electronics

MDPI

Article

A Deep Learning Model for Snoring Detection and Vibration Notification Using a Smart Wearable Gadget

Tareq Khan

School of Engineering, Eastern Michigan University, Ypsilanti, MI 48197, USA; tareq.khan@emich.edu

Received: 6 August 2019; Accepted: 2 September 2019; Published: 4 September 2019

Abstract: Snoring, a form of sleep-disordered breathing, interferes with sleep quality and quantity, both for the person who snores and often for the person who sleeps with the snorer. Poor sleep caused by snoring can create significant physical, mental, and economic problems. A simple and natural solution for snoring is to sleep on the side, instead of sleeping on the back. In this project, a deep learning model for snoring detection is developed and the model is transferred to an embedded system—referred to as the listener module—to automatically detect snoring. A novel wearable gadget is developed to apply a vibration notification on the upper arm until the snorer sleeps on the side. The gadget is rechargeable, and it is wirelessly connected to the listener module using low energy Bluetooth. A smartphone app—connected to the listener module using home Wi-Fi—is developed to log the snoring events with timestamps, and the data can be transferred to a physician for treating and monitoring diseases such as sleep apnea. The snoring detection deep learning model has an accuracy of 96%. A prototype system consisting of the listener module, the wearable gadget, and a smartphone app has been developed and tested successfully.

Keywords: Bluetooth low energy; convolutional neural network; deep learning; Mel frequency cepstral coefficients; raspberry pi; smartphone app; snoring sound; tilt sensor; vibration notification; wearable gadget

1. Introduction

One of the world's longest studies of adult life, conducted by the Harvard Medical School, revealed that close relationships, more than money or fame, are what keep people happy throughout their lives [1]. Snoring doesn't only interfere with the snorer's sleep, it often creates bitterness and resentment between couples. About 40% of adult men and 24% of adult women are habitual snorers [2]. Snoring starts when the muscles surrounding the throat relax during sleep. This narrows the airway, triggering vibrations that cause snoring. Snoring generally occurs when a person sleeps on the back. A natural cure for snoring problem is to sleep on the side [3]. In this project, a convolutional neural network (CNN)-based deep learning model for snoring detection is developed, validated and tested. The model is then transferred to an embedded system—referred to as the listener module—to automatically detect snoring. A novel wearable gadget is developed to apply a vibration notification on the upper arm until the snorer sleeps on the side. Unlike an alarm sound, as the vibration is only applied to the snorer, the gadget doesn't disturb any other person who is sleeping near the snorer. The gadget is low power, rechargeable, and it is wirelessly connected to the listener module using Bluetooth low energy. A smartphone app—connected with the listener module using home Wi-Fi—is developed to start/stop a snoring session, and to log the snoring events with timestamps. The overall operation of the proposed system is shown in Figure 1. The needs and significance of the proposed gadget are mentioned below:

- About 75% of people who snore suffer from disrupted breathing during sleep for short periods—known as obstructive sleep apnea (OSA) [3]. Severe sleep apnea carries a significant risk

of early death, but even mild to moderate sleep disorders can be related to heart disease, diabetes, reduced sexual function, obesity, gastroesophageal reflux disease, arrhythmia (irregular heart rhythm), headache, nocturia (wake at night several times to urinate) and stroke [4]. The proposed gadget will notify the snorer by vibration and compel them to sleep on their side. Moreover, the logged snoring data can be used to diagnose and monitor OSA and other diseases by the physicians, as discussed in [5–8].

- Snoring interferes with sleep quality and the sleep quantity of the individual who snores and anyone who shares the same sleeping space. Poor quality and insufficient sleep reduce memory, thinking skills, and the ability to manage conflict. Lack of sleep can make a person irritable, short-tempered, and depressed [9]. The proposed gadget will reduce snoring, increase sleep quality, and facilitate better mental health.
- In the United States, insufficient sleep and sleep disorders account for $411 billion in economic losses and represent 2.28% of the country's gross domestic product (GDP) annually [10]. This device can improve sleep quality and reduce economic loss.

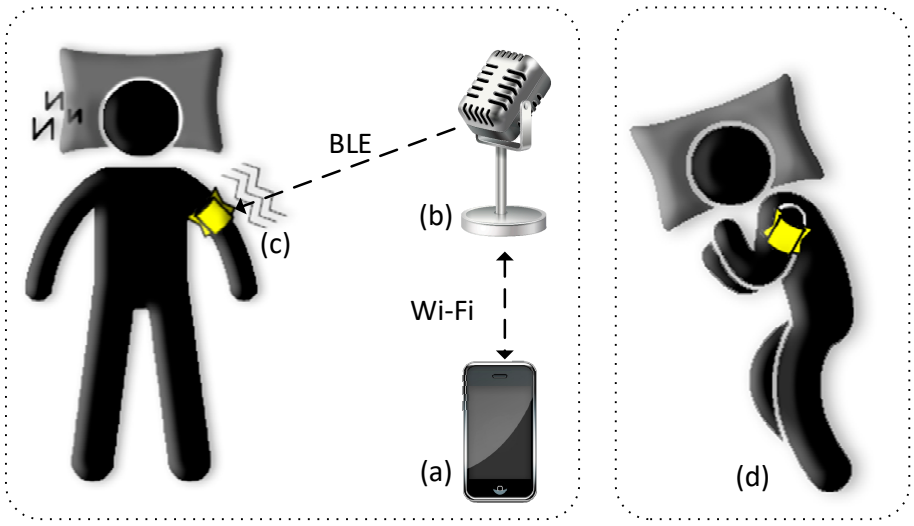


Figure 1. The proposed snoring detection and vibration notification system. The user configures the system and starts a snoring session using the smartphone app (**a**)—which connects with the listener module using Wi-Fi. The listener module (**b**) detects snoring sound and sends command using Bluetooth low energy (BLE) to the wearable gadget (**c**) to generate vibration notification to the snorer. Snoring events with a timestamp are logged in smartphone (**a**). The vibration stops automatically after the snorer changes sleeping position to the side (**d**).

Some available anti-snoring mouthpieces on the market [11–14] hold the lower jaw forward to maintain an open airway to reduce snoring. However, these mouthpieces are often uncomfortable to wear, hamper the natural mouth function, and also require regular cleaning. Theravent [15] uses a strip that is put on the nose. The micro-valves in the strip create pressure in the airway when breathing out, keeping it open to reduce vibration. Smart Nora [16] uses hardware that is placed on a nearby table, listens for snoring. Once detected, another piece of hardware placed under the pillow starts to move in order to stimulate the throat muscles for breathing. The smartphone app, SnoreLab [17], records snoring sound while sleeping, however, it does not do any action to stop snoring. Compared to the existing anti-snoring devices, the proposed novel wearable device is more comfortable and will not hinder the natural mouth or nose functions, as it can be worn on the upper arm. After the listener module detects snoring, the wearable gadget applies vibration notification on the upper arm until the snorer sleeps on the side. The smartphone app also logs snoring data and generates bar graphs - that can be used to monitor and to treat snoring. Some snoring sound analysis and detection work

using the Fourier transform method is found in the literature [18–26]. Compared to these works, the proposed work uses a deep learning model to classify snoring sounds from non-snoring sounds.

2. Materials and Methods

In this paper, a deep learning classifier to detect snoring sound is modeled first. Then the proposed system as shown in Figure 1 is designed and developed. They are briefly described below.

2.1. Deep Learning Model for Snoring Detection

2.1.1. Dataset Generation

A dataset of 1000 sound samples is developed in this project. The dataset contains 2 classes—snoring sounds and non-snoring sounds. Each class has 500 samples. The snoring sounds were collected from different online sources [27–31]. The non-snoring sounds were also collected from similar online sources. Then silences were trimmed from the sound files and the files were split to equal-sized one-second duration files using WavePad Sound Editor [32]. Thus, each sample has a duration of one second.

Among the 500 snoring samples, 363 samples consist of snoring sounds of children, adult men and adult women without any background sound. The remaining 137 samples consist of snoring sounds having a background of non-snoring sounds. Background non-snoring sounds were mixed with the snoring sounds using [32]. The 500 non-snoring samples consist of background sounds that might be available near the snorer. Ten categories of non-snoring sounds are collected, and each category has 50 samples. The ten categories are baby crying, the clock ticking, the door opened and closed, total silence and the minor sound of the vibration motor of the gadget, toilet flashing, siren of emergency vehicle, rain and thunderstorm, streetcar sounds, people talking, and background television news.

2.1.2. Feature Extraction

The first step to classify sound is to extract the features. In this paper, the Mel frequency cepstral coefficients (MFCCs) [33–35] are calculated for each sample. The motivating idea of MFCC is to compress information into a small number of coefficients based on an understanding of the human ear. To calculate MFCC, the time-domain audio signal is first divided into 20–40 ms frames. Then, for each frame, the power spectrum is calculated. Then triangular-shaped Mel filterbanks are calculated and applied to the power spectra, and spectrogram is obtained. Human ears are much better at discriminating small changes in pitch at low frequencies (below 1 kHz) than they are at high frequencies. So, in Mel filterbank, the first 10 filters are placed linearly around 100, 200, ... 1000 Hz. Above 1 kHz, these bands are placed with logarithmic Mel-scale. Then the logarithm of all filterbank energies and then their discrete cosine transform (DCT) are calculated to decorrelate the filter bank coefficients.

