Audio Keyword Reconstruction from On-Device Motion Sensor Signals via Neural Frequency Unfolding

TIANSHI WANG*, University of Illinois at Urbana-Champaign
SHUOCHAO YAO*, George Mason University
SHENGZHONG LIU, University of Illinois at Urbana-Champaign
JINYANG LI, University of Illinois at Urbana-Champaign
DONGXIN LIU, University of Illinois at Urbana-Champaign
HUAJIE SHAO, University of Illinois at Urbana-Champaign
RUIJIE WANG, University of Illinois at Urbana-Champaign
TAREK ABDELZAHER, University of Illinois at Urbana-Champaign

In this paper, we present a novel deep neural network architecture that *reconstructs* the high-frequency audio of selected spoken human words from low-sampling-rate signals of (ego-)motion sensors, such as accelerometer and gyroscope data, recorded on everyday mobile devices. As the sampling rate of such motion sensors is much lower than the Nyquist rate of ordinary human voice (around 6kHz+), these motion sensor recordings suffer from a significant frequency aliasing effect. In order to recover the original high-frequency audio signal, our neural network introduces a novel layer, called the *alias unfolding layer*, specialized in expanding the bandwidth of an aliased signal by reversing the frequency folding process in the time-frequency domain. While perfect unfolding is known to be unrealizable, we leverage the sparsity of the original signal to arrive at a sufficiently accurate statistical approximation. Comprehensive experiments show that our neural network significantly outperforms the state of the art in audio reconstruction from motion sensor data, effectively reconstructing a pre-trained set of spoken keywords from low-frequency motion sensor signals (with a sampling rate of 100-400 Hz). The approach demonstrates the potential risk of information leakage from motion sensors in smart mobile devices.

CCS Concepts: • **Computing methodologies → Artificial intelligence**; • **Computer systems organization → Embedded and cyber-physical systems**; • **Human-centered computing → Ubiquitous and mobile computing**.

Additional Key Words and Phrases: Motion sensors;Deep learning; Time frequency analysis

ACM Reference Format:
Tianshi Wang, Shuochao Yao, Shengzhong Liu, Jinyang Li, Dongxin Liu, Huajie Shao, Ruijie Wang, and Tarek Abdelzaher. 2021. Audio Keyword Reconstruction from On-Device Motion Sensor Signals via Neural Frequency Unfolding. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 3, Article 132 (September 2021), 29 pages. https://doi.org/10.1145/3478102

*Both authors contributed equally to this research.

Authors' addresses: Tianshi Wang, tianshi3@illinois.com, University of Illinois at Urbana-Champaign; Shuochao Yao, shuochao@gmu.edu, George Mason University; Shengzhong Liu, sl29@illinois.edu, University of Illinois at Urbana-Champaign; Jinyang Li, jinyang7@illinois.edu, University of Illinois at Urbana-Champaign; Dongxin Liu, dongxin3@illinois.edu, University of Illinois at Urbana-Champaign; Huajie Shao, hshao5@illinois.edu, University of Illinois at Urbana-Champaign; Ruijie Wang, ruijiew2@illinois.edu, University of Illinois at Urbana-Champaign; Tarek Abdelzaher, zaher@illinois.edu, University of Illinois at Urbana-Champaign.

2474-9567/2021/9-ART132 $15.00
https://doi.org/10.1145/3478102

1 INTRODUCTION

This paper presents a deep learning model for *reconstructing* selected spoken keywords from (ego-)motion sensor data. When a device, such as a mobile phone or a smart watch, is in the vicinity of sound produced by nearby human speech, the sound will cause device vibration. On-device accelerometers and gyroscopes can perceive these vibration signals, albeit at a much lower frequency. We show how to "unfold" the recorded accelerometer and gyroscope signals in the frequency domain. The unfolding can be trained to reconstruct specific keywords making them recognizable with a probability that significantly exceeds random guessing (e.g., over 85% accuracy, for each of the 10 digits).

Reconstruction, as opposed to machine classification, has the potential to leverage human audio skills to complement the machine in signal identification. We show that a malicious app with access to motion sensors installed on a victim's smart device can reproduce an audio-like time-series from which a person can identify a set of pre-determined keywords. This opens up possibilities for leaking information, such as credit card numbers (e.g., when ordering delivery), bank accounts, birth dates, zip codes, and social security numbers. For example, installing the app on a small-business employee's phone used for order/reservation processing can quickly amass credit card numbers used for the reservations. Importantly, a person recovering these numbers from the reconstructed audio will generally not need to understand the rest of the conversation. The paper is a cautionary note whose purpose is to motivate viewing accelerometers and gyroscopes as potential audio recording devices when it comes to permission management on our mobile devices in the near future.

The key innovation presented in this paper lies in the development of a new neural network architecture that *unfolds aliased signals in the frequency domain*, thereby allowing approximate sound reconstruction. The approach is inspired by recently published work on advantages of designing neural networks to operate directly on frequency domain signals [27, 41, 42], when the underlying task has strong frequency domain interpretations. Our deep neural network ingests the low sampling-rate signals recorded by the accelerometer and gyroscope and outputs a corresponding high sampling-rate audio.

Previous research has suggested the possibility of using motion sensors to recognize hotwords [3, 29, 44], and detect speakers' identity and gender [29]. In essence, these recognition applications perform *classification* tasks, where motion sensor signals are categorized into some pre-defined categories. Reconstruction received comparatively less attention [5]. Our results show improved performance over previous reconstruction solutions in terms of audio waveform similarity metrics.

Reconstruction of audio signals from lower sampling rate (ego-)motion sensor data (as opposed to *classification*) offers two main challenges. First, waveform reconstruction from low sampling rate motion sensor signals involves recovering the lost high frequency information. As the sampling rate of motion sensors (100-400 Hz) is much lower than the Nyquist frequency of human speech (6000 Hz) [1], the motion sensor signals suffer from a strong aliasing effect. Theoretically, an undersampled signal no longer has complete information to recover its original waveform. Fortunately, human speech has unique acoustic patterns over time and frequency, which significantly narrows down feasible solutions for frequency unfolding. With sufficient data, a deep neural network can exploit this sparsity property to learn the right unfolding patterns for a specific set of keywords and recover those keywords to an adequate degree of approximation, even when the testing set includes new instances and speakers not in the training set. A challenge is to wisely design this network in order to facilitate effective learning of data unfolding in the frequency domain.

Second, motion sensors deform (and not just downsample) the sound waves. In microphones, the vibrations produced by human speech hit the diaphragm, a very thin film specifically designed to be sensitive to acoustic stimuli. Motion of the diaphragm converts the mechanical energy of an acoustic wave into electrical signals. In contrast, for motion sensors, such as accelerometers, the sound wave must first *move* the device, causing vibration and thus a reading on its internal accelerometer that captures the vibration signal. Since the inertia

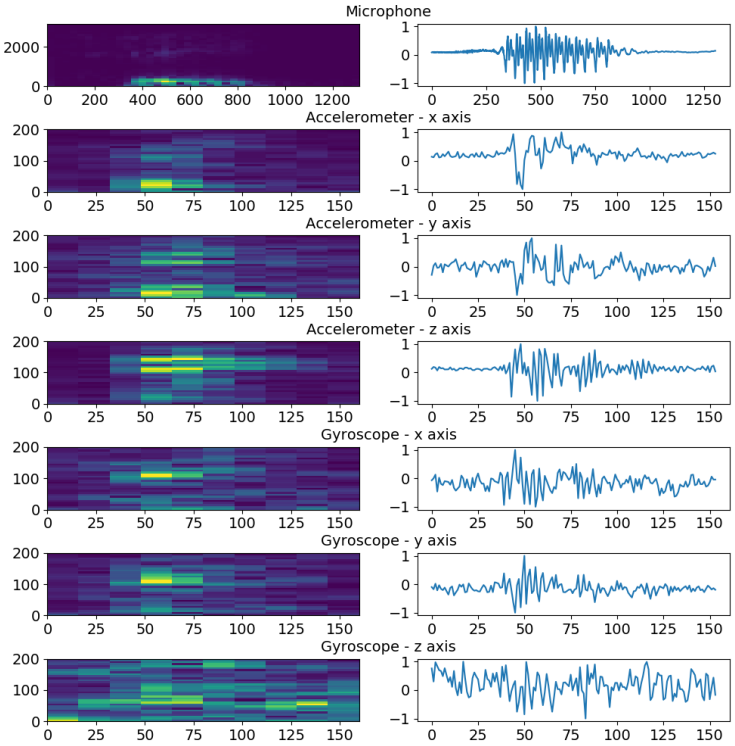


Fig. 1. Different responses of the microphone and the motion sensors to the same human speech sound wave

of the device is much higher than that of a microphone's diaphragm, it acts as a significant low-pass filter (i.e., offers a higher signal attenuation with increased frequency). Unlike aliasing (caused by down-sampling) that simply shifts high-frequency components lower in the spectrum, a low-pass filter attenuates those components, causing further information loss. As shown in Figure 1, the motion sensors have very different responses to the same nearby human speech sound wave compared with the microphone. Our deep neural network must not only upsample the motion sensor signals, but also undo the inertial filtering caused by the device.

To resolve the above challenges, our contribution is two-fold. First, we introduce an innovative neural network layer called the *alias unfolding layer*. It upsamples low sampling-rate signals into higher sampling-rate signals. The alias unfolding layer doubles the sampling rate of an aliased signal by reversing folding in the frequency domain. It first uses multi-resolution Short-time Fourier transform (STFT) to convert the input signal time series into spectrograms, and then decomposes the original time-frequency values of the spectrograms into a high-frequency part and a low-frequency part, based on learnable parameters. By concatenating the two parts along the frequency dimension and transforming it back to the time domain, the output signal doubles in resolution. In our speech reconstruction task, we stack multiple alias unfolding layers to reach the target sampling rate, and use multiple spectrograms within each layer (that differ in their time-frequency resolution) to optimize across hyperparameters of the Short-time Fourier transform itself.

Second, we propose using the STFNet [42] time/frequency convolution layers to manipulate the intermediate results between two alias unfolding layers. We customize the conventional convolution operation to adapt to

frequency domain operation. One alias unfolding layer and multiple STFNet time/frequency convolution layers make up a frequency expanding module, which is the building block of our neural network.

The training process is driven by a customized *hierarchical MFCC-DTW loss function*. It not only evaluates loss based on the final output but also considers the intermediate results, giving extra supervision for the reconstruction process. The rest of this paper is organized as follows. Section 2 presents related work. Section 3 reviews the needed background underlying our algorithms. Section 4 describes our neural network design. An experimental evaluation is presented in Section 5. Section 6 discusses the limitations of the work. The paper concludes with Section 7.

2 RELATED WORK

2.1 Speech Recognition by Motion Sensors

Several prior publications attempted to recognize/detect speech from motion sensor signals over the past decade. Unlike ours, most prior attempts solved classification problems. L. Zhang *et al.* [44] propose using the accelerometer in a smartphone to detect spoken hotwords such as “Okay Google" and “Hi Galaxy". They demonstrate the feasibility of using an accelerometer for voice control. Y. Michalevsky *et al.* [29] introduce a system that uses gyroscope signals to recognize speakers’ gender, uncover their identity, and recognize 11 isolated digits. Their experiments disclose the potential privacy concern that malicious mobile phone apps can steal gender and identity information, and even eavesdrop on sensitive data, such as credit card numbers. However, their accuracy at recognizing the 11 digits is only 26% in the speaker-independent case and 65% in the speaker-dependent case. Han *et al.* [16] leverage a Time Interleaved Analog-Digital-Conversion (TI-ADC) technique to approximate a high sampling-rate by using multiple non-acoustic sensors with lower sampling rates to reconstruct human speech. In their experiments, they use $2 \sim 16$ homogeneous non-acoustic sensors (either geophone, accelerometer or gyroscope) of a 500 or 1000 Hz sampling-rate to approximate a 2~8 kHz total sampling-rate, assuming the multiple sensors have perfect offset under time synchronization. The number of sensors and their relatively high sampling-rate places this approach beyond the capabilities of ordinary smart mobile devices. Anand *et al.* [3] exploit speech detection and recognition based on accelerometer signals generated from a mobile phone’s loudspeaker while playing audio in the loudspeaker mode. They implement a speech recognition function that relies on word isolation and keyword search, which is inherently a classification problem. Closest to our work is recent research [5] that used STFT to convert sensor signals into spectrograms, and adopted an encoder-decoder neural network to reconstruct high frequency speech signals. In section 5.2, we compare our model to it, showing a significant improvement in *reconstruction quality*.

