# Deep Learning vs. Professional Healthcare Equipment: A Fine-grained Breathing rate monitoring Model

**Bang Liu, Xili Dai*, Haigang Gong, Zihao Guo, Nianbo Liu, Xiaomin Wang, Ming Liu**
Big Data Research Center and Department of Computer Science and Engineering
University of Electronic Science and Technology of China, Chengdu, China.
*Corresponding author: daixili_cs@163.com

## Abstract

In mHealth field, accurate breathing rate monitoring technique has benefited a broad array of healthcare related applications. Many approaches try to use smartphone or wearable device with fine-grained monitoring algorithm to accomplish the task, which can only be done by professional medical equipments before. However, such schemes usually result in bad performance in comparison to professional medical equipments. In this paper, we propose DeepFilter, a deep learning based fine-grained breathing rate monitoring algorithm that works on smartphone and achieves professional-level accuracy. DeepFilter is a bi-directional Recurrent Neural Network (RNN) stacked with convolutional layers and speeded up by batch normalization. Moreover, we collect 16.17GB breathing sound recordings data of 248 hours from 109 and another 10 volunteers to train and test our model respectively. The results show a reasonably good accuracy of breathing rate monitoring.

## 1 Introduction

The emergence of mHealth draws much attention both in industry and academy[Organization, 2011]. Google, Microsoft and Apple conduct a series of work on mHealth from hardware to software. Google is the first one to get involved in mHealth. In April 2012, Google released Google Glass[Arnold, 2013] and applied it to healthcare in July 2013[Albrecht *et al.*, 2014]. Pristine declared to develop medical application for Google Glass. After that, Google accomplished the acquisition of a biotech company Liftlabs, which invented an electronic spoon to help Parkinson patients have food. In 2015, Google X announced it was working on wearable suits which can exam cancer cell of users. In addition, Microsoft Band, Apple Watch, Fitbits and Jawbone, and more smart wearable devices bloom up everywhere.

There exists a broad array of healthcare related applications on sleep monitoring by smart wearable devices[Tomlinson *et al.*, 2012]. They often aim at fine-grained breathing rate monitoring as a kind of non-obtrusive sleep monitoring for the understanding of users sleep quality. Since inadequate and irregular sleep can lead to serious health problems such as fatigue, depression, cardiovascular disease and anxiety[Parish, 2009], breathing rate monitoring is critical to detect early signs of several diseases such as diabetes and heart disease [Shaw *et al.*, 2008]. The breathing rate monitoring can also be applied to the sleep apnea diagnosis and treatment, treatment for asthma[Braun *et al.*, 2012] and sleep stage detection[Chung *et al.*, 2007]. Thus, fine-grained breathing rate monitoring is important to facilitate these healthcare related applications.

Traditionally, one's breathing rate can be captured by professional medical equipments, as monitoring machines in hospitals. In most cases, such machines are too expensive, too complex, and too heavy for daily use of ordinary people. A possible solution is to achieve accurate sleep monitoring via smartphone or other devices with recognition algorithm [Hossain *et al.*, 2015], which is more and more popular in current healthcare related applications. For example, [Ren *et al.*, 2015] exploits the readily available smartphone earphone placed close to the user to reliably capture the human breathing sound. It can not work if the earphone is apart from the user. [Liu *et al.*, 2015] tracks the vital signs of both breathing rate and heart rate during sleep, by using off-the-shelf WiFi without any wearable or dedicated devices. However, the wearable devices can not achieve approximative performance in comparison to professional medical equipments. The latter often has a much lower signal-to-noise ratio(SNR), and can achieve a much higher accuracy in breathing rate monitoring.

In this paper, we aim at developing a fine-grained breathing rate monitoring algorithm that works on smartphone and achieves professional-level accuracy. We propose a deep learning model as DeepFilter, which can filter the breathing from low SNR data. We empirically exploit the framework of deep learning and apply it to the fine-grained breathing rate monitoring on smartphone. The deep learning model combines several convolutional layers and a bi-directional recurrent layer, and is trained in an end-to-end manner using the cross entropy loss function. In addition, batch normalization is adapted to speed up the training. Moreover, we collect 16.17GB breathing sound recordings data of 248 hours from 109 and another 10 volunteers to train and test our model respectively. The results show a reasonably good accuracy of breathing rate monitoring.

The main contributions of this paper are highlighted as fol-

lows:

- As our best knowledge, we are the first to apply deep learning to fine-grained breathing rate monitoring, with low-SNR data recognition.
- We run real experiments on smartphone and verify the availability and performance of our model, which directly promotes the sleep monitoring applications in our daily lives.

## 2 Related Work

Since our scheme involves accurate sleep monitoring and deep learning, we mainly discuss the previous work on the two aspects.

Medical-based sleep monitoring systems are often developed for clinical usage. In particular, Polysomnography [Kushida CA, 2005] is used in medical facilities to perform accurate sleep monitoring by attaching multiple sensors on patients, which requires professional installation and maintenance. It can measure many body functions during sleep, including breathing functions, eye movements, heart rhythm, and muscle activity. Such systems incur high cost and are usually limited to clinical usage. DoppleSleep[Rahman *et al.*, 2015], a contactless sleep sensing system that continuously and unobtrusively tracks sleep quality, by using commercial off-the-shelf radar modules.