In this work, the sound sample is divided into 30 ms frames. The number of filters in the filterbank was chosen to be 32, the number of FFT points set to 512, and the number of cepstral coefficients was set to 32. The MFCCs are calculated using SpeechPy [36] library. Figure 2 shows a snoring class sample and a non-snoring class sample in the time domain and its MFCC representation. After the MFCC is calculated, the one-dimensional time-domain sound signal becomes a two-dimensional signal of size 32×32—that can be treated as an image. We can then apply image classification deep learning architectures—such as CNN—to classify the sample images. Thus, a dataset of 1000 MFCC images is created from the sound samples to be fed to the deep learning network.

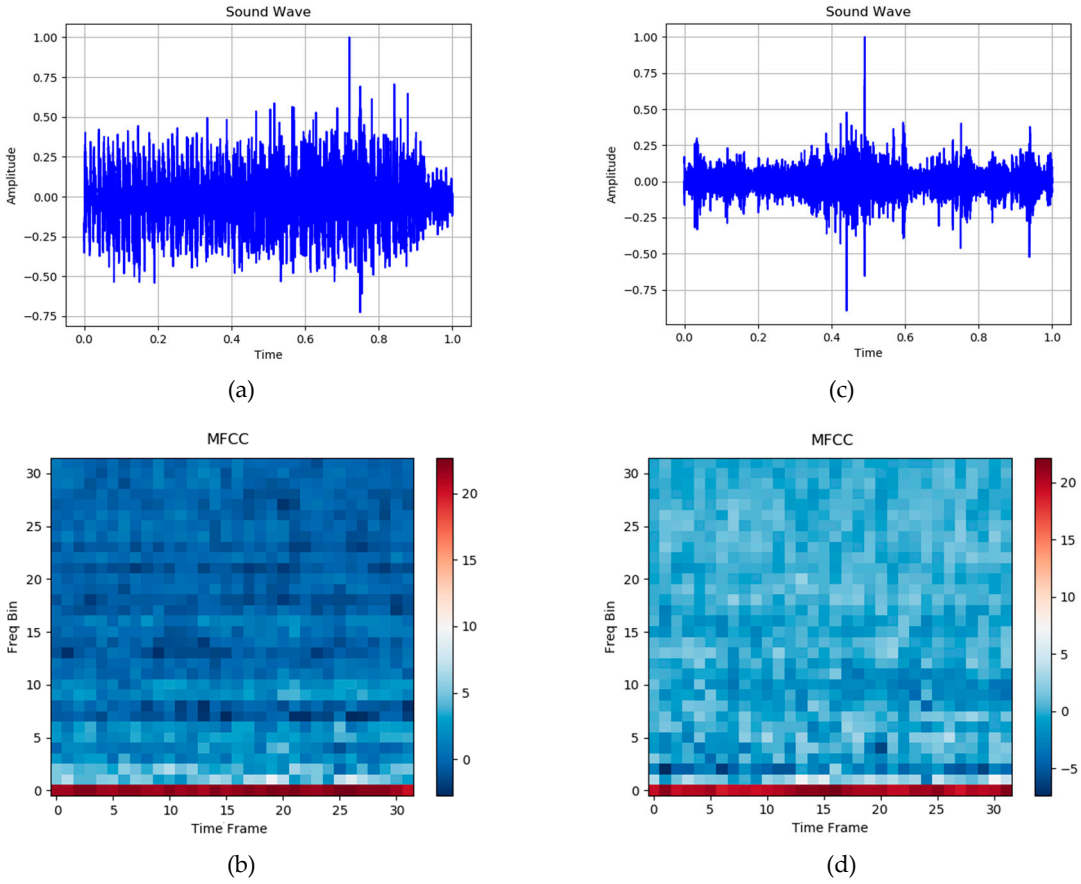


Figure 2. (**a**) Time-domain snoring sound after amplitude normalization; (**b**) Mel frequency cepstral coefficients (MFCC) of the snoring sound in (**a**); (**c**) time-domain non-snoring sound of toilet flashing after amplitude normalization; (**d**) MFCC of the non-snoring sound in (**c**);

2.1.3. Convolutional Neural Network Architecture

A deep learning network, as shown in Figure 3, is used to classify the sound as snoring or as non-snoring. The different layers and optimizer of the network are briefly described below.

- MFCC Image: The MFCC image is a tensor of size (32, 32, 1). The data type of the image pixel is converted to floating-point. To normalize the pixel values, the mean and the standard deviation of the training dataset is calculated and saved in a file. Then from all images of the dataset, the mean is subtracted from each pixel value and then each pixel value is divided by the standard deviation.
- Convolutional Layer: A 2-D convolutional layer applies sliding convolutional filters to the input. The layer convolves the input by moving the filters along the input vertically and horizontally and computing the dot product of the weights and the input and then adding a bias term [37]. In the proposed model, four convolutional layers are used having filter sizes of 3×3. The filters are initialized with random values and they are learnable network parameters. For instance, in Figure 3, in the *conv2d_1* layer, there are 32 filters of size 3×3 with padding— thus they produce 32 output layers having the same height and width of the input layer. In the proposed model, *conv2d_1* and *conv2d_3* use padding to make the output size as same as the input size; whereas *conv2d_2* and *conv2d_4* do not use padding.
- Activation Layer: The convolutional layers and the dense layers (except the last dense layer) are followed by a nonlinear activation function—the rectified linear unit (ReLU) [38]. A ReLU layer performs a threshold operation on each element, where any value less than zero is set to zero.

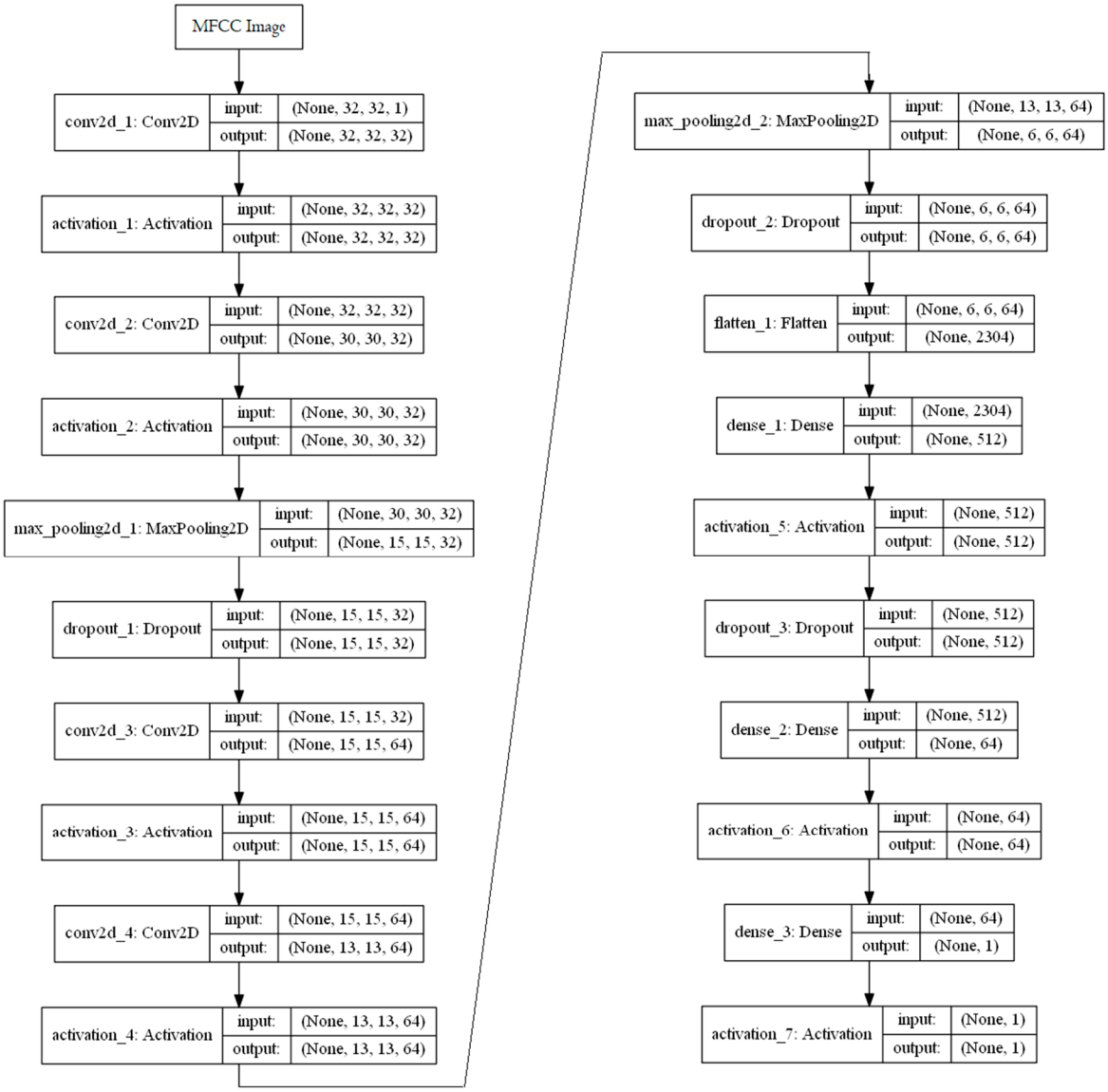


Figure 3. The proposed convolutional neural network architecture.

