sMFCC : Exploiting Sparseness in Speech for Fast Acoustic Feature Extraction on Mobile Devices – a Feasibility Study

Shahriar Nirjon[1], Robert Dickerson[1], John Stankovic[1], Guobin Shen[2], Xiaofan Jiang[3]
[1]Department of Computer Science, University of Virginia
[2]Microsoft Research Asia, Beijing, China
[3]Intel Labs China, Beijing, China
{smn8z, rfd7a, jas9f}@virginia.edu, jacky.shen@microsoft.com, fred.jiang@intel.com

ABSTRACT

Due to limited processing capability, contemporary smartphones cannot extract frequency domain acoustic features in real-time on the device when the sampling rate is high. We propose a solution to this problem which exploits the sparseness in speech to extract frequency domain acoustic features inside a smartphone in real-time, without requiring any support from a remote server even when the sampling rate is as high as 44.1 KHz. We perform an empirical study to quantify the sparseness in speech recorded on a smartphone and use it to obtain a highly accurate and sparse approximation of a widely used feature of speech called the Mel-Frequency Cepstral Coefficients (MFCC) efficiently. We name the new feature the sparse MFCC or sMFCC, in short. We experimentally determine the trade-offs between the approximation error and the expected speedup of sMFCC. We implement a simple spoken word recognition application using both MFCC and sMFCC features, show that sMFCC is expected to be up to 5.84 times faster and its accuracy is within $1.1\% - 3.9\%$ of that of MFCC, and determine the conditions under which sMFCC runs in real-time.

Keywords

Smartphone, Speech, Sparse FFT, MFCC

1. INTRODUCTION

All major smartphone platforms these days support numerous voice driven applications such as – voice commands (e.g. to launch an application or call some contact), voice-enabled search (e.g. Google's voice search), voice recognizing personal assistant (e.g. iPhone's SiRi), and voice-based biometrics. There are also non-voice sound driven applications, such as the music matching service from Shazam [22]. All of these applications require fast acoustic feature extraction both in time-domain and frequency-domain in order to offer fast, real-time services. While using only the time-domain acoustic features is sufficient in a limited number of applications, the frequency-domain features are a must for a robust and accurate encoding of acoustic signals.

State-of-the-art smartphone applications and platforms that extract acoustic features are primarily of two kinds. The first kind records the audio and sends it to a remote server over the Internet for further processing. This method has several limitations such as the requirement for an uninterrupted Internet connectivity and high bandwidth, and the associated expense of sending a large chunk of audio data over the cellular network. The second kind, on the other hand, performs the entire signal processing task inside the phone. But the limitation of this approach is that in order for fast and real-time feature extractions, they must limit the sampling rate to the minimum. For example – SpeakerSense [14] and SoundSense [15] limit their maximum sampling rate to 8 KHz. Hence, the quality of sampled speech suffers from the aliasing problem and the extracted features are often of low quality [2]. Although a sampling rate of 8 KHz satisfies the Nyquist criteria for human speech ($300-3300$ Hz), practically the higher the sampling rate the better it is in producing high quality samples. Furthermore, for non-speech acoustic analysis, a sampling rate of 44.1 KHz is required to capture the range of frequencies in human hearing ($20 - 20000$ Hz). But the problem is – there is no efficient algorithm that extracts frequency domain acoustic features inside the phone in real-time at such high sampling rates.

In this paper, we propose a novel solution to this problem which enables the extraction of frequency domain acoustic features inside a smartphone in real-time, without requiring any support from a remote server even when the sampling rate is as high as 44.1 KHz. We are inspired by a recent work [9, 8] coined sparse Fast Fourier Transform (sFFT) – which is a probabilistic algorithm for obtaining the Fourier Transformation of time-domain signals that are sparse in the frequency domain. The algorithm is faster than the fastest Fourier Transformation algorithm under certain conditions. Our goal in this paper is to analyze the feasibility of applying the sFFT to extract a highly accurate and sparse approximation of a widely used feature for speech, called the Mel-Frequency Cepstral Coefficients (MFCC) on the phone. Besides speech recognition, MFCC features are widely used in many other problems such as speaker identification [19], audio similarity measure [10], music information retrieval [13], and music genre classification. However, we limit our scope in this work to speech data only.

