12-6-2021

# Pranayama Breathing Detection with Deep Learning

Bikash Shrestha
*University of Missouri-St. Louis*, shresthabikash637@gmail.com

# Pranayama Breathing Detection with Deep Learning

by

Bikash Shrestha

A Thesis
Submitted to The Graduate School of the

University of Missouri-St. Louis
in partial fulfillment of the requirements for the degree

Master of Science

In

Computer Science

December 2021

## Abstract

Yoga, a complementary health approach, according to a 2017 National Health Interview Survey by the Center for Disease Control and Prevention (CDC), is a choice of around 14.3% adults in the US. Kapalbhati pranayama, a yoga practice of alternating fast exhales and longer passive inhales, is understood to improve our health. Incorrect and irregular practices, however, can cause injuries and adverse effects. To avoid these undesired effects, it is essential to maintain a pace fit for the practitioner. In the absence of any tools to observe a pace of practice, this work develops a deep learning method that listens to a person's breathing sound and outputs the pace/counts of kapalbhati breathings in real-time. Our deep learning model, based on our own accuracy definition, is 93% accurate. The model, when packaged as an application, can be used by practitioners with a plan/prescription to observe and adjust their pace and counts per session. In this paper, we elaborately discuss the data collection, cleaning, labeling, deep learning architecture and training, challenges in development, and quantitative evaluation of our results. We plan to release our code publicly along with an application that the public can use. We would like to caution that this work aims to provide an example tool for maximizing benefits and minimizing the adverse effects and injuries from kapalbhati practice, but not to promote yoga practice.

## Acknowledgements

I would like to thank Dr. Badri Adhikari for his mentorship, ideas, guidance, and valuable time throughout my research work. It would not have been possible without his help. I would also like to thank my friends and other people who took out their valuable time to provide kapalbhati recordings for this project. Additionally, I would also like to thank Keerthi Vakkalagadda and Gopi Chandrakesan for helping in data collection and data labeling. Last but not least, I am thankful to Dr. Sharlee Climer and Dr. Lav Gupta for being part of this project and for providing useful insights and suggestions.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Yoga exercises involving controlled rhythmic breathing, similar to all other physical exercises, if practiced incorrectly can lead to side effects and injuries. These yoga practices, colloquially known as ‘pranayama’, may cause side effects such as hernia and high blood pressure [15], if practiced incorrectly or at a non-prescribed pace. To avoid these undesired effects, practitioners would greatly benefit from software/mobile-applications that provide real-time feedback. No such applications currently exist, for any type of pranayama breathing exercises. *In this work, we investigate if a deep learning method can be developed to accurately predict the breathing pace of a practitioner using audio as the only input.* We focus on a specific type of pranayama known as ‘kapalbhati’.

In this work, we develop a method to predict the counts and pace of kapalbhati breathing using deep learning in real time. For our investigations we rely on the existing deep learning technologies and widely used spectrogram features extracted from audio. In the absence of any other method to predict the pace of kapalbhati breathing, this work is the first contribution as a novel application of deep learning. The dataset we created and our investigations/findings relating to the effect of noise on accuracy are our additional contributions.

# Chapter 2

# Background and related work

In this section, we provide brief surveys of research related to pranayamas, the kapalbhati pranayama, and deep learning method developed for audio inputs.

Several studies have found pranayamas to be effective for improving health. One study involving 60 male and female patients aged 20-60 years has shown that slow breathing exercise has a significant effect on blood pressure [27]. Slow-paced 'bhastrika' pranayama are also demonstrated to decrease the systolic and diastolic blood pressure significantly [32]. Similarly, the 'buteyko' breathing technique improves the symptoms and reduces bronchodilator use in patients with asthma, according to a study with 90 asthma patients [7]. 'Sukha' pranayama is also understood to have significant cardiovascular effects, a study with 23 hypertensive patients shows [3]. In general, these findings suggest that, pranayama, if practiced as prescribed, can have many benefits.

Kapalbhati pranayama is a rythmic breathing practice where each breathing cycle involves a relaxed passive inhale followed by a fast exhale. It's practice is believed to clear the lungs, the nasal passage, and the mind of the practitioner. It's practice is also understood to result in many health benefits such as improving blood circulation in the head [29], reducing blood sugar level [33] reducing obesity [18], and reducing pulse rate and stress [39]. But to receive these benefits, it should be practiced correctly and with proper guidance. It is important to follow a prescribed pace, i.e., repeat the breathing cycle only a certain number of times in a minute. Generally, 60-120 breathing cycles per minute are prescribed for young and healthy individuals [36].