- Max Pooling Layer: A max-pooling layer performs down-sampling by dividing the input into rectangular pooling regions and computing the maximum of each region [39]. In the proposed model, the dimensions of the pooling region are set to 2×2.
- Dropout Layer: A dropout layer randomly sets input elements to zero with a given probability. This operation effectively changes the underlying network architecture between iterations and helps prevent the network from overfitting [40]. No learning takes place in this layer. In the proposed network architecture, three dropout layers are used to prevent overfitting of training data within a few epochs. The dropout probabilities for layers 1–3 are 0.25, 0.25, and 0.50, namely.
- Flatten Layer: A flatten layer collapses the spatial dimensions of the input and make it a single column vector. In this model, the flatten layer converts the (6, 6, 64) tensor to a one-dimensional vector of size 2304.
- Dense Layer: The dense layer or a fully connected (FC) layer calculates the dot product of the input and a weight matrix, and then adds a bias vector [41,42]. The weight matrix and bias are initialized with random values and they are learnable network parameters.
- Loss Function and Optimizer: The last fully connected layer, dense layer 3, combines the features to classify the images. Therefore, the output size of the last dense layer is set to one for binary

classification and it is followed by the *Sigmoid* [43] score function. A loss function quantifies the agreement between the predicted scores and the ground truth labels and an optimizer tries to reach the global minima where the loss function attains the least possible value for the network parameters. In the proposed model, the *binary_crossentropy* loss is calculated and *RMSprop* [44] optimizer is used.

2.2. Prototype System Architecture

The system architecture comprising of the listener module, the wearable gadget, and the smartphone app—as shown in Figure 2—is designed and developed. The different modules of the system are briefly described below.

2.2.1. Listener Module

The listener module can be attached on the bed headboard or placed on a table near the snorer. It receives the user's command from the smartphone to start/stop the snoring detection. When it detects snoring, it sends the snoring status to the wearable gadget to vibrate. The hardware and firmware of this module are briefly described below.

2.2.1.1. Hardware

The block diagram of the listener module hardware is shown in Figure 4. The single-board computer, Raspberry Pi (RPi) [45], is used as the processor. It contains a 1.2 GHz 64-bit quad-core ARMv8 microprocessor, 1 GB of RAM, micro secure digital (SD) card slot supporting up to 32 GB, onboard Wi-Fi and BLE module, and other built-in hardware peripherals. A microphone with a built-in sound card [46] is connected with the RPi using the USB interface. The power supply for the RPi board is supplied using a 110 V AC to 5.1V DC adapter [47].

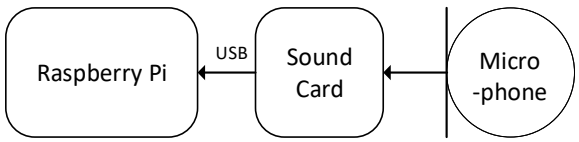


Figure 4. Block diagram of the listener module hardware.

2.2.1.2. Firmware

A Debian-based Linux operating system, Raspbian [45], is installed on a 16 GB SD card of the RPi board. The firmware is developed in Python language and the necessary packages such as Tensorflow, Keras, SpeechPy, Bluepy, and Sounddevice are installed. The deep learning model (H5 file) for snoring detection, the mean and the standard deviation file of the training dataset are transferred in its SD card. The firmware is built on two layers—the driver layer and the application layer. The driver layer consists of low-level firmware for accessing different hardware peripherals. The application layer access the hardware by calling the functions of the driver layer.

A pseudocode of the application layer is shown in Figure 5. The program waits for the smartphone to get connected using Wi-Fi socket. The RPi is configured as the server and the smartphone is configured as the client for the socket connection. The Wi-Fi Internet Protocol (IP) address of the RPi is made static at 192.168.1.55 by editing the *dhcpcd.conf* file [48]—so that smartphone does not need to find the IP address of the RPi each time it wants to connect. Once the connection is made, the smartphone sends the BLE media access control (MAC) address of the wearable gadget to the RPi. After receiving the BLE MAC address, the RPi connects with the wearable gadget using the *Bluepy* library. Then, the program enters in a loop and it continues until the user presses the stop button in the smartphone.

```
while True
        WaitForSocketConnection (HOST_IP, PORT)
        BLE_MAC := GetBLE_MAC ()
        ConnectWithWearableGadget (BLE_MAC)

        Finish := 0
        while not (Finish)
            isSnore := detectSnore ()
            if (isSnore != isSnorePrev)
                SendBLE(isSnore)
                SendLogData (isSnore)
                isSnorePrev := isSnore

             sleep(SAMPLE_DELAY)
             Finish := GetStopMSG ()
        CloseConnections()
```

Figure 5. Pseudocode of the application layer firmware of the listener module.

Inside the loop, the program first detects snoring. To detect snoring, it records the sound for one second at a sampling rate of 48000 using the *Sounddevice* library. Then, the MFCC of the sound signal is calculated using the *SpeechPy* library and then the values are normalized. The signal is then classified using the deep learning model as snoring or non-snoring using the *Keras* library. The model outputs the snoring probability of the recorded sound—thus a probability greater than or equal to 0.5 is classified as snoring sound. Most human snore during the inspiratory phase and the snoring sound has a duration between 1 and 2 seconds [49]. Most adults breathe in the range of 10–15 times in a minute and a snoring episode repeats in 4–6 seconds [50]. In order to avoid false alarms, six sound signals—each having a one-second duration—are recorded and classified one by one. If two of the six sounds are classified as snoring, then a snoring detection is considered. If the snoring detection status is changed, then the status is sent to the wearable gadget using BLE. The status along with the date & time stamp is also sent to the smartphone using Wi-Fi for data logging.

2.2.2. Wearable Gadget

The wearable gadget is a low power electronic device that is worn on the upper hand of the snorer. The device connects with the listener module using BLE. When snoring is detected, it generates vibration until the snorer sleeps on the side. A key design challenge of the gadget is lowering the power consumption—as it is powered by a battery. The hardware and firmware of the device are briefly described below.

2.2.2.1. Hardware

The block diagram of the hardware unit of the wearable gadget is shown in Figure 6. A low-power, small-size system-on-chip (SoC) micro-controller with BLE, nRF52832 [51], is used as the processing and wireless communication unit. The micro-controller contains an industry-standard ARM Cortex M4F MCU, 512 kB in-system programmable flash memory, 64 kB RAM, 2.4 GHz BLE transceiver, general-purpose input/output (GPIO), Timer, and many other peripherals. It has low-power sleep modes and it is suitable for systems where ultra-low power consumption is required to increase battery life.

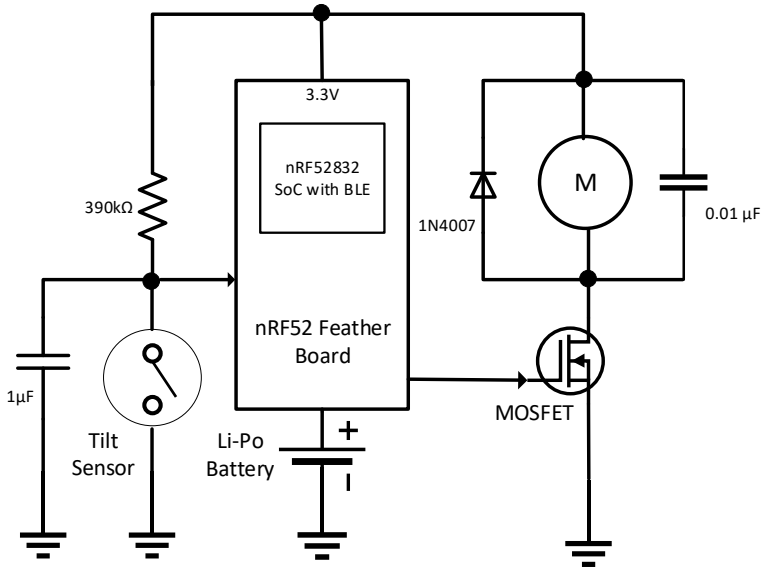


Figure 6. Block diagram of the wearable gadget hardware.