We perform an empirical study involving 10 participants (5 male and 5 female participants) where we collect more than 350 utterances of speech per person, recorded at different sampling rates. In our study, we quantify the sparseness of speech and show that human voice is suitable for applying sFFT to compute frequency domain acoustic features efficiently. We analyze the sensitivity of sFFT in approximating MFCC, and based on this, we design an algorithm to efficiently extract MFCC features using the sFFT instead of the traditional FFT. We name this new feature the sparse MFCC or sMFCC, in short. We experimentally determine the trade-offs between the approximation error and the expected speedup of sMFCC.

As a proof of concept, we implement a simple spoken word recognition application using both MFCC and sMFCC features and compare their accuracy and expected running time on our collected data. As the research is still in progress, we only implement the MFCC feature extraction on the smartphone while the rest of the analysis done on a PC. In future, we plan to complete the porting of the entire app to the smartphone. We also plan to explore other types of applications that require general purpose acoustic feature extraction on smartphones. The main contributions of this paper are:

- A study on 10 smartphone users to quantify the sparseness of speech data recorded on smartphones and to analyze the feasibility of using sFFT for frequency domain feature extraction.
- We describe an efficient algorithm for computing a highly accurate and sparse approximation of MFCC features which exploits the sparseness in speech.
- We implement a simple spoken word recognition application using both MFCC and sMFCC features, show that sMFCC is expected to be upto 5.84 times faster and its accuracy is within $1.1\% - 3.9\%$ of that of MFCC, and determine the conditions under which sMFCC runs in real-time.

2. BACKGROUND

2.1 Mel-Frequency Cepstral Coefficients

The Mel-Frequency Cestrum Coefficients (MFCC) is one of the most popular short-term, frequency domain acoustic features of speech signals [4]. The MFCC have been widely used in speech analysis because of their compact representation (typically, each speech frame is represented by a 39-element vector), close resemblance to how human ear responds to different sound frequencies, and their less susceptibility to environmental noise.

The MFCC feature extraction starts with the estimation of power spectrum which is obtained by taking the square of the absolute values of the FFT coefficients. However, prior to computing the power spectrum, typically each speech frame passes through a pre-emphasis filter which is followed by a windowing process. The log-power spectrum is used instead of the power-spectrum as human hearing works in decibel scales. The log-power spectrum then goes through a filtering process. A filter-bank with around 20 triangular band-pass filters is applied to reduce the dimensionality. These filters follow a Mel-scale which is linear up to 1 KHz and logarithmic for the larger frequencies – resembling human hearing. Finally, a discrete cosine transform (DCT) is performed to compress the information and to make the vectors uncorrelated. Only the lower-order coefficients (typically 13) are used and the rest are discarded. The 13-MFCCs plus the deltas and double deltas constitute a 39-element feature vector for each speech frame.

2.2 Sparse Fast Fourier Transform

The discrete Fourier transform (DFT) is one of the most significant algorithms in the digital signal processing domain. The fastest algorithm that computes DFT of an n-dimensional signal is $O(n \log n)$-time. However, a recent algorithm [9, 8] coined sparse FFT (sFFT) has broken this bound for a special case of DFT where the signals are sparse. A signal is considered sparse if most of its Fourier coefficients are small or very close to zero. For a small number k of non-zero Fourier coefficients, sFFT computes the Fourier transformation in $O(k \log n)$-time.

The basic idea of sFFT is to hash the Fourier coefficients into a small number of bins. The signal being sparse in the frequency domain, it is less likely that each bin will have more than one large coefficient. The binning process is done in $O(B \log n)$ where B is the number of bins – by at first permuting the time-domain signals and then filtering them. Each bin at this point ideally has only one large Fourier coefficient, and only such 'lonely' coefficients are taken into the solution. The process is repeated $\Theta(\log k)$ times, each time varying the bin size $B = k/2^r$ where the integer $r \in [0, \log k]$, so that all k coefficients are obtained. The overall running time of the algorithm is dominated by the first iteration, and hence the time complexity is $O(k \log n)$. The algorithm is probabilistic, but for exact k-sparse signals (i.e. at most k of the coefficients are significant), the algorithm is optimal, as long as $k = n^{\Omega(1)}$.