Next, we provide a brief survey of several related deep learning methods that take audio as the primary input feature. McClure et al [26] recently developed a breathing analysis system using the data from wearable sensors which are used by the one dimensional convolutional neural network (CNN) [22] to detect different types of breathing patterns. For normal breathing, they have achieved a mean F1-score of 92% and for central sleep apnea, their method has a mean F1-score of 87%. Similarly, for other breathing patterns like coughing, yawning, sighing, and obstructive sleep apnea, they have achieved 72%, 63%, 57%, and 51% respectively. Also, a group from Radboud University Nijmegen [28] developed deep learning-based methods to estimate respiratory rate and breathing capacity with a sensitivity of 91.2%. After the COVID-19 pandemic hit the world, many researchers also investigated the use of deep learning for COVID detection from X-rays and breathing patterns. Sait et al [35] developed a deep learning based multimodal system using a CNN-based architecture, Inception-v3, along with multi-layered perceptron (MLP) to detect COVID-19 using breathing sounds and X-rays. Their system, CovScanNet, uses the spectrogram of the breathing sound of the user to detect abnormalities in the breathing signal with a preliminary accuracy of 80%. Another study done by Alkhodari et al [1] has also shown the use of breathing recordings in detecting the COVID using the combination of CNN and bidirectional LSTM. They have used mel frequency cepstral coefficients (MFCC) along with other hand-crafted features like sample entropy, spectral entropy, kurtosis and skewness, fractal dimension, and zero-crossing rate for their deep learning model. They have developed a system that uses smartphone-based breathing sound to discriminate between COVID infected vs healthy subjects with a maximum sensitivity of 94.21% and specificity of 94.96%. Their system was able to identify 18 asymptomatic subjects among 120 COVID patients with 100% accuracy. Similarly, Laguarta et al. [21] has developed a deep learning-based method to detect COVID using the cellphone recorded coughing sound of the user. They have used MFCC features obtained from the coughing sound as an input to the CNN based architecture with one Poisson biomarker layer and 3 pre-trained ResNet50 networks in parallel which has 98.5% accuracy. Their method was also able to detect COVID positive asymptomatic patients with 100% accuracy similar to Alkhodari's work.

Most of the deep learning applications in audio may be grouped into two categories: those related to speech (speech recognition) and those related to other sounds (sound recognition). Speech recognition is one of the very popular applications where a tremendous amount of research has been done on the use of deep learning for speech processing to convert the speech into sequence of words/texts. This field has grown significantly over the past few years and this has been used in many popular speech-based applications like Google Assistant developed by Google, Siri developed by Apple, Alexa developed by Amazon, Cortana developed by Microsoft and so on. This has become one of the popular ways to interact with machines in the current world. A group of scientists from Google has developed a state-of-the-art speech recognition system with sequence-to-sequence models [6]. They introduced a multi-head attention based architecture which has better performance than a single-head attention. They were able to improve the word error rate (WER) of 12,500 hours of voice search task from 9.2% to 5.6% which is better than the best conventional system that has 6.7% WER. Hori et al [16] developed a novel word-based RNN language model using LibriSpeech and WSJ where their model was able to achieve the best WER of 5.1% on the WSJ Eval92 test set for their end-to-end speech recognition system. Similarly, in the study done by Rao et al [34], they have shown that the use of recurrent neural network transducers on their end-to-end speech recognition system was able to achieve WER of 8.5% on voice search and 5.2% on voice dedication task. Residual CNN-based [40]; Similar study done by Wang et al. where they have used residual convolutional neural networks with connectionist temporal classification loss function. Their system was able to achieve 14.91% WER on the WSJ dev93 test dataset. More recently, attention-based models are found to be superior to the traditional type of CNN and RNN architecture. Chan et al [5] developed a system called Listen, Attend and Spell (LAS) using attention-based RNN which was able to achieve a WER of 14% on google voice search without an external language model and 10.3% with a language model. Most recently, transformer based architectures have been very popular in the deep learning community. They have shown outstanding performance in Natural language processing and now they are also performing very well in other domains. Here is a work done by Gulati et al [14] where they have used transformer based architecture in their automatic speech recognition (ASR) system. Their method was able to achieve a WER of 4.3% without language model and 3.9% with an external language model on the LibriSpeech dataset. A comprehensive study on various deep learning techniques that are used for speech emotion recognition has been done in this [19] work. The study has shown that deep learning has significantly improved the performance as compared to the traditional algorithms. A study done by Gu et al [13] has shown the use of a deep multimodal framework using CNN and LSTM to predict human emotions based on the spoken sentence. Their method was able to achieve a weighted accuracy of 60.4% for five emotions on the IEMOCAP dataset. Similarly, Yu et al [43] have used an attention-based LSTM architecture in their speech emotion recognition system where their method was able to achieve a weighted accuracy of 68% on the same IEMOCAP dataset.