In this project, the nRF52 Feather [52] board is used which consists of the nRF52832 microcontroller, USB-Serial converter for efficient programming and debugging, a connector for 3.7 V lithium polymer (Li-Po) battery, onboard 3.3 V regulator, and a battery charging circuit. The programming and charging circuits only get power when the board is connected with USB and do not consume power when the battery is connected only. When both battery and USB are connected, the charging circuit starts to charge the battery from USB power. A 3.7 V Li-Po rechargeable battery [53] with a capacity of 500 mAh is used as the power source for this hardware unit.

A tilt sensor switch [54] is used to detect the snorer's sleeping orientation—whether they are sleeping on their back or sleeping on their side. A metal ball inside the sensor makes contact with the sensor terminals when the sensor is upright; disconnects the terminals when the sensor is tilted. The sensor is connected to a GPIO interrupt pin of the microcontroller and the interrupt is configured to be triggered whenever it changes. A 1 µF capacitor is connected from the MCU pin to the ground to filter out the bounce signal from the sensor. An external pull-up resistor of 390 kΩ - as shown in Figure 6 - is used to make the pin high when the sensor terminals are open. The nRF52832 has internal pull-up resistors (R_{pu}) having a value of approximately 13 kΩ [51]. If the internal pull-up resistors are enabled, this branch through the switch will continuously consume 3.3 ÷ 13 k = 0.25 mA current whenever the sensor terminals are closed. This is a significant current. To solve this problem, the internal pull-up resistor was disabled and a higher value external pull-up resistor of 390 kΩ is used. In this case, the branch consumes only 3.3 ÷ 390 k = 8.46 µA current (measured 8.24 µA in real-time) when the sensor terminals are closed. Experiments showed that if a higher value of external resistor such as 1 MΩ is used, then too much voltage drops and the microcontroller pin does not recognize an interrupt event. Thus 390 kΩ external pull-up resistor is a good design choice. As the input resistance of the microcontroller pin is high, the sink current is approximately zero (measured 0.01 µA in real-time). Now, when the snorer changes the sleep position on the back, the sensor terminals get open. This causes the microcontroller pin to go high and it triggers a rising interrupt event. When the snorer changes the sleep position on the side, the sensor terminals get closed. This causes the microcontroller pin to go low and it triggers a falling interrupt event.

A vibration motor [55] is used to generate vibration notification—so that the snorer changes the sleeping position to the side. A metal-oxide-semiconductor field-effect transistor (MOSFET) [56] is used to connect and disconnect the motor's ground pin from the power supply ground. The gate of the transistor is controlled by a GPIO pin of the microcontroller as shown in Figure 6. A 0.01 µF capacitor is connected across the motor in parallel to absorb noises generated from the motor brush. When the

motor is turned off, a negative spike of voltage is generated across it. The diode protects the circuit against this by shorting such reverse current from the motor.

2.2.2.2. Firmware

In this proposed system, the listener module sends data to the wearable gadget using BLE. The listener module is configured as *central* and the wearable gadget is configured as *peripheral* [57]. A high-performance Bluetooth 5 qualified protocol stack for the nRF52832 SoC—referred to as *SoftDevice*—is implemented that contains complete stack with Generic Access Profile (GAP), generic attribute profile (GATT), link and other layers [58]. A custom *service*, having a Universally Unique ID (UUID) of 9ECA-DC24-0EE5-A9E0-93F3-A3B5-0100-406E, is created in the GATT profile. Under this service, a custom *characteristic* having a UUID of 9ECA-DC24-0EE5-A9E0-93F3-A3B5-0200-406E is created. The characteristic has a data length of 1 byte—for sending the Boolean snoring status. It's *write* and *write without response* properties were set so that the listener module can write data in this characteristic. A call-back function is also assigned with the characteristic so that an event is raised whenever data is written in this characteristic.

A flowchart of the wearable gadget firmware is shown in Figure 7. Once the listener module receives the BLE MAC of the wearable gadget from the smartphone, it requests a connection, and then a connection is established with the wearable gadget. A real-time operating system (RTOS), known as *FreeRTOS* [59], is used in the firmware. The firmware is designed by an using event response—known as *call-backs*—without always running codes inside a loop. The firmware goes to low power sleep mode and waits for an event to happen to wake-up.

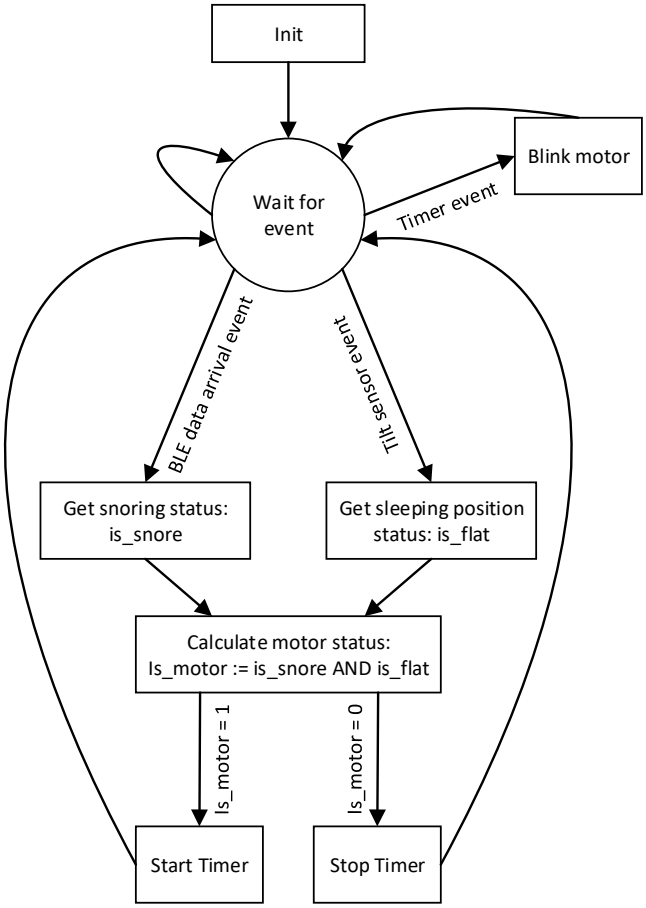


Figure 7. Flowchart of the wearable gadget firmware.

Whenever the listener module sends the snoring status to the wearable gadget, a BLE data arrival event occurs. Whenever the snorer changes sleeping position, the tilt sensor interrupt event occurs. On those events - the gadget wakes up from sleep mode, updates the snoring and sleeping position status namely, and recalculates the vibration motor status. The vibration motor status is set to true when snoring is detected, and the person is sleeping flat on their back. If the vibration motor status is true, then a timer having an interval of 1 second is started. It triggers a timer event every after one second and the vibration motor is turned ON and OFF on alternate seconds. If the vibration motor status is false, then the motor is turned OFF and the timer is stopped. After starting or stopping the timer, the program goes to sleep mode and waits for an event.

To make the program power efficient, the following configurations were done:

- The frequency of the RTOS tick interrupt (referred to as configTICK_RATE_HZ) was reduced from 1024 Hz to 4 Hz to reduce power. This change caused 200 µA current reduction. The tick interrupt is used to measure time. Therefore, a higher tick frequency means the timer can measure time to a higher resolution. The RTOS scheduler will share processor time between tasks of the same priority by switching between the tasks during each RTOS tick. A high tick rate frequency will therefore also have the effect of reducing the 'time slice' given to each task [59]. The system clock of the microcontroller (referred to as SystemCoreClock) is 64 MHz and it is not changed. Thus, the microcontroller executes instructions at a high speed without sacrificing the performance.
- No floating-point numbers were used because the floating-point unit (FPU) consumes significant power. To take the microcontroller in low power mode, the exception flags and the pending FPU interrupts were cleared [60].
- The DC/DC converter of the microcontroller is enabled instead of low dropout (LDO) regulator [61].
- The advertising BLE connection interval was set to 20 ms and the transmission power level was set to 0 dB to balance between power consumption and performance. We did an experiment by reducing the transmission power level to -30 dB—however, no significant reduction of current consumption was noticed, but created a problem during connection.

2.2.3. Smartphone App

The smartphone app is developed for the Android platform. The first screen of the app contains a list-view box showing information about each sleeping session—snoring duration, the percentage of the snoring time in the session, and the session start date and time. It also contains a button to start and to stop a session, and a label showing the connection status with the listener module.

The app contains a settings menu for configuring the listener module's IP and wearable gadget's BLE MAC. The IP of the listener module can be found by visiting the router devices list webpage or using the Fing app [62]. As the IP of the listener module was made static, the user will only need to configure the IP once. To get the BLE MAC of the wearable gadget, the user presses the 'Start Scan' button to find the nearby BLE devices that are advertising. The nearby BLE devices are added in a list-view box and the user can select the required MAC address from the list. The listener module's IP and the wearable gadget's BLE MAC address are saved in a settings file.

