosthetics and augmented reality. As the psychophysical dimensions of the material properties cover a wide range of percepts, embedded tactile perception systems require efficient signal feature extraction and classification techniques to process signals collected by tactile sensors in real-time. For this purpose, we developed two embedded systems, one that served as a vibrotactile stimulator system and one that recorded and classified the vibrotactile signals collected by its sensors. The quality of the collected data was first verified offline using Fourier transform for feature extraction and then applying powerful machine learning classifiers such as support vector machines and neural networks. We implemented the proposed memory-less signal feature extraction method in order to achieve real-time processing as the data is being collected. The experimental results have shown that the proposed method significantly reduces the computational complexity of feature extraction and still has led to high classification accuracy even when fed to the less complex classifiers such as random forests that can be easily implemented on embedded systems. Finally, we have also shown that low-cost, highly accurate, and real-time tactile texture classification can be achieved using the proposed approach with an ensemble of sensors.

INDEX TERMS Signal processing algorithms, edge computing, tactile sensors, texture analysis, machine learning.

I. INTRODUCTION

With the recent advances in hardware design and VLSI technology, mobile embedded systems such as IoT and Edge devices have started to offer artificial intelligence (AI) services [1], [2]. Considerable scientific and technological efforts have been devoted to developing tactile sensing embedded systems with prospective applications in many fields, such as telehealth systems (e.g. remote examination, palpation, and surgery), smart prosthetics, and robotics with the sense of touch [3]–[9]. Pinker [10] described the complexity of human tactile capabilities as *"Think of lifting a milk carton. Too loose a grasp, and you drop it; too tight, and you crush it; and with some gentle rocking, you can even use the tugging on your fingertips as a gauge of how much milk is inside!"* Unlike humans that can effortlessly perform object perception and manipulation tasks; tactile enabled embedded systems are still primitive and need more research and development for tasks such as discriminating material properties (such as texture, hardness, roughness, and friction [11], [12], dexterous object manipulation/grasping [6], [13]–[15], slip detection [16], and so on). With high performance platforms, such as GPUs and AI-accelerators, raw signals coming from tactile sensors can be processed using high complexity time-frequency transforms and/or deep learning systems for feature extraction and AI processing [17]–[19]. However, for embedded and Edge/IoT systems the real-time processing of such streaming input signals of various modalities with different frequency spectrums is a big challenge [20]–[22]. As forwarding the streaming raw data to servers for cloud

The associate editor coordinating the review of this manuscript and approving it for publication was F. K. Wang.

processing might not be optimal due to one or more of the cost, energy consumption, and privacy issues, tactile enabled embedded systems must be equipped with efficient feature extraction and AI processing methods to be able to detect patterns of interest in real-time. The goal of feature extraction on embedded platforms is to transform the raw signals read from sensors into a more descriptive domain such that simpler AI processing algorithms can perform at accuracy levels that are close to those obtainable in high performance offline platforms. As tactile intelligence ultimately requires real-time processing of information collected by an array of various types of sensors with dense spatial arrangement for operating multiple points of contact [23], [24], in this paper, to process our experimental tactile dataset obtained via single contact point with the textured surface, the proposed feature extraction algorithm was designed to fit on a low-cost tiny embedded device. Ultimately, such methods co-designed under the resource constraints of the embedded and Edge/IoT devices are expected to serve better for more advanced tactile information processing tasks, such as classifying the aforementioned wider variety of texture classes and tactile experiences. Ideally, such a real-time feature extraction method should be memory-less (without buffering the signal values) and thus apply fewer memory accesses to fetch data; otherwise, dynamic power consumption will be higher due to the digital signal activities over the internal and external interconnects of the embedded device.

The deep learning approach has received increased attention due to the fact that the features it learns to extract in its early layers have similar properties to those extracted by biological neurons in the primary visual cortex (V1) [17], [18] (V1-like features include Gabor-like edge filters, gratings, and color blobs [25]). In our earlier work [24], [26], [27], we have developed a neurocomputational model; self-organized on natural images, the model learnt to extract features that closely matches structural and functional properties of Layer 4 of the cat primary visual cortex [26]. Kursun and Favorov [24] have shown that these features can be used to perform efficient texture image classification and as in deep convolutional neural networks (CNN) [17], [18], they can be used by subsequent cortical areas to develop gradually more complex and perceptually advanced features. Texture classification using tactile information can also benefit from extraction of such robust and perceptually salient general purpose features. As some of the prominent features that neurons in our CNN-like model learn correspond to average power in various frequency bands in their local receptive fields, in this study, we investigate the effectiveness of such bandpower features for real-time tactile information processing on embedded and Edge/IoT systems. Developing such intelligent embedded devices for tactile processing is a relatively new and emerging field with many general applications of tactile technology [7], [8], [13], [14], [28] and more specifically in neuroscience for real-time animal neurophysiological experiments that can utilize wearable tactile devices and developing diagnostic tests of neuropathies [27], [29]–[33].

To test the effectiveness of the proposed feature extraction method for tactile information processing, we developed an embedded system to record a pilot dataset [34] of tactile signals collected by a set of sensors. The dataset is first used offline to evaluate the discriminative power of these features in classifying various materials with different textures. Secondly, another embedded system is developed to evaluate the realtimeness of the tactile information classification based on the proposed CMB features. The embedded system continuously applies CMB in the time domain and computes the bandpowers of the input tactile signals in frequency bands of gradually increasing ranges. Even with a small number of bands, the embedded system achieves high classification accuracy by applying the proposed CMB feature extraction method and a Random Forest classifier (an ensemble of rule-based decision trees) in real-time.

The Fourier Transform (FT) of a signal can be used to decompose a signal to its frequency components and provides a very high resolution and lossless description (features) of the input signal. However, its computational and space complexity might overwhelm low-cost embedded systems. Comparable levels of signal classification accuracy is achievable using features extracted in the time domain as in the frequency domain [17]–[19]. In order to achieve real-time processing on tiny embedded systems, we can exploit the trade-offs between the descriptiveness of the representation and its computational complexity by performing the computations of the feature extraction in the time domain, instead of the frequency domain. These simpler features can be characterized as lossy and lower resolution approximations to the frequency/power spectrum of the signal [19], [35]. Computing the total power of a signal can be considered as one of the simplest features of such a low resolution approximation to spectral analysis. The total-power feature can be computed by summing the squares of all frequency harmonics in the wideband decomposed by the Fourier transform. Moreover, this sum can also be computed in the time domain per Parseval's theorem [36] (as described in more detail in Section II). That is, performing the Fourier transform is not required to compute the total power of a signal; instead, it can be computed by summing up the squares of the signal amplitudes across the given time window. In addition to the wideband computation suggested by Parseval's theorem, extending this idea further by computing the bandpowers in cumulative frequency bands will complement/enrich the set of features extracted in the time domain and help obtain finer approximations to the power spectrum. We called this method cumulative multi-bandpower (CMB) feature extraction method.

The contributions of this paper are the following:

- Design and development of a data collection and texture classification embedded system with tactile sensors (the collected dataset is available at [34]),
- Development of a novel feature extraction method for signal processing in embedded and IoT systems and comparison with the Fourier transform.

The rest of this paper is organized as follow. Section II discusses the required background including Parseval's theorem and the basics of exponential smoothing and low-pass filtering. Section III describes our proposed embedded system and the collected tactile dataset. The proposed CMB feature extraction method is introduced and discussed in Section IV. The real-time embedded classifier and the classification results based on the CMB features is presented in Section V. Finally, Section VI concludes the paper.

II. BACKGROUND

We review Parseval's theorem and its relation to the exponential smoothing as background for the proposed CMB (cumulative multi-bandpower) feature extraction method. Combining Parseval's theorem and the exponential smoothing technique, CMB leads to a simple yet efficient implementation on the proposed embedded system for tactile signal processing/classification. CMB is designed to avoid the computational and space complexity of the Fourier transform at runtime due to the computational/memory limitations of target embedded systems. We review Parseval's theorem and how it can be used to extract a set of simple yet powerful features (bandpowers in various frequency ranges) for use in embedded platforms.