Some smartphone apps, such as Sleep as Android, Sleep cycle alarm clock and iSleep [Hao *et al.*, 2013], can perform low cost sleep monitoring by using the smartphone built-in microphone and motion sensors. These Apps however only support coarse-grained monitoring, such as the detection of body movements, coughing and snoring. [Chen *et al.*, 2013] utilizes the phone usage features such as the duration of phone lock to measure sleep duration. The Respiratory app [AppCrawir, 2014] derives a persons respiratory rate by analyzing the movements of the users abdomen when placing the phone between the users ribcage and stomach. ApneaApp[Nandakumar *et al.*, 2015], a contactless sleep apnea event detection system that works on smartphone, which does this by transforming the phone into an active sonar system that emits frequency-modulated sound signals and listens to their reflections to capture the breathing movement. [Ren *et al.*, 2015] exploits the readily available smartphone earphone placed close to the user to reliably capture the human breathing sound. [Liu *et al.*, 2015] proposes to track the vital signs of both breathing rate and heart rate during sleep by using off-the-shelf WiFi without any wearable or dedicated devices. There is still a gap between the performance of professional equipment and that of the approaches above.

Recently, deep neural networks are first used for better phone recognition [Mohamed *et al.*, 2012], in which traditional Gaussian mixture models are replaced by deep neural networks that contain many layers of features and a very large number of parameters. Convolutional networks have also been found beneficial for acoustic models. Recurrent neural networks are beginning to be deployed in state-of-the-art recognizers[Graves *et al.*, 2013] and work well with convolutional layers for the feature extraction[Hannun *et al.*, 2014]. We are inspired by the good performance of previous work on speech recognition, and introduce deep learning algorithm into the problem of fine-grained breathing rate monitoring[Sainath *et al.*, 2015].

## 3 DeepFilter

In this section, we introduce the whole framework of DeepFilter, and investigate the training of the model in detail.

### 3.1 The Framework

Figure. 1 shows the framework of DeepFilter. Our model is a RNN that begins with several convolutional input layers, followed by fully connected layer and multiple recurrent (uni or bidirectional) layers, and ended with output layer. The network is trained end-to-end and is added batch normalization with cross entropy loss function.

In speech recognition, "end-to-end" is often used to support the training without aligned training labels, which doesnt involve the frame-level cross entropy loss function. In breathing rate monitoring, we first create frame-aligned training labels, and translate it into a classification problem to decide whether the frames belong to inhaling/exhaling or not. Since one inhaling/exhaling event may involve several frames, we exploit recurrent layers to process input sequence, for the recurrent layers can capture the sequence information to improve the performance. Thus, the input/output sequence of our model is similar to that of speech recognition. The only difference is that the input and output sequences of speech recognition have different lengths while that of our model have the same length.

To one sample $x^i$ and label $y^i$, sequence frames are sampled from training set $\chi = \{X^1, X^2, \cdots, X^N\}$, which generates some voice recordings of size $N$. Assuming $X^i$ is one voice recording of 136 seconds (the sampling rate is 44100Hz), and 40ms is a frame (which is an empirical value that always used as window size in speech recognition). It can be divided into samples $X^i = \{x^1, x^2, \cdots, x^n\}$ and each sample $x^j = (x_1^j, x_2^j, \cdots, x_T^j)$ combines $T = 50$ frames. And one frame $x_t^j = (x_{t,1}^j, x_{t,2}^j, \cdots, x_{t,f}^j)$ is a $f = 1764$ dimension vector($\frac{44100}{1000/40} = 1764$). Thus, a voice recording of 136 seconds can be divided into $n = 68$ samples (136/2=68, 40ms*50=2s). Each sample $x^j$ has a corresponding label sequence $y^j = (y_1^j, y_2^j, \cdots, y_T^j)$, $y_t^j = \{0, 1\}$, in which a frame without breathing is set to 0, and otherwise is set to 1. Generally, the goal of our processing is to convert an input sequence $x^j$ into a 0-1 sequence.

The data described above is suitable as input for a RNN. However, our model is a RNN with several convolutional input layers, which requires the input should be in a two-dimension structure. Thus, we split one 40ms frame into 4 frames with 10ms. Each 10ms frame is translated from time domain to frequency domain through FFT, which produces a 220-dimension vector. Now, we translate one-dimension 40ms frame into a two-dimension spectrogram with the size of 220*4.

The main idea of our scheme is to differ the breathing events from the low SNR recordings. It needs to support the

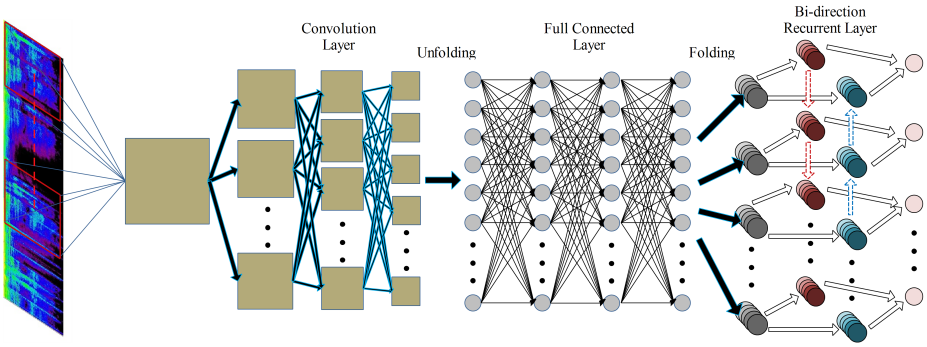



Figure 1: The deep learning framework of our model. The model is a bi-directional RNN, which begins with three convolutional layers (the third layer is mean pooling) and is followed with four fully connected layers. The data is translated into spectrogram as input to the network and is trained end-to-end.

high-frequency signals for learning the fine-grained features in deep learning model. Thus, we use the sampling rate of 44100Hz, which is the highest sampling rate of most smartphone on the market.