After configuration, the user presses the 'start' button to get connected with the listener module using home Wi-Fi. This is done using socket connection where the listener module is configured as a *server* and the smartphone is configured as a *client*. Once the connection is made, the smartphone app sends the BLE MAC address to the listener module using the socket connection. After receiving the BLE MAC, the listener module connects with the wearable gadget and starts detecting snoring sound as discussed in Section 2.2.1. Whenever a change is the snoring status occurs, the listener module sends the snoring status with the date and time stamp to the smartphone using the Wi-Fi socket connection. After receiving the data, the app stores them in a log file and keep track of total snoring duration of that session. When the user presses the 'stop' button to end the sleep session, the socket is disconnected, the session duration and snoring duration are calculated and appended in a file with session start date and time, and session information is displayed in the front screen's list view box.

To monitor snoring durations and progress, a graph generation function is also added in the smartphone app where the user can select any date, and the total snoring time of each date in the last seven days can be displayed as a bar graph. This is calculated from the data stored in the log file.

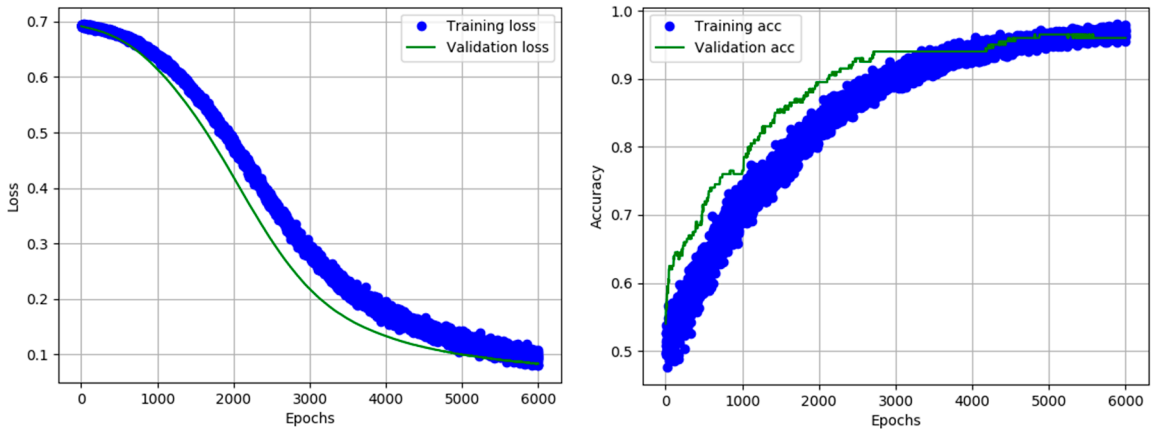
3. Results

3.1. Snoring Detection Deep Learning Model Results

The dataset of 1000 sounds was first converted to 1000 MFCC images. Then the images were randomly shuffled preserving the ground truth label correspondence. Then the dataset is split into three—700 images for training, 200 images for validation, and 100 images for testing. The 100 images for testing were kept separate until the model is trained and validated. Then the final accuracy of the model was evaluated with the unseen test images.

The deep learning network, as shown in Figure 3, was constructed in Python using Keras library. Keras is a high-level neural networks application programming interface (API), written in Python and it runs on top of TensorFlow [63]. The model was trained on a desktop computer with Intel Core i7 (6 Cores) 4.6 GHz microprocessor, 32 GB RAM and NVIDIA GeForce GTX1060 graphics processing unit (GPU).

The model was trained and validated simultaneously for 6000 epochs with a learning rate of 1×10^{-5}. The training and validation time was approximately seven minutes and 33 s. The plots of loss w.r.t. epochs and accuracy w.r.t. epochs are shown in Figure 8a,b namely for both training and validation dataset. From Figure 8, we see that both training and validation loss decrease, and that accuracy increases with more epochs. After 6000 epochs, the training loss is 0.10 and the validation loss is 0.08. Both the training and validation dataset gets an accuracy of approximately 0.96 after 6000 epochs.



(**a**) (**b**)

Figure 8. (**a**) Loss vs epochs for training and validation datasets; (**b**) Accuracy vs epochs for training and validation datasets.

After the training and validation are done, the model and its total 1,278,049 learned parameters (i.e., filters, weights, and biases) were saved in an H5 file. The size of the model in the disk is 9.80 MB. Then the model was tested with an unseen test set of 100 MFCC images. For the test set, the loss was 0.13 and accuracy was 0.96. Here, we see that the model has similar accuracy for the test set when compared with the training and validation set—indicating that the model is well generalized.

3.2. Prototype System Results

A prototype of the proposed system comprising of the listener module, wearable gadget, and a smartphone app has been developed and tested successfully. A photograph of the listener module is

shown in Figure 9. The hardware is designed with a Raspberry Pi board interfaced with a microphone with a sound card—is shown in Figure 9. The hardware was programmed according to the discussion in Section 2.2.1.2 and was configured to run the program automatically on boot. On the RPi, the average pre-processing time of one recorded sound—which includes MFCC generation and normalization—is approximately 53 ms and the classification time by the deep learning model is approximately 47 ms.

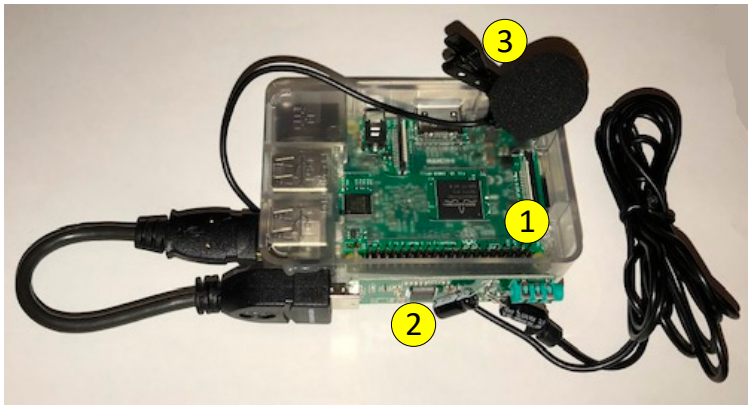
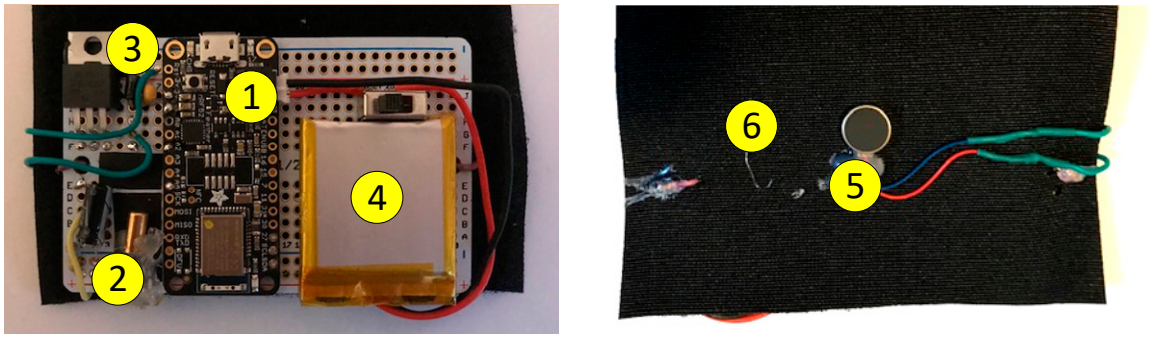


Figure 9. Photograph of the listener module—(1) Raspberry Pi; (2) Sound card; (3) Microphone.

The photograph of the wearable gadget on a proto-board is shown in Figure 10.



(**a**) (**b**)

Figure 10. Photograph of the wearable gadget—(**a**) Top view: (1) SoC microcontroller with BLE and battery charger; (2) Tilt sensor; (3) MOSFET; (4) Li-Po rechargeable battery. (**b**) Bottom view: (5) Vibration motor; (6) Gadget sewed on loop style non-adhesive nylon strips fabric.

The gadget has a size of 8 × 5 cm. An ammeter was connected through the positive wire of the battery to measure the current consumption. The measured current consumptions of the gadget at different states are shown in Table 1.