Several deep learning methods have also been developed for non-speech related audio tasks. A group studied lung sound classification using deep learning. In this study [4] the group has developed a semi-supervised deep learning algorithm to automatically classify lung sounds into two classes: wheeze and crackle. 11,627 audio files were collected from 284 patients with pulmonary disease from different auscultation locations. Their algorithm has achieved the ROC curve with area under the curve (AUC) of 0.86 for wheeze class and 0.74 for crackle class. Another group classified birds' sounds using deep learning. In this study [41] the group has developed a deep learning method to classify 14 bird species using their sound. Their proposed deep learning method has achieved the F1-score of 94.36%. Along with deep learning when other hand crafted features like acoustic features and visual features are added by using late fusion, the final method has achieved the F1-score of 95.95%. Another group classified various environmental sounds. Li et al. [23] has done a comparison of various deep learning methods such as DNN, RNN, CNN, and an Ensemble of all models (Late fusion method) for environmental sound detection. The work has also shown the comparison of deep learning methods with the traditional methods. Overall the late fusion method has significantly improved performance with 88.1% mean cross-validation accuracy and 88.2% test accuracy than other individual methods which ranges from 72-83% accuracy. A study done by Piczak [31] has shown the use of CNN for classifying environmental sounds for three different datasets which contain various numbers of sound classes ranging from 10 to 50 classes. Deep learning methods have also been developed for music genre classification and for recommending music. A study done by Kim et al [20] has shown the use of multimodal deep learning where CNN and RNN have been used to classify the genre of the music. Another study done by Elbir et al [9] has also shown the use of CNN with acoustic features to develop a music genre classification and music recommendation system. Their method MusicRecNet was able to achieve 81.8% of mean accuracy in the GTZAN dataset. A multi label music genre classification system [30] was developed by Oramas et al where they have developed a system to classify music into 250 genre classes. Their CNN based method has an accuracy of 88.8% as compared to 79.2% accuracy with traditional architecture. Similarly, Jiang et al [17] has developed a music recommendation system using recurrent neural network (RNN) where they make comparisons of different songs by similarity to do the recommendation for the next song. Their audio-based approach was able to achieve 76.5% accuracy whereas the lyrics-based approach was able to achieve 86.4% accuracy. Also, Fessahaye et al. [10] has proposed a new approach to improve music recommendation systems. In their work they have developed a deep learning based algorithm called Tunes Recommendation System (T-RECSYS) that uses a hybrid of content-based and collaborative filtering which has a precision of 88% on a Spotify Recsys Challenge dataset. Similarly, Dong et al [10], has developed a music recommendation system using fusion deep learning models which uses CNN based models that was able to achieve 90.2% accuracy on a publicly available dataset.

Overall, we see that deep learning in audio has developed significantly in recent years. Many state-of-the-art deep learning techniques have been developed for audio signal processing such as in speech recognition, music classification, and environmental sound detection. However, no methods have been developed to detect pranayama sounds.

The rest of the thesis is organized as follows. In our methods section, we describe how we collected, cleaned, and labelled kapalbhati recordings, the architecture of our deep learning model, and the definition of our accuracy metric. We also discuss how spectrogram features obtained from audio are helpful in obtaining highly accurate predictions. In our results section, we discuss how we optimized various hyperparameters used for model training, comparison with human benchmarks, and how noise affects the model's accuracy. We also describe the design and implementation of the web application we developed for packaging our model.

# Chapter 3

# Methods

## 3.1 Collecting, cleaning, and labeling data

To train a machine learning model that recognizes kapalbhati breathing strokes, a collection of kapalbhati sound recordings is required. Such a dataset, to the best of our knowledge, is currently absent. In order to build a dataset, we first looked for recordings at Youtube (youtube.com) and FreeSound (freesound.org), and found about ten relevant recordings. Although finding readily available recordings saved us some time, to build a diverse and comprehensive dataset, these were insufficient. To add more samples, we recorded additional 30-50 seconds kapalbhati recordings from 23 people including our friends and relatives. These 33 audio clips were our positive examples, i.e., recordings that have kapalbhati breathing strokes. Analyzing these recordings we found that most of the breathing strokes (quick exhalations) are 2/10 seconds long and that the gap between two consecutive strokes is 8/10 seconds (see **Figure 1** and **Figure 2** in Appendix for a distribution). In addition to using these positive examples for training our model, we hypothesized that adding random background noise with random intensities would make our models robustly accurate. For the background noise, we used a random subset of 2000 audio clips from the Google's AudioSet [11]. AudioSet is a large-scale publicly available dataset containing thousands of audio events. The 33 audio clips (positive examples) and 2000 event audio clips in AudioSet were our development (training/validation) dataset. At a sampling rate of 22050 Hz, all these audio recordings, some of which were stereo or multi-channel, were converted to mono-channel signals.

## 3.2 Audio amplitudes to spectrograms and labels to continuous real numbers

An audio recording can be converted to an amplitude waveform, a vector of floating points. For example a one-second audio signal, at a sampling rate of 22050 Hz, translates to a vector of 22050 numbers. Such amplitude vectors, which can be seen as a time series data, can be directly fed to a deep learning model. However, from these amplitude data, deep learning models can learn little. Instead, a more effective method is to transform these one-dimensional amplitude arrays to two-dimensional time-frequency domain representation called spectrograms (see **Figure 3.1**). The use of spectrograms have proven more effective for many other problems including emotion recognition from speech [2, 24], sound classification [8, 31], and music classification [20, 9, 30]. In a spectrogram, which is a 2D array, one dimension (say, length) represents time and the other (say, height) represents frequency. Short-time fourier transform (STFT) is typically used to obtain spectrograms from an audio. A normal fourier transform converts a signal to its component frequencies only to keep the frequency-related information, i.e., at the cost of losing time-related information. But, in STFT, a signal is splitted into windows of time and a fourier transform is run for each window to preserve both information: time and frequency. When transforming an audio signal of a certain duration to a spectrogram, the length and height of the spectrogram are dictated by two parameters of the STFT algorithm: frame length and frame step. Frame length defines the size of the chunk in which the signal is divided into. Increasing frame length increases the frequency resolution because there is more data to derive frequency information. Frame step, on the other hand, which is also known as hop size, defines the overlap between two adjacent chunks. For an audio of certain duration translated to $S$ amplitude samples, and STFT parameters frame length $Fl$ and frame step $Fs$, the dimensions of the resulting spectrogram may be calculated using:

$(number\ of\ frequency\ bins,\ number\ of\ frames) = (\frac{Fl}{2} + 1, \frac{S-Fl}{Fs} + 1)$.

For example, if an amplitude waveform of a two second audio has 44100 samples (sampled at 22050 Hz), the selected frame length is 1024, and the selected frame step is 512, then the number of frequency bins will be 513 and the number of frames will be 85. Therefore, the resulting shape of the spectrogram will be (85, 513).

While a 2D spectrogram array (frequency x frames) serves as an input $X$ for training a deep learning model, we also need a vector of length 'frames' as an output label $Y$. Generating labels for all recordings involved listening to all of the audio recordings and manually marking the start and end positions of each kapalbhati breathing stroke in the audio. This process of marking start/end of each kapalbhati breathing was much easier than we had anticipated (see Appendix for a detailed description of how Audacity was used). For an audio, if we zero-initialized the output label vector $Y$, the start/end markers can be used to create bands of positive values in $Y$. While having 0s and 1s in $Y$ would enable us to frame the problem as a binary classification problem, we suspected that it would be challenging for a deep learning model to predict a '1' at the beginning and end regions of a single kapalbhati breathing stroke. Hence, we transformed a band of 1s to a bell-shaped distribution of continuous values with a peak (with value 1) towards the middle of the band. Our $Y$ would then be a real-valued vector with all numbers between 0 and 1: the values gradually peak to one and drop back down for each kapalbhati breathing stroke in the audio. This binary to continuous-value transformation enabled us to frame the "$X$ to $Y$ prediction" as a regression problem (see Figure 12). Hence, a deep learning that receives the spectrogram $X$ of length $Fl$ would have to predict the values in vector $Y$ of length $Fl$ where the values (0 to 1) would capture the rise and fall of each kapalbhati breathing stroke.

## 3.3 Deep learning architecture and training

Our system uses a convolutional neural network (CNN) based architecture. Since a spectrogram is an image like data, we opted for two-dimensional (2D) CNN which are commonly used for problems in computer vision. Our network architecture is inspired by the network used by google Deepmind's method Alphafold, [12]. The main reason behind using such a model is to preserve the time related data while training. We do not want to shrink the network because the predicted signal from our model should match the length of the spectrogram. The **Figure 3.2** shows the architecture of the CNN used in our system. As we can see from the figure, the length of the input is constant in every layer of the network. The main advantage of doing this is, it makes the model dynamic. The model can accept audio signals of any length because of this architecture. The model accepts an input of size $(L \times B \times C)$ where $L$ and $B$ represent the shape of the input spectrogram, and C