Based on Parseval's relation, the average energy of a signal recording, $x[n]$, can be determined either by adding up the energy of the signal per each sample (i.e., $\sum \mid x[n] \mid^2$) at the time domain, or by taking the energy of signal in the frequency domain as summation of $\mid X(j\omega) \mid^2 / 2\pi$ as shown in Eq. 1.

$$\sum_{n=-\infty}^{+\infty} \mid x[n] \mid^2 = \frac{1}{2\pi} \int_{2\pi} \mid X(j\omega) \mid^2 d\omega \tag{1}$$

As Parseval's theorem relates the signal's total power in the time and frequency domains, it allows keeping track of the power in real-time without the need of keeping a sliding window of past signal samples for transforming into the frequency domain. Therefore, the theorem offers a method for staying in the time domain yet being able to do useful feature extraction in the frequency domain.

The exponential smoothing offers an efficient, memoryless approach to apply low-pass filter on the stream of samples of a given signal [36]. The exponential moving average filter on the signal $x[n]$ is defined as in Eq. 2:

$$\begin{aligned} y[0] &= x[0] \\ y[n] &= \alpha x[n] + (1-\alpha) y[n-1], \quad n > 0 \\ &= \alpha \sum_{m=0}^{n} (1-\alpha)^n \times x[n-m] \end{aligned} \tag{2}$$

As the smoothing factor α of the exponential smoothing decreases, high frequencies are attenuated. The angular cutoff frequency, ω_c, can be taken as the half-power point (or −3dB-point) and computed as in Eq. 3, which can be converted to ordinary frequency as $\omega_c f_s / 2\pi$. Figure 1 plots the cut-off

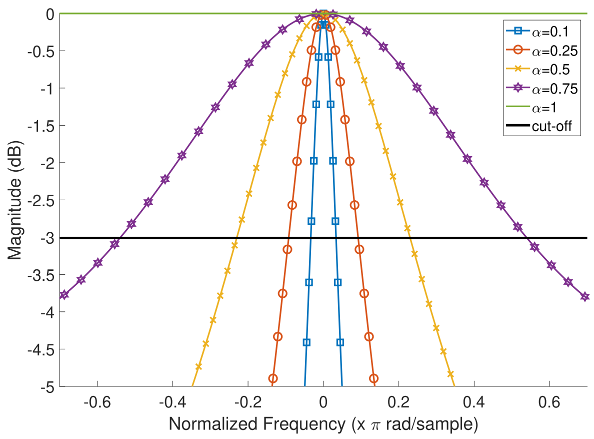


FIGURE 1. The frequency response (magnitude response) of exponential filtering. Lower smoothing factors, α, yield low-pass filters with lower cut-off frequencies. The cut-off is the angular frequency (in π rad/sample) at which the DTFT magnitude goes below the plotted −3dB attenuation level.

frequencies corresponding to various α values.

$$\omega_c = arccos(\frac{\alpha^2 + 2\alpha - 2}{2\alpha - 2}) \tag{3}$$

III. DATA COLLECTION SYSTEM

In this section, we describe the proposed data collection embedded systems. Figure 2a shows the overall architecture of the data collection system. We have developed two embedded systems, one that serves as a vibrotactile stimulator system and one that records and classifies these tactile signals collected from tactile sensors. The vibrotactile stimulator system serves for data collection in a controlled environment; it controls a stepper motor that rotates the drum. Considering the fact that the psychophysical dimensions of the material properties cover a wide range of percepts (such as roughness, softness, warmness, and friction) [11] and they require complex spatiotemporal analysis, we have limited our study to the machine perception/discrimination of various textures that can be sensed by sensors attached to a probe/stick touching the material surfaces via a single touch point. As the probe rubs against the surface of the textured material on the stimulator, the sensors attached to the probe capture the vibrotactile signals for real-time classification. The probe is 3D printed with high printing density so that it transmits the vibrations at its tip without distortion.

The stimulator system consists of a control unit, a motor driver module, and the rotating drum module. The control unit, reads the experiment specifications (including the speed and direction of rotation) to control the drum accordingly.

Figure 2b shows the physical implementation of the system. The diameter of the drum is 7 cm and it rotates at a linear speed of 5 cm s^{-1} which was chosen as a typical touch velocity. For each texture, 20 seconds of recordings are collected (corresponding to nearly five rotations of the drum).

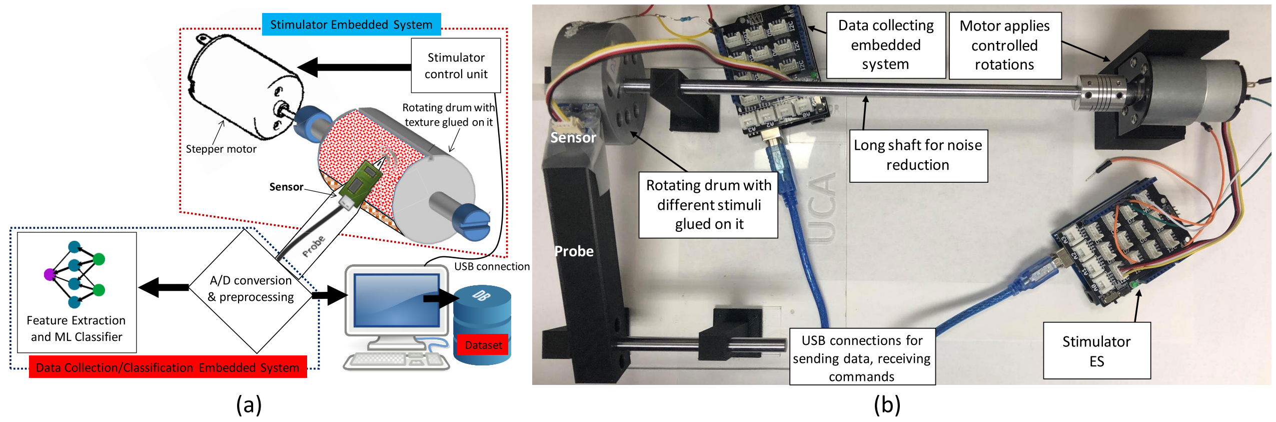

Stimulator Embedded System
Stimulator control unit
Stepper motor
Rotating drum with texture glued on it
Sensor
Probe
USB connection
A/D conversion & preprocessing
Feature Extraction and ML Classifier
DB
Dataset
Data Collection/Classification Embedded System
(a)
Data collecting embedded system
Motor applies controlled rotations
Sensor
Long shaft for noise reduction
Rotating drum with different stimuli glued on it
Probe
USB connections for sending data, receiving commands
Stimulator ES
(b)

FIGURE 2. Block diagram of the proposed embedded systems (a) for sensor stimulation and tactile data collection/classification. The right-hand side (b) shows the physical implementation of the embedded systems.

For this study, we have explored a number of commercial-off-the-shelf sensors and embedded boards. To acquire multimodal tactile information, we studied various sensors, including accelerometers, piezo sensors (e.g. Piezoelectric Polymer sensor), motion sensors (e.g. Vibration module based on the vibration sensor SW-420) and microphones. For the recording embedded system, we have used the Arduino UNO SoC board to read and collect data in a flexible sampling rate. Based on the limited memory and performance budget of the embedded system used in this study, we have chosen the 3-dimensional accelerometer sensor (MMA-7660 from NXP Company [37]) and an electret condenser microphone (CMA-4544PF-W from CUI Company [38]) as the sources of recordings in our tactile dataset. Although using a richer combination of sensors on a more expensive embedded board would achieve even higher classification accuracy, developing efficient methods for real-time signal processing on embedded systems will help improve the throughput of both low-cost and advanced embedded systems.

We have used the on-chip analog to digital converter (ADC) of the AVR processor available on the Arduino board. The AVR's ADC is set to work with the maximum available clock speed which is the Arduino's Clock/128 = 16 MHz/128 = 125 kHz. Based on the technical details of the AVR's ADC, each analog to digital conversion operation takes 13 ADC clocks that makes a total of 104 μs for each conversion. This yields the highest available sampling rate of 9615 Hz on our Arduino data collecting system. Knowing this limitation, we have set up the sampling rate of data collecting system to 200 Hz for the accelerometer to collect the motion data and to 8 kHz for the microphone to collect the sound data.