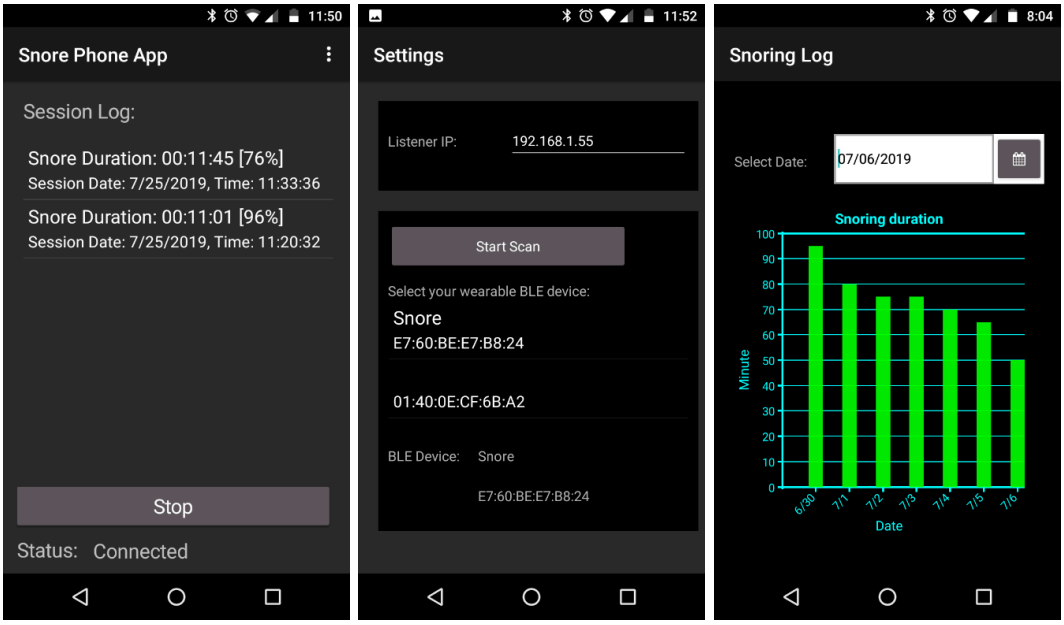
Table 1. Measured current consumption of the gadget at different states.

State	Current Consumption (μA)
Advertising	~195
Idle	240
Generating vibration	635

The BLE will advertise only when it needs to be discovered. Once discovered and connected with the listener module, the device consumes 240 μA when it is waiting for an event in idle state. It consumes 635 μA when the vibration motor is on. The prototype uses a Li-Po rechargeable battery having a capacity of 500 mAh. Assuming the gadget is powered on for eight hours and the snoring

happens for four hours in one day—the gadget will be in the idle state for 6 hours and generate vibration for 2 h as the vibration motor is blinked with a frequency of 0.5 Hz. In this case, the battery life of the gadget is approximately $500 \div (2 \times 0.635 + 6 \times 0.240) = 184$ days.

Some screenshots of the smartphone app are shown in Figure 11. The first screen of the app contains a list-view box showing information about each sleeping session, a button to start and to stop a session, and a label showing the connection status with the listener module—as shown in Figure 11a. Figure 11b shows the settings screen for configuring listener module's IP and wearable gadgets BLE MAC address. Figure 11c shows the bar graph of snoring duration on last seven days of the selected date.



(**a**) (**b**) (**c**)

Figure 11. Screenshots of the smartphone app—(**a**) front screen showing session log, start/stop session button and connection status with listener module; (**b**) setting screen for configuring listener module's IP and wearable gadgets BLE MAC address; (**c**) bar graph of snoring duration on last seven days of the selected date.

3.3. Experimental Setup and Results

The wearable gadget was put on the upper left arm of a mannequin as shown in Figure 12. The gadget was approximately 30° tilted towards the floor so that the tilt sensor terminals are disconnected when the mannequin is sleeping on the back. A speaker was attached at the place of the nose of the mannequin and it was connected with a laptop to generate snoring and non-snoring sounds. The experimental setup is shown in Figure 13. The listener module was attached near the front wall of the mannequin. The smartphone app was used to start the sleep session. Snoring sounds were played using the laptop through the speaker placed at the nose. The listener module successfully detected the snoring and sent snoring status to the wearable gadget using BLE. The wearable gadget vibrated as long as the snoring sound was playing and stopped vibrating when the snoring sound was stopped. The vibration also stopped when the sleeping position was changed to the side as shown in Figure 13b. Non-snoring sounds such as silence, street sound, thunderstorm, door opening were played, and the listener module classified them as non-snoring sounds.

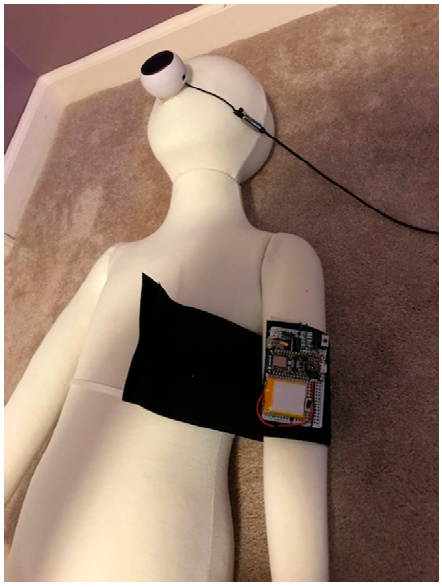
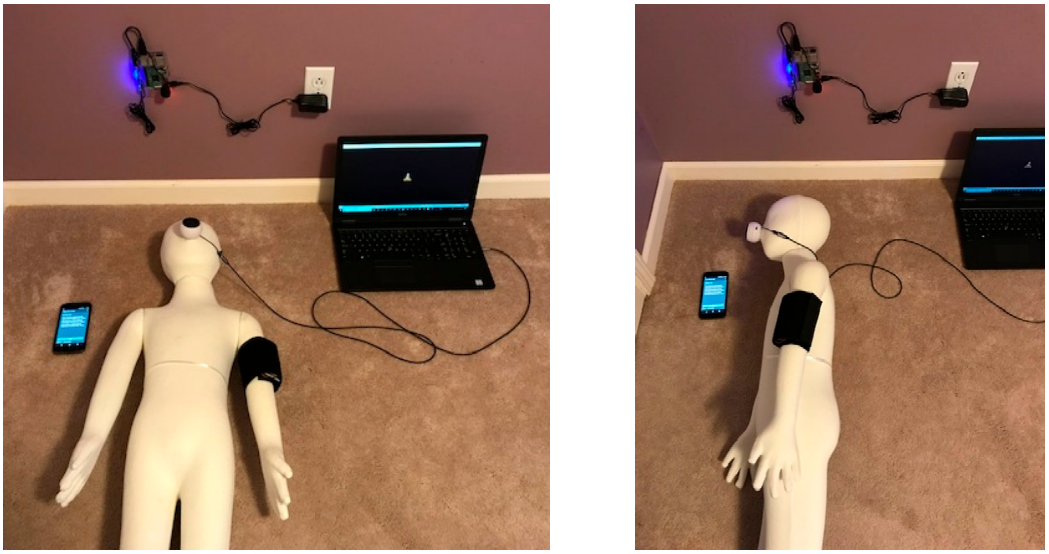


Figure 12. Wearable gadget attached at the upper left arm of a mannequin.



(**a**) (**b**)

Figure 13. Photograph of the experimental setup. (**a**) The wearable gadget generates vibration when the listener module detects snoring and the mannequin is sleeping on the back. (**b**) Vibration stops when the mannequin's sleeping position is changed to the side.

3.4. Comparison with Other Works

A comparison with other related works is shown in Table 2. Here we see that the proposed wearable device is more comfortable than the mechanical devices as it will not hinder the natural mouth or nose functions. The listener module detects snoring using CNN-based deep learning method and have a good detection accuracy of 96%. The wearable gadget applies vibration alert on the upper arm until the snorer sleeps on the side to prevent snoring. The smartphone app also logs snoring data and generates bar graphs that can be used to monitor and to treat snoring related diseases.

Table 2. Comparison with other works.

	Snoring Detection	Snoring Detection Accuracy	Snoring Prevention	Obstruct Any Natural Body Function?	Data Logging in Smartphone
Mechanical devices [11–14]	No	N/A	Mechanical device on mouth moving the lower jaw slightly forward to maintain an open airway	Yes, Mouth	No
Theravent [15]	No	N/A	Strip on nose creates pressure in the airway to reduce vibration.	Yes, Nose	No
Smart Nora [16]	Yes	– [1]	Hardware placed under the pillow starts to move to stimulates the throat muscles for breathing	No	No
SnoreLab [17]	Yes	– [1]	No	No	Yes
R. Nonaka, et. al. [26]	Yes, using logistic regression classifier with auditory image modeling features	97.30%	No	No	No
E. Dafna, et. al. [49]	Yes, AdaBoost classifier with 34 time and spectral features	98.40%	No	No	No
H. Romero, et. al. [64]	Yes, using deep learning with bottleneck features.	91.11%	No	No	No
T. Emoto, et. al. [65]	Yes, deep neural network	75.10%	No	No	No
B. Arsenali, et. al. [66]	Yes, Recurrent Neural Network with MFCC feature	95.00%	No	No	No
Proposed	Yes, CNN based deep learning with MFCC feature	96.00%	Vibration on arm until sleep on the side	No	Yes

[1] not reported.