2.2 Audio Super-resolution

Audio super-resolution, also known as speech bandwidth extension, is about extending the bandwidth of an audio signal in order to gain better audio quality. Audio super-resolution is a research topic that has been studied for decades, and its most common application is speech enhancement, which aims to transmit narrowband speech to wideband speech in telephone systems. The extension of bandwidth is usually based on some models that map the narrowband features onto wideband spectral information [6, 20, 34]. Early approaches including [4, 12, 18, 35, 43] employ the codebook mapping technique, which uses one codebook recording a representative set of the feature vectors of narrowband speech frames and another codebook recording the spectral information of the counterpart wideband speech frames. When doing bandwidth extension, input narrowband information is classified to an entry in the narrowband codebook based on the best-matching principle, and the extension band information is generated by finding the counterpart in the wideband codebook. Linear mapping is another type of model adopted for bandwidth extension [10, 30]. These models use a linear transfer model, which is obtained through off-line training, to estimate the wideband spectral information based on the input narrow band features.

Furthermore, probabilistic models including Gaussian Mixture Models (GMMs) [31, 38] and Hidden Markov Models (HMMs) [7, 19, 20, 40] are taken to introduce the non-linear mapping between narrowband and wideband features.

More recent methods adopt neural networks to predict the missing high frequency information of a low resolution audio signal in a data-driven manner. Some solutions use fully-connected neural networks for bandwidth extension [2, 13, 15, 21]. For example, Li *et al.* [23] propose a deep neural network (DNN) for expanding 4kHz bandwidth speech to 8kHz. The neural network takes the low-frequency spectrum features as input and predicts the entire wideband spectrum. Other work leverages convolutional neural networks (CNNs) to better exploit the local motif features of the input data [14, 22]. For example, Lim *et al.* [25] introduce a CNN called Time-Frequency Network (TFNet) for audio upsampling. TFNet contains two branches: one takes Fast Fourier Transformed (FFT) data as input and learns frequency domain features; the other branch takes raw time series data and learns time domain features. The outputs from the two branches are jointly used for generating the upsampling audios. A state-of-the-art neural network for audio super-resolution is presented in [22]. Inspired by auto-encoders, the authors introduce a convolutional neural network architecture with a bottleneck design, known to encourage the model to learn features hierarchically. In our evaluation section, we take this network as a baseline, showing the advantage of our method over this audio super-resolution approach. Other types of networks include recurrent neural networks (RNNs) [26], dilated convolutional neural networks [14], and generative adversarial networks (GANs) [11, 24, 32].

Like our reconstruction task, audio super-resolution involves restoring the high frequency part of an undersampled time series signal. However, there are two major differences between our task and audio super-resolution. First, audio super-resolution tasks typically focus on bandwidth extension by a factor in the range of 2-6 (e.g., a 4 kHz speech signal to an 8 kHz signal in [9]). In contrast, we start at a signal in the 100-400 Hz range and extend its bandwidth by a factor of 16-64. Second, audio super-resolution upsamples audio to audio, while we have to address the additional distortion introduced by inertial low-pass filtering of the recording device.

3 BACKGROUND

3.1 Nyquist Rate and Aliasing

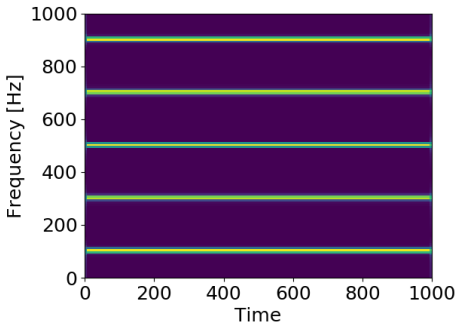
The Nyquist rate refers to the lowest sampling rate required for error-free reconstruction of a finite bandwidth signal, $x(t)$. If the bandwidth of $x(t)$ is B, the Nyquist rate for this signal is $2B$. If the sampling rate is below the Nyquist rate, the reconstructed function will suffer from aliasing, which prevents correct reconstruction.

Informally, aliasing means that some of the high frequency part of the original signal is folded onto the low frequency part (in the frequency domain). Accordingly, multiple frequency components, when sampled at lower rates, map exactly to the same frequency, where they are additively superimposed. Restoring the original components from the resulting superposition is no longer unique. Generally, the perceived frequency to which any signal frequency f maps when sampled at any sampling rate f_s can be calculated by:

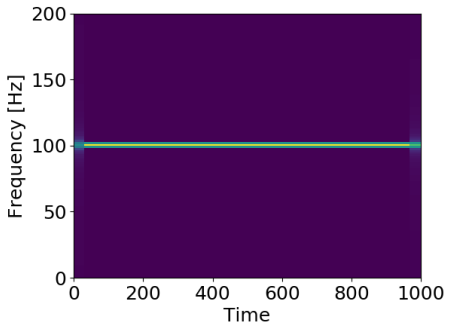
$$f_{perceived} = |f - f_s \cdot ROUND(\frac{f}{f_s})| \tag{1}$$

Equation (1) shows that for an undersampled signal, the high frequency part that exceeds $f_s/2$ is folded onto range $0 \sim f_s/2$. Figure 2 shows an example to illustrate this frequency folding effect. Figure 2.a and 2.b are the STFT spectrograms of the same signal but under different sampling rates. The brighter pixels in the spectrograms indicate bigger magnitudes at the corresponding frequency components. This signal contains five components of frequency 100 Hz, 300 Hz, 500 Hz, 700 Hz and 900 Hz, respectively. In Figure 2.a, when the sampling rate is 2000 Hz, exceeding the Nyquist rate for the highest frequency wave (900 Hz), five bright bands appear at the correct positions on the frequency axis in the left spectrogram. If the sampling rate drops to 400 Hz, according to

Equation 1, all the 5 waves overlay at 100 Hz. Indeed, as shown in Figure 2.b, only one bright band appears at 100 Hz on the frequency axis. This effect is a key challenge that our neural network must overcome, since the motion sensor signal suffers from significant aliasing.



(a) The STFT spectrogram when sampling rate = 2000 Hz



(b) The STFT spectrogram when sampling rate = 400 Hz

Fig. 2. An example to show the aliasing effect. The signal contains 5 constant frequency waves (100 Hz, 300 Hz, 500 Hz, 700 Hz and 900 Hz) showing different spectrograms when oversampling(2.a) and undersampling(2.b)

3.2 STFNet-Convolution Operation

We incorporate the STFNet-convolution operation, proposed by Yao et al. in [42], into our deep neural network architecture. The idea behind the STFNet-convolution operation is to use a trainable kernel to convolve on the STFT spectrogram representation of a signal. We denote the STFT spectrogram as tensor $X \in \mathbb{C}^{t \times f \times d}$ where t is the length of the time dimension, f is the length of the frequency dimension, and d is the length of the feature dimension. The convolution kernel is $W \in \mathbb{C}^{1 \times m \times d \times o}$ where m is the kernel size along the frequency dimension, and o is the output feature dimension. There are two main differences between the STFNet-convolution operation and the conventional convolution operation used for images in the computer vision domain. First, as an STFT spectrogram has real and imaginary parts, STFNet-convolution maintains two sets of convolution kernels for real and imaginary parts, separately. Another difference is the padding technique. Conventional convolution operations either take *no padding* or *zero padding*. For the STFNet-convolution operation, on one hand, one does not want to do *no padding* because it shrinks the output and loses time-frequency domain information, making the STF spectrogram irreversible to the original length time sequence. On the other hand, zero padding adds zeros to the spectrogram which is not a neutral operation in the frequency domain (i.e., it changes the reconstructed signal) [42]. To keep the shape of inputs and outputs consistent without introducing extra information, STFNet-convolution does padding by appending the previously cut out "duplicated" frequency parts back following the conjugate symmetry property of Discrete Fourier Transform (DFT) for real-valued time series:

$$X_{[\cdot,\tau-k,\cdot]} = X_{[\cdot,-k,\cdot]} = X^*_{[\cdot,k,\cdot]} \tag{2}$$

where τ represents the original length of the frequency components in the STFT spectrogram. X^* is the complex conjugation. The STFNet-convolution operation can thus be expressed as:

$$Y = SpectralPad(X) \circledast W \tag{3}$$

4 DEEP NEURAL NETWORK DESIGN

In this section, we describe the details of our deep neural network design. We first look into the four key components of the network: (i) the alias unfolding layer, (ii) the STF frequency convolution layer, (iii) the residual STF time convolution module and (iv) the hierarchical MFCC-DTW loss function. Finally, we combine them together and give an overview of our complete network architecture.

4.1 Alias Unfolding Layer

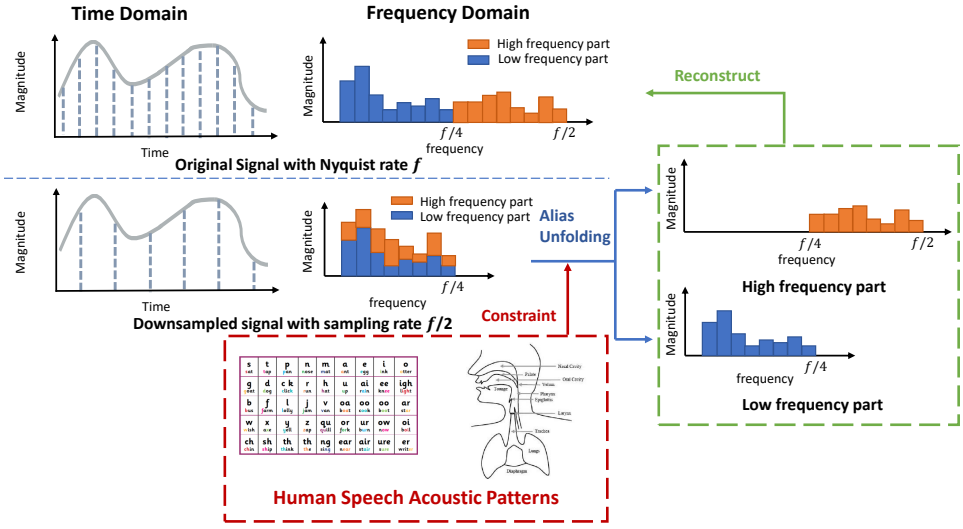


Fig. 3. The intuition of designing alias unfolding layer. The undersampled signal can be separated into low-frequency part and high-frequency part. Though theoretically there are infinite ways of separation, the search space is constrained by human speech acoustic patterns in our case.

The alias unfolding layer aims to upsample low sampling-rate signals into high sampling-rate signals. As we described earlier in Section 3.1, the greatest challenge in upsampling low sampling-rate motion sensor signals to high sampling-rate human speech signals lies in recovering from the aliasing effect. Figure 3 shows a downsampled signal with a sampling rate $f/2$ that is half of the Nyquist rate f of its original signal. In this case, the right half high-frequency part in the spectrogram of the original signal will fold onto the left half low-frequency part. In order to reconstruct the original signal from the downsampled signal, we need to reverse this folding process, which involves separating the low-frequency part and high-frequency part from the overlay spectrogram and concatenate them together to reproduce the original signal spectrogram. Theoretically, the number of ways to separate the two parts is infinite. However, as the target signal is spoken keywords, the search space is constrained by human speech acoustic patterns.

Based on the above intuition, the alias unfolding layer takes a time series as the input, converts it into an STFT specotrgram, expands its frequency bandwidth by 2x, and finally applies inverse STFT to convert it back to a time series. In Figure 4, we take the first alias unfolding layer in the network as an example to illustrate this design. The input is the motion sensor signal in time sequence form with length 512. Since we have 3 axes for each motion sensor, x, y and z, the number of channels is 3. Next, we pass the time sequence through multi-resolution STFT. Here, we have 3 different resolutions. The top spectrogram in Figure 4 has the highest time resolution and

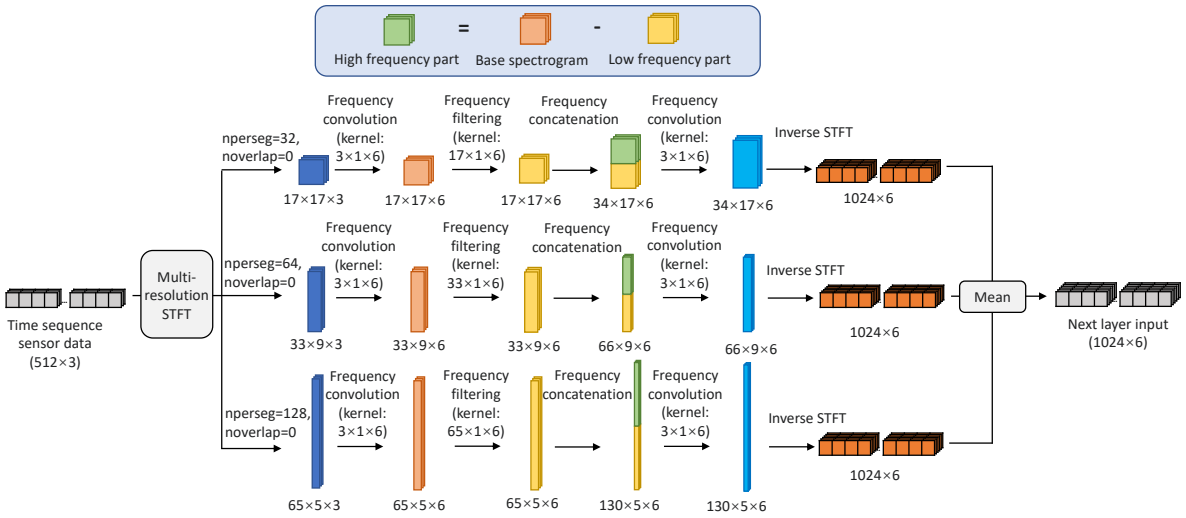


Fig. 4. The internal structure of an alias unfolding layer. It doubles the frequency of the input time series data.

lowest frequency resolution (17×17), while the bottom one has the lowest time resolution and highest frequency resolution (65×5). Next, each spectrogram undergoes a frequency convolution with kernel shape (3×1×6). This operation does not change the time-frequency resolution, but increases the number of channels to 6 so as to increase the scale of the neural network. The values of the convolution kernels are learnable parameters. The convolution outputs then go through filtering operations on the frequency axis. Again, this step does not change the time-frequency resolution. It extracts the values of high frequency parts from the original spectrograms. The low frequency part can then be calculated by subtracting the high frequency part (which is the yellow one in figure), from the spectrograms (which is the red one). Concatenating the low frequency part and high frequency part together, we get the upsampled spectrogram. The frequency resolution is doubled. Following this stage, the output goes through another frequency convolution. An inverse Short-time Fourier transform brings the spectrograms back to the time domain. Finally, we use a mean operation to combine the outputs from the multi-resolution process. At this point, the output can be used as the input for the next layer (like in Figure 4), or we can use mean operation on the feature dimension and get a 1-D time sequence as the final reconstructed audio output.