For every sampling instance, the proposed data collection system reads a new data sample, the data is sent through serial USART communication to the computer for populating the tactile dataset. The 3-dimensional accelerometer sensor that we used for data collection measures movements in X, Y, and Z directions and gives a total of 3 values per

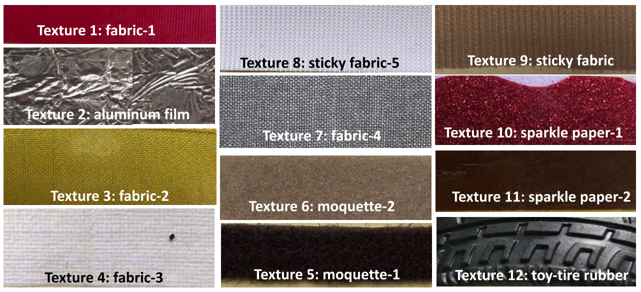


FIGURE 3. The 12 texture classes used in our data collection and classification.

sample, from which the acceleration can also be computed. Movement recordings are integer values that are sent to the base computer for storage. The statistics of recorded values from the sensors are given in Table 1. As the readings to be transferred to the computer (for populating the dataset) have different orders of magnitude, the transfer time is not fixed as the Arduino's USART communication converts data to string before the transfer. This negatively alters the data sampling period of the sampling loop. To avoid sampling rate variation, we have set one of the ARV on-chip timers to interrupt the processor every specific amount of time (determined based on the desired sampling frequency) that calls the sensor reading and data transfer routines.

We have used commercial off-the-shelf embedded boards and electrical components (AVR-based embedded boards, stepper motors, etc.) as well as our own designed and 3D printed mechanical components (including the rotating drum glued with different texture strips). The collected tactile dataset has 12 texture classes and Figure 3 shows an exemplary subset of texture strips that used for the experiments. Textures include sandpapers of various grits, Velcro strips with various thicknesses, aluminum foil, and rubber bands of various stickiness. The dataset collected is available upon request and can be found at [34].

To validate the data collection system, we analyzed the recordings of the 12 texture classes using discrete time Fast

Fourier Transform (FFT). As outlined in Section V, using one to three seconds of contact with these texture materials were sufficient to distinguish them. The recordings are first tested offline and shown to include discriminatory information for texture classes. We cropped a number of training examples from each texture; we used 256-sample windows for each example (that corresponds to about one second recording of the accelerometer). Then, we applied the Fourier transform for feature extraction and tested how discriminable the classes are using various machine learning algorithms, including K-nearest-neighbor (KNN), support vector machines (SVM), and random forests (RF) [39], [40]. Window sizes smaller than 256-samples (e.g. 128) correspond to less than half a second of the accelerometer recordings (and much shorter for the sound signals sampled at 8 kHz) and result in significantly lower classification accuracy. However, even with a window length of 128-samples, the FFT-based implementation on the embedded system failed due to the data memory limitation. A method that avoids the window-based buffering of sample readings has been proposed and discussed in the next section.

IV. PROPOSED CUMULATIVE MULTI-BANDPOWER FEATURE EXTRACTION METHOD

Embedding an efficient feature extraction method into the hardware platform shown in Figure 2 enables it to collect and real-time classify the texture data. Instead of buffering the samples of sensor readings in time windows for spectral analysis in the frequency domain, we propose a simple (memory-less) yet efficient (discriminatory) signal feature extraction method easily implemented on our intelligent embedded system for tactile classification. Such feature extraction methods are of significant importance for real-time, energy-efficient, mobile embedded systems. The memory and performance constraints of an embedded implementation may prohibit the use of resource demanding feature extraction methods such as the Fourier transform and deep learning. Our proposed feature extraction method computes a small yet descriptive statistics of power spectral density of the streaming signals coming from the vibrotactile sensors. Parseval's theorem described in Section II allows the computation of the total power of a signal in the time domain. Combining this idea with exponential smoothing of the input signals, the band-powers in various cumulative frequency bands can form a rich set of features extracted in the time domain in real-time. The proposed method, called cumulative multi-bandpower (CMB) feature extraction method, is described below.

Let x[n] denote the discrete readings obtained from a given sensor at time step n (e.g., the data coming from one of the three channels of the accelerometer sensor and the feature extraction can be performed in parallel in each dimension separately).

Parseval's theorem (Eq. 1) states that the total energy (thus, average power) of a signal can be calculated either using the amplitudes in the time domain or spectral power in the frequency domain. More specifically, summing power-per-sample across time (i.e. sum of the squares of the amplitudes of the data samples) is another way of computing the total spectral power across frequency. On one hand, FFT returns the power spectrum that precisely describes the distribution of power into individual frequency components of the given signal; on the other hand, working in time domain with summations (accumulation of powers of samples, $x[n]^2$, provides a memory-less mechanism that can help the embedded system avoid complex and computationally demanding FFT approach. To take advantage of both approaches, instead of using average power as the single feature over the whole frequency spectrum, we propose to extract an array of such features receiving their incoming samples from smoothened versions of the signal (i.e. low-pass filtered data samples with various pass-band/cut-off characteristics). We approximate the low-pass filters using exponential smoothing (see Eqs. 2 3) as defined in Eq. 4:

$$S^{\alpha_k}[n] = (1-\alpha_k) \times S^{\alpha_k}[n-1] + \alpha_k \times x[n] \tag{4}$$

for a set of K smoothing factor values, $0 < \alpha_k < 1$, for $k = 1, \ldots, K$. Using lower smoothing factors, α_k, computes low-pass filters with lower cut-off frequencies (i.e. lower values of α_k actually increase the level of smoothing; see Figure 1 for the relationship between exponential smoothing and low-pass filters). Let us assume smoothing factors are sorted in decreasing order and let us include an additional $\alpha_0 = 1$, which does not perform any smoothing, $S^{\alpha_0}[n] = x[n]$, and it will be used for computing the average power of the signal as suggested by Parseval's theorem: $1 = \alpha_0 > \alpha_1 > \alpha_2 > \ldots > \alpha_K > 0$. Note that Eq. 4 can be computed memory-less without the need for storing the past $S^{\alpha_k}[n]$ values. In fact the computational code performs the following assignment.

$$S^{\alpha_k} \leftarrow (1-\alpha_k) \times S^{\alpha_k} + \alpha_k \times x[n] \tag{5}$$

Having such an array of S^{α_k} data values, the Parseval's theorem can now be applied to sum up the squares of these values in order to calculate average powers in gradually narrower bands of frequencies as shown in Figure 1 (due to gradually lower cut-off frequencies these consecutive low-pass filters have). Let F^{α_k} denote the (average power) feature extracted for a given alpha value as in Eq. 6:

$$F^{\alpha_k}[n] = \frac{1}{n} \sum_i |S^{\alpha_k}[i]|^2 \tag{6}$$

Note that we can avoid buffering $F^{\alpha_k}[n]$ values and again use exponential smoothing to estimate the sum of power-per-sample, ${S^{\alpha_k}}^2$, in our calculations:

$$F^{\alpha_k} \leftarrow (1-\beta) \times F^{\alpha_k} + \beta \times |S^{\alpha_k}|^2 \tag{7}$$

Also note that using exponential smoothing applies exponentially decreasing weights over time that fits well with the transient nature of the incoming data as the data changes from one texture class to another. These set of $F^{\alpha_k}, k = 0, 1, \ldots, K$, features can discriminatory signals based on not only their frequency components but also their amplitudes

TABLE 1. Statistics of the collected dataset including mean, standard deviation, and data range for signals recordings. The sources of data include sound recorded by the microphone and X, Y, and Z channels of the accelerometer.