4. Discussion

In the dataset, each sample has a duration of one second. When 2D MFCC is generated from the one second 1D sample, it generates 32 timeframes i.e., 32 columns. The size of the MFCC is 32 × 32. These dimensions are powers of two—thus, they're good for computer memory and the input layer of the deep learning model. It is possible to make each sample larger. However, this will make the size of the input layer—causing an increase in model size, training time, and latency (i.e., classification time). Moreover, most human snore during the inspiratory phase and the snoring sound has a duration of 1–2 s [49]. Thus, choosing one second sample size is a good choice.

CNN has the ability to extract feature from the time domain sound wave, thus it is possible to directly feed the 1D sound wave to the CNN—without calculating the MFCC. However, each snoring sound is one second in duration and recorded at a sampling rate of 48,000. Thus, the 1D signal has a size of 1 × 48,000. This will make the input layer size of the deep learning model large and will increase the model size, training time, and the latency (i.e., classification time). Using MFCC, the input layer size of the model is 32 × 32 = 1024, which is about 47 times smaller than the time domain signal. This reduces the model size, training time, and latency. The only tradeoff in this approach is to spend approximately 53 ms for MFCC generation.

There are several popular CNN architectures found in the literature. Some examples of such models are LeNet, AlexNet, VGG Net, NiN, DenseNet, FractalNet, GoogLeNet with Inception units, and Residual Networks [67]. For instance, VGGNet is slow to train and the network architecture weights themselves are quite large. Due to its depth and number of fully connected nodes, VGG is over 533 MB for VGG16 and 574 MB for VGG19. Even though ResNet is much deeper than VGG16 and VGG19, the model size is about 102 MB for ResNet50. The weights for Inception V3 are smaller than both VGG and ResNet, coming in at 96 MB [68]. The large size makes deploying these models in an embedded system difficult. Picking the right network architecture requires running several

architectures by changing network layers and hyper-parameters - then choosing the best architecture. This is more of an art than a science [69]. In this work, several architectures were tested by rearranging the layers and changing the network hyper-parameters - and the proposed model was found to be working well in terms of accuracy, training time, and model size. The model has a good accuracy of 96%, the training time of only 7 minutes and 33 s, and the model size of only 9.8 MB. Moreover, some of these popular CNN models are for multiclass classification problems—aimed for specific dataset such as ImageNet, whereas, the proposed model is designed for binary classification problem - aimed to classify the new snoring dataset.

The proposed system worked as expected. However, during the prototype testing, we found that the listener module sometimes classifies an unknown sound—which was not in the non-snoring dataset—as snoring sound and thus creating a false alarm. This problem can be solved by adding more samples of non-snoring sounds in the dataset. Another way to solve this problem is to design the system in such a way that the model will learn the possible snoring and non-snoring sound in the house at run time dynamically as trained by the user.

It is possible to program the smartphone app to detect snoring and then send the snoring status to the wearable gadget using BLE—omitting the listener module from the proposed system. However, at the time writing this paper, a well maintained and stable compiler to run python libraries with deep learning models in Android is hardly available. We plan to implement the listener module to the smartphone as future work.

To find out the user acceptance of the technology, a group of survey participants can be hired who will use the prototype for several months and then complete a feedback form. This will provide information about the strengths and weaknesses of the design from the user's perspective.

Another test with participants is important in a sleep lab - is to observe the behavior of the snorers when vibration notification is generated. The snorer might notice the vibration and sleep on the side—as expected. It may happen that the snorer is not alert by the vibration during sleep and do nothing. It may also happen that the snorer is alert by the vibration but do unexpected actions such as removing the gadget. As future work, we plan to make several prototypes of the system and apply for Institutional Review Board (IRB) approval to perform tests with human subjects in a sleep lab. Once approved, the tests for user acceptance of the technology and the tests on user behavior will be conducted.

It should be noted that although the deep learning model showed 96% accuracy for unseen snoring sounds, clinical trials are required with several participants to verify the accuracy in real-time - as snoring sound may differ from individual to individual. Moreover, an option to retrain the deep learning model with the individual user's snoring sound should be added in the listener module. This will adjust the model to that particular user and can increase detection accuracy and user satisfaction. We plan to do the clinical trials with participants after IRB approval and implement real-time training option of the model with the individual user's snoring sound as future work.

Some future works include making a larger dataset and using data augmentation for training the deep learning model, adding another tilt sensor in the gadget to detect sleeping on both left and right sides, making the gadget size smaller by using surface-mounted device (SMD) components, and making a flexible printed circuit board (PCB) for the gadget.

5. Conclusions

In this paper, a deep learning model for snoring detection is trained, validated and tested. A prototype system comprising of a listener module for snoring detection, a low power wearable gadget to notify the snorer, using vibration, to sleep on their side, and a smartphone app to log snoring data is developed and tested successfully.

Author Contributions: Conceptualization, methodology, software, validation, analysis, investigation, resources, writing—original draft preparation, writing—review and editing, visualization, supervision, project administration, funding acquisition—by T.K.

Funding: This research was funded by the Summer Research/Creative Activity (SRA) award of Eastern Michigan University.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mineo, L. Good Genes Are Nice, But Joy Is Better. The Harvard Gazette. Available online: https://news. harvard.edu/gazette/story/2017/04/over-nearly-80-years-harvard-study-has-been-showing-how-to-live-a-healthy-and-happy-life/ (accessed on 6 August 2019).
2. Snoring—Overview and Facts. Available online: http://sleepeducation.org/essentials-in-sleep/snoring/overview-and-facts (accessed on 6 August 2019).
3. Melone, L. 7 Easy Fixes for Snoring. Available online: https://www.webmd.com/sleep-disorders/features/easy-snoring-remedies#1 (accessed on 6 August 2019).
4. Vann, M.R. 11 Health Risks of Snoring. Available online: https://www.everydayhealth.com/news/eleven-health-risks-snoring/ (accessed on 6 August 2019).
5. Lee, G.S.; Lee, L.A.; Wang, C.Y.; Chen, N.H.; Fang, T.J.; Huang, C.G.; Cheng, W.N.; Li, H.Y. The Frequency and Energy of Snoring Sounds Are Associated with Common Carotid Artery Intima-Media Thickness in Obstructive Sleep Apnea Patients. *Sci. Rep.* **2016**, *6*, 30559. [CrossRef]
6. Alencar, A.M.; da Silva, D.G.V.; Oliveira, C.B.; Vieira, A.P.; Moriya, H.T.; Lorenzi-Filho, G. Dynamics of snoring sounds and its connection with obstructive sleep apnea. *Phys. A Stat. Mech. Its Appl.* **2013**, *392*, 271–277. [CrossRef]
7. Smith, D.L.; Gozal, D.; Hunter, S.J.; Gozal, L.K. Frequency of snoring, rather than apnea–hypopnea index, predicts both cognitive and behavioral problems in young children. *Sleep Med.* **2017**, *34*, 170–178. [CrossRef] [PubMed]
8. Yunus, F.M.; Khan, S.; Mitra, D.K.; Mistry, S.K.; Afsana, K.; Rahman, M. Relationship of sleep pattern and snoring with chronic disease: Findings from a nationwide population-based survey. *Sleep Health* **2018**, *4*, 40–48. [CrossRef] [PubMed]
9. Breus, M.J. How to Keep Snoring from Hurting Your Relationship. Available online: https://www.psychologytoday.com/us/blog/sleep-newzzz/201412/how-keep-snoring-hurting-your-relationship (accessed on 6 August 2019).
10. Why Sleep Matters: Quantifying the Economic Costs of Insufficient Sleep. Available online: https://www.rand.org/randeurope/research/projects/the-value-of-the-sleep-economy.html (accessed on 6 August 2019).
11. SnoreRx. Available online: https://www.snorerx.com/ (accessed on 6 August 2019).
12. ZQuiet. Available online: https://zquiet.com/ (accessed on 6 August 2019).
13. Good Morning Snore Solution. Available online: https://goodmorningsnoresolution.com/ (accessed on 6 August 2019).
14. VitalSleep Anti-Snoring Mouthpiece. Available online: https://www.vitalsleep.com/anti-snoring-device-by-vital-sleep.html (accessed on 6 August 2019).
15. Theravent. Available online: https://www.theraventsnoring.com (accessed on 6 August 2019).
16. Smartnora. Available online: https://www.smartnora.com/ (accessed on 6 August 2019).
17. SnoreLab. Available online: https://www.snorelab.com (accessed on 6 August 2019).
18. Agrawal, S.; Stone, P.; McGuinness, K.; Morris, J.; Camilleri, A.E. Sound frequency analysis and the site of snoring in natural and induced sleep. *Clin. Otolaryngol. Allied Sci.* **2002**, *27*, 162–166. [CrossRef] [PubMed]
19. Shiomi, F.K.; Pisa, I.T.; Campos, C.J.R. Computerized analysis of snoring in Sleep Apnea Syndrome. *Braz. J. Otorhinolaryngol.* **2011**, *77*, 488–498. [CrossRef] [PubMed]
20. Pevernagie, D.; Aarts, R.M.; De Meyer, M. The acoustics of snoring. *Sleep Med. Rev.* **2009**, *14*, 131–144. [CrossRef] [PubMed]
21. Calabrese, B.; Pucci, F.; Sturniolo, M.; Guzzi, P.H.; Veltri, P.; Gambardella, A.; Cannataro, M. A System for the Analysis of Snore Signals. *Procedia Comput. Sci.* **2011**, *4*, 1101–1108. [CrossRef]
22. Al-Mardini, M.; Aloul, F.; Sagahyroon, A.; Al-Husseini, L. Classifying obstructive sleep apnea using smartphones. *J. Biomed. Inform.* **2014**, *52*, 251–259. [CrossRef] [PubMed]
23. Koo, S.K.; Kwon, S.B.; Moon, J.S.; Lee, S.H.; Lee, H.B.; Lee, S.J. Comparison of snoring sounds between natural and drug-induced sleep recorded using a smartphone. *Auris Nasus Larynx* **2018**, *45*, 777–782. [PubMed]