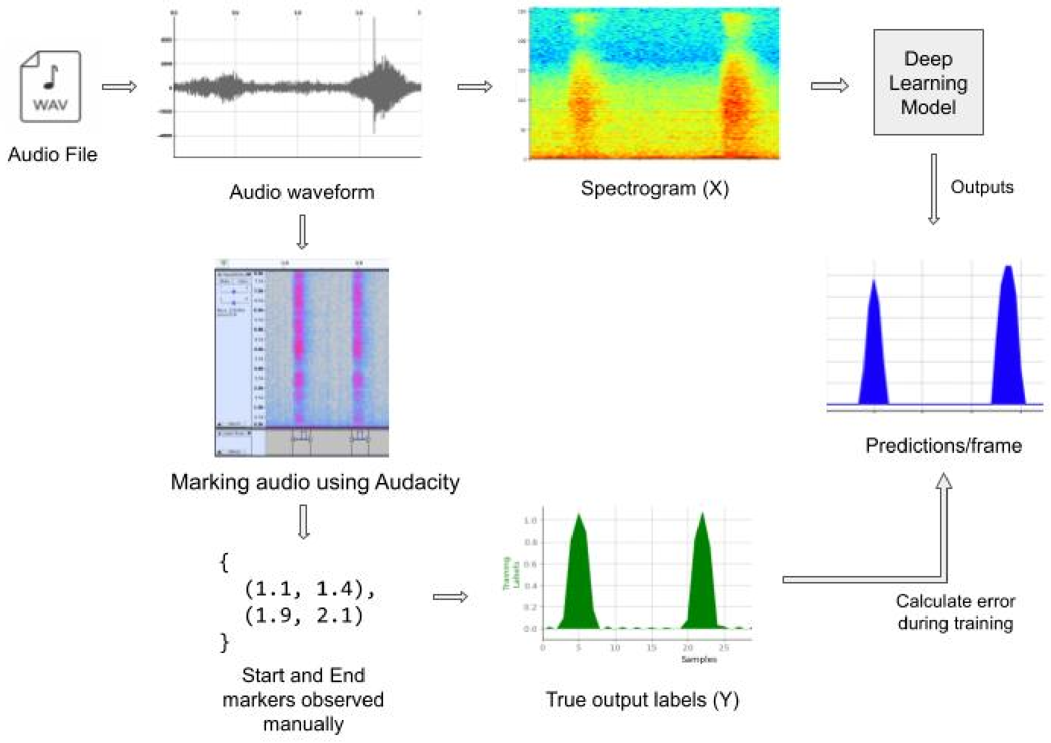


**Figure 3.1:** Overall setup for model training and data flow. Spectrograms obtained from an audio waveform and a vector of continuous values obtained using the start/end markers of each kapalbhati stroke serve as the input (X) and output label (Y) for training a deep learning model.

is the channel size which is always 1 in our case. For the model where B changes, we reduce the number of convolutional layers from the network to ensure that the last convolutional layer will have an input size of $(L \times L \times 1)$. Since the size of $B$ depends on the shape of the spectrogram, and the shape of the spectrogram depends on the frame length and frame size (more details in **Section 2.2**), we have to change the layers for the different networks accordingly.

Our networks mainly consist of a 2D convolutional layer with $3 \times 3$ filter size and a rectified linear unit (ReLU) as an activation function, a Batch Normalization layer, a Max Pooling layer, and a Dropout layer. These layers are combined together in an alternate fashion which can be seen in the **Figure 3.2**. At the end of the network a flattening layer is added to ensure that the output of the last CNN layer is a single dimensional vector which is the predicted signal from the model. The total parameters of the different networks that are implemented in this work ranges from 290,000 to 297,000. Other hyperparameters used for development were–optimizer: adam, epochs: 512, batch size: 4, loss function: mean squared error (MSE).

While training the model, on each batch 4 training samples were selected where 2 of the samples were randomly selected from positive samples and the remaining 2 samples were selected from the negative samples. This makes the model learn both positive and negative features of the data at the same time to prevent it from being biased towards one of them. In addition to that, data augmentation was also applied to the training samples by adding real natural background noise to the positive samples. The background noise was also randomly selected and added to the positive sample with a random signal-to-noise ratio (SNR). The signal-to-noise ratio value was between 5 to 40, where lower SNR means more prominent background noise and higher SNR means more prominent positive signal (see **Appendix** for a detailed process to compute SNR).

Most of our training experiments were a three-fold cross validation. In other words, all audio samples were divided randomly into three equal subsets–two for training and one for validation. With these three folds (subsets of the audio), three training experiments were run: training using first and second subsets and validation using the third, training using the first and third, and training using second and third. When evaluating accuracy, these three folds are assessed separately or when needed, averaged.

In a Ubuntu server (with hardware settings: dual 4215R 3.2GHz CPUs, 128 GB of RAM, and a NVIDIA Quadro RTX 6000 GPU with 24 GB GPU-memory, and SSD hard drives) one epoch of training a the model that accepts spectrograms of sizes (85, 513) took around 49 seconds to complete. When the spectrograms were decreased to (43, 513), it only took eight seconds. Our entire dataset, including the 2000 audios from the AudioSet occupy 215 MB of disk space. Python programming, along with Tensorflow, Keras, NumPy, matplotlib, librosa, scipy, and pydub libraries, were used for our experiments.

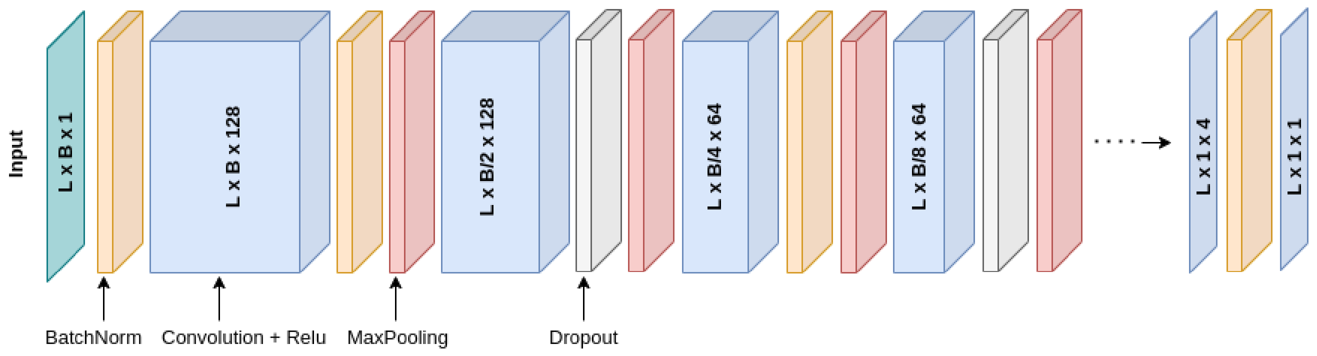


**Figure 3.2:** Our network is a CNN based architecture with alternating combinations of Convolutional+Relu, BatchNormalization/Dropout, and MaxPooling layers. The input has a size of (L x B x 1) where (L x B) is the shape of the spectrogram.