4.2 STF Frequency Convolution Layer

The STF frequency convolution layer is based on the STFNet-Convolution operation from Yao et al. [42]. The STFNet-Convolution operation is effective at learning local motifs in the frequency domain and has outstanding performance in extracting features in physical sensor data. The advantage of frequency-domain convolution is that it extracts localized spectral features independently from their location in the frequency spectrum. Since aliasing (folding) essentially moves parts of the spectrum to a different position in the frequency domain, we conjecture that an operation that looks for features independently of their location might be of help. Another reason for features to shift in the frequency domain is because of differences in utterance speed and pitch. A higher-pitched voice pronouncing the same words, for example, will create similar features to a deeper voice in the frequency domain, but shifted towards higher frequencies. A convolution operation can recognize those features regardless of their position in the spectrum.

An STF frequency convolution layer takes a vector $v \in \mathbb{R}^{l \times d}$ as input, where l is the length of the time series, and d is the feature dimension. The STF frequency convolution layer first uses multi-resolution STFT with 3 different window sizes $[\tau_1, \tau_2, \tau_3]$ to transform v into 3 spectrograms $\mathbf{STFT}^{\tau_1}(X)$, $\mathbf{STFT}^{\tau_2}(X)$ and $\mathbf{STFT}^{\tau_3}(X)$. Then each spectrogram is separately passed into an STFNet-Convolution operation. Each output spectrogram then goes through an inverse STFT and becomes time series v'_1, v'_2, v'_3, while each vector is in shape of $2l \times d$. Finally, we average the three vectors v'_1, v'_2 and v'_3, and have $v' \in \mathbb{R}^{2l \times d}$ as the output.

4.3 Residual STF Time Convolution Module

It often happens in neural networks that the introduction of additional layers offers no better or even worse performance, compared to shallower counterparts. This is known as network *degradation*. It happens when the additional layers introduce higher training errors. Residual networks were introduced to solve the degradation problem. Residual networks use *skip connections* (i.e., connections among non-consecutive layers), offering the freedom to skip ineffective layers (depending on learned weights) thereby preventing degradation. We incorporate residual network design into the STF time convolution module.

Recall also, as argued earlier, that our challenge is not only the conversion from a low sampling rate signal to a high sampling rate signal, but also a compensation for inertial distortion of the signal (thanks to the low-pass-filtering effect of device inertia). We need to learn an appropriate signal compensation. The time-domain convolution operation can approximate an arbitrary filter and hence is helpful for removing the inertial distortion.

As shown in Figure 5, a residual STF time convolution module consists of 3 residual units. In each residual unit, two STF time convolution layers are stacked. A residual unit is expressed as: $y = F(x, \{W_0, W_1\}) + x$, where x and y are the input and output vectors of the residual unit, $F(x, W_0, W_1)$ is the residual mapping to be learned, W_0 is the convolution kernel matrix for the first STF time convolution layer and W_1 is for the second layer.

The STF time convolution layer is the building block of the residual STF time convolution module. The input of an STF time convolution layer is the vector x_i, where $i \in \{0, 1\}$ denotes the layer in the residual unit. The input x_i is first converted to spectrogram $X_i \in \mathbb{C}^{t \times f \times d}$ with STFT. The spectrogram X_i then undergoes convolutions with kernel $W_i \in \mathbb{C}^{m \times 1 \times d \times o}$, where W_i is the convolution kernel matrix, m is the kernel size, d is the input feature dimension and o is the output feature dimension. Here, convolution operations only apply along the time axis. Unlike in STFNet-convolution operation, where the spectrogram X_i needs special padding technique to keep the input and output in the same shape, STF time convolution layers just use conventional zero-padding to ensure the shape of the output tensor is not shrunk. This is because padding zeros along the time axis does not introduce additional information since zeros are interpreted as no information in the time domain. After time convolutions, the output X'_i goes through an inverse STFT and becomes vector x'_i. If $x_0 = x$ which means it is the output of the first STF time convolution layer in a residual unit, it will be passed into the second layer. For x'_1 which is the output of the second STF time convolution layer in a residual unit, it will be summed up with x and become the final output of the residual unit y.

4.4 Hierarchical MFCC-DTW Loss

As our deep neural network upsamples the motion sensor signals in a hierarchical manner, we need to ensure that each level of the frequency expanding module learns the correct features and outputs results that are close to the ground truth audio signal of the same sampling rate. Thus, we add supervision to the output of each frequency expanding module. We define the loss function $\mathcal{L}(\hat{d_1}, d_1, ...\hat{d_N}, d_N)$ as follows:

$$\mathcal{L}(\hat{d_1}, d_1, ..., \hat{d_N}, d_N) = \frac{\sum_{i=0}^{N} 2^i L(\hat{d_i}, d_i)}{\sum_{i=0}^{N} 2^i} \tag{4}$$

where $\hat{d_i}$ is the output of the ith frequency expanding module, d_i is the ground truth audio downsampled to the same sampling rate as $\hat{d_i}$, N is the number of frequency expanding modules, and $L(\hat{d_i}, d_i)$ is the loss we defined for evaluating the difference between the reconstructed result $\hat{d_i}$ and ground truth d_i. The factor, 2^i, used as the weight assigned to the loss which comes from the ith frequency expanding module. We assign a higher weight to the loss of the frequency expanding module that is closer to the end of the neural network, since (intuition suggests that) the quality of intermediate results that are closer to the end of the network has a higher impact on the quality of the final output audio.

In $L(\hat{d_i}, d_i)$, $\hat{d_i}$ is supposed to have similar features as the ground truth d_i, by metrics of human perception. Therefore, we use Mel-frequency cepstral coefficients (MFCC) to extract features that can well represent human speech. During data collection and preprocessing, we found that time offsets exist between input motion sensor signals and their corresponding ground-truth audio signals. Hence, we choose dynamic time warping (DTW) to calculate the difference between the MFCC features of $\hat{d_i}$ and d_i, as DTW can measure similarity of shape between two temporal sequences. In practice, to reduce the computational complexity, we constrain the offset of two matching points in DTW to be less than 1/16 of the time axis length. As a result, the loss function between $\hat{d_i}$ and d_i can be formulated as:

$$L(\hat{d_i}, d_i) = \boldsymbol{DTW}(\boldsymbol{MFCC}(\hat{d_i}), \boldsymbol{MFCC}(d_i)) \tag{5}$$

4.5 Network Architecture Overview

The input to the neural network is a fused signal, **s**, composed of accelerometer and gyroscope recordings from a smart mobile device.

Let R_1 be the sampling rate, and T_1 be the number of samples of either time series. The accelerometer time series thus comprises samples denoted $(x_{0,t}, y_{0,t}, z_{0,t})$, from $t = 0$ to T_1/R_1. Similarly, the gyroscope time-series comprises samples denoted $(x_{1,t}, y_{1,t}, z_{1,t})$ for the same time interval. Each sample represents the readings of the x, y, and z axes of the respective sensor. In practice, the recordings from the two motion sensors are not synchronized, so we interpolate the signals to realize alignment. The output of the network is an audio signal $\mathbf{y} = \{a_{1/R_2}, ..., a_{T_2/R_2}\}$, where R_2 is the target sampling rate ($R_2 > R_1$), T_2 is the number of samples, and a_t is the audio sample at time t. We denote the upsample ratio by $r = R_2/R_1$. Since the sampling rate of motion sensors is no more than 400 Hz, whereas intelligible human speech requires around 5000 Hz [33], we require $r > 10$.

In our neural network, we upsample the motion sensor signals in a hierarchical manner. In each alias unfolding layer, we aim at expanding the frequency bandwidth by 2. To get the final audio signal with sampling rate, R_2, we must concatenate $log_2(R_2/R_1)$ alias unfolding layers. For simplicity, we constrain R_2 to be a power of 2 multiple of R_1. As detailed shortly, in addition to the alias unfolding layers that upsample the signals, we have STF frequency/time convolution layers to handle noise and undo the uneven attenuation in the frequency spectrum caused by device inertia (separately from aliasing). An alias unfolding layer, an STF frequency convolution layer and several STF time convolution layers make up a frequency expanding module, which is the building block of our neural network. Parameter optimization of the deep neural network is driven by the loss function, $\mathcal{L}(d_1, \hat{d_1}, ...d_N, \hat{d_N})$, where d_i is the intermediate result output from the ith frequency expanding module, $\hat{d_i}$ is the ground truth audio measured at the same sampling rate as d_i, and N is the number of frequency expanding modules (i.e., the number of times we do unfolding). The goal of the network is to minimize the loss function $\mathcal{L}(d_1, \hat{d_1}, ...d_N, \hat{d_N})$.

Figure 5 illustrates our network architecture. The first frequency expanding module is blown-up to show internal detail. The remaining frequency expanding modules are similar, so their details are not shown. We separate input **s** into s_{acc} and s_{gyro}, then process the data from the two types of sensors separately. Each motion sensor signal is fed into a frequency expanding module. Within a frequency expanding module, an alias unfolding

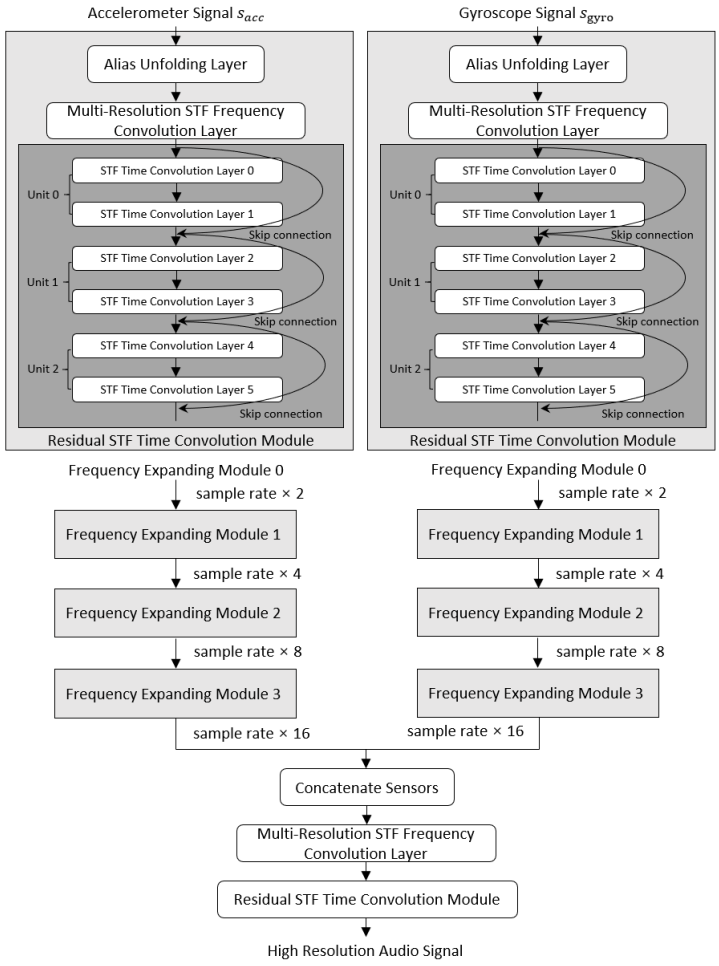


Fig. 5. Deep neural network architecture overview. The signals from the two motion sensors first separately go through a series of frequency expanding modules and are jointly processed in the end.

layer increases the sampling rate of the input signal by 2x. Following is one STF frequency convolution layer and three STF time convolution layers. While this design choice is adjustable, we found that concatenating three STF time convolution layers as a residual sub-network yields better performance. Several expanding modules are stacked. In the figure, we set the target sampling rate to be 16x of the original sampling rate, so we stack four frequency expanding modules in total (although this is adjustable depending on the ultimately ratio desired). When the two sensor data streams finish the aforementioned processing in parallel, we concatenate the two upsampled time series on their feature dimension, and pass into another STF frequency convolution layer and a residual STF time convolution module. At this point, the output is the time series with 16x higher resolution than the input signal, $\mathbf{s}$. We average this time series on its feature dimension, and provide $\mathbf{y}$ as the final high resolution audio signal output.