Data Class	Accelerometer X-Channel			Accelerometer Y-Channel			Accelerometer Z-Channel			Sound Channel		
	Mean	Std. Dev.	Data Range	Mean	Std. Dev.	Data Range	Mean	Std. Dev.	Data Range	Mean	Std. Dev.	Data Range
Class 1	8.39	1.29	[-0.99, 13]	-0.57	1.99	[-5, 21]	18.59	2.17	[0, 25]	-1.4E-05	4.3E-02	[-0.8, 0.7]
Class 2	7.90	4.58	[-13, 25]	-0.65	2.55	[-10, 28]	17.60	5.73	[-2, 30]	-1.2E-05	1.1E-01	[-1, 0.9]
Class 3	7.50	5.76	[-16, 28]	-0.58	2.29	[-6, 22.1]	18.86	2.77	[0, 30]	-1.7E-05	4.1E-02	[-0.6, 0.6]
Class 4	6.74	11.99	[-31, 30]	-0.68	5.14	[-18, 28]	14.08	10.35	[-15, 31]	-1.6E-05	1.7E-02	[-0.1, 0.1]
Class 5	6.70	6.38	[-18, 28]	-0.98	2.37	[-8, 21.2]	19.02	3.45	[0, 29]	-1.3E-05	4.0E-02	[-0.2, 0.2]
Class 6	6.08	3.62	[-14, 25]	-0.92	2.44	[-13, 21.8]	19.35	2.45	[0, 27]	-1.4E-05	8.5E-02	[-1, 1]
Class 7	3.87	9.48	[-32, 31]	-0.21	5.69	[-22, 24]	17.33	3.85	[0, 28]	-1.5E-05	1.9E-02	[-0.1, 0.1]
Class 8	3.61	10.20	[-32, 31]	-0.86	5.64	[-19, 21.1]	18.82	2.50	[0, 26]	-1.5E-05	1.1E-02	[-0.1, 0.1]
Class 9	5.95	4.42	[-16, 21]	-0.89	3.37	[-19, 22.2]	18.85	2.76	[0, 26]	-8.8E-06	5.8E-02	[-0.5, 0.4]
Class 10	5.96	2.13	[-12, 19]	-0.93	2.24	[-7, 19.3]	19.32	2.30	[0, 25]	-1.1E-05	5.5E-02	[-0.4, 0.4]
Class 11	5.62	2.70	[-18, 28]	-0.84	2.36	[-16, 25]	19.10	2.97	[-6, 28]	-1.6E-05	6.5E-02	[-1, 1]
Class 12	5.79	1.67	[-6, 21]	-0.89	1.96	[-5, 19.8]	19.55	2.12	[0, 29]	-1.6E-05	4.1E-02	[-0.8, 0.6]
Average	**6.18**	**6.48**	**[-32, 31]**	**-0.75**	**3.47**	**[-22, 28]**	**18.37**	**4.50**	**[-15, 31]**	**-1.4E-05**	**5.7E-02**	**[-1, 1]**

and DC-offsets (c_0 of FFT or the average amplitude). However, the following simple normalizations can be incorporated into the feature extraction for obtaining features sensitive only to frequency variations.

To achieve DC-offset invariance, first we modify Eq. 6 as Eq. 8:

$$F^{\alpha_k}[n] = \sum_i |S^{\alpha_k}[i] - x[i]|^2, \quad k = 1, \ldots, K \tag{8}$$

For $k = 0$, we do not change the formulation of Eq. 6

$$F^{\alpha_0}[n] = \sum_i |S^{\alpha_0}[i]|^2 = \sum_i |x[i]|^2 \tag{9}$$

As before, these F values can be obtained using memoryless computation by exponential smoothing:

$$F^{\alpha_k} \leftarrow (1-\beta) \times F^{\alpha_k} + \beta \times |S^{\alpha_k} - x[n]|^2 \tag{10}$$
$$F^{\alpha_0} \leftarrow (1-\beta) \times F^{\alpha_0} + \beta \times |S^{\alpha_0}|^2, \tag{11}$$

Finally, the normalized features, R, are computed as:

$$R^{\alpha_k} = \frac{F^{\alpha_k}}{F^{\alpha_1}}, \quad k = 1, \ldots, K \tag{12}$$

It is straightforward to show that R^{α_k} features are both amplitude-scaling and DC-offset invariant. We can show that scaling the amplitude of the signal, $x[n]$, by a factor, m, and changing the DC-offset by a constant, $c = c_0$, does not change these R features. Let $x_{m,c}[n] = m \times x[n] + c$ represent this new signal and let $F^{\alpha_k}_{m,c}$ and $R^{\alpha_k}_{m,c}$ represent the unnormalized and normalized features of $x_{m,c}$, respectively.

$$\begin{aligned} F^{\alpha_k}_{m,c}[n] &= \sum_i |S^{\alpha_k}_{m,c}[i] - x_{m,c}[i]|^2 \\ &= \sum_i |(m \times S^{\alpha_k}[i] + c) - (m \times x[i] + c)|^2 \\ &= \sum_i |m(S^{\alpha_k}[i] - x[i])|^2 \\ &= m^2 \sum_i |S^{\alpha_k}[i] - x[i]|^2 \\ &= m^2 \times F^{\alpha_k}[n], \quad k = 1, \ldots, K \end{aligned} \tag{13}$$

Since all $F'^{\alpha_k}[n]$ features for all α values are scaled up by a factor of m^2, the scaling can be cancelled out by using the ratios of the F^{α_k} values to each other. A good strategy to control the magnitude of the features would be to normalize each F^{α_k} by the previous feature, $F^{\alpha_{k-1}}$. For simplicity of formulating the theory and the subsequent discussion, we choose to normalize by F^{α_1} and compute the normalized features, $R^{\alpha_k}_{m,c}$, of $x_{m,c}$ as:

$$\begin{aligned} R^{\alpha_k}_{m,c}[n] &= \frac{F^{\alpha_k}_{m,c}[n]}{F^{\alpha_1}_{m,c}[n]} \\ &= \frac{m^2 F^{\alpha_k}[n]}{m^2 F^{\alpha_1}[n]} \\ &= R^{\alpha_k}[n], \quad k = 1, \ldots, K \end{aligned} \tag{14}$$

Clearly, $R^{\alpha_1} = 1$ and it can be omitted. Moreover, augmenting the set of $K-1$ normalized features, $R^{\alpha_k}, k = 2, \ldots, K$, with F^{α_0} and F^{α_1} has the same descriptive power as the set of $K+1$ unnormalized features, $F^{\alpha_k}, k = 0, 1, \ldots, K$. Having F^{α_0} and F^{α_1} with the scaling and DC-offset invariant normalized features can help machine learning classifiers detect average power and amplitude variations as they also might be valuable sources of information.

As a first demonstration of the proposed feature extraction method, we use simple sine waves with various frequency, amplitude, phase, and DC-offsets. For this aim, we define the following six functions with $w_1 = 2\pi \times 600$ and $w_2 = 2\pi \times 250$ (frequencies of 600 Hz and 250 Hz):

- $x_1(t) = \sin(w_1\, t)$,
- $x_2(t) = \sin(w_2\, t)$,
- $x_3(t) = 0.2\, x_1 + 0.7\, x_2$,
- $x_4(t) = 2 \sin(w_1\, t)$,
- $x_5(t) = \sin(w_1\, t + \omega_0)$, for $\omega_0 = 83$, and
- $x_6(t) = 10 + \sin(w_1\, t)$

These signals, $x_1(t)$ through $x_6(t)$, are sampled for 1 s at a rate of $F_s = 4$ kHz and their Fourier transforms show frequency harmonics at either/both 600 Hz and 250 Hz, as expected. Figure 4 shows that the proposed features, R^{α_k}, have frequency sensitivity as the plots for x_1, x_2, and x_3 have different feature values. Moreover, the figure shows

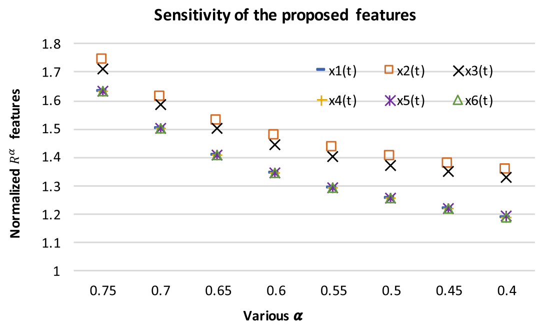


FIGURE 4. Normalized R^α features, $F^{\alpha k}/F^{\alpha k-1}$, detect the differences in the power spectrum while showing invariance to scaling, DC-offset, and phase differences as demonstrated on signals x_1 through x_6.

that these features are invariant to amplitude-scaling and DC-offset as the plots for x_1, x_4, and x_6 are perfectly identical. The features are also very robust to phase changes as the differences between the features of x_1 and x_5 are negligibly small (≈0.0001dB) in Figure 4, which demonstrates that the changes in the feature values are due to the difference in the power spectrum of signals.