### 3.2 Batch Normalization for Deep Bidirectional RNNs

To efficiently absorb data, we increase the depth of the network by adding more convolution and full connected layers. However, it becomes more challenging to train the network using gradient descent as the size and the depth increase, even the Adagrad algorithm could achieve limited improvement. We add batch normalization[Ioffe and Szegedy, 2015] to train the deeper network faster. Recent research has shown that batch normalization can speed convergence, though not always improving generalization error. In contrast, we find that when applied to very deep RNNs, it not only accelerates training, but also substantially reduces final generalization error.

In a typical feed-forward layer containing an affine transformation followed by a non-linearity $f(\cdot)$, we insert a batch normalization transformation by applying $f(Wh + b) \rightarrow f(\mathcal{B}(Wh))$ where

$$\mathcal{B}(x) = \gamma \frac{x - E(x)}{(Var[x] + \epsilon)^{1/2}} + \beta \tag{1}$$

The terms **E** and **Var** are the empirical mean and variance over a mini-batch respectively. The learnable parameters $\gamma$ and $\beta$ allow the layer to scale and shift each hidden unit as desired. The constant $\epsilon$ is small and positive, and is included only for numerical stability. In our convolutional layers, the mean and variance are estimated over all the temporal output units for a given convolutional filter on a mini-batch. The batch normalization transformation reduces *internal covariate shift* by insulating a given layer from potentially uninteresting changes in the mean and variance of the layers input.

A recurrent layer is implemented as:

$$\begin{aligned} \overrightarrow{h}_t^l &= f(W^l h_t^{l-1} + \overrightarrow{U}^l \overrightarrow{h}_{t-1}^l + b^l) \\ \overleftarrow{h}_t^l &= f(W^l h_t^{l-1} + \overleftarrow{U}^l \overleftarrow{h}_{t+1}^l + b^l) \end{aligned} \tag{2}$$

where $\overrightarrow{h}_t^l$ and $\overleftarrow{h}_t^l$ are computed sequentially from $t = 1$ to $t = T$ and from $t = T$ to $t = 1$ respectively. And the $l + 1$ (non-recurrent) layer takes both the forward and backward units as inputs $h_t^{l+1} = f(W^{l+1} h_t^l + b^{l+1})$, where $h_t^l = \overrightarrow{h}_t^l + \overleftarrow{h}_t^l$, and the activation function $f(x) = \min\{\max\{0, x\}, 20\}$ is clipped ReLu.

There are two ways of applying batch normalization to recurrent operation.

$$\begin{aligned} h_t^l &= f(\mathcal{B}(W^l h_t^{l-1} + U^l h_{t-1}^l)) \\ h_t^l &= f(\mathcal{B}(W^l h_t^{l-1}) + U^l h_{t-1}^l) \end{aligned} \tag{3}$$

The first one indicates that the mean and variance statistics are accumulated over a single time-step of the mini-batch, which is ineffective in our study. We find that the second one works well, and [Cooijmans *et al.*, 2016] has explained the reason.

### 3.3 Convolutions

Temporal convolution is commonly used in speech recognition to efficiently model temporal translation invariance for variable length utterances. Convolution in frequency attempts to model spectral variance due to speaker variability more concisely than what is possible with large fully connected networks. We experiment with the use of convolutional layers from one to three. These are both in the time-and-frequency domain (two-dimension) and in the time-only domain (one-dimension).

Some previous works[Amodei *et al.*, 2015] demonstrated that multiple layers of two-dimension convolution improve results substantially than one-dimension convolution. And convolution has good performance on noisy data. A key point and necessary of low-SNR data recognition is denoise[Amodei *et al.*, 2015]. Thus, the convolution component of the model is the key point for it to work well on low-SNR data.

### 3.4 Bidirectional

Recurrent model with only forward recurrences routinely performs worse than similar bidirectional ones, so that implying some amount of future context is vital for good performance. However, bi-directional recurrent models are challenging to

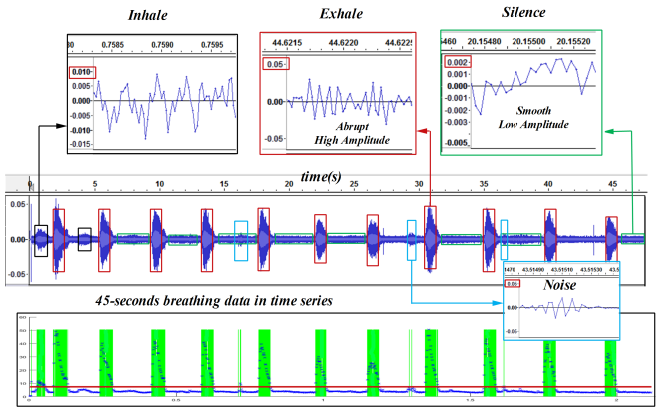



Figure 2: A snapshot shows how we label the data in semi-automatically, which is motivated from the observation of the frequency and the amplitude of the data.