5 EVALUATION

In this section, we evaluate the efficacy of our neural network at reconstructing keywords from motion sensor signals. First, we introduce the experimental setup, hardware device specifications, datasets, and baselines. Then, we analyze the performance of the neural network under various settings and compare with baselines. Finally, an ablation study is included to verify the necessity of our design decisions.

5.1 Experimental Setup

The experimental setup includes two pieces of hardware: a speaker for playing human speech audio and a smart mobile device for receiving motion sensor signals (and microphone signals as ground truth for training). The microphone signals are used for labeling motion sensor data. They are not used as input data for the neural network. In the following experiments, unless otherwise stated, we use the regular setting. In the regular setting, we use a Google Pixel 1 smart phone as the mobile device. The volume of sound played by the speaker is 80 dB. The speaker and the smart phone are placed on a table. The distance between the smart phone and the speaker is 5 inches. The recording environment is in an office room with background noise of 40 dB. In some experiments, the settings will be changed in order to analyze the influence of different factors including sampling rates, speech volume, models of the mobile devices, and the distances between the speaker and the mobile device. Such changes will be noted as needed. It is worth noting that except for assuming the human speech sound comes from an external source, we also set up experiments of using the internal speaker of the mobile device to play the human speech in Section 5.7.

In our experimental setup, the mobile device is only used for recording motion sensor signals (and microphone signals as ground truth for training). As the microphone and motion sensors are not genuinely synchronized, we record the timestamp of each data sample and apply cubic spline interpolation on them to achieve approximate synchronization. The recordings will later be transferred to a server desktop for model training and prediction.

5.2 Hardware Devices

The default smart phone that we use to collect motion sensor signals is a Google Pixel 1 (2016). It is equipped with a Bosch Sensortec BMI160 low-power IMU [17]. The maximum sampling rate for its accelerometer and gyroscope is 400 Hz. We also tested another smart phone, Google Nexus 5 (2013). Its IMU model is InvenSense MPU-6515 six-axis MEMS MotionTracking device [28]. The maximum sampling rate for its accelerometer and gyroscope is 200 Hz. Moreover, we tested a wearable mobile device, the Fitbit Versa Lite smart watch, to collect motion sensor data. The Fitbit Versa Lite has an accelerometer and gyroscope with maximum sampling rate of 100 Hz. The speaker we used for playing human speech audio is Marshall Stanmore II. We trained the deep neural network on a desktop with an Nvidia TITAN V GPU. The CPU model of the desktop is Intel Core i7-5820K CPU 3.30GHz.

5.3 Datasets

We used two datasets in the experiments:

(1) Spoken 10 Digits Dataset [8]: This dataset contains 30,000 audio samples of 10 spoken digits (0∼9). The 30,000 audio samples are recorded by 60 different speakers. Among the speakers, 48 are males and 12 are females. The speakers have different accents including English, German, Chinese, and Spanish. Each speaker says each digit 50 times, generating 500 samples for the ten digits. The audio is recorded in a quiet environment, and the recording quality is good.
(2) Google Speech Commands Dataset [39]: This dataset contains over 50,000 one-second-long utterances of 20 short speech commands spoken by thousands of different people. Each speaker says each word for 1∼5 times. This dataset is obtained from a crowd sourcing program, so the quality of audio recording is not guaranteed. The audio contains different levels of noise, and some snippets are completely garbled. The

speakers have different accents and include males and females. Compared to the spoken 10 digits dataset, the speech commands dataset has a much higher diversity and a lower quality. Thus, it suitably challenges the generalization ability and robustness of our reconstruction model.

We randomly divide each dataset into a 90% training set and a 10% testing set. Each motion sensor sample of a spoken word is padded to the shape of $512 \times 2 \times 3$ (data length $\times$ 2 sensors $\times$ 3 axes) as the input of the neural network.

5.4 Baselines

We perform two sets of baseline comparisons:

5.4.1 Recognition/Classification Baselines. This set compares us with recognition/classification baselines in terms of accuracy. Classification is a simpler problem than reconstruction. Thus, we do not aim to beat classification accuracy. Rather, we aim to approach it with our reconstruction scheme. We indeed demonstrate that our reconstruction model outperforms basic recognition/classification models and nearly ties a state-of-the-art model. The baselines are:

(1) *K-nearest neighbors (KNN)*: This is a very basic non-neural network-based classification model. We use STFT to transfer the time series recordings on the 6 axes (2 sensors $\times$ 3 axes) into spectrograms separately and flatten the spectrograms into an 1-d array as the feature. We select 3 nearest neighbours for classification.
(2) *3-layer LSTM neural network (LSTM)*: Long short-term memory (LSTM) is a recurrent neural network architecture that is specialized in learning features in time series data. We design a simple neural network consisting of 3 bidirectional LSTM layers with unit number of 50 for each layer. We insert a batch normalization layer and a dropout layer (dropout rate = 0.8) between each LSTM layer. We use the Adam optimizer with the learning rate of 10^{-4}. We empirically set the other hyperparamters to optimize the prediction accuracy of this neural network. The input of the network is the STFT spectrograms of the sensor signals, which is in the shape of $time\_dim \times feature\_dim$, where $feature\_dim = frequency\_dim \times 2 \times 6$. Here, 2 stands for the real and imaginary parts of the STFT spectrogram values, while 6 is the number of axes (2 sensors $\times$ 3 axes). This baseline represents a basic neural network.
(3) *DenseNet*: DenseNet is the state-of-the-art deep neural network used for speech recognition from motion sensor signals [5]. This network sets up connections from each layer to all the other layers in a feed-forward fashion, so that each layer can utilize the feature-maps from all of its preceding layers. DenseNet can alleviate the gradient-vanishing problem and reduce the number of training parameters. We follow the preprocessing steps introduced in [5] to adapt DenseNet to suit our application. First, we interpolate the raw motion sensor signals with sampling rate 400 Hz into 1000 Hz in order to have more evenly distributed samples. Next, we use STFT to convert the motion sensor signals into 6 spectrograms, while each spectrogram is in the shape of $frequency\_dim \times time\_dim$. Then we quantify the values in each spectrogram into $0 \sim 255$ and fit them into a tensor in the shape of $frequency\_dim \times time\_dim \times 6$.

5.4.2 Reconstruction Baselines. The second set of comparisons is with reconstruction baselines, done in terms of both accuracy and audio quality metrics. We show that our model significantly outperforms compared baselines. The reconstruction baselines are:

(1) *Cubic B-spline interpolation (Spline)*: This is a naive technique for upsampling a time series signal.
(2) *Audio super-resolution neural network (AudioSup)*: We use the state-of-the-art audio super-resolution neural network proposed by Kuleshov et al. [22]. This neural network has an encoder-decoder architecture. The encoder consists of several convolutional layers that downsample the input signal to extract hierarchical features. The decoder reverses the downsampling step by using subpixel shuffling layers. All the operations

in this network happen in time domain. This neural network is not originally designed for audio reconstruction from motion sensors. However, since the audio super-resolution task shares many similarities with our reconstruction task, it can apply to our use-case nearly without modification.

(3) *AccelEve reconstruction network (AccelEve)*: [5] This network also aims at reconstructing human speech audio from motion sensor signals. It preprocesses the accelerometer signals into STFT spectrograms, and takes the spectrograms as inputs. The operations in the neural network thus happen in the time-frequency domain. This neural network also has an encoder-decoder architecture. The connection between the encoder and decoder parts comprises multiple residual blocks. To have a fair comparison, we do the preprocessing steps mentioned in [5], that are similar to the procedure done for DenseNet.

5.5 Evaluation Method

We evaluate the accuracy of all models using standardized metrics of audio similarity. In addition, where applicable, we also evaluate the quality of reconstructed audio on Mechanical Turk. In the latter case, we determine the fraction of times that a human listener can correctly tell the word from its reconstructed audio playback. We report the latter set of results only for those models where humans were able to correctly name more word utterances than random chance (e.g., more than 10% of single digits).

For standard metrics, we first use the signal-to-noise ratio (SNR) as a metric for measuring the difference between two audio signals in the time domain. It is defined as:

$$SNR(\hat{x}, x) = 10log\frac{||x||_2^2}{||\hat{x} - x||_2^2} \tag{6}$$

where $\hat{x}$ is the reconstructed audio signal and x is the ground truth audio signal. For SNR, the higher the value, the better the quality of the reconstructed audio. The second metric that we use is the signal-to-distortion ratio (SDR), which is also a metric measures the difference in the time domain. Specifically, it evaluates the energy ratio of the target signal compared to the distortions [37]. It is defined as:

$$SDR = 10log\frac{||s_{target}||^2}{||e_{interf} + e_{noise} + e_{artif}||^2} \tag{7}$$

The terms s_{target}, e_{interf}, e_{noise}, e_{artif} are all decomposed from the estimated signal by orthogonal projections, where s_{target} represents the target signal with allowed distortion. And e_{interf}, e_{noise}, e_{artif} represent the interferences, noise and artifacts error respectively. The higher the value, the better the quality of the reconstructed audio. We use the implementation of SDR from [36]. The third metric we use is the log-spectral distance (LSD), which measures the difference between two audio signals in the time-frequency domain. It is defined as:

$$LSD(\hat{x}, x) = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{F}\sum_{f=1}^{F}(P(\hat{x})_{t,f} - P(x)_{t,f})^2} \tag{8}$$

$$P(x) = log|STFT(x)|^2 \tag{9}$$

where $\hat{x}$ and x are the reconstructed audio and ground truth respectively. T and F are the time and frequency dimension length of the STFT spectrogram of $\hat{x}$ and x. For LSD, smaller values mean better reconstructed audio quality.

To measure the accuracy of human recognition of words reconstructed by different schemes, we hire volunteers on Amazon Mechanical Turk to manually classify the reconstructed audio into 0-9 digits (for the 10 digits dataset) or 20 voice commands (for the voice command dataset). Each reconstructed word or digit is recognized by three

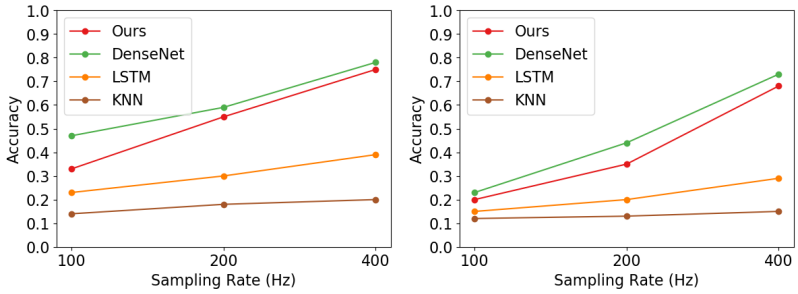
different volunteers. The final accuracy is the average of the accuracy provided by the three volunteers. Before the volunteers start working, we provide a set of labeled reconstruction results as a warm-up.

Finally, to compare with classification solutions (as opposed to reconstruction solutions) we compare the accuracy of reconstructed word recognition by Amazon Mechanical Turk workers to machine classification accuracy of several compared schemes.

5.6 General Performance

To test the performance of our model, we first compare human recognition accuracy of words reconstructed by our model to the machine classification accuracy of *several classification baselines* under the regular experimental setting. We then compare human recognition accuracy of *different reconstruction techniques*. Finally, we compare performance in terms of *standardized audio accuracy metrics*.

In Figure 6, we compare the accuracy of our model to three recognition baselines that are based on machine classification (not audio signal reconstruction). The horizontal axis is the sampling rate of the input motion sensor data, and the vertical axis is the classification accuracy. Note that, the accuracy for our model is the percentage of correctly identified words by human listeners on Amazon Mechanical Turk, whereas the accuracy for the three baselines is their machine classification accuracy. As we stated before, classification is an easier task than reconstruction. Thus, we do not aim to beat classification performance. Rather, we aim to approach it.