The algorithm of the feature extraction algorithm is given in Algorithm 1. The time and space complexity of the CMB feature extraction algorithm are both linear in K, $O(K)$, where K is the number of cumulative bands used by CMB (corresponding to the number of alpha values). Fast Fourier Transform (FFT), on the other hand, has $O(nlogn)$ time complexity and $O(n)$ space complexity, where n is the length of the FFT-window with $n \gg K$, especially when the sampling rate is high. This difference makes CMB more applicable than FFT in real-time with a small compromise in accuracy.

Algorithm 1 Proposed Memory-Less Cumulative Multi-Bandpower (CMB) Feature Extraction Algorithm

```
1  function CMB_Features(x, $\alpha[0..K]$, $S[0..K]$, $F[0..K]$, $\beta$)
       // Input:
       // x: Data value at the current time step, $x[n]$ of Eq. 4
       // $\alpha[0..K]$: Smoothing factors
       // $S[0..K]$: For each smoothing factor $\alpha$, smoothed data value of the previous time step that correspond to $S^{\alpha_k}[n-1]$ of Eq. 4
       // $F[0..K]$: Unnormalized features at the previous time step that correspond to $F^{\alpha_k}[n-1]$ of Eq. 6
       // $\beta$: Time constant for estimating the running average of $F[0..K]$
       // Output:
       // $R[1..K]$: Normalized features that correspond to $R^{\alpha_k}[n]$ of Eq.12
       // $F[0..K]$: Updated values of the unnormalized features given as input
       // $S[0..K]$: Updated values of the smoothed data values given as input
       //
2  for ($k \leftarrow 0$ to $K$) do
3      $S[k] \leftarrow (1-\alpha[k]) \times S[k] + \alpha[k] \times x$
4      if ($k == 0$) then
5          $F[k] \leftarrow (1-\beta) \times F[k] + \beta \times x^2$
6      else
7          $F[k] \leftarrow (1-\beta) \times F[k] + \beta \times |S[k]-x|^2$
8          $R[k] \leftarrow F[k]/F[1]$
9      end
10 end
```

V. EXPERIMENTAL RESULTS

To evaluate the proposed embedded system for tactile classification, we have performed a wide range of experiments. In the first experiment, we have implemented and tested various classifiers on the texture classification task using both the proposed CMB features and the FFT features for comparison. We have used the following six classifiers [39], [40]:

- Random forest classifier, which is referred to as *RF* in the figures. The *RF* classifier is a majority voting ensemble of a number of decision trees. We have varied the number of trees for optimization purposes. With improved generalization capabilities, *RF* is one of the simplest yet accurate classifiers in machine learning [40], [41].
- Support vector machine classifier with the radial basis function (RBF) kernel, which is referred to as $RBF-SVM$ in the figures. As a nonlinear maximum margin classifier, $RBF-SVM$ is one of the most successful classifiers in machine learning (especially with small/mid size datasets).
- Linear support vector machine classifier, which is referred to as $Linear-SVM$ in the figures. $Linear-SVM$ is less powerful than $RBF-SVM$ but it has good generalization due to margin maximization. Moreover, it is easy to train, scales to large number of samples, and the discriminant can be computed explicitly.
- K-nearest neighbors classifier using the Euclidean distance metric, which is referred to as $KNN-Euclidean$ in the figures. Other distance metrics have also performed comparable in our experiments. This algorithm does not perform any training, it only stores the training dataset and measures the distance of a test example to these training examples to make its inference). *KNN* can serve as a good baseline for accuracy but even for $K = 1$, it is inefficient due to high time and space complexity.
- Multi-layer perceptron classifier, which is referred to as *MLP* in the figures. *MLP* has a number of neurons in each one of its hidden layers. The hidden units in each hidden layer extract nonlinear combinations of the inputs from the previous layer in order to define sufficiently nonlinear discriminants. With more layers and neurons, the number of parameters increase and the generalization reduces [39].
- Logistic regression (referred to as $Logistic-R$ in the figures) is a statistical method that models the probability of classes (dependent variables) using a linear combination of features (predictors or independent variables).

We picked classifiers that are suitable for our embedded implementation and that are also straightforward to optimize

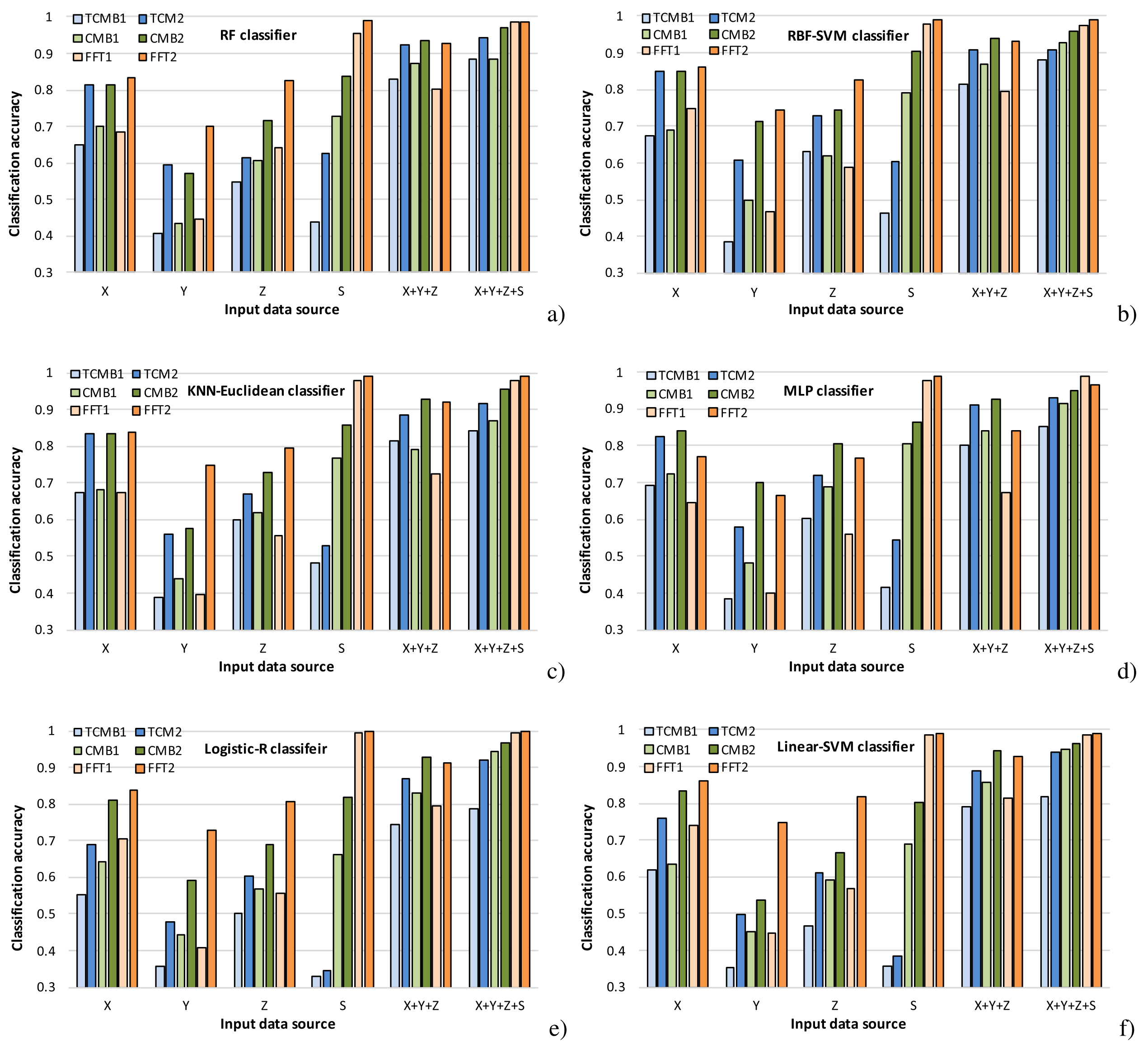


FIGURE 5. Test accuracies of classification using various combinations of accelerometer and sound data. The classifiers are (a) Random Forest, (b) RBF Support Vector Machine, (c) 1-Nearest Neighbor, (d) Multi-Layer Perceptron, (e) Logistic Regression, and (f) Linear Support Vector Machine.