3. MOTIVATION

Smartphones allow a fixed number of sampling rates to capture raw audio signals from the microphone. Ideally the choice of an appropriate sampling rate should be driven by the application's QoS requirements. But often the developers are forced to choose a lower sampling rate than the desired one due to the limited processing power of the device. For example, in general-purpose acoustic processing, a 44.1 KHz Nyquist sampling rate is required to capture the range of frequencies in human hearing ($20 - 20000$ Hz). Even in speech processing problems, oversampling at 16 KHz helps avoid aliasing, improves resolution and reduces noise [23]. But at higher sampling rates, the real-time performance of the smartphone gets worse and the developers are forced to select the minimum rate compromising the quality of sampled speech.

Figure 1 compares the time to compute MFCC feature vectors from audio records of different durations at 3 different sampling rates. The experiment is done on a Nexus S smartphone running Android 2.3 that supports 8 KHz, 22.05 KHz, and 44.1 KHz sampling rates. We see that, the time to compute MFCC features is always longer than the duration of the recorded audio when the sampling rate is higher than 8 KHz. For example, the computation of MFCC vectors of a 4-second recording takes on average 7.02 s at 22.05 KHz, and 11.85 s at 44.1 KHz. Therefore, at these higher rates, the application is not capable of real-time performance.

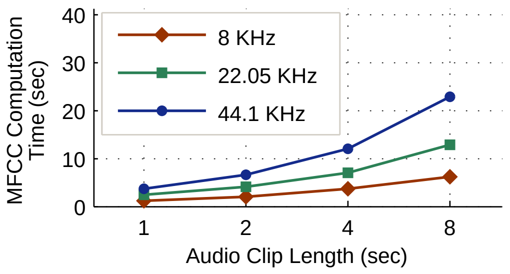

Figure 1: The MFCC computation time is $2-4$ times longer than the audio clip length when the sampling rate is high.

Our goal is therefore to investigate the problem: whether or not it is possible to compute MFCC feature vectors in real-time on a smartphone when the data rate is high? In an attempt to answer this question, we study the nature of human speech recorded on a smartphone. We hypothesize that the sparseness in speech can be exploited to compute a close approximation of MFCC feature vectors on a smartphone in real-time. While the focus of this work is on speech, an analysis of the general-purpose acoustic signals based on similar principles is under investigation and we leave it as our future work.

4. THE SPARSE MFCC ALGORITHM

The idea of sparse MFCC algorithm is to compute a sparse approximation of MFCC features from a given frame of discrete time-domain signals x_n of length n. The algorithm uses a modified version of sFFT as a subroutine. We denote the new feature by sMFCC to signify its relation to sFFT. Like the sFFT to FFT, sMFCC is an approximation to MFCC, where the approximation error is defined by,

$$error(k) = 1 - \mathbf{M\hat{F}CC} \cdot \mathbf{sMF\hat{C}C(k)} \tag{1}$$

where, $\mathbf{M\hat{F}CC}$ and $\mathbf{sMF\hat{C}C(k)}$ are unit vectors, and their scalar product is subtracted from unity to obtain the approximation error. sMFCC is expressed as a function of the sparseness parameter k, which is one of the key parameters to the sFFT algorithm. The following two sections describe the sMFCC extraction algorithm in detail.