(a) 10 digits (b) Speech commands

Fig. 6. Accuracy compared with recognition baselines

As shown in Figure 6.a and 6.b, DenseNet, as the state-of-the-art speech recognition model for motion sensor data, has the highest accuracy among the four models. Our reconstruction network comes as a close second after DenseNet. The low classification accuracy of LSTM and KNN shows that this classification task is not simple. Compared with LSTM and KNN models, our model has much better reconstruction accuracy than their classification accuracy over all different sampling rates and for both datasets. As explained in section 5.3, the Google speech commands dataset has more classes, higher diversity among speakers and less audio quality than the 10 digits dataset. Therefore, the data points shown in Figure 6.b generally have lower accuracy than the data points in Figure 6.a.

As our model reconstructs high resolution audio signals from low resolution motion sensor signals in a hierarchical manner, to have an intuitive view, we visualize the intermediate results of the reconstruction process of an audio sample "six" /sɪks/ layer by layer. In Figure 7, from top to bottom are the ground truth and the reconstructed result at sampling rate of 6400 Hz, 3200 Hz, 1600 Hz, 800 Hz, and 400 Hz. The first two columns of Figure 7 show the spectrograms and waveforms of ground truth. The last two columns in Figure 7 are spectrograms and waveforms of our reconstruction result. We can see that the reconstruction result closely matches the ground

truth. Looking from the bottom to the top, the reconstructed spectrograms gradually approximate ground-truth more faithfully, showing that our hierarchical reconstruction model is working correctly. The spectrograms at 6400 Hz of our results and the ground truth are quite similar, indicating that the final reconstruction result is close to the original audio.

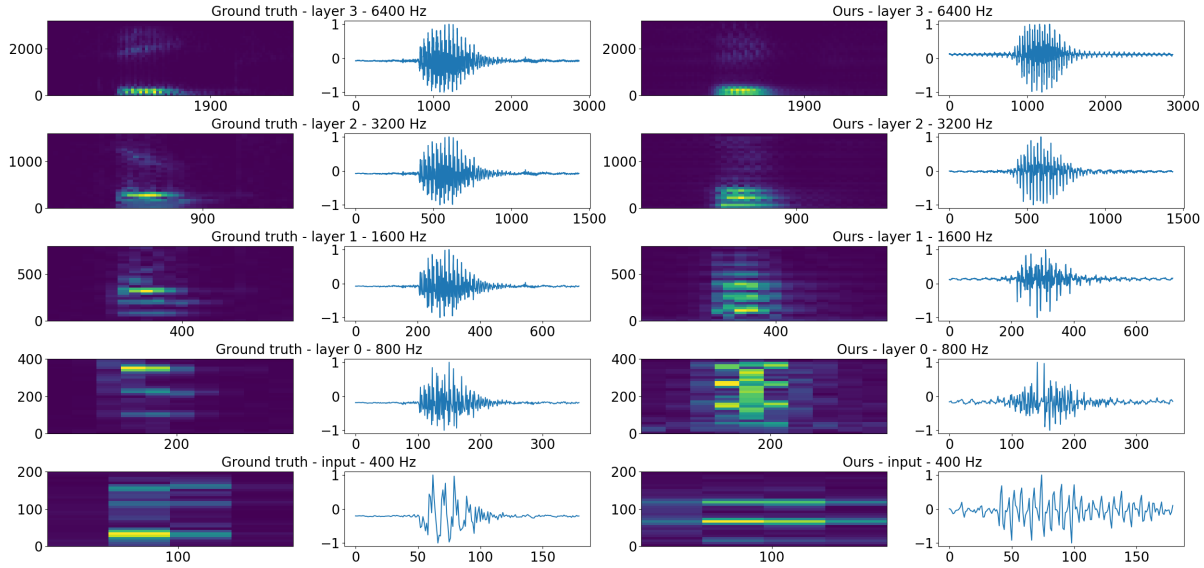


Fig. 7. Intermediate results of the reconstruction process of our model. From bottom to top, the reconstructed spectrograms gradually approximate ground truth, showing that our hierarchical reconstruction model is working correctly.

Next, we switch to comparisons with other audio reconstruction baselines, namely Spline, AudioSup, and AccelEve, described earlier. In this experiment, the motion sensors were set to a sampling frequency of 100-400Hz. The compared approaches were used to reconstruct the audio. Mechanical Turk workers were asked to identify the reconstructed spoken words. Recognition of words reconstructed with our approach performed as shown in Figure 6. The recognition accuracy for the other approaches was comparable to random chance and therefore not shown.

To visually inspect the reconstructed waveforms, Figure 8 compares our model and the three audio reconstruction baselines that attempt to reproduce the original audio waveform from motion sensor data. For spline interpolation and audio super-resolution neural network, the reconstructed signals are clearly inferior. We can see that the reconstruction results lack the high frequency information, whereas the low frequency magnitudes are not successfully unfolded. For AccelEve, it has better reconstruction results. However, human listeners are unable to beat random classifiers at recognizing the reconstructed speech. In order to offer a more principled comparison, we thus turn to standard metrics of audio reconstruction quality, SNR, LSD and SDR defined earlier.

As shown in Figure 9, the reconstructed results from our model has the highest SNR and SDR, and the lowest LSD. As we mentioned before, SNR and SDR measures the similarity between two audio signals in the time domain, while LSD measures their similarity in the time-frequency domain. Compared with the baselines, our model has the advantage in the quality of reconstruction in both the time and frequency domain.

Finally, to tease apart challenges caused by downsampling from those caused by inertial distortion, we add experiments on the effect of downsampling only. Specifically, we replace motion sensor signals with audio signals

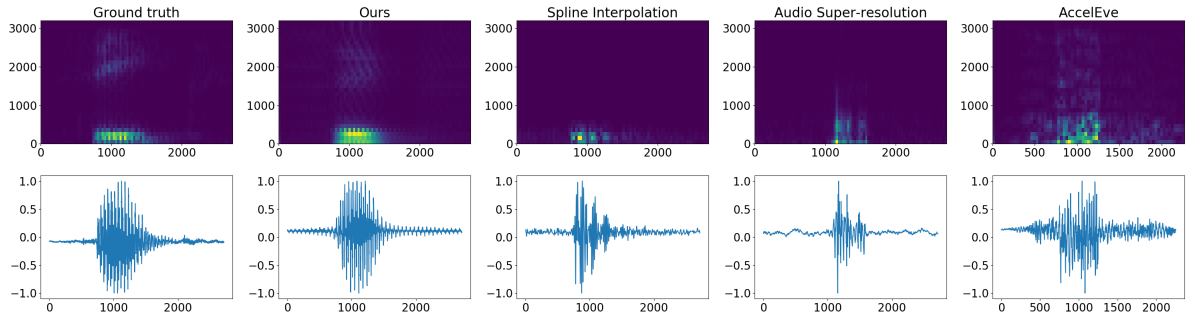
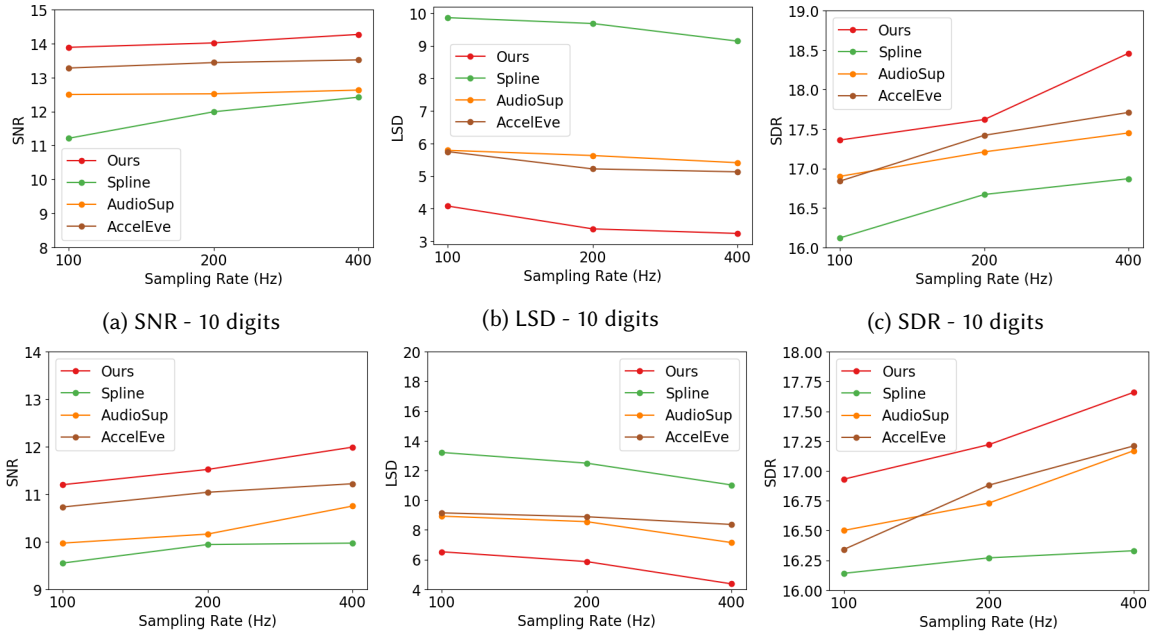


Fig. 8. Reconstruction results compared with reconstruction baselines. Compared with the baselines, ours has the closest reconstruction to the ground truth.

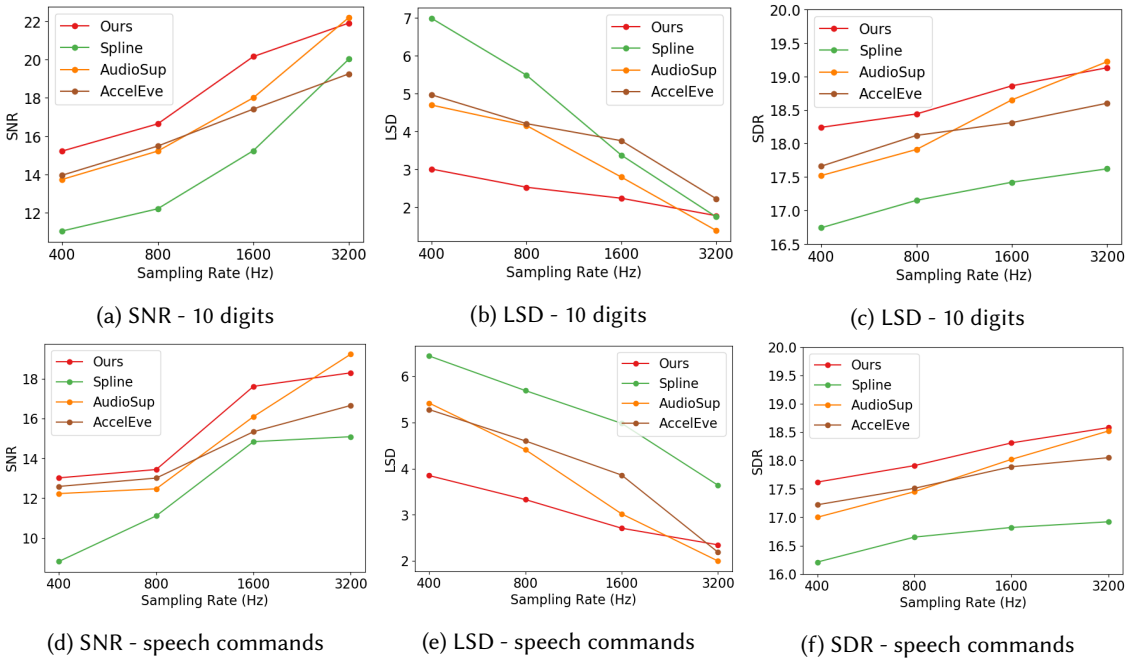


(a) SNR - 10 digits (b) LSD - 10 digits (c) SDR - 10 digits

(d) SNR - speech commands (e) LSD - speech commands (f) SDR - speech commands

Fig. 9. Audio quality metrics compared with reconstruction baselines: motion sensor signals as inputs

that are directly downsampled from the ground truth speech. Keeping the target sampling rate at 6400 Hz, we gradually increase the sampling rate of the input (downsampled) audio from 400 Hz to 800 Hz, 1600 Hz and 3200 Hz. As shown in Figure 10, the reconstruction results of the three baselines have lower SNR and SDR, and higher LSD than our model when sampling rate is between 400 $\sim$ 1600 Hz. When the sampling rate is at 3200 Hz, other approaches become competitive. This demonstrates that past research is good at enhancing signal resolution when the multiplicative enhancement factor is limited (e.g., from 3200 Hz to 6400 Hz). When the multiplicative factor increases (e.g., from 400 Hz to 6400 Hz), our approach outperforms others.