without requiring many hyperparameters so that we can keep the focus on feature extraction (not the selection/optimization of an advanced classifier). The default settings were generally preferred to avoid over-fitting. For KNN, we used $K = 1$, for linear SVM we used the default value for the regularization parameter ($C = 1$). For $RBF - SVM$, we used automatic scaling for the gamma value (gamma is inversely proportional to the RBF radius). For MLP, we used a single hidden layer with 100 neurons with ReLU activation function with a learning rate of 0.001 and momentum = 0.9. For random forests, we used 100 trees and the percentage of features to consider for the best split was set to 50%.

Snipping recordings of various lengths and applying feature extraction, we created the training and test sets for the machine learning classifiers. We created two datasets, one with shorter snips of 1 to 1.5 seconds in length (chosen randomly in that range to mimic variations in a typical finger swipe/touch) and one with longer snips of 2 to 3 seconds. We have repeated these experiments 30 times. Each training set contains 120 example snips (10 per class) and the test sets contain 480 examples (40 examples per class). To extract features from the snips, we apply and compare the following three feature sets and use them as input to the aforementioned classifiers. For the implementation of the classifiers on PC for offline analysis, including optimization/validation of the classifier hyperparameters, we used scikit-learn Python module for machine learning [42].

TABLE 2. True positive and true negative rates for the classification methods using the CMB features.

Classifier		Classes used in classifications with the proposed CMB features												Average
		class 1	class 2	class 3	class 4	class 5	class 6	class 7	class 8	class 9	class 10	class 11	class 12	
Random Forest	TP	100	99.8	86.9	99.6	80.9	76.8	79.8	70.6	78.8	79.3	67.9	80.8	**83.4**
	TN	100	99.5	98.9	99.9	98.9	97.8	97.3	98.3	97.9	98.0	98.6	96.7	**98.5**
RBF-SVM	TP	99.4	99.8	86.7	98.4	87.4	82.2	74.9	77.4	89.1	86.5	64.9	88.1	**86.2**
	TN	98.6	100	98.4	99.8	98.4	98.3	98.0	98.6	98.6	98.8	99.1	98.3	**98.7**
KNN-Euclidean	TP	94.3	98.4	82.3	93.6	80.3	73.3	75.7	81.9	82.8	83.7	73.3	86.0	**83.8**
	TN	99.0	99.9	98.0	99.6	98.5	98.3	98.3	98.6	98.2	99.0	96.9	97.9	**98.5**
MLP	TP	92.8	99.3	71.6	95.7	65.9	62.1	69.6	74.5	73.3	68.1	70.8	79.7	**76.9**
	TN	97.2	99.7	98.4	98.6	98.4	98.9	96.4	95.9	98.9	97.7	98.4	96.3	**97.9**
Logistic R	TP	98.3	100	83.5	98.6	80.7	71.7	74.0	79.0	81.5	79.3	73.3	84.3	**83.7**
	TN	98.9	99.9	98.2	99.8	98.5	98.4	98.0	98.0	97.9	98.3	98.4	97.8	**98.5**
Linear-SVM	TP	99.4	99.8	87.4	98.4	87.3	81.6	74.2	76.0	88.2	85.7	68.7	87.7	**86.2**
	TN	98.7	100	98.4	99.8	98.3	98.4	97.9	98.5	98.7	98.9	98.8	98.4	**98.7**

- The proposed CMB features. These features are referred to as *CMB*1 and *CMB*2 in the figures for the short ($\approx$1 s) and for the long ($\approx$2 s) recordings, respectively. CMB works in time domain by updating its features in real-time and the length of the snips does not complicate its computations.
- The Fourier transform (FFT) features. The frequency spectrum of the snips are computed by FFT and fed as the feature vector to the classifiers. In the figures, for the short and long recordings, FFT features are referred to as *FFT*1 (using one-second windows) and *FFT*2 (using two-second windows), respectively.
- The third set of features is a tiny subset of our proposed CMB features, hence we named it Tiny-CMB (TCMB). Here, we use CMB only with the two most extreme α values, $\alpha = 1$ that computes the total power of the signal i.e., $F^1[n] = \frac{1}{n}\sum_i x[i]^2$, (as also mentioned in Eg. 9) and $\alpha = 0$ that computes the variance of the signal ($F^0[n] = \frac{1}{n}\sum_i (\overline{x} - x[i])^2$, where $\overline{x} = \frac{1}{n}\sum_i x[i]$). From the power and the variance, the square of the average of the signal window can also be computed, $\overline{x}^2 = F^1 - F^0$, which makes TCMB features interesting from the machine learning perspective as using the mean and the standard deviation of classes play an important role in inference in machine learning [39]. In the figures, we refer to the features as the short and long recordings as *TCMB*1 and *TCMB*2, respectively.

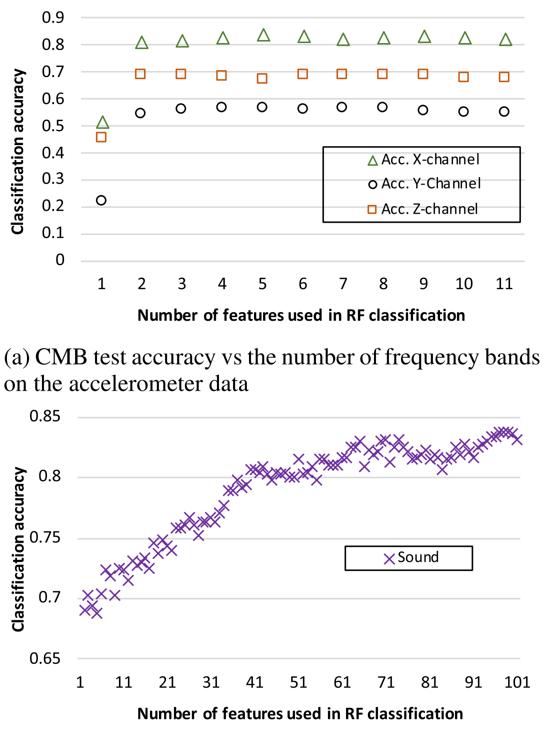
Figure 5 compares the classification accuracies obtained with most classifier-feature-sensor combinations on the short and long recording datasets. The sensors include X, Y, and Z channels of the accelerometer and the sound (denoted by S in the figures). The figure shows that increasing the length of data recordings for feature extraction improves the classification accuracy in most cases for the CMB features. However, for the FFT features, longer windows have led to no improvement or even accuracy loss in some cases e.g., in Figure 6d when all channels of data are used. That small degradation might be due to the curse of dimensionality phenomenon [39]: Using longer recordings leads to very high dimensional feature vectors (frequency components) for FFT. However, for the proposed CMB features, using longer or shorter recordings does not change the number of features (which depends only on the number of α values). The results summarized in the figure also suggests that the proposed CMB feature extraction method offers accuracy levels comparable to those of FFT and achieves that without the memory/computation demands that FFT has.

Tables 2 and 3 shows the details of classification results with the proposed CMB and FFT features in terms of the true positive and true negative rates for each class. Here, the true positive rate (TP in the tables) of a class refers to the proportion of the actual textures of that class that are correctly identified as such by the classifier; while the true negative rate (TN) measures the proportion of texture examples of other classes that are correctly predicted to be nonmembers of that class by the classifier. The tables are color-coded in a way that values closer to 100% are dark green; and as the values get lower, the color gets closer to dark red. Comparing the corresponding cells of these two tables, we see that the FFT features offer better TP/TN. This comes from the fact the FFT provides more powerful (full resolution view of the spectrum) features that lead to more accurate classifications. However, the FFT implementation was not feasible on our tiny embedded platform. On the other hand, the CMB features can provide comparable levels of TP/TN in almost all cases with its simple implementation on the embedded board.