4.1 Estimation of Sparseness

Since the value of k is a key input to the sFFT algorithm and sFFT is used in our computation, the first step of sMFCC algorithm is to find the optimum value of k, denoted by k^*, for which the MFCC approximation error is within a small, non-negative threshold δ, i.e.,

$$k^* = \min_{\mathrm{error(k)}<\delta} k \tag{2}$$

In order to obtain k^*, we first compute the MFCC using the standard FFT algorithm which runs in $O(n \log n)$. We then perform an iterative $O(n)$ search for $k \in [1, n]$ until we find the optimum k^*. The computation of sMFCC(k), for $k \in [1,\ n]$, is optimized by precomputation. We precompute the FFT, keep the FFT coefficients sorted in non-increasing order, and take only the largest k coefficients while making other coefficients zero – while computing sMFCC(k). Note that, this step of our algorithm does not use sFFT and runs in $O(n \log n)$. The shape of the function $error(k)$ (Figure 4 in Section 6.2) however suggests that, instead of a linear search over all values of k, we could expedite the process with a binary search. However, k is estimated once per utterance, i.e. using the first $5-10$ frames once voice activity is detected, and hence the amortized cost of this step is not significant.

4.2 Computing sMFCC

Once we obtain the sparsity parameter k, we compute the sMFCC for each frame in 3 steps which we describe next.

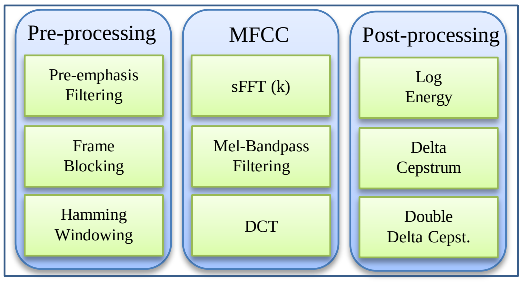

Figure 2: The computational tasks involved in the sMFCC feature extraction process is shown.

Pre-processing: The time-domain signals are first passed through a high-pass pre-emphasis filter (Eq. 3) to amplify the high-frequency formants that are suppressed in speech. We then segment the signals into frames of 64 ms with an overlap of 1/3 of the frame size. A hamming window (Eq. 4) is applied to each frame to ensure the continuity between the first and last points which is required for FFT.

$$\begin{aligned} x[i] &= x[i] - 0.95 \times x[i-1] && (3)\\ hamm(i) &= 0.54 - 0.46\cos(\frac{2\pi i}{n-1}) && (4)\end{aligned}$$

MFCC: We modify the sFFT algorithm to fit into our algorithm. Recall that, sFFT tries to extract k Fourier coefficients in $\log k$ iterations to guarantee the retrieval of all k coefficients. However, in our experience, sFFT returns most (at least 75%) of the k coefficients in a single iteration. We, therefore, modify the sFFT by running it for only a single iteration with a slightly larger sparseness parameter of $k = \min(n, \lceil 4k^*/3 \rceil)$ to speed up the process. Once the Fourier coefficients are obtained, we follow the standard procedure of MFCC [5]. We apply 20 triangular band-pass filters (called Mel-banking) to obtain 20 log energy terms, perform a DCT to compress them, and take the first 13 coefficients to constitute a 13-element sMFCC vector M.

Post-processing: The 13-element sMFCC vector is augmented to include the delta and double delta cepstrums to add dynamic information into the feature vector, and thus we obtain a 39-element feature vector.The deltas Δ and double deltas Δ^2 are computed using the following two equations,

$$\begin{aligned} \Delta_i &= M_{i+2} - M_{i-2} && (5)\\ \Delta_i^2 &= M_{i+3} - M_{i-1} - M_{i+1} + M_{i-3} && (6)\end{aligned}$$

5. EXPERIMENTAL SETUP

We perform an empirical study involving 10 volunteers, in which, we record their speech using a smartphone in home environments. Each participant was given a list of 86 English words and a paragraph from a book. The wordlist includes 10 digits, 26 characters of the English alphabet, 25 mono-syllable and 25 poly-syllable words. Participants were asked to utter each word 4-times – clearly and at regular pace. There were about a 2-second gap between two spoken words so that we could extract and model each word separately. The group of participants is comprised of undergraduate and graduate students, researchers, professionals, and their family members. Their ages are in the range of $20-60$, and they have diversities in speaking style and ethnicity. The smartphone we used during the data collection is a Nexus S phone running Android 2.3.6 OS. It has a 1 GHz Cortex A8 processor, 512 MB RAM, 1 GB internal storage, and 13.31 GB USB storage. The execution time of MFCC is measured on the smartphone using Android's API, and speedup of sMFCC is the ratio of running times of MFCC and sMFCC.