## 3.4 Defining accuracy for effective model evaluation

For the evaluation of our model, we developed a separate pipeline to calculate the accuracy of the model. The accuracy calculation includes multiple steps. Each audio file in the validation dataset was split into 3 seconds clips. A new data generator was developed to clip the audio in a contiguous fashion and process it to generate features for the model to output the prediction. The clipping is done in such a way that every clip will be 3 seconds long but the last clip will depend on the total length of the audio. For example, if the whole audio is 11 seconds long, then there will be three 3 second clips but the last clip will be 2 seconds long. Each clip was converted into a spectrogram and the respective y labels of that part of the audio were also calculated. Along with the output signal (y labels), the discrete labels (timestamps) of each breathing cycle in the given three-second audio clips were also retrieved. These discrete labels play a vital role in the accuracy calculation. The data generator returns three values: a spectrogram, an output continuous y labels, and discrete timestamps labels. The spectrogram is fed into the model which outputs the predicted signal. The predicted signal is then provided to a function called $find\_peaks$ from the Scipy library that calculates the positions of the peaks in the signal. The function outputs the list of indices that represent the position of each peak in the signal. We find the accuracy of the prediction by comparing these positions with the original timestamps. But before comparing it, we had to convert each timestamp into their respective sample numbers because the timestamps are in seconds and the positions are the sample indices of the predicted signal. The total number of samples in the output signal is equal to the length of the spectrogram (see **Section 2.2**). The true output label has a start sample number and end sample number for each breathing cycle. A hashmap was created using the new true output labels where the range between start sample number and end sample number represents positive and the remaining range is considered as negative. Now, each predicted peak position was compared to this hashmap in such a way where if the predicted peak position is in between the positive range, then it is counted as true prediction (i.e True Positive) else if there is no predicted peak position in that range, it is counted as false prediction (i.e False Negative). Similarly, for the range in the true output labels where there is no breathing (i.e negative), if there is no predicted peak in that range, then it is considered as true prediction. (i.e True Negative) else if there is a predicted peak, it is considered as false prediction (i.e False Positive). All of these processes were repeated for each 3 seconds clip and merged together to create a single confusion matrix for the given audio. Using this confusion matrix of each audio, we calculated the accuracy, and F1-Score for the validation datasets with all positive audio signals.

## 3.5 Implementation - Application development

To test our deep learning model, we developed a web application that records the breathing audio of the user from the device's microphone in real-time and shows the count of the breathing cycle on the screen. In this app, the user clicks the start button before practicing kapalbhati. After clicking the button, the app immediately starts to record the sound of the surroundings. The user can only see the count on the screen. But internally, the app is designed in a way where it continuously records the audio but in every two seconds, it sends the recorded 2 seconds audio to our pipeline where the audio is converted into a spectrogram and fed to the model for predicting the output signal. And using that predicted output signal, we calculate the number of breathing cycles in the recorded audio which is then displayed to the user on the screen.

For developing this app, we have used Flask micro-framework which is a Python framework. Flask uses the Werkzeug WSGI toolkit and the Jinja2 web templating engine for Python which helps to render dynamic web pages. The routing, debugging, and web server from Werkzeug, together with the templating from Jinja, make Flask a lightweight and easy-to-use full-stack web application framework. This enabled us to quickly develop the web framework to test our deep learning model with real-world use cases. A python server was created to host the application. There are multiple API endpoints that take the HTTP request from the user as well as send an HTTP response to the user in the form of HTML contents which is shown on the browser.

# Chapter 4

# Results

## 4.1 Generating the optimal spectrogram for high accuracy

As we discussed earlier (see **Methods**), an audio input translates to an spectrogram defined by two parameters: frame length and frame step. Frame length determines the spectrogram's number of frequency bins and frame step decides the number of frames. While the common practice in applications of deep learning for audio-related tasks is to use frame length as 256, 1024, 2048 and frame step as 128, 512, [25, 31, 42] (In general practice frame step size is selected to be 50% of the frame length, [25, 31, 38, 37]), during the initial prototype development with these parameter values, high accuracy was not obtained. This low performance hinted to us that this combination of frame length and step size may not be ideal for our problem. Consequently, to determine the optimal frame length and frame step that delivers highest possible accuracy, we tested various frame length and step combinations. Specifically, we tested all pairwise combinations of frame length in the set 64, 128, 256, 512, 1024, 2048 and frame step in the set 256, 512, 1024, 2048, i.e, 24 ($6 \times 4$) combinations to test. Each test involved a three-fold cross validation (CV) training experiment. A three-fold CV experiment results in three accuracy values: one for each fold's validation set. The average of these three accuracies obtained for each combination were used for ranking the 24 combinations. In **Figure 4.1** we summarize the average accuracies and the accuracies for all the individual folds. We find that the top four combinations of frame length and step combinations are (512, 1024), and (2048, 1024), (128, 512), and (512, 512), and their corresponding average accuracies are 93.4%, 92.2%, 92.2%, and 91.8%, respectively. We used these four combinations of frame step and length for experiments discussed in the subsequent sections. Of the top four best combinations of frame length and step, three contradict the intuition that overlapping frame steps (i.e, having frame length greater than frame step) is more informative. Use of the F1-score metric instead of accuracy, showed similar results (see **Figure 5** in Appendix).