In the next experiment, we studied the accuracy of the CMB-based classification with respect to the number of features (i.e. the number of α values that controls the number of cumulative bands). Figure 6 presents the results obtained in this set of experiments; the results on the accelerometer sensors are given in Figure 6a and the results on the sound signals collected by the microphone sensor are given in Figure 6b. We see that increasing the number of bands (features) help the sound data a lot more for achieving high classification accuracy. However, as the accelerometer carries information in a limited band of frequencies, its accuracy reaches its peak at a faster rate (using only a few α values). This observation means that the optimum number of bands in the CMB

TABLE 3. True positive and true negative rates for the classification methods using the FFT features.

Classifier		Classes used in classifications with FFT features												Average
		class 1	class 2	class 3	class 4	class 5	class 6	class 7	class 8	class 9	class 10	class 11	class 12	
Random Forest	TP	100	90.7	83.4	99.7	82.9	90.9	56.0	60.3	90.3	77.1	70.0	76.1	**81.4**
	TN	99	99.0	98.0	99.6	98.7	99.1	96.7	96.5	99.0	97.9	98.7	97.2	**98.3**
RBF-SVM	TP	100	99.8	92.0	99.9	87.0	91.8	60.8	61.8	94.4	85.8	68.2	78.7	**85.0**
	TN	100	100	98.8	99.7	99.0	99.3	96.8	96.8	99.4	97.5	99.5	97.0	**98.6**
KNN-Euclidean	TP	100	99.6	90.1	100	86.3	90.3	54.8	63.1	93.6	77.7	70.3	77.1	**83.6**
	TN	100	99.8	98.7	99.7	99.0	99.0	96.8	96.3	99.4	97.8	98.5	97.0	**98.5**
MLP	TP	100	99.6	92.4	99.7	85.5	88.7	66.1	61.7	88.1	70.7	72.7	84.1	**84.1**
	TN	100	100	98.9	100	99.0	98.8	96.5	97.3	98.9	98.4	99.3	95.4	**98.6**
Logistic R	TP	100	100	94.6	100	86.8	83.5	60.6	60.3	96.1	25.9	67.3	99.2	**81.2**
	TN	100	100	98.8	99.9	98.8	99.5	96.5	97.1	99.1	99.8	98.8	91.1	**98.3**
Linear-SVM	TP	100	99.8	89.7	99.9	91.5	82.8	60.9	62.3	95.3	76.6	60.9	79.8	**83.3**
	TN	100	100	99.2	99.7	98.7	99.2	96.8	96.9	99.3	97.2	98.8	96.0	**98.5**



(a) CMB test accuracy vs the number of frequency bands on the accelerometer data

(b) CMB test accuracy vs the number of frequency bands on the sound data

FIGURE 6. Accuracy of the random forest classifier vs the number of frequency bands (α values) of the CMB method. Separate plots are shown for each one of the four sensor channels (X, Y, Z, and S) collected in parallel. The increase in the accuracy saturates faster for the accelerometer data shown in (a) that has simpler frequency distribution than the more complex sound data shown in (b).

feature extraction method indeed depends on the nature of the distribution of the useful information on the frequency bands. Embedded system designers may use this opportunity in their designs to minimize the number of bands needed by the feature extraction for the given application.

As the random forest classifier [41] can be expressed as an ensemble of nested *if* $-$ *then* $-$ *else* statements, it is a good candidate to be implemented on embedded systems with minimum data memory requirements and faster execution for inference after its training. In the next experiment, we have worked on optimizing the random forest classifier to better meet the the resource limitations of our Arduino-based implementation. We have used Arduino-UNO, one of the tiniest Arduino boards, in the implementation of the proposed CMB feature extraction based random forest classifier to show that the hardware requirements for extracting the features and the subsequent classification are minimal. We tried 10 random forest classifiers starting with a single decision tree and by doubling the number of trees up to 1024. The execution time (on both the embedded and PC implementation) and the classification accuracy of these random forest classifiers are reported in Table 4. Our experiment revealed that the Arduino system is not able to host more than 32 decision trees due to the lack of sufficient code/data memory. Nevertheless, even with a low number of trees in the random forest (such as 16 and 32, which are implementable on the host embedded system), a good classification accuracy is achieved. Compared to the implementation on our high-end PC with GPU, the embedded implementation needs about 20 times longer runtime for classification due to its limited computational power and lower clock frequency.

TABLE 4. Performance of the proposed CMB features for the random forest classifier implemented on an Arduino-based embedded system.

	Number of trees in the random forest										
	1	2	4	8	16	32	64	128	256	512	1024
Arduino runtime (µs)	1.14	1.76	9.56	18.5	36.62	72.88	--	--	--	--	--
PC runtime (µs)	1.10	1.19	1.38	1.86	2.70	4.35	7.45	13.92	27.17	51.84	101.91
Classification accuracy (%)	87.1	87.1	92.9	94.4	96.0	96.5	96.7	95.8	96.7	96.5	96.5

VI. CONCLUSIONS

In this study, we developed an intelligent embedded system equipped with vibrotactile sensors to populate a tactile dataset and to classify tactile signals as they are collected in real-time. For feature extraction, we also proposed a novel power spectral feature extraction method that we called CMB (cumulative multi-bandpower). The proposed feature extraction method CMB is based on the memory-less total power computation suggested by Parseval's theorem and the memory-less low-pass filtering achieved by the exponential smoothing approach to compute the band-powers of the input signals in cumulative frequency bands. Therefore, CMB works in the time domain and achieves real-time computations on the streaming tactile input signals.

For the classification of the textures, CMB features are fed to a random forest classifier implemented as an ensemble of (majority voting) rule-based decision trees. Although the combination of more descriptive Fourier transform and more powerful support vector machines achieved a higher accuracy for offline classification, its implementation for real-time online processing on the embedded board was not feasible due to data and code memory limitations. Nonetheless, our embedded implementation of the combination of the CMB features and the random forest classifier achieves comparable classification accuracy especially when multiple sensors are fused for classification.

REFERENCES

[1] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: Wireless communication meets machine learning," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 19–25, Jan. 2020.

[2] M. Yazici, S. Basurra, and M. Gaber, "Edge machine learning: Enabling smart Internet of Things applications," *Big Data Cognit. Comput.*, vol. 2, no. 3, p. 26, Sep. 2018.

[3] J. C. Gwilliam, Z. Pezzementi, E. Jantho, A. M. Okamura, and S. Hsiao, "Human vs. robotic tactile sensing: Detecting lumps in soft tissue," in *Proc. IEEE Haptics Symp.*, Mar. 2010, pp. 21–28.

[4] M. P. Menikdiwela, K. M. I. S. Dharmasena, and A. M. H. S. Abeykoon, "Haptic based walking stick for visually impaired people," in *Proc. Int. Conf. Circuits, Controls Commun. (CCUBE)*, Dec. 2013, pp. 1–6.

[5] M. Schopfer, H. Ritter, and G. Heidemann, "Acquisition and application of a tactile database," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 1517–1522.

[6] H. Soh and Y. Demiris, "Incrementally learning objects by touch: Online discriminative and generative models for tactile-based recognition," *IEEE Trans. Haptics*, vol. 7, no. 4, pp. 512–525, Oct. 2014.

[7] C. Chi, X. Sun, N. Xue, T. Li, and C. Liu, "Recent progress in technologies for tactile sensors," *Sensors*, vol. 18, no. 4, p. 948, Mar. 2018.

[8] A. Moringen, W. Aswolinkiy, G. Buscher, G. Walck, R. Haschke, and H. Ritter, "Modeling target-distractor discrimination for haptic search in a 3D environment," in *Proc. 7th IEEE Int. Conf. Biomed. Robot. Biomechatron. (Biorob)*, Aug. 2018, pp. 845–852.