6. EXPERIMENTAL RESULTS

We conduct four sets of experiments. First, we quantify the sparseness of speech in our empirical dataset. Second, we show the approximation error in sMFCC. Third, we establish the condition for speedup in sMFCC. Finally, we describe a simple spoken word recognizer to quantify the cost and benefits of sMFCC over MFCC.

6.1 Sparseness in Speech

The sparseness of signal is defined by the number of negligible Fourier coefficients in its spectrum. A coefficient is considered negligible if it contains a very small amount of signal power. Sparseness in audio signals depends on the audio type. In this paper, we study clean speech signals only.

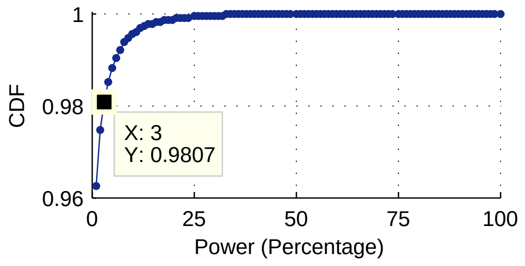

Figure 3: 98.07% **of the Fourier coefficients in our dataset contains only** 3% **or less power.**

Figure 3 shows the cumulative distribution function (CDF) of power in the Fourier spectrum of the utterances in our dataset. To obtain this plot, we compute the FFT of all the utterances in our dataset, take the squared magnitude of FFT to obtain the signal power, construct a 100-bin histogram where each bin corresponds to a range of powers, compute the fraction of Fourier coefficients that are in each bin, and compute the CDF. Each point on the plot tells us, what fraction of the signals have power less than or equal to the range corresponding to the X-coordinate. For example, the marked point on the plot denotes that 98.07% of the Fourier coefficients in each utterance of our dataset contains only 3% or less power. The rest 1.93% coefficients that are significant are permuted (see [9, 8] for the details) in the frequency domain so that the spectrum becomes extremely sparse and ideal for the application of sFFT.

6.2 Sparse Approximation Error in sMFCC

The quality of sMFCC features depends on the choice of an appropriate k. The larger the value of k, the closer it is in approximating MFCC. In this experiment, we analyze the sensitivity of k to the MFCC approximation error.

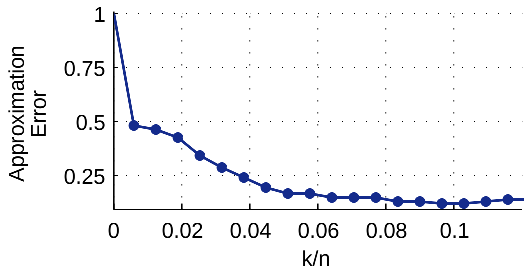

Figure 4: The higher the value of k, the better approximation of MFCC we get.

Figure 4 shows the approximation error for the range of sparseness $k \in [0.00625, 0.125]$. We consider this range since it contains most of the significant Fourier coefficients and is also important for the discussion of speedup in Section 6.3. Each point on the plot corresponds to the mean approximation error of sMFCC for a given k, where the mean is taken over all 64 ms frames in all the utterances in our dataset. The frame size is $n = 4096$ samples, which is the next power of 2 that holds a 64 ms frame at 44.1 KHz. This figure guides us in choosing the parameter k in our sMFCC computation algorithm if we want to keep the MFCC approximation error below a desired threshold. A very close approximation ($<$ 1% error) is possible by choosing $k/n = 0.2$ or higher. However, such close approximation may not be required in an actual application which we will see in Section 6.4.1. The reason is that human speech being sparse, even at a smaller k/n ratio, the absolute value of approximation error is not high.