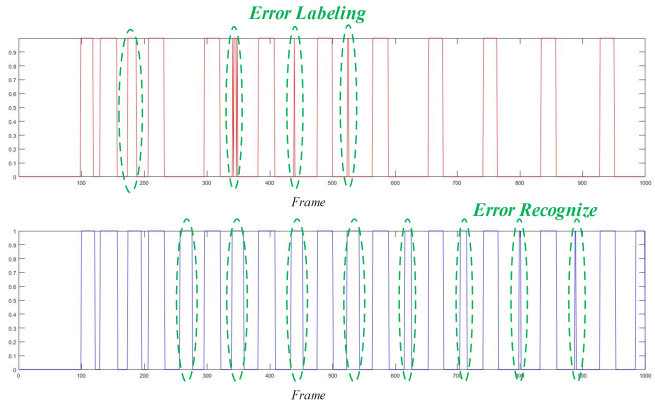



Figure 3: Post Processing. A comparison between truth and test which shows the necessity of post processing.

be deployed in an online, low-latency setting because they cannot stream the transcription process as the utterance arrives from the user. The DeepSpeech2 of Baidu[Amodei *et al.*, 2015] supports a special layer called lookahead convolution, which is a unidirectional model but without any loss in accuracy comparing to a bi-direction one. In our study, we use bi-direction recurrent model because we do not have the constraint of working in real time.

# 4 Fine-Grained Breathing Monitoring

## 4.1 Training Data

Large-scale deep learning systems require an abundance of labeled data. For our system we need a lot of labelled low-SNR data, but it is difficult to label the low-SNR data. To train our large model, we first collect high-SNR data (the data collected in a quiet surrounding in which breathing can be heard clearly), which is much easier to be labelled. After that, we combine the labelled data with pink noise to lower the SNR of the data through data synthesis, which has been successfully applied to data extensions of speech recognition.

Fig. 2 is one high-SNR sound recording of 45 seconds. The 11 red boxes mark 11 expirations, and the black, green and blue boxes mark 2 inspirations, 10 silences and 3 noises respectively. When we enlarge the frames of expiration, inspiration, silence and noise, it is obvious that the differences between adjacent points in a silence frame is much lower than that in other three types of frame on average. And its amplitudes are also minimum, only $10^{-3}$. Then, we give each frame a Frame Index $l(x^i)$ as follows:

$$l(x^i) = \sum_{t=1}^{n} |x_t^i| + |x_{t+1}^i - x_t^i| \tag{4}$$

where $x^i$ is the $i$-th frame in data, $n = 1764$. The sub-figure on the bottom of Fig. 2 shows the label indexes of one sound recording (x-axis is frame, y-axis is corresponding Frame Index). To facilitate observation, we add green lines on the frames whose index is larger than the threshold. It makes the label indexes can distinguish the silence frame from other three types of frame well. For labelling the data, we erased the inspiration and noise manually (since it is a high-SNR data, the volunteers are required to keep quiet during recording), and the threshold is also given manually. All the above actions are conducted in a sound processing software **Audacity**[Audacity, 2015], and the thresholds and labels are determined on **Matlab**.

It took us 3 weeks for labeling about 7458 sound recordings (one recording is about 2 minutes, and the total recordings from 109 people are about 248 hours). Finally, we use **Audacity** to complete the data synthesis discussed above.

During the labelling, we find the number of breathing frames is larger than that of non-breathing frames, which may reduce the classification performance. Thus, we add a weight in loss function as follows:

$$Ł(\widehat{y}, \mathbf{y}) = \frac{1}{T}\sum_{t=1}^{T}[\eta y^t \ln \widehat{y}^t + (1 - y^t)\ln(1 - \widehat{y}^t)] \tag{5}$$

## 4.2 Post-Processing

Fig. 3 is a snapshot of 1000 continuous frames randomly chosen from training data, in which the top figure is the ground truth, while the bottom figure shows the recognition results from our model (the x-axis is the frame, and the y-axis is the label value). As shown in the figure, breathing is continual and periodical, while the incorrectly labelled data and false-recognized frames are abrupt and discrete. Thus, we can define post-processing as follows.

First, we can regard a breathing event as the continuous frames with breathing label. Thus, we delete the breathing events whose frame number is less than a threshold, which is the key point of post-processing. In this study, we choose a value of 50 because a breathing time less than 0.5s is abnormal according to our tests. As shown in Fig. 3, the post-processing could remove the green dotted line cycle. The effect of post-processing is shown in Table. 2, and the values of TTR(Test ground-Truth Rate) can demonstrate its efficiency (TTR is a metric described in Section 5.3).