24. Markandeya, M.N.; Abeyratne, U.R.; Hukins, C. Characterisation of upper airway obstructions using wide-band snoring sounds. *Biomed. Signal Process. Control* **2018**, *46*, 201–211. [CrossRef]
25. Hara, H.; Tsutsumi, M.; Tarumoto, S.; Shiga, T.; Yamashita, H. Validation of a new snoring detection device based on a hysteresis extraction algorithm. *Auris Nasus Larynx* **2017**, *44*, 576–582. [CrossRef] [PubMed]
26. Nonaka, R.; Emoto, T.; Abeyratne, U.R.; Jinnouchi, O.; Kawata, I.; Ohnishi, H.; Akutagawa, M.; Konaka, S.; Kinouchi, Y. Automatic snore sound extraction from sleep sound recordings via auditory image modeling. *Biomed. Signal Process. Control* **2016**, *27*, 7–14. [CrossRef]
27. Snoring Sounds. Available online: https://www.soundsnap.com/tags/snoring (accessed on 6 August 2019).
28. Breathing and Snoring Sound Effects. Available online: https://www.zapsplat.com/sound-effect-category/breathing-and-snoring (accessed on 6 August 2019).
29. People Snoring Sound Effects. Available online: https://www.fesliyanstudios.com/royalty-free-sound-effects-download/people-snoring-189 (accessed on 6 August 2019).
30. 10 Minutes Snoring Sound. Available online: https://www.youtube.com/watch?v=1deTKPX1j8c (accessed on 6 August 2019).
31. Actual Sound of a Man Snoring. Available online: https://www.youtube.com/watch?v=SOxwffK0xUc (accessed on 6 August 2019).
32. WavePad Audio Editing Software. Available online: https://www.nch.com.au/wavepad/index.html (accessed on 6 August 2019).
33. Davis, S.; Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. Acoust. Speech Signal Process.* **1980**, *28*, 357–366. [CrossRef]
34. Fayek, H. Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between. Available online: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html (accessed on 6 August 2019).
35. Mohamed, A. Deep Neural Network Acoustic Models for ASR. Ph.D. Thesis, University of Toronto, Toronto, ON, Canada, 2014. Available online: https://tspace.library.utoronto.ca/bitstream/1807/44123/1/Mohamed_Abdel-rahman_201406_PhD_thesis.pdf (accessed on 6 August 2019).
36. SpeechPy. Available online: https://speechpy.readthedocs.io/en/latest/intro/introductions.html (accessed on 6 August 2019).
37. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
38. Vinod, N.; Hinton, G.E. Rectified linear units improve restricted Boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
39. Nagi, J.; Ducatelle, F.; di Caro, G.A.; Ciresan, D.; Meier, U.; Giusti, A.; Nagi, F.; Schmidhuber, J.; Gambardella, L.M. Max-Pooling Convolutional Neural Networks for Vision-based Hand Gesture Recognition. In Proceedings of the IEEE International Conference on Signal and Image Processing Applications (ICSIPA2011), Kuala Lumpur, Malaysia, 16–18 November 2011.
40. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
41. Xavier, G.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
42. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 13–16 December 2015; pp. 1026–1034.
43. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006.
44. A Look at Gradient Descent and RMSprop Optimizers. Available online: https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b (accessed on 6 August 2019).
45. Raspberry, Pi. Available online: https://www.raspberrypi.org (accessed on 6 August 2019).
46. USB Lavalier Lapel Microphon. Available online: https://www.amazon.com/Lavalier-Microphone-Cardioid-Condenser-K053/dp/B077VNGVL2 (accessed on 6 August 2019).
47. DC Power Supply. Available online: https://www.sparkfun.com/products/13831 (accessed on 6 August 2019).
48. How to Give Your Raspberry Pi a Static IP Address. Available online: https://thepihut.com/blogs/raspberry-pi-tutorials/how-to-give-your-raspberry-pi-a-static-ip-address-update (accessed on 6 August 2019).

49. Dafna, E.; Tarasiuk, A.; Zigel, Y. Automatic Detection of Whole Night Snoring Events Using Non-Contact Microphone. *PLoS ONE* **2013**, *8*, e84139. [CrossRef] [PubMed]
50. Mesquita, J.; Solà-Soler, J.; Fiz, J.A.; Morera, J.; Jané, R. All night analysis of time interval between snores in subjects with sleep apnea hypopnea syndrome. *Med. Biol. Eng. Comput.* **2012**, *50*, 373–381. [CrossRef] [PubMed]
51. nRF52832 SoC. Available online: https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52832 (accessed on 6 August 2019).
52. Adafruit Feather nRF52 Bluefruit LE—nRF52832. Available online: https://www.adafruit.com/product/3406 (accessed on 6 August 2019).
53. Lithium Ion Polymer Battery—3.7v 500mAh. Available online: https://www.adafruit.com/product/1578 (accessed on 6 August 2019).
54. Tilt Sensor. Available online: https://www.sparkfun.com/products/10289 (accessed on 6 August 2019).
55. Vibration Motor. Available online: https://www.amazon.com/tatoko-12000RPM-Wired-Phone-Vibration/dp/B07L5V5GYG (accessed on 6 August 2019).
56. MOSFET N-CH 30V 24A. Available online: https://www.digikey.com/product-detail/en/infineon-technologies/IRL2703PBF/IRL2703PBF-ND/811700 (accessed on 6 August 2019).
57. Davidson, R.; Townsend, K.; Wang, C.; Cufí, C. *Getting Started with Bluetooth Low Energy Tools and Techniques for Low-Power Networking*; O'Reilly Media: Sevastopol, CA, USA, 2014.
58. S132 SoftDevice. Available online: https://www.nordicsemi.com/Software-and-Tools/Software/S132 (accessed on 6 August 2019).
59. Free RTOS Customization. Available online: https://www.freertos.org/a00110.html (accessed on 6 August 2019).
60. Floating Point Unit (FPU) of nrf52. Available online: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v12.0.0%2Fhardware_driver_fpu.html&cp=4_0_9_2_4 (accessed on 6 August 2019).
61. Optimizing Power on nRF52 Designs. Available online: https://devzone.nordicsemi.com/nordic/nordic-blog/b/blog/posts/optimizing-power-on-nrf52-designs (accessed on 6 August 2019).
62. Raspberry Pi IP Address. Available online: https://www.raspberrypi.org/documentation/remote-access/ip-address.md (accessed on 6 August 2019).
63. Keras: The Python Deep Learning library. Available online: https://keras.io (accessed on 6 August 2019).
64. Romero, H.E.; Ma, N.; Brown, G.J.; Beeston, A.V.; Hasan, M. Deep Learning Features for Robust Detection of Acoustic Events in Sleep-disordered Breathing. In Proceedings of the ICASSP 2019—2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 810–814.
65. Emoto, T.; Abeyratne, U.R.; Kawano, K.; Okada, T.; Jinnouchi, O.; Kawata, I. Detection of sleep breathing sound based on artificial neural network analysis. *Biomed. Signal Process. Control* **2018**, *41*, 81–89. [CrossRef]
66. Arsenali, B.; van Dijk, J.; Ouweltjes, O.; den Brinker, B.; Pevernagie, D.; Krijn, R.; van Gilst, M.; Overeem, S. Recurrent Neural Network for Classification of Snoring and Non-Snoring Sound Events. In Proceedings of the 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Honolulu, HI, USA, 17–21 July 2018; pp. 328–331.
67. Alom, M.Z.; Taha, T.M.; Yakopcic, C.; Westberg, S.; Sidike, P.; Nasrin, M.S.; Hasan, M.; Van Essen, B.C.; Awwal, A.A.S.; Asari, V.K. A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics* **2019**, *8*, 292. [CrossRef]
68. Rosebrock, A. ImageNet: VGGNet, ResNet, Inception, and Xception with Keras. Available online: https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/ (accessed on 20 August 2019).
69. Chollet, F. *Deep Learning with Python*, 1st ed.; Manning Publications: Shelter Airland, NY, USA, 2017; p. 60.