[9] R. S. Dahiya, P. Mittendorfer, M. Valle, G. Cheng, and V. J. Lumelsky, "Directions toward effective utilization of tactile skin: A review," *IEEE Sensors J.*, vol. 13, no. 11, pp. 4121–4138, Nov. 2013.

[10] S. Pinker and W. W. Norton. (2009). *How Mind Works*. [Online]. Available: https://books.google.com/books?id=5cXKQUh6bVQC

[11] S. Okamoto, H. Nagano, and H. N. Ho, "Psychophysical dimensions of material perception and methods to specify textural space," in *Pervasive Haptics: Science, Design, and Application*, H. Kajimoto, S. Saga, and M. Konyo, Eds. Tokyo, Japan: Springer, 2016, pp. 3–20.

[12] P. A. Schmidt, E. Maël, and R. P. Würtz, "A sensor for dynamic tactile information with applications in human–robot interaction and object exploration," *Robot. Auto. Syst.*, vol. 54, no. 12, pp. 1005–1014, Dec. 2006.

[13] W. Duchaine, "Why tactile intelligence is the future of robotic grasping," *IEEE Spectr. Automat.*, to be published.

[14] A. Schmitz, Y. Bansho, K. Noda, H. Iwata, T. Ogata, and S. Sugano, "Tactile object recognition using deep learning and dropout," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Nov. 2014, pp. 1044–1050.

[15] A. D. Berger and P. K. Khosla, "Using tactile data for real-time feedback," *Int. J. Robot. Res.*, vol. 10, no. 2, pp. 88–102, Apr. 1991.

[16] R. S. Fearing, "Tactile sensing mechanisms," *Int. J. Robot. Res.*, vol. 9, no. 3, pp. 3–23, Jun. 1990.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[18] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019.

[19] L. Hertel, H. Phan, and A. Mertins, "Comparing time and frequency domain for audio event recognition using deep learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 3407–3411.

[20] Z. Takhirov, J. Wang, V. Saligrama, and A. Joshi, "Energy-efficient adaptive classifier design for mobile systems," in *Proc. Int. Symp. Low Power Electron. Design*, 2016, pp. 52–57.

[21] S. Venkataramani, A. Raghunathan, J. Liu, and M. Shoaib, "Scalable-effort classifiers for energy-efficient machine learning," in *Proc. 52nd Annu. Design Autom. Conf.*, 2015, pp. 1–6.

[22] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Commun. Mag.*, vol. 48, no. 9, pp. 140–150, Sep. 2010.

[23] H. Sun and G. Martius, "Machine learning for haptics: Inferring multi-contact stimulation from sparse sensor configuration," *Frontiers Neurorobotics*, vol. 13, p. 51, Jul. 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2019.00051

[24] O. Kursun and O. V. Favorov, "Suitability of features of deep convolutional neural networks for modeling somatosensory information processing," *Proc. SPIE, Pattern Recognit. Tracking*, vol. 10995, May 2019, Art. no. 109950G.

[25] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328.

[26] O. V. Favorov and O. Kursun, "Neocortical layer 4 as a pluripotent function linearizer," *J. Neurophysiology*, vol. 105, no. 3, pp. 1342–1360, Mar. 2011.

[27] O. V. Favorov, O. Kursun, and M. Tommerdahl, "Role of feed-forward inhibition in neocortical information processing: Implications for neurological disorders," in *The Physics of the Mind and Brain Disorders* (Springer Series in Cognitive and Neural Systems), I. Opris and M. Casanova, Eds. Cham, Switzerland: Springer, 2017, pp. 383–397.

[28] O. Oballe-Peinado, J. A. Hidalgo-Lopez, J. Castellanos-Ramos, J. A. Sanchez-Duran, R. Navas-Gonzalez, J. Herran, and F. Vidal-Verdu, "FPGA-based tactile sensor suite electronics for real-time embedded processing," *IEEE Trans. Ind. Electron.*, vol. 64, no. 12, pp. 9657–9665, Dec. 2017.

[29] *Cortical Metrics LLC*. Accessed: May 13, 2020. [Online]. Available: https://www.corticalmetrics.com

[30] O. V. Favorov, E. Francisco, J. Holden, O. Kursun, L. Zai, and M. Tommerdahl, "Quantification of mild traumatic brain injury via cortical metrics: Analytical methods," *Mil. Med.*, vol. 184, no. S1, pp. 228–236, Mar. 2019.

[31] U. Kursun, O. Kursun, and O. Favorov, "A novel thermal tactile stimulator device for quantitative sensory testing," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Aug. 2016, p. 3441.

[32] O. Kursun, M. M. Ilhan, A. Cinar, M. E. Isenkul, C. O. Sakar, A. E. Gursoy, E. Tasan, and O. V. Favorov, "Analyzing relations among measurements of diabetic neuropathy," in *Proc. Int. Conf. Appl. Informat. Health Life Sci.*, 2014, pp. 114–115.

[33] O. Kursun, B. Sakar, M. Isenkul, C. Sakar, F. Gurgen, S. Delil, and O. Favorov, "Analysis of effects of Parkinson's disease on the somatosensory system via cm-4 tactile stimulator," in *Proc. Int. Conf. Appl. Informat. Health Life Sci.*, 2013, pp. 57–60.

[34] O. Kursun and A. Patooghy. (2020). *Texture Dataset Collected By Tactile Sensors*. [Online]. Available: http://dx.doi.org/10.21227/kwsy-x398

[35] Z.-Q. J. Xu, Y. Zhang, and Y. Xiao, "Training behavior of deep neural network in frequency domain," in *Proc. Int. Conf. Neural Inf. Process.* Sydney, NSW, Australia: Springer, 2019, pp. 264–274.

[36] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals & Systems*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[37] *3-Axis Orientation/Motion Detection Sensor*, document MMA7660FC, NXP Semiconductor, 2012. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MMA7660FC.pdf

[38] *Electret Condenser Microphone Sensor*, document CMA-4544PF-W, CUI Devices, 2013. [Online]. Available: https://www.mouser.com/datasheet/2/670/cma-4544pf-w-1309465.pdf

[39] E. Alpaydin, *Introduction to Machine Learning*, 3rd ed. Cambridge, MA, USA: MIT Press, 2014.

[40] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3133–3181, 2014. [Online]. Available: http://jmlr.org/papers/v15/delgado14a.html

[41] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in PyThon," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

OLCAY KURSUN received the B.S. and M.S. degrees in computer engineering from Bogazici University, Istanbul, Turkey, in 1998 and 2000, respectively, and the Ph.D. degree in computer science from the University of Central Florida, in 2004, on developing cortex-inspired neural networks. He worked as an Associate Professor at Istanbul University, until 2016. After more than a decade of international collaboration, in 2016, he joined his collaborators' neuroscience lab with the Department of Biomedical Engineering, University of North Carolina, Chapel Hill, where he continued developing machine learning and computational neuroscience algorithms. Since Fall 2017, he has been a Faculty Member with the Department of Computer Science, University of Central Arkansas, where he has co-founded the Intelligent and Embedded Systems Laboratory. His research interests include machine learning and pattern recognition with particular interest in deep learning, multiview machine learning, biological neural models and their applications in neuroscience, biomedical engineering, and embedded systems.

AHMAD PATOOGHY (Member, IEEE) received the Ph.D. degree in computer engineering from the Sharif University of Technology, Tehran, Iran in 2011. After his Ph.D. he joined the Iran University of Science and Technology, Tehran, as an Assistant Professor, from 2011 to 2017, and later worked as a Senior Researcher at Boston University, Boston, MA, USA, from 2017 to 2018. He is currently leading the Intelligent and Embedded Systems Laboratory, University of Central Arkansas, where he conducts research on the security and reliability of cyber-physical systems, hardware design and acceleration for deep/spiking neural networks, and architectural design for security and reliability. He has published more than 80 conference papers and journal articles, and served as a reviewer for wide variety of IEEE, ACM, Elsevier, and Springer journals. He has also served as Panelist of the National Science Foundation for reviewing grant proposals.