# 5 Experiment

## 5.1 Training

We train our deep RNN on 2 work stations with the configuration as: Intel Xeon processor E5-2620, 32GB memory,

| Model Configuration | | |
|---|---|---|
| DeepFilter 1 | DeepFilter 2 | DeepFilter 3 |
| 6 weight layers | 7 weight layers | 6 weight layers |
| input (882×50 sequence) | input (220×4×50 spectrogram) | |
| one-dimension-conv9-16 | two-dimension-conv3-16<br>two-dimension-conv3-32 | two-dimension-conv3-16<br>two-dimension-conv3-32 |
| mean-pool | | |
| FC-512 | | |
| FC-512 | | |
| FC-512 | | |
| Uni-128<br>Uni-128 | Uni-128<br>Uni-128 | Bi-128 |
| $\alpha = 1e^{-3}$ | | |
| Output: Sigmod | | |
| Loss: Cross-Entropy | | |

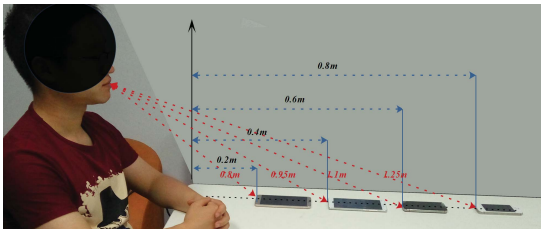
Table 1: Model parameters. The convolutional layer parameters are denoted as "⟨ convolutional type⟩ conv⟨ receptive field size⟩-⟨ number of channels⟩". And the parameters of recurrent layer are denoted as "⟨ Uni or Bi ⟩-⟨ number of hidden units⟩ ", where "Uni" denoted unidirectional recurrent and "Bi" denoted bi-directional recurrent. The ReLU activation function is not shown for brevity.

standard AnyFabric 4 GB Ethernet adapter, and Nvidia GTX Titan GPU with 12GB memory. We use a PC as *parameter server* and 2 work stations as *workers*. Each individual machine sends gradient updates to the centralized parameter repository, which coordinates these updates and sends back updated parameters to the individual machines running the model training. We use a public deep learning library Tensorflow[**?**] to implement our system.

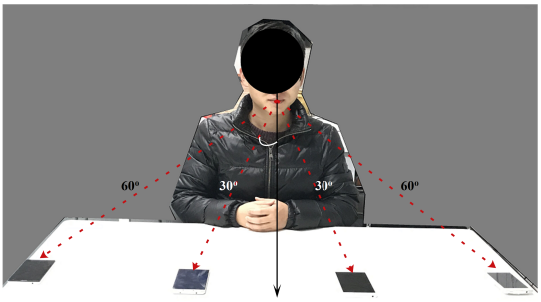
There are four deep learning models trained in our study. The baseline model is a unidirectional RNN with 4 hidden layers, and the last hidden layer is recurrent layer. The framework is 882-2000-1000-500-100-1, and learning rates $\alpha = 10^{-5}$ without momentum. Training data is described in prior section, and we take 40 ms as a frame and 50 frames as a group($T = 50$). Another three models are also RNNs, but begin with convolution layers. The detailed model parameters are listed in Table. 3. The third line in the table is the number of hidden layers respectively. The convolution in one-dimension and that in two-dimension have different inputs. For one-dimension convolution, the input is a 40ms frame and is translated to frequency domain with 882 dimensions. The input of two-dimension convolution is a 40ms frame too, and is translated into spectrogram(4×220). Then, one or two convolutional layers are followed with a mean pooling layer, and the mean pooling size is $3 \times 2$($4 \times 4$ for one-dimension mean pooling). All models have three fully connected layers, and each layer has 512 units. They are ended with two unidirectional recurrent layers except DeepFilter 3, which is ended with one bi-directional recurrent layer. All the models are trained through Adagrad with initial learning rate $\alpha = 1e^{-3}$.

## 5.2 Experimental Data

Fig. 4 shows the procedure of data collection. In Fig. 4(a), the volunteer sits in the front of desk, and four smartphones are placed on the desk with a distance of 0.2m, 0.4m, 0.6m, and 0.8m from the margin of desk respectively (the distances from volunteer's nose to smartphones are far enough, which




(a) Four smartphones in different distances




(b) Four smartphones in different angles

Figure 4: Collecting the test data

reach 0.6m, 0.85m, 1.0m, and 1.2m respectively). The further the distance is, the lower the SNR is, and the more difficult the labelling of data is. We make four smartphones to possess the same label by collecting data in synchronization, while it is easy to label the nearest one. In Fig. 4(b), the volunteer sits in the front of desk, with four smartphones on the desk in a distance of 0.4m from the margin of desk, with 4 different angles $\pm 30°$ and $\pm 60°$. We collect 10 volunteers' breathing data, each including 2 minutes tests as Fig. 4 (a) and (b) respectively, and label them finally.

We find some differences on smartphone with different manufacturers. A funny discovery is from iphone 4s. iphone 4s has much worse ability to collect sound recording like breathing, since the build-in microphone has a filter function

that can filter the low frequency noise before recording. This function is developed to improve the quality of voice conversations. We test some smartphones from different manufacturers such as One Plus, Huawei Honor, MI, MEIZU, OPPO, vivo, and finally find that 4 MI smartphones can collect more intact data. Consequently, we choose 4 MI smartphones with the same band, for removing the unexpecting hardware diversities in experiments.