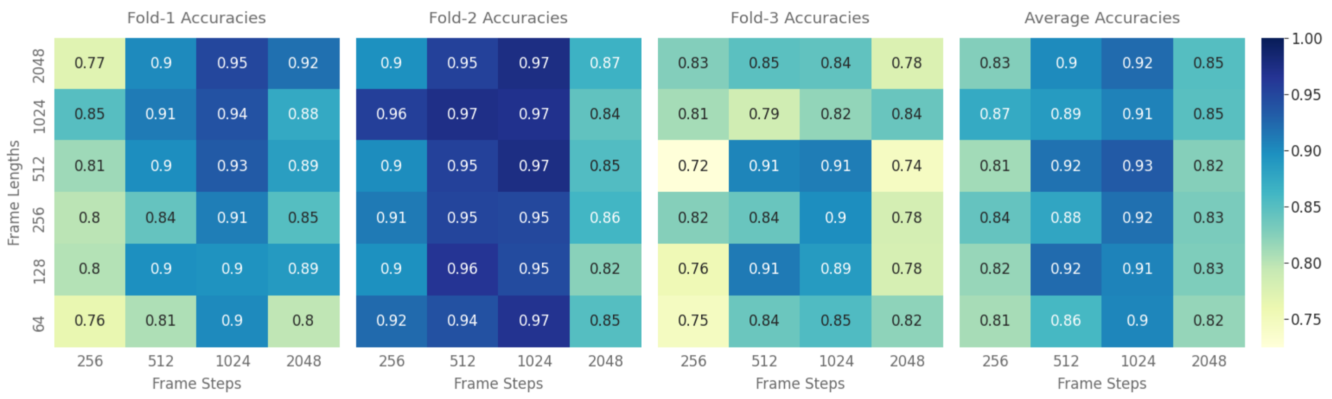
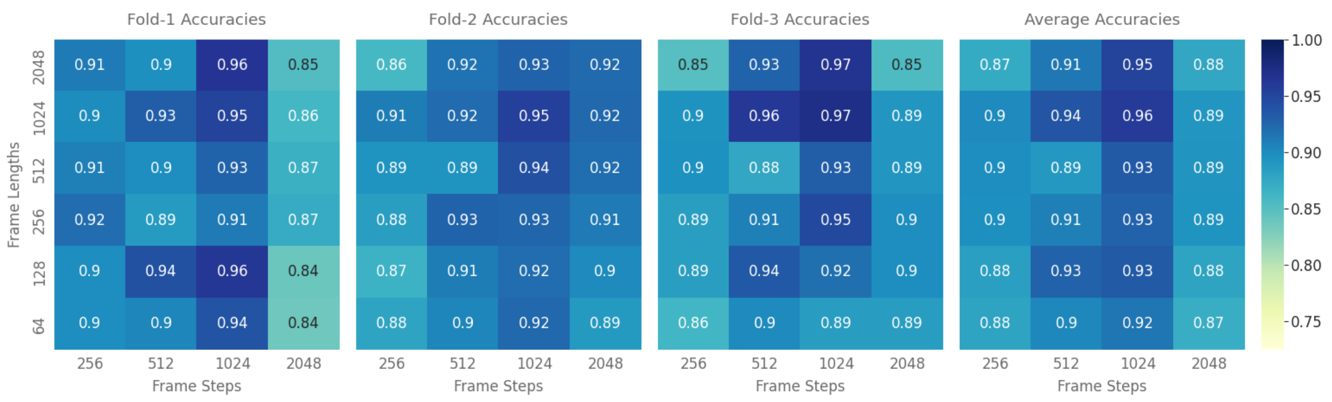


**Figure 4.1:** Effect of frame length and step on the accuracy across the three folds (first three columns) and the average of the three folds (last column) using accuracy for all the 24 combinations of cross validation experiments. Each cell in the first three heat maps represents the accuracy for the fold and the cells in the last heatmap show averages. The average accuracy of fold one, two, and three are 86.7%, 92.1%, and 82.4%, respectively, suggesting fold three is the most difficult one followed by fold one.