6.3 Speedup in sMFCC

Sparseness in speech is the source of expected speedup in sFFT and hence in sMFCC as well. Figure 5 shows the speedup in sMFCC for the range of sparseness $k \in [0.00625, 0.125]$. We observe the maximum speedup of 5.84 when the sparsity parameter is at its minimum. As we consider more and more FFT coefficients while computing sMFCC, the speedup decreases and after $k/n = 6.769\%$ the regular MFCC becomes faster than its sparse counterpart. This limitation comes from the fundamental bound of sFFT, which says, sFFT is faster than FFT when $k/n < 3\%$ [8]. However, our modified version of sFFT is faster for the reason we discussed earlier in Section 4.2, and hence we have a larger bound of 6.796%.

6.4 A Simple Spoken Word Recognizer

The goal of this experiment is to analyze the tradeoff between the accuracy and expected speedup of sMFCC features in an application scenario. To do so, we implement a

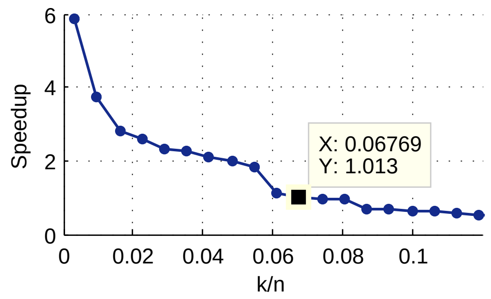

Figure 5: sMFCC has a better running time than MFCC as long as the sparseness $k/n < 6.769\%$.

simple spoken word recognizer that is essentially a speech-to-text program for a single word from a fixed vocabulary. The recognizer consists of two parts: a smartphone app and a word recognizer running on a PC. The word recognizer running on a PC is for proof of concept, our envisioned use-case is however to run it on the phone.

At first, the user turns on the application on the phone and presses the 'speak now' button. The smartphone then starts sampling the microphone at 44.1 KHz and keeps producing speech frames until the user presses the 'stop' button. Each frame goes through the MFCC feature extraction process which happens in real-time. Each spoken word produces a number of frames, and a 39-element MFCC feature vector is obtained for each frame. We take the mean and the standard deviation of each of the 39 MFCC coefficients over all frames to obtain a single 78-element feature vector which is used in the classification step. The feature vectors are then sent to a PC for classification and further analysis. The ground truth is obtained by taking notes manually. We train a Support Vector Machine (SVM) classifier in order to recognize the words. A 3-fold cross validation is used to determine the accuracy of the classifier where 75% of each user's data is taken for training and the rest is used for validation.

6.4.1 *Accuracy*

We compare the accuracy of the sMFCC-based SVM classifier with that of the MFCC-based one. The baseline MFCC-based classifier is essentially a special case of sMFCC-based one with a sparseness $k/n = 1$, and has a recognition accuracy of 85.85%. Figure 6 compares the recognition accuracy of sMFCC-based classifier to the baseline for the same range of k/n we have been using throughout the paper. We observe that, the accuracy of sMFCC-based classifier is initially 3.9% lower than the baseline, and the two accuracies becomes practically identical once k/n reaches 0.12. However, from the discussion in Section 6.3 we know that, sMFCC runs faster than MFCC only when the k/n ratio is within the 6.679% bound. For this boundary case, sMFCC shows an accuracy of 84.75%, which is only 1.1% lower than the baseline. In summary, with sMFCC-based classifier for our simple word recognition problem, we can achieve a faster running time than the baseline with a very small (1.1%-3.9%) sacrifice in recognition accuracy.

6.4.2 *Computation Time*

The MFCC feature extraction process runs once per spoken word. Hence, the execution time depends on the duration of the spoken word which varies from person-to-person and from word-to-word. In our dataset, the duration of

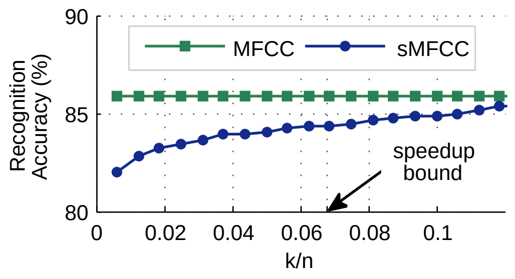

Figure 6: The recognition accuracy of sMFCC-based classifier is within $1.1\% - 3.9\%$ of the MFCC-based one inside the speedup zone.

speech ranges from the minimum of 400 ms to the maximum of 2.88 s. We, therefore, compute the expected feature extraction time $E[\phi_{feat}(d_i)]$ using the following equation,

$$E[\phi_{feat}(d_i)] = \sum f_i \times \phi_{feat}(d_i) \tag{7}$$

where, d_i is the duration of speech, f_i is the frequency of utterances with duration d_i, and $\phi_{feat}(d_i)$ is the feature extraction time (either MFCC or sMFCC).