### 5.3 Results

There are four metrics to measure the performance of an algorithm in Table. 2. TPR(True Positive Rate), TNR(True Negative Rate), WAR(Whole Accuracy Rate) and TTR. There are two classes of samples in our data set, breathing frames (positive samples) and non-breathing frames (negative samples). TPR is a recognition accuracy on positive samples, while TNR is a recognition accuracy on negative samples. And WAR is the recognition accuracy on the whole data set. After recognized from deep learning model, breathing frames are calculated into TTR. Here TTR is a measure of breathing rate, which is defined as $\frac{a}{b}$, where "a" is the breathing rate calculated by post-processing, and "b" is the ground-truth breathing rate. Table. 2 also lists the TPR and TNR of four distances respectively, for the number of positive samples is quadruple the number of negative ones. Actually, the recognition accuracy of negative samples is much higher than that of positive samples.

We can obtain five results from Table. 2 as follows. First, the deep learning models exhibit advantages on precise recognition in comparison to SVM and LR. The superiority increases with the decrease of SNR of the data. Second, the convolution exhibits a good performance according to the results of our models and baseline. The two-dimension convolution is better than one-dimension convolution, which demonstrates the ability of feature representation of convolution. And the frequency domain (two-dimension) provides more information than time domain (one-dimension). It is said that convolutional layer brings the most improvement on accuracy of recognition. Third, bi-directional recurrent layer is better than unidirectional one, according to the results of DeepFilter 2 and DeepFilter 3. It means that not only the history information, but also the future information can improve the accuracy of recognition. In practical, the improvement of bi-directional recurrent layer is limited, which is not much than that of convolution. Fourth, the results of baseline, DeepFilter 1, DeepFilter 2 and DeepFilter 3 demonstrate that the convolution are performed well in both time and frequency domains. And the performance on negative samples(TNR) is much better than that of positive samples in deep learning models. Finally, DeepFilter 3 obtains the best result in most cases, especially for lower-SNR data. And the results from DeepFilter 1 to DeepFilter 3 demonstrate that convolutional layer and recurrent layer are not conflict, but boost each other in our problem.

In Table. 2, there are significant differences between 40cm and 60cm with respect to the recognition accuracy. Since the SNR of breathing recordings decreases exponentially with the increasing of distance, TTR can directly indicate the performance of fine-grained breathing rate monitoring. As we see, the TTR of DeepFilter 3 is close to that of DeepFilter 2, and the TTR values of 6 algorithms are less than other three metric values. It implies that one breathing event is separated by some misclassified frames, in which each part is less than 5 frames. Since such breathing events are removed by post-processing, the TTR values of 6 algorithms are less than other three metric values.

Table 3 lists four related vital sign monitoring methods, including ApneaAPP[Nandakumar *et al.*, 2015], FMBS[Ren *et al.*, 2015], WiFi-RF[Liu *et al.*, 2015], DoppleSleep[Rahman *et al.*, 2015]. All the four vital sign monitoring methods belong to contactless sleep monitoring systems, which involve breathing frequency detection. The column "Modalities" lists the method of the system used. "RiP-L" and "RiP-U" are the lower bound and the upper bound of accuracy of the system in the corresponding papers, and "bmp" is the unit which means the difference between the rate detected by the system and the actual rate. "RiO-TTR" is the result that we reproduce the four methods with our data on the metric "TTR". Different from our work, WiFi-RF and DoppleSleep need extra device to assist breathing monitoring while our system only requires the off-the-shelf smartphone. ApneaAPP uses Frequency Modulated Continuous Wave (FMCW) to capture the breathing frequency. We reproduced the method but obtained poor results (as shown in first line of the last column of Table. 3, $<$0.2 means less than 0.2). The reason lie in that FMCW needs the device to transmit ultrasonic, and the higher of frequency of ultrasonic is used, the better results are found. Most smartphones only support ultrasonic less than 20kHz, which can not provide enough accuracy in practical (only 0.2 accuracy can be achieved in our reproduced scheme, which is much lower than the 0.98 listed in paper). FMBS is the only method using voice recording, which is most similar to our work. FMBS uses earphone to reinforce voice recordings of the users during the night, and adopts envelope detection to assist the breathing detection. But it is a coarse-grained breathing detector, which only achieves a low TTR value as 0.53 when it is running on our voice recordings. As so far, there are no any other works using smartphone on fine-grained breathing monitoring of voice recordings. WiFi-RF and DoppleSleep use WiFi radio frequency and extra device to capture breathing respectively, which may achieve acceptable accuracy but are not suitable for breathing monitoring based on smartphones.

We can find that the TTR of DeepFilter is 0.77 in Table 2, which is better than FMBS's[Ren *et al.*, 2015] 0.53. It means our deep learning algorithm can achieve comparable performance of several professional devices.

We also list the recognition accuracies of four distances in Fig. 4(a) and four angles in Fig. 4(b). As shown in Table 2, the accuracy is reducing with the raising of distance. An interesting phenomenon of accuracy is dramatically declined on 80cm, which maybe an explanation of critical point of SNR. We will conduct further study on it in our future work. And the results of accuracy ratio affected by different angles are listed in Table 4, which shows that a tiny influence on accuracy by different angles.