(a) SNR - 10 digits (b) LSD - 10 digits (c) LSD - 10 digits

(d) SNR - speech commands (e) LSD - speech commands (f) SDR - speech commands

Fig. 10. Audio quality metrics compared with reconstruction baselines: downsampled audio as input

5.7 Influence Factors

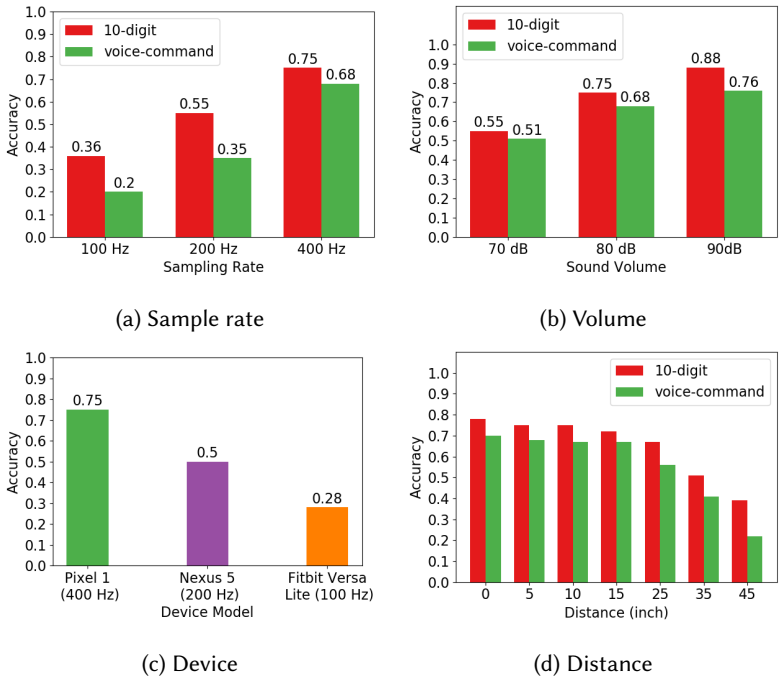
In this section, we investigate the factors that influence performance of our model. The influence factors we tested include:

(1) **Sampling rate**: The sampling rate of the motion sensors.
(2) **Volume**: The loudness of voice played from the speaker, measured at the position of the receiver mobile device.
(3) **Device**: The model of device that records the motion sensor signals.
(4) **Distance**: The distance between the mobile device and the speaker. When distance equals 0, we refer to a scenario, where the audio is played from the internal speaker of the mobile device itself. While changing the distance, we keep the volume measured near the device always being 80 dB.

Figure 11.a shows the performance of our model as a function of sampling rates. The higher the sampling rate, the better the performance of the model. This is intuitive, since data with higher sample rate contains richer information about the original signal.

Figure 11.b shows the model accuracy under different speaker volumes. Higher volume means stronger vibrations to the mobile device and clearer signals received by the motion sensors. Results show that volume has a huge impact on model performance. When volume increases to 90 dB, the reconstruction accuracy increases to 88% for the 10 digits dataset and 76% for the voice command dataset. When volume drops to 70 dB, the accuracy of the 10 digits dataset decreases to 55%, and 51% for the voice command dataset.

In Figure 11.c, we compare performance on different devices. Pixel 1 has the highest maximum sampling rate for motion sensors among the three devices. Therefore, it has the highest reconstruction accuracy. For Nexus 5, its maximum motion sensor sampling rate is 200 Hz. It has a lower accuracy than Pixel 1. The reason why



(a) Sample rate (b) Volume

(c) Device (d) Distance

Fig. 11. Influence of different factors

Pixel 1 with sample rate of 200 Hz still has higher accuracy than Nexus 5 might be related to the difference in performance between the motion sensor models on the two mobile phones. The Fitbit has the lowest motion sensor sampling rate, and the lowest accuracy. Except for the low sampling rate, another disadvantage of the Fitbit Versa Lite is that it has very limited computing power and storage. During signal recording, it has to buffer the motion sensor signals, while transferring the data to the host mobile phone to prevent storage from overflowing. Based on our observation, the parallel execution of the two tasks puts a heavy burden on the computing resource of the Fitbit Versa Lite, and impacts the stability of signal sampling, leading to lower quality of motion sensor data.

Figure 11.d shows the influence of the distance between the speaker and the device. Our model reaches its best performance when the sound is played through the speaker embedded in the device itself. Within 0 to 15 inches, as the distance grows, the performance slightly drops or remains. This indicates that the signal strength received by the motion sensors is approximately proportional to the sound volume near the device. As long as the sound volume remains the same, changing distance does not influence the performance much. However, the performance decays fast when the distance reaches 25 to 45 inches. It demonstrates that the signal strength received by the motion sensors also relies on the vibration transferred by the solid surface. When the distance reaches a certain threshold, the vibration caused by the speaker and transferred through the solid table surface decays faster than the sound wave transferred through the air.

Table 1 summarizes the audio quality metrics under the influence of different factors.

Table 1. Metric values under the influence of different factors

Influence Factor	Device Model	Volume (dB)	Sampling Rate (Hz)	Distance (inch)	10-digit			voice-command		
					SNR	LSD	SDR	SNR	LSD	SDR
Sampling rate	Pixel 1	80	100	5	13.89	4.08	17.36	11.20	6.52	16.93
	Pixel 1	80	200	5	14.02	3.38	17.62	11.52	5.86	17.22
	Pixel 1	80	400	5	14.27	3.24	18.46	11.99	4.36	17.66
Volume	Pixel 1	70	400	5	13.53	3.47	18.01	11.74	4.51	17.21
	Pixel 1	80	400	5	14.27	3.24	18.46	11.99	4.36	17.66
	Pixel 1	90	400	5	16.19	3.01	18.65	12.84	3.69	18.02
Device	Pixel 1	80	400	5	14.27	3.24	18.46	11.99	4.36	17.66
	Nexus 5	80	200	5	13.86	4.03	17.21	N/A	N/A	N/A
	Fitbit	80	100	5	10.46	4.41	16.99	N/A	N/A	N/A
Distance	Pixel 1	80	400	on-device	14.33	3.21	18.69	12.10	4.39	17.61
	Pixel 1	80	400	5	14.27	3.24	18.46	11.99	4.36	17.76
	Pixel 1	80	400	10	14.25	3.33	18.40	11.80	4.51	17.80
	Pixel 1	80	400	15	14.24	3.41	18.29	11.62	4.82	17.65
	Pixel 1	80	400	25	14.01	3.73	18.01	11.52	5.52	17.12
	Pixel 1	80	400	35	13.52	3.96	17.54	10.85	5.88	16.75
	Pixel 1	80	400	45	13.34	4.02	17.19	10.65	6.16	16.33

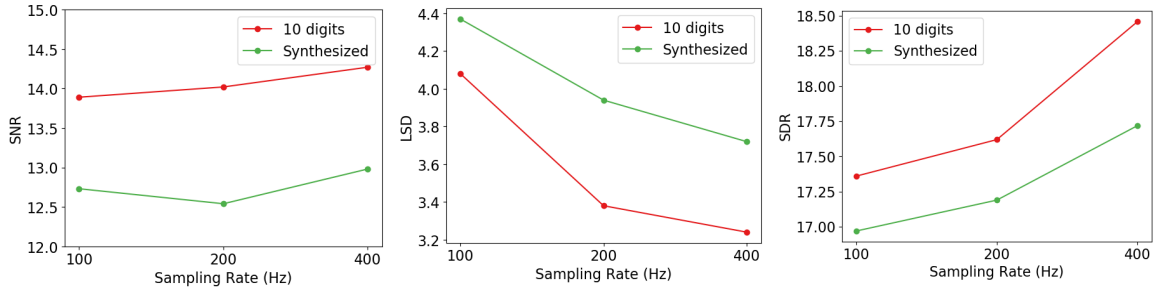
5.8 Generalizability across Vocabulary

In general, our model trained with a set of keywords *will not generalize* to new keywords that include sounds not present in the training data. There is simply no basis for the network to learn the unfolding of signals not previously encountered. However, the model does have limited generalizability in the sense of being able to reproduce sounds made of a concatenation of units previously encountered during training. To demonstrate this effect, we broke up the 10-digit data set into smaller component units used for training. Specifically, we segmented the signal of each spoken digit into 8 overlapping chunks, where each chunk had the shape of 128 (motion sensor samples) × 2 (sensors) × 3 (axis). We made this modification in order to refine the granularity of the features learned by the neural network, so that the model can attain some generalizability on vocabulary. We then designed a small synthetic word testing set, shown in Table 2. The test set contains 7 words. Each word is synthesized from compositions of different sounds in the 10 digits dataset. For example, the synthetic word "fox" (/fɑks/) is composed roughly of the first part of the digit "four" (/fɑ/) and the second part of digit "six" (/ks/). To create the synthetic words, we selected 10 speakers and manually segmented their pronunciation of the 10 digits into parts that were then synthesized to produce the aforementioned seven words not in the original data. This process resulted in 70 synthetic samples in total (10 speakers × 7 words). We recorded the motion sensor signals of the 70 synthetic words under the regular experimental setup (defined in Section 5.1) and used them for testing.

Table 2. Seven synthetic words: the underlines represent that the vocal composition is taken by the synthetic word

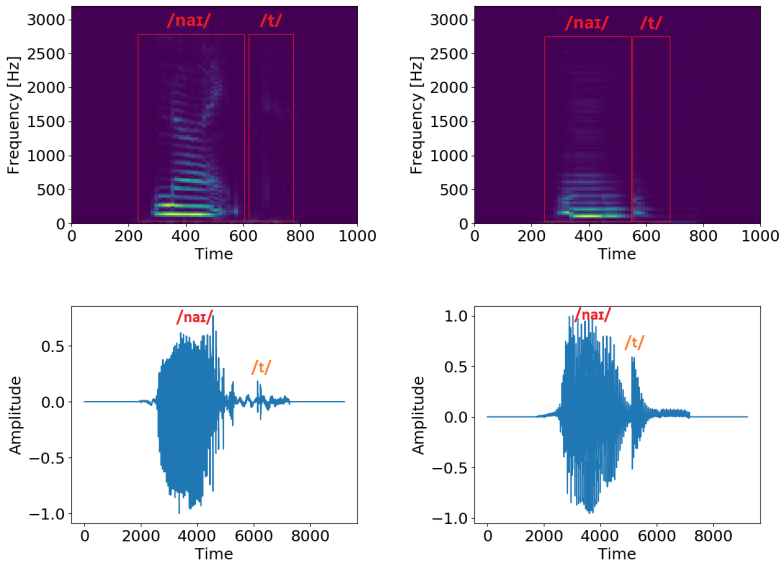
Synthetic word	Fox	Night	Reef	Rose	Swan	Tooth	Vent
Vocal compositions	Four + Six	Nine + Eight	Three + Five	Zero + Six	Six + One	Two + Three	Seven + Eight

The results of the above experiment are shown in Figure 12, compared to the reconstruction of digits in the original data set. As might be expected, the reconstruction results of synthetic words are slightly worse compared with the in-vocabulary digits. For an example, Figure. 13 shows the spectrogram and waveform of a synthetic word sample "night" (composed by nine and eight). The reconstruction result has a distorted shape and loses some high frequency information in the spectrogram, but it successfully recovers the two peaks of the corresponding phonemes /naɪ/ and /t/. It demonstrates that our model has a certain generalizability on vocabularies, but performance still needs improvement. We will further discuss this limitation in generalizability in Section 6.



(a) SNR (b) LSD (c) SDR

Fig. 12. Audio quality metrics of synthetic words compared with 10 digits



(a) Ground truth (b) Reconstruction result

Fig. 13. Reconstructed synthetic word "night" (nine + eight) compared with ground truth. Our model successfully recovers the two peaks of the corresponding phonemes /naɪ/ and /t/.

5.9 Generalizability across Speakers

In this section, we explore the generalization ability of our model to new speakers (but same vocabulary). Hence, we use the 10 digits dataset, while leaving one speaker out as the testing set, and gradually increase the number of speakers in the training set. Since each speaker has 500 samples, this setting leaves 500 samples in the testing set, and N×500 samples in the training set, where N is the number of speakers in the training set. As shown in Table 3, when more speakers are added to the training set, accuracy improves. The growth in accuracy gradually slows down when enough speakers are added into the training set.

Table 3. Accuracy when different number of speakers are given

Speakers in Training Set	Accuracy
5	37%
10	42%
20	44%
30	46%

5.10 Generalizability across Devices

The weakest part of our approach, at present, lies in its (lack of) generalizability across devices. Different devices have different inertial features, thereby impacting the nature of inertial filtering applied by the device to the sound wave when measured by motion sensors. As a result, the recorded waveforms generally seem "out of vocabulary" to the upsampling network. To investigate generalizability across devices, we designed an experiment that implements the threat model described below.

Threat model: Consider an employee at a call center, where customers must verify their identity upfront by revealing a piece of sensitive information. For example, consider the Internal Revenue Service (IRS), where customers calling in must state their social security number before proceeding with the rest of the call. In a typical exchange, a customer dials-in to the call center. The call is patched through to an employee. We assume, in this case, that the employee is using some (agency-issued) smart phone that has been infected by malware that records and transmits motion sensor data. The employee greets the customer and asks for their social security number for identity verification. The customer dictates their number. The employee thanks the customer and asks how they may help them. The rest of the call is irrelevant. The infected phone transmits the recorded motion sensor waveform to the attacker who then deciphers the social security number.