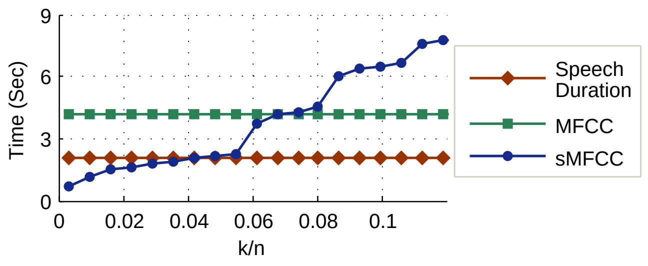

Figure 7: The sMFCC feature extraction algorithm runs in real-time as long as $k/n < 4.835\%$.

Figure 7 shows the mean speech duration and the expected computation times of MFCC and sMFCC feature extraction process. We see that, the expected MFCC computation time is 4.21 s, which is about 2 times higher than the duration of speech (2.11 s). On the other hand, the expected computation time for sMFCC varies with k/n: it increases as k/n increases, is lower than the duration of speech as long as $k/n < 4.835\%$, and crosses the MFCC computation time when k/n reaches the speedup limit of 6.679%. Hence, applications that require fast and real-time feature extraction should set the k parameter such that k/n is below the real-time limit of 4.835%. At this limit, the accuracy of the sMFCC-based word recognizer is 83.97%, which is only 1.88% lower than the accuracy of the baseline MFCC-based classifier.

Discussion: We show the tradeoff between accuracy and speedup for single word recognition problem from a limited vocabulary. The developer of the app should decide what k/n value to pick for his application. For different apps, the most suitable value of k/n will be different, and need to be chosen from a similar tradeoff curve.

7. RELATED WORK

Sparseness in data is exploited in many application domains such as learning decision trees [12], compressed sensing [6], analysis of Boolean functions [21], large scale time series data analysis [18], similarity search [1], and homogeneous multi-scale problems [3]. In our work, we perform an empirical study on the sparseness in speech data collected on smartphones with the goal of exploiting the sparseness to expedite the MFCC feature computation.

MFCC is a widely used feature for analyzing acoustic signals [5, 17, 19, 13, 10]. MFCC is used for spoken word recognition [5], voice recognition [17], speaker identification [19], music modeling [13], and music similarity measure [10]. [7] presents a nice comparison of different MFCC implementations. However, in our work, we introduce a sparse MFCC (sMFCC) which is a sparse approximation of MFCC and is efficient to extract.

Comparison of several speech recognition techniques on mobile devices are described in [11]. [14] performs speaker identification, [15] classifies sound into voice, music or ambient sound, and [16] classifies conversion. But all of these applications limit their sampling rate to its minimum. [20] uses a high sampling rate to extract heart beats from acoustic signal, but the system is not fully real-time. In this work, we analyze the feasibility of using sMFCC features that is expected to run in real-time and at higher data rates.

8. CONCLUSION AND FUTURE WORK

In this work, we propose an algorithm that exploits sparseness in speech to extract a highly accurate and sparse approximation of MFCC features (sMFCC) efficiently inside a smartphone in real-time when the sampling rate is as high as 44.1 KHz. We implement a simple spoken word recognition application using both MFCC and sMFCC features, show that sMFCC is upto 5.84 times faster than MFCC and its accuracy is within $1.1\% - 3.9\%$ of that of MFCC, and determine the conditions under which sMFCC runs in real-time.