The precision of polysomnography requires over 99% which is the standard in sleep quality monitoring. In prac-

| | 20cm (0.8m) | | 40cm (0.95m) | | 60cm (1.1m) | | 80cm (1.25m) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TPR | TNR | TPR | TNR | TPR | TNR | TPR | TNR | TPR | TNR | WAR | TTR(bmp) |
| SVM | 0.897 | 0.923 | 0.787 | 0.920 | 0.402 | 0.843 | 0.316 | 0.820 | 0.601 | 0.876 | 0.807 | 0.53 |
| LR | 0.867 | 0.913 | 0.757 | 0.932 | 0.413 | 0.789 | 0.342 | 0.829 | 0.594 | 0.865 | 0.798 | 0.53 |
| Baseline | 0.994 | **0.988** | 0.948 | 0.916 | 0.509 | 0.925 | 0.461 | 0.947 | 0.715 | 0.944 | 0.886 | 0.64 |
| DeepFilter 1 | 0.996 | 0.970 | 0.980 | 0.986 | 0.667 | **0.932** | 0.639 | 0.921 | 0.820 | **0.952** | 0.919 | 0.71 |
| DeepFilter 2 | **0.998** | 0.965 | 0.981 | **0.988** | **0.740** | 0.910 | 0.703 | 0.900 | 0.855 | 0.943 | 0.921 | **0.77** |
| DeepFilter 3 | **0.998** | 0.976 | **0.998** | 0.976 | 0.733 | 0.900 | **0.710** | **0.921** | **0.859** | 0.943 | **0.922** | **0.77** |

Table 2: The results from 6 models: Support Vector Machine(SVM), Logistic Regression(LR), DRNN(baseline), Model 1, Model 2 and Model 3. All of them are trained and tested on two data set. The rightmost 4 columns are average scores over the 4 distances

| Related Methods | Modalities | RiP-L(bmp) | RiP-U(bmp) | RiO-TTR(bmp) |
|---|---|---|---|---|
| ApneaAPP[Nandakumar *et al.*, 2015] | Ultrasonic | 0.98< | **0.996**> | 0.2> |
| FMBS[Ren *et al.*, 2015] | Voice Recording | 0.5< | 0.95> | 0.53 |
| WiFi-RF[Liu *et al.*, 2015] | Radio Frequency | 0.6< | 0.8> | – |
| DoppleSleep[Rahman *et al.*, 2015] | Radar Module | 0.786< | 0.893> | – |

Table 3: The comparisons of several breathing rate monitoring techniques. "RiP-L" and "RiP-U" are the lower and upper bound of accuracy in their papers respectively. "RiO-TTR" is the result that we obtained by reproducing the methods with our data on metric "TTR".

| Angles | $+30°$ | $-30°$ | $+60°$ | $-60°$ |
|---|---|---|---|---|
| Accuracy Ratio | 95.01% | 94.50% | 91.67% | 92.81% |

Table 4: The results of four angles.

Figure 5: Sleep Test.

tical, DeepFilter can achieve 90% accuracy in 1 meter, but lower than 80% in 2 meters. The 2-meter monitoring distance is sufficient for most fine-grained breathing rate monitoring applications. In most cases, the apps can work well within 1 meter as monitoring distance.

### 5.4 Realistic Test

Fig. 5 shows the procedure of realistic sleep test, and the results are shown in Table 5. The volunteer lies on the bed, and the smartphone is placed at the side of the pillow. The smartphone records the breathing of the volunteer, and another camera records the procedure. The test lasts 7 hours, and finally we collect enough dirty data. It includes the snore, the voice of turning, and other unknown voices from outside. We choose three voice clips that include three poses shown in Fig. 5, and obtain the relatively clean data (the environment is usually quite during sleep, so that high-SNR data is easily to find). The three clips last 23, 25 and 22 minutes respectively. We labeled the data by hand, and run our model to validate the effectiveness. The results shown in Table 5 prove that our method achieves fine-grained breathing monitoring in realistic sleep. And we will conduct more realistic experiments on smartphones and other mobile devices in the near future.

| | TPR | TNR | WAR | TTR |
|---|---|---|---|---|
| flat | 80.00% | 92.10% | 90.10% | 0.77 |
| right | 94.60% | 98.80% | 97.20% | 0.84 |
| left | 78.70% | 87.90% | 86.20% | 0.69 |

Table 5: The results of sleep test.

## 6 Conclusion

In this paper, we try to apply deep learning as fine-grained breathing rate monitoring technique to smartphone for peoples daily sleep quality monitoring. We propose DeepFilter, a bi-directional RNN stacked with convolutional layers and speeded up by batch normalization, to perform this task. The desirable results of experiments and realistic sleep test prove that our model achieves professional-level accuracy, and deep learning based breathing monitoring apps on smartphones are promising. Our work also extends the use of deep learning to low-SNR data recognition, which is inspiring for more data processing applications.

In our future work, we will exploit more data to train DeepFilter. It implies that DeepFilter needs to suit more smartphones from different manufacturers. And more robust algorithms of post-processing should be developed. Then, we will also try more deep learning models to solve our problem. And we will deploy the approaches on smartphones and other mobile devices in the near future.