We carried out the above call scenario six times, where an individual would call the infected phone and engage in the above exchange. The infected phone was a model not used in training. Each time the call was repeated, a different (made up) social security number was used. The conversation was recorded using motion sensors sampled at 400 Hz. The motion sensor file was then sent for audio reconstruction using a neural network trained with the 10-digit dataset. The caller was not in the dataset. To evaluate the results, two different individuals (the attackers) listened to the reconstructed audio and wrote down what they thought the numbers were. If they failed to recognize a digit, a "?" was inserted. Table 4 shows the results. Three key observations can be seen:

Table 4. Comparing real and perceived numbers by two listeners.

Listener 1	Listener 2	Ground Truth
??1-57-670	661-55-674	021-52-4913
186-59-1990	986-59-1990	590-52-5694
056-73-84?8	0?6-73-74?8	201-72-8902
567-89-1567	967-85-1566	999-49-2868
613-95-8855	613-95-8857	802-33-9211
885-58-5887	885-58-5887	283-25-3020

- *Null suppression:* In all six calls, it was possible for the attackers to determine which part of the audio was the 9-digit social security number simply from the pattern of sounds and silences in the recording (since

dictating a number creates a pattern of sounds and pauses that is distinct from normal speech). In other words, suppression of irrelevant words (i.e., null suppression) was not a challenge *for the attacker*.[1]

- *Intelligibility:* As seen in Table 4, the two listeners had roughly 80% agreement on the digits they thought they heard.
- *Correctness:* Surprisingly, despite the above (relatively) high degree of agreement, only 16% of the individual digits (18 out of 108) in the original phone conversations where actually named correctly by the attackers. This number is not significantly higher than random chance. (The probability of guessing 18 out 108 digits correctly by chance is about 1.5%.)

The above experiment suggests two observations. First, the neural network learns to reconstruct the inertial distortion caused by the training devices, which can be different from that caused by the testing device, resulting in out of distribution inputs. Second, the neural network learns to reconstruct input signals into intelligible patterns, but it is not good at suppressing low-confidence inputs. Thus, out of distribution data produce outputs that are still mapped to intelligible patterns that individuals recognize, although these outputs are incorrect. As we discuss later, there is a trade-off between training the network to "guess" specialized intelligible patterns from arbitrary inputs and training it generalized upsampling functions that might involve higher degrees of output approximation.

5.11 Ablation Studies

To examine our neural network design, we conduct ablation studies by replacing or removing some components of the original neural network, and observing the performance of the resulting model variations. All the experiments in this section are performed in the regular experimental settings on the spoken 10 digits dataset. To have a more intuitive idea about the quality of reconstruction results from each model variation, we visualize the reconstruction process of an input sample of digit “six” (phonetic symbols: /sɪks/) in Figure 14. As the input sampling rate is 400 Hz while target sampling rate is 6400 Hz, we have 4 layers in total. In Figure 14 and Table 5, we use the label "original" to indicate that the result is from our original model. The accuracy presented in Table 5 is the recognition accuracy by human listeners on Amazon Mechanical Turk.

Table 5. Performance of model variations in the ablation studies

Variation Name	**Accuracy**	**SNR**	**LSD**	**SDR**
Original	75%	14.27	3.24	18.46
Deconvolution	72%	10.41	3.48	18.34
As-image	71%	10.58	4.76	18.42
MSE-loss	N/A	8.33	7.67	17.33
No-intermediate-supervision	N/A	8.65	8.43	17.61

5.11.1 Alias Unfolding Versus Deconvolution. In our design, the alias unfolding layer expands the frequency resolution of the input spectrogram by 2. An alternative to expanding the frequency bandwidth is to perform a deconvolution operation. Deconvolution, also known as transposed convolution, is a common method for expanding the dimension of a given 2-D feature matrix. In the first ablation study, we implement a deconvolution layer to substitute the alias unfolding layer. The deconvolution layer replaces the frequency filtering, unfolding, and concatenation operations in the alias unfolding layer with a deconvolution operation, having a kernel size 1×3 and a stride number of 2. It upsamples the frequency bandwidth of the input spectrogram by 2, just like

[1]As we show later, recognition of out-of-distribution data as such was, however, a challenge for the *neural network*.

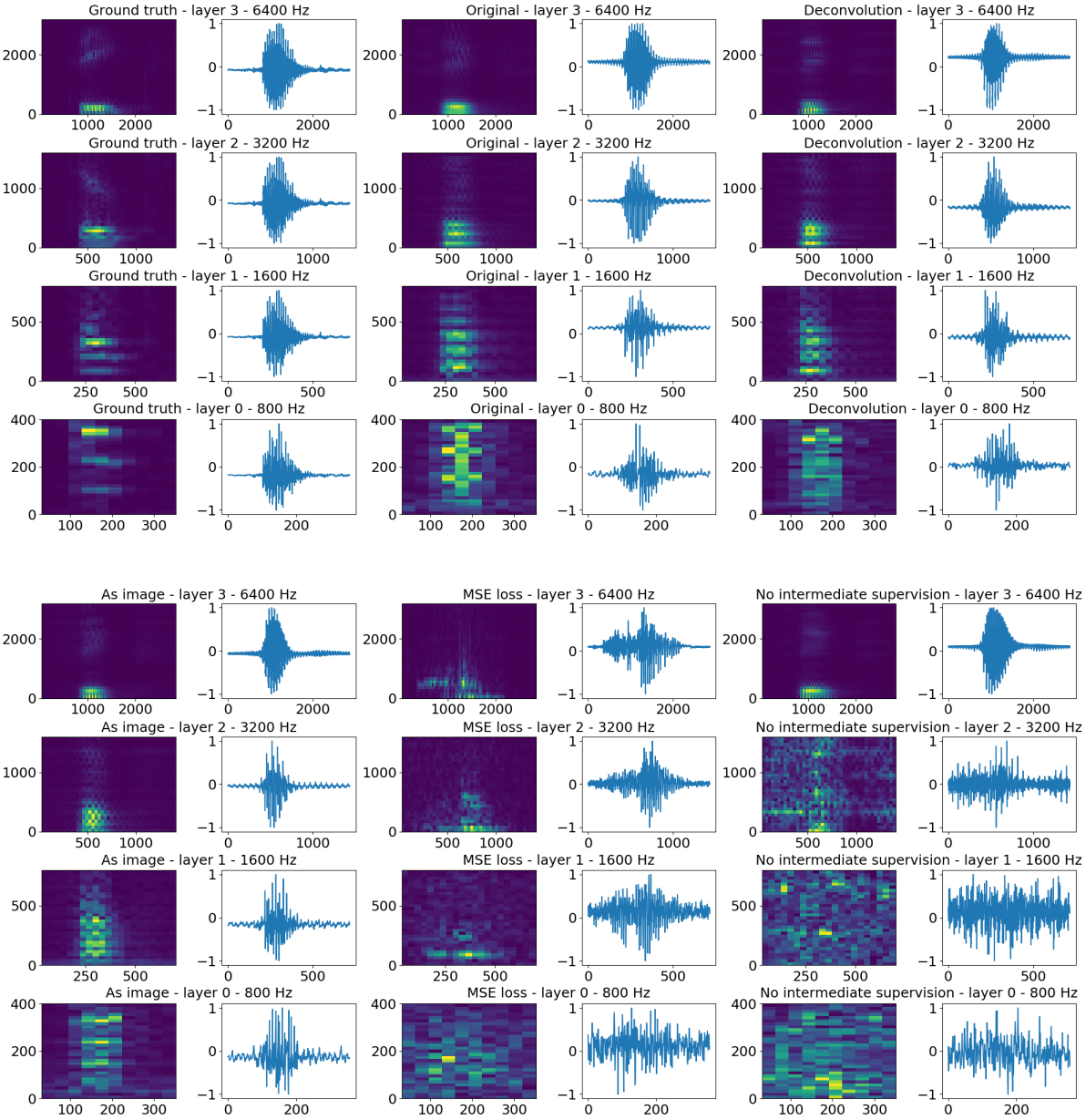


Fig. 14. Ablation studies. The reconstruction process of each model variation for an input sample of digit “six”.

the original alias unfolding layer. However, there is an essential difference between the alias unfolding layer and the deconvolution layer. In alias unfolding layer, we split the folded spectrogram into low a frequency part and a high frequency part by subtracting the learned low frequency part from the folded (as shown in Figure 4).

Consequently, the sums of magnitude at every corresponding position in the two unfolded spectrograms are equal to the previous folded spectrogram. This guarantees that the frequency unfolding process follows laws of physics for aliased signal unfolding. In contrast, the deconvolution layer has no insight about the underlying physics. Its frequency expanding process makes no constraint on the output values. As shown in Table 5, replacing alias unfolding layers with deconvolution layers impairs the performance of our neural network. The "deconvolution" model has a lower accuracy, lower SNR and SDR, and higher LSD than our original model.

5.11.2 Spectrograms Versus Images. In the time/frequency convolutional layers of the original neural network, we use a *spectral padding* (Equation 3) technique to keep the consistency in dimensions of the input and output spectrograms. The insight is that, unlike images, conventional zero padding is inappropriate for spectrograms (it is not a neutral operation). In this ablation study, we instead treat spectrograms as images, and apply ordinary zero padding to them in the convolutional layers. Table 5 (row noted "as-image") shows that even though image-based treatment can still produce intelligible audio, the quality is worse than that of the original model in all aspects.

5.11.3 MFCC-DTW Loss Versus MSE Loss. Mean squared error (MSE) measures the average of the squared errors of two time series. Compared with MFCC-DTW loss, MSE has three shortcomings. First, MSE measures similarity between reconstruction results and the ground truth in the time domain. But for humans to recognize sounds, the information in the frequency domain is also very important. MFCC-DTW loss assesses the reconstructed audio quality in both time and frequency domain explicitly, providing a more comprehensive loss score for back propagation in the neural network. Second, when comparing two time series not aligned on the time axis, MSE is less robust, since it does not explicitly tolerate offsets or differences in speed. In practice, the input sensor signals are not perfectly aligned with the ground truth speech audio. Thus, DTW has an advantage in calculating the loss in time domain. Third, MSE has no insight about the human auditory system. In contrast, MFCC loss evaluates the reconstructed audio quality based on human perceptual response to sounds. Experimental results show that the "MSE" variation has worse performance than the original model. In Figure 14, the reconstructed result from the "MSE" variation suffers more distortion in both time and frequency domains than what the original model. The output audio is not intelligible.

5.11.4 Remove Intermediate Supervision. The original model apply supervision at intermediate layers during the reconstruction process by including them in the loss function. In this study, we change the hierarchical MFCC-DTW loss function from Equation 4 to:

$$\mathcal{L}(\hat{d_N}) = L(d_i, \hat{d}_i) \tag{10}$$

which means only taking the final output loss as the total loss. Experimental results show that the output speech audio from variation "no-intermediate-supervision" are not intelligible, as their audio quality is too low. In Figure 14, we can see the intermediate results of the reconstruction process are poor. Without supervision on each layer, the intermediate reconstruction results are far different from the ground truth.

6 DISCUSSION

In this section, we discuss the limitations of our work and analyze impediments to full human speech reconstruction. As shown in Section 5.9, the performance of our model degrades when the test speaker is not in the training dataset, although this degradation can be ameliorated by increasing the diversity of speakers in training data. Different mobile devices exhibit different distortions in the recorded motion sensor waveform due to their different weights, shapes, and sensor models. The resulting device generalizability challenge may be partially mitigated by analytical device-specific preprocessing (ahead of upsampling) that is informed by the dynamics of the recording device. The preprocessing would undo the inertial filtering by inverting the transfer function from the down-sampled audio signal to the actual accelerometer signal shaped by device inertia. This transfer function

is measurable, but for the malware to reverse that transfer function automatically, it would have to have access to both sound and acceleration (at least occasionally) so that it can compute the needed transfer function to begin with. Other influence factors mentioned in Section 5.7 also impact model performance. The diversity of these factors makes training a particularly challenging problem. An interesting direction might be to resort to data augmentation designed to reduce sensitivity of training to specific features affecting the input signal, such as sound volume or device inertia. We delegate this investigation to future work.