Our future work includes three main directions. First, we plan to improve our empirical study by adding more participants, investigating the continuous speech recognition problem rather than just single word recognition, incorporating more complex classifiers, and considering background noise. Second, we plan to complete porting the entire application to the smartphone which includes on-line training and classification modules. Third, we plan to explore other types of sounds such as music and environmental sounds, and other acoustic features.

Acknowledgments

This work was supported, in part, by NSF Grant EECS-0901686.

9. REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *International Conference on Foundations of Data Organization and Algorithms*, pages 69–84, 1993.

[2] K. Brandenburg. *Perceptual Coding of High Quality Digital Audio*, volume 437. Springer US, 2002.

[3] I. Daubechies, O. Runborg, and J. Zou. A sparse spectral method for homogenization multiscale problems. *Multi-scale Modeling Simulation*, 6(3):711–740, 2007.

[4] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357 – 366, aug 1980.

[5] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357 – 366, aug 1980.

[6] D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[7] Z. Fang, Z. Guoliang, and S. Zhanjiang. Comparison of different implementations of mfcc. *Journal of Computer Science and Technology*, 16(6):582–589, nov 2001.

[8] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Nearly optimal sparse fourier transform. In *44th ACM Symposium on Theory of Computing 2012 (STOC '12)*, NewYork, NY.

[9] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Simple and practical algorithm for sparse fourier transform. In *ACM-SIAM Symposium on Discrete Algorithms 2012 (SODA '12)*, Kyoto, Japan.

[10] J. Jensen, M. Christensen, M. Murthi, and S. Jensen. Evaluation of mfcc estimation techniques for music similarity. In *European Signal Processing Conference 2006 (EUSIPCO '06)*.

[11] A. Kumar, A. Tewari, S. Horrigan, M. Kam, F. Metze, and J. Canny. Rethinking speech recognition on mobile devices. In *2nd International Workshop on International User Interfaces for Developing Regions (IUI4DR)*, Palo Alto, CA, 2011.

[12] E. Kushilevitz and Y. Mansour. Learning decision trees using the fourier spectrum. In *ACM Symposium on Theory of Computing 1991 (STOC '91)*.

[13] B. Logan. Mel frequency cepstral coefficients for music modeling. In *International Symposium on Music Information Retrieval 2000 (ISMIR '10)*.

[14] H. Lu, A. J. B. Brush, B. Priyantha, A. K. Karlson, and J. Liu. Speakersense: energy efficient unobtrusive speaker identification on mobile phones. In *9th International Conference on Pervasive Computing 2011 (Pervasive '11)*, pages 188–205, San Francisco, CA.

[15] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *7th International Conference on Mobile Systems, Applications, and Services 2009 (MobiSys '09)*, pages 165–178, Poland.

[16] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *6th ACM conference on Embedded network sensor systems 2008 (SenSys '08)*, pages 337–350, Raleigh, NC, USA.

[17] L. Muda, M. Begam, and I. Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *Journal of Computing*, 2(3):138–143, 2010.

[18] A. Mueen, S. Nath, and J. Liu. Fast approximate correlation for massive time series data. In *ACM SIGMOD International Conference on Management of Data 2012 (SIGMOD '10)*, pages 171–182, Indiana, USA.

[19] K. Murty and B. Yegnanarayana. Combining evidence from residual phase and mfcc features for speaker recognition. *Signal Processing Letters, IEEE*, 13(1):52 – 55, jan. 2006.

[20] S. Nirjon, R. Dickerson, Q. Li, P. Asare, J. Stankovic, D. Hong, B. Zhang, G. Shen, X. Jiang, and F. Zhao. Musicalheart: A hearty way of listening to music. In *10th ACM Conference on Embedded Networked Sensor Systems 2012 (SenSys '12)*, Toronto, Canada.

[21] R. O'Donnell. Some topics in analysis of boolean functions (tutorial). In *ACM Symposium on Theory of Computing 2008 (STOC '08)*.

[22] A. Wang. An industrial-strength audio search algorithm. In *International Symposium on Music Information Retrieval 2003 (ISMIR '03)*.

[23] J. Watkinson. *The Art of Digital Audio*. Newton, MA, USA, 1993.