## References


[Albrecht *et al.*, 2014] Urs Vito Albrecht, Ute Von Jan, Joachim Kuebler, Christoph Zoeller, Martin Lacher, Oliver J Muensterer, Max Ettinger, Michael Klintschar, and Lars Hagemeier. Google glass for documentation of medical findings: Evaluation in forensic medicine. *Journal of Medical Internet Research*, 16(2):192–198, 2014.

[Amodei *et al.*, 2015] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro,

Jingdong Chen, Mike Chrzanowski, Adam Coates, and Greg Diamos. Deep speech 2: End-to-end speech recognition in english and mandarin. *Computer Science*, 2015.

[AppCrawir, 2014] AppCrawir. Respiratory rate. http://appcrawlr.com/ios/respiratory-rate, 2014.

[Arnold, 2013] Tyler Arnold. Google glass. *Alaska Business Monthly*, 56:1307–1321, 2013.

[Audacity, 2015] Audacity. Audacity. http://www.audacityteam.org/, 2015.

[Braun *et al.*, 2012] Phillip X. Braun, Claire F. Gmachl, and Raed /A/. Dweik. Bridging the collaborative gap: Realizing the clinical potential of breath analysis for disease diagnosis and monitoringctutorial. *IEEE Sensors Journal*, 12(11):3258–3270, 2012.

[Chen *et al.*, 2013] Zhenyu Chen, M. Lin, Fanglin Chen, and N. D. Lane. Unobtrusive sleep monitoring using smartphones. pages 145–152, 2013.

[Chung *et al.*, 2007] Gih Sung Chung, Byoung Hoon Choi, Ko Keun Kim, Yong Gyu Lim, Jin Wook Choi, Do Un Jeong, and Kwang Suk Park. Rem sleep classification with respiration rates. In *International Special Topic Conference on Information Technology Applications in Biomedicine*, pages 194–197, 2007.

[Cooijmans *et al.*, 2016] Tim Cooijmans, Nicolas Ballas, Laurent, and Aaron Courville. Recurrent batch normalization. 2016.

[Graves *et al.*, 2013] A. Graves, A.-R. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[Hannun *et al.*, 2014] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, and Adam Coates. Deep speech: Scaling up end-to-end speech recognition. *Computer Science*, 2014.

[Hao *et al.*, 2013] Tian Hao, Guoliang Xing, and Gang Zhou. isleep: unobtrusive sleep quality monitoring using smartphones. In *ACM Conference on Embedded Networked Sensor Systems*, pages 1–14, 2013.

[Hossain *et al.*, 2015] H M. Sajjad Hossain, Nirmalya Roy, and Abdullah Al Hafiz Khan. Sleep well: A sound sleep monitoring framework for community scaling. In *IEEE International Conference on Mobile Data Management*, pages 44–53, 2015.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Computer Science*, 2015.

[Kushida CA, 2005] Morgenthaler T Alessi CA Bailey D Coleman J Jr Kushida CA, Littner MR. Practice parameters for the indications for polysomnography and related procedures: an update for 2005. *Sleep*, 28(4):499–521, 2005.

[Liu *et al.*, 2015] Jian Liu, Yan Wang, Yingying Chen, Jie Yang, Xu Chen, and Jerry Cheng. Tracking vital signs during sleep leveraging off-the-shelf wifi. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '15, pages 267–276, New York, NY, USA, 2015. ACM.

[Mohamed *et al.*, 2012] Abdel Rahman Mohamed, George E. Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio Speech and Language Processing*, 20(1):14–22, 2012.

[Nandakumar *et al.*, 2015] Rajalakshmi Nandakumar, Shyamnath Gollakota, and Nathaniel Watson. Contactless sleep apnea detection on smartphones. In *International Conference on Mobile Systems, Applications, and Services*, pages 45–57, 2015.

[Organization, 2011] World Health Organization. mhealth: new horizons for health through mobile technologies. *Social Indicators Research*, volume 64(7):471–493(23), 2011.

[Parish, 2009] J. M. Parish. Sleep-related problems in common medical conditions. *Chest*, 135(2):563–72, 2009.

[Rahman *et al.*, 2015] Tauhidur Rahman, Alexander T. Adams, Ruth Vinisha Ravichandran, Mi Zhang, Shwetak N. Patel, Julie A. Kientz, and Tanzeem Choudhury. Dopplesleep: a contactless unobtrusive sleep sensing system using short-range doppler radar. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 39–50, 2015.

[Ren *et al.*, 2015] Y. Ren, C. Wang, J. Yang, and Y. Chen. Fine-grained sleep monitoring: Hearing your breathing with smartphones. In *INFOCOM 2015*, pages 1194–1202, April 2015.

[Sainath *et al.*, 2015] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *IEEE International Conference on Acoustics*, pages 4580–4584, 2015.

[Shaw *et al.*, 2008] Jonathan E. Shaw, Naresh M. Punjabi, John P. Wilding, K. George M. M. Alberti, and Paul Z. Zimmet. Sleep-disordered breathing and type 2 diabetes: a report from the international diabetes federation taskforce on epidemiology and prevention. *Diabetes Research and Clinical Practice*, 81(1):2–12, 2008.

[Tomlinson *et al.*, 2012] Mark Tomlinson, Mary Jane Rotheramborus, Leslie Swartz, and Alexander C. Tsai. Scaling up mhealth: Where is the evidence? *Plos Medicine*, 10(2):458–459, 2012.