A more interesting limitation of our current model lies in its limited ability to generalize to vocabulary that is out of the training dataset. Generalizing to new vocabulary requires the model to break-down training words into smaller elementary vocal components (e.g. phonemes or even finer-grained components) from which other new words can be constructed. A trade-off exists in the choice of training unit. If we take whole words as training units, the deep neural network will learn to convert these chunks of motion sensor inputs into entire audio words. This whole word-to-word conversion restricts model generalizability. What is worse, the neural network then learns to act closer to a classifier. That is to say, it will tend to convert new signals (e.g., new words or signals recorded by new devices) into some pattern that exists in the training set, regardless of confidence, leading to intelligible but often incorrect outputs, as shown in Section 5.10. A possible approach to increase generalizability is to segment individual words into multiple overlapping (smaller) chunks, and take each chunk as a training unit. In this case, the neural network learns how to convert finer-grained features, such as the vocal components inside each word. As we show in Section 5.8, using such finer-grained signals as inputs makes our model generalize better to new words (synthesized from phonemes present in words in the training data) at the expense of a lower reconstruction quality. A trade-off is thus observed between generalizability and intelligibility of the reconstructed results.

The above tradeoff occurs because when we use finer-grained inputs, the input length becomes shorter, offering less information for the neural network to do signal disambiguation. Given the reduced information, the likelihood of unfolding the input waveform into the correct audio signal decreases. To improve this trade-off between generalizability and intelligibility, one needs to optimally set the granularity of the inputs and the parameters of the reconstruction network. This exploration might enable real-time reconstruction of streaming speech.

Finally, we have shown that top classification solutions beat our reconstruction-based approach. A legitimate question is therefore: why bother with reconstruction at all? Why not rely on classification, which is a simpler problem? In the authors' opinion, the answer lies in the greater potential of reconstruction-based solutions for generalizability to a broad vocabulary (even despite the limitations mentioned above). As mentioned in the introduction, reconstruction relies on the human auditory system as the eventual ultimate "word classifier". Thus, a reconstruction system needs merely to approximate sounds of vocal components (e.g., phonemes). The rest will be done by the human. The system does not need to know, for example, if a person said "son", "sun", or "sonne" if they sound the same. A keyword classification system, in contrast, would have to worry about finding the right word or label. The number of classes in keyword-based classification systems therefore can grow very large, leading to a higher potential for misclassifications. In contrast, for languages where multitudes of words are made up of a much smaller number of basic sounds (e.g., on the order of 50 phonemes in the English language), reconstruction might eventually be a more scalable solution. The system needs to learn the reconstruction of only a small number of basic elements to capture a large vocabulary of words. The paper improves over prior reconstruction-based solutions, thus contributed a step in that direction.

7 CONCLUSION

In this paper, we investigated the feasibility of reconstructing spoken keywords from low sampling rate motion sensors by means of unfolding signals in the frequency domain using a neural network designed for this purpose. A key building block of this network is the *alias unfolding layer* that reverses the frequency folding process in the time-frequency domain with learnable parameters. Assisted with the time/frequency convolutional layers,

customized specifically for learning features from spectrograms, our neural network generates audio-like signals from motion sensor signals sampled at 100-400 Hz. It is shown to be superior to the state of the art according to several reconstruction quality metrics as well as human-in-the-loop experiments. Our neural network might be the next step in the quest for successful recovery of intelligible human speech from low sampling-rate motion sensor signals recorded on mobile devices. The approach proposed in this paper, however, suffers generalizability limitation to out-of-distribution data. While some of these limitations can be trivially addressed by increasing the diversity of training data, an interesting research question is whether a combination of signal processing (at the right input granularity) and neural-network-based up-sampling can significantly ameliorate the generalizability bottlenecks. The work joins a series of previous results that call for seriously considering privacy implications of unrestricted sharing of motion sensor data.

ACKNOWLEDGMENTS

Research reported in this paper was sponsored in part by the Army Research Laboratory under Cooperative Agreement W911NF-17-20196 and in part by NSF under award CPS 20-38817. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, NSF, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] 2019. Voice frequency. https://en.wikipedia.org/wiki/Voice_frequency

[2] Johannes Abel, Maximilian Strake, and Tim Fingscheidt. 2018. A simple cepstral domain DNN approach to artificial speech bandwidth extension. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5469–5473.

[3] S Abhishek Anand, Chen Wang, Jian Liu, Nitesh Saxena, and Yingying Chen. 2019. Spearphone: A speech privacy exploit via accelerometer-sensed reverberations from smartphone loudspeakers. *arXiv preprint arXiv:1907.05972* (2019).

[4] V. Atti, V. Krishnan, D. Dewasurendra, V. Chebiyyam, S. Subasingha, D. J. Sinder, V. Rajendran, I. Varga, J. Gibbs, L. Miao, V. Grancharov, and H. Pobloth. 2015. Super-wideband bandwidth extension for speech in the 3GPP EVS codec. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5927–5931.

[5] Zhongjie Ba, Tianhang Zheng, Xinyu Zhang, Zhan Qin, Baochun Li, Xue Liu, and Kui Ren. [n.d.]. Learning-based practical smartphone eavesdropping with built-in accelerometer. In *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium*. 23–26.

[6] Dhananjay Bansal, Bhiksha Raj, and Paris Smaragdis. 2005. Bandwidth expansion of narrowband speech using non-negative matrix factorization. In *Ninth European Conference on Speech Communication and Technology*.

[7] Patrick Bauer and Tim Fingscheidt. 2009. A statistical framework for artificial bandwidth extension exploiting speech waveform and phonetic transcription. In *2009 17th European Signal Processing Conference*. IEEE, 1839–1843.

[8] Sören Becker, Marcel Ackermann, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. 2018. Interpreting and Explaining Deep Neural Networks for Classification of Audio Signals. *CoRR* abs/1807.03418 (2018). arXiv:1807.03418

[9] Yan Ming Cheng, Douglas O'Shaughnessy, and Paul Mermelstein. 1994. Statistical recovery of wideband speech from narrowband speech. *IEEE Transactions on Speech and Audio Processing* 2, 4 (1994), 544–548.

[10] Samir Chennoukh, A Gerrits, G Miet, and R Sluijter. 2001. Speech enhancement via frequency bandwidth extension using line spectral frequencies. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, Vol. 1. IEEE, 665–668.

[11] Chris Donahue, Julian McAuley, and Miller Puckette. 2018. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208* (2018).

[12] Julien Epps and W Harvey Holmes. 1999. A new technique for wideband enhancement of coded narrowband speech. In *1999 IEEE Workshop on Speech Coding Proceedings. Model, Coders, and Error Criteria (Cat. No. 99EX351)*. IEEE, 174–176.

[13] Jianqing Gao, Jun Du, and Enhong Chen. 2018. Mixed-bandwidth cross-channel speech recognition via joint optimization of dnn-based bandwidth expansion and acoustic modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27, 3 (2018), 559–571.

[14] Yu Gu and Zhen-Hua Ling. 2017. Waveform Modeling Using Stacked Dilated Convolutional Neural Networks for Speech Bandwidth Extension.. In *INTERSPEECH*. 1123–1127.

[15] Archit Gupta, Brendan Shillingford, Yannis Assael, and Thomas C Walters. 2019. Speech bandwidth extension with wavenet. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 205–208.

[16] Jun Han, Albert Jin Chung, and Patrick Tague. 2017. Pitchln: eavesdropping via intelligible speech reconstruction using non-acoustic sensor fusion. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*. 181–192.
[17] Scott Havard. [n.d.]. *Google Pixel XL Teardown.* https://www.ifixit.com/Teardown/Google+Pixel+XL+Teardown/71237
[18] Rongqiang Hu, Venkatesh Krishnan, and David V Anderson. 2005. Speech bandwidth extension by improved codebook mapping towards increased phonetic classification. In *Ninth European Conference on Speech Communication and Technology*.
[19] Peter Jax and Peter Vary. 2000. Wideband extension of telephone speech using a hidden Markov model. In *2000 IEEE Workshop on Speech Coding. Proceedings. Meeting the Challenges of the New Millennium (Cat. No. 00EX421)*. IEEE, 133–135.
[20] Peter Jax and Peter Vary. 2003. On artificial bandwidth extension of telephone speech. *Signal Processing* 83, 8 (2003), 1707–1719.
[21] Juho Kontio, Laura Laaksonen, and Paavo Alku. 2007. Neural network-based artificial bandwidth expansion of speech. *IEEE transactions on audio, speech, and language processing* 15, 3 (2007), 873–881.
[22] Volodymyr Kuleshov, S Zayd Enam, and Stefano Ermon. 2017. Audio super-resolution using neural nets. In *ICLR (Workshop Track)*.
[23] Kehuang Li, Zhen Huang, Yong Xu, and Chin-Hui Lee. 2015. DNN-based speech bandwidth expansion and its application to adding high-frequency missing features for automatic speech recognition of narrowband speech. In *Sixteenth Annual Conference of the International Speech Communication Association*.
[24] Sen Li, Stéphane Villette, Pravin Ramadas, and Daniel J Sinder. 2018. Speech bandwidth extension using generative adversarial networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5029–5033.
[25] Teck Yian Lim, Raymond A Yeh, Yijia Xu, Minh N Do, and Mark Hasegawa-Johnson. 2018. Time-frequency networks for audio super-resolution. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 646–650.
[26] Zhen-Hua Ling, Yang Ai, Yu Gu, and Li-Rong Dai. 2018. Waveform modeling and generation using hierarchical recurrent neural networks for speech bandwidth extension. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26, 5 (2018), 883–894.
[27] Shengzhong Liu, Shuochao Yao, Yifei Huang, Dongxin Liu, Huajie Shao, Yiran Zhao, Jinyang Li, Tianshi Wang, Ruijie Wang, Chaoqi Yang, et al. 2020. Handling missing sensors in topology-aware iot applications with gated graph neural network. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 3 (2020), 1–31.
[28] Brittany McCrigler. [n.d.]. *Google Nexus 5 Teardown.* https://www.ifixit.com/Teardown/Nexus+5+Teardown/19016
[29] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. 2014. Gyrophone: Recognizing speech from gyroscope signals. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 1053–1067.
[30] Yoshihisa Nakatoh, Mineo Tsushima, and Takeshi Norimatsu. 1997. Generation of broadband speech from narrowband speech using piecewise linear mapping. In *Fifth European Conference on Speech Communication and Technology*.
[31] Amr H Nour-Eldin and Peter Kabal. 2011. Memory-based approximation of the Gaussian mixture model framework for bandwidth extension of narrowband speech. In *Twelfth Annual Conference of the International Speech Communication Association*.
[32] Santiago Pascual, Antonio Bonafonte, and Joan Serra. 2017. SEGAN: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452* (2017).
[33] Pery Pearson. [n.d.]. Sound Sampling. http://www.hitl.washington.edu/projects/knowledge_base/virtual-worlds/EVE/I.B.3.a.SoundSampling.html
[34] N Prasad and T Kishore Kumar. 2016. Bandwidth extension of speech signals: A comprehensive review. *International Journal of Intelligent Systems and Applications* 8, 2 (2016), 45–52.
[35] Yasheng Qian and Peter Kabal. 2002. Wideband speech recovery from narrowband speech using classified codebook mapping. In *Australian Int. Conf. Speech Science, Technology*. 106–111.
[36] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. 2014. mir_eval: A transparent implementation of common MIR metrics. In *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer.
[37] Emmanuel Vincent, Shoko Araki, Fabian Theis, Guido Nolte, Pau Bofill, Hiroshi Sawada, Alexey Ozerov, Vikrham Gowreesunker, Dominik Lutter, and Ngoc QK Duong. 2012. The signal separation evaluation campaign (2007–2010): Achievements and remaining challenges. *Signal Processing* 92, 8 (2012), 1928–1936.
[38] Yingxue Wang, Shenghui Zhao, Yingying Yu, and Jingming Kuang. 2015. Speech bandwidth extension based on GMM and clustering method. In *2015 Fifth International Conference on Communication Systems and Network Technologies*. IEEE, 437–441.
[39] Pete Warden. 2017. Speech Commands: A public dataset for single-word speech recognition. *Dataset available from http://download.tensorflow.org/data/speech*$_c$*ommands*$_v$0.01.*tar.gz* (2017).
[40] Can Yağlı, MA Tuğtekin Turan, and Engin Erzin. 2013. Artificial bandwidth extension of spectral envelope along a Viterbi path. *Speech communication* 55, 1 (2013), 111–118.
[41] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. 2017. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 351–360.
[42] Shuochao Yao, Ailing Piao, Wenjun Jiang, Yiran Zhao, Huajie Shao, Shengzhong Liu, Dongxin Liu, Jinyang Li, Tianshi Wang, Shaohan Hu, et al. 2019. Stfnets: Learning sensing signals from the time-frequency perspective with short-time fourier neural networks. In *The*

World Wide Web Conference. ACM, 2192–2202.

[43] Yuki Yoshida and Masanobu Abe. 1994. An algorithm to reconstruct wideband speech from narrowband speech based on codebook mapping. In *Third International Conference on Spoken Language Processing*.

[44] Li Zhang, Parth H Pathak, Muchen Wu, Yixin Zhao, and Prasant Mohapatra. 2015. Accelword: Energy efficient hotword detection through accelerometer. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 301–315.