The distribution of data in the three folds are not equally distributed with hard and easy combinations of kapalbhati recordings in the experiments described previously. Therefore, we ran another set of experiments for all the 24 combinations by reshuffling the data. **Figure 4.2** shows the results for our new experiments with reshuffled data with equally distributed data in each fold. There is a slight increment in the accuracy but in overall the performance of the different combinations are similar to the previous experiments. The new top four combinations from these experiments are (1024, 1024), (1024, 2048), (512, 1024), and (1024, 512) with 96%, 95.3%, 93.8%, and 93.4% average accuracies.



| Frame Lengths | Fold-1 256 | Fold-1 512 | Fold-1 1024 | Fold-1 2048 | Fold-2 256 | Fold-2 512 | Fold-2 1024 | Fold-2 2048 | Fold-3 256 | Fold-3 512 | Fold-3 1024 | Fold-3 2048 | Avg 256 | Avg 512 | Avg 1024 | Avg 2048 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2048 | 0.91 | 0.9 | 0.96 | 0.85 | 0.86 | 0.92 | 0.93 | 0.92 | 0.85 | 0.93 | 0.97 | 0.85 | 0.87 | 0.91 | 0.95 | 0.88 |
| 1024 | 0.9 | 0.93 | 0.95 | 0.86 | 0.91 | 0.92 | 0.95 | 0.92 | 0.9 | 0.96 | 0.97 | 0.89 | 0.9 | 0.94 | 0.96 | 0.89 |
| 512 | 0.91 | 0.9 | 0.93 | 0.87 | 0.89 | 0.89 | 0.94 | 0.92 | 0.9 | 0.88 | 0.93 | 0.89 | 0.9 | 0.89 | 0.93 | 0.89 |
| 256 | 0.92 | 0.89 | 0.91 | 0.87 | 0.88 | 0.93 | 0.93 | 0.91 | 0.89 | 0.91 | 0.95 | 0.9 | 0.9 | 0.91 | 0.93 | 0.89 |
| 128 | 0.9 | 0.94 | 0.96 | 0.84 | 0.87 | 0.91 | 0.92 | 0.9 | 0.89 | 0.94 | 0.92 | 0.9 | 0.88 | 0.93 | 0.93 | 0.88 |
| 64 | 0.9 | 0.9 | 0.94 | 0.84 | 0.88 | 0.9 | 0.92 | 0.89 | 0.86 | 0.9 | 0.89 | 0.89 | 0.88 | 0.9 | 0.92 | 0.87 |

**Figure 4.2:** Effect of frame length and step on the accuracy across the three folds (first three columns) and the average of the three folds (last column) using accuracy for all the 24 combinations of cross validation experiments on the reshuffled dataset. The average accuracy of fold one, two, and three are 90.4%, 91%, and 90.5%, respectively. These additional experiments were performed to distribute the dataset equally in each fold with difficult and easy samples.

## 4.2 Comparing the model's accuracy with a human listener

It is a common practice in deep learning to ask how accurate a model is compared to a human (i.e., human baseline). For comparing the accuracy of our model with a human, we needed kapalbhati audios with a range of difficulty. We started with five random audios, each around 30 seconds long. To create audios with high difficultly, to some of these audio, we added random intensity of short noise clips at random locations. Also, it is easy for a human listener to observe the pattern and make correct guesses even at the presence of noise, so this noise addition helps to create a better evaluation dataset. These five audios were sent to a human volunteer to listen and provide us the total number of kapalbhati breathing strokes they heard in each audio. These audios were also run through the deep learning models to make predictions.

On these five audios, we find that the deep learning models perform similar to a human listener. Our results summarized in **Table 4.1** show that in most cases, the number of kapalbhati breathing strokes detected by the models are similar to the counts detected by the human. Also, we find that our first model (out of the three folds) performs generally better than the other two. These results are encouraging, given that our models are trained only using a few dozen 30-second kapalbhati recording.

**Table 4.1:** Comparison of the model's accuracy with a human listener. The number of kapalbhati breathing strokes in the audio (actual truth), reported by a human listener, and predicted by the three models ($\text{Model}_{\text{fold1}}$, $\text{Model}_{\text{fold2}}$, and $\text{Model}_{\text{fold3}}$) are reported. First fold predictions outperform the human baseline, second fold predictions are comparable whereas third fold predictions do not reach human baseline.

| **Audio** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| Actual number of kaplabhati strokes | 37 | 37 | 37 | 37 | 37 |
| Human listener (baseline) | 37 | 0 | 20 | 29 | 31 |
| $\text{Model}_{\text{fold1}}$ | 37 | 1 | 22 | 30 | 34 |
| $\text{Model}_{\text{fold2}}$ | 37 | 2 | 18 | 24 | 29 |
| $\text{Model}_{\text{fold3}}$ | 37 | 0 | 18 | 16 | 20 |

## 4.3 Effect of training window on accuracy

The next parameter to tune, for training the deep learning models, was the training window, i.e., how many seconds of audio to look at a time? While a training window of two seconds, chosen empirically, resulted in a more than 90% accuracy, we tested two additional window sizes: one and three seconds. For each of these three window sizes (one/two/three seconds), we repeated our three-fold CV experiments with each of the top four combinations of frame length and frame step (discussed in the previous section), totaling in 12 ($3 \times 4$) CV experiments. Compared to the training using two second clips, the training using one second clips was much faster and the one using three second clips was much slower. Results summarized in **Figure 4.3** show that regardless of speed, both one- and three-second clipping are less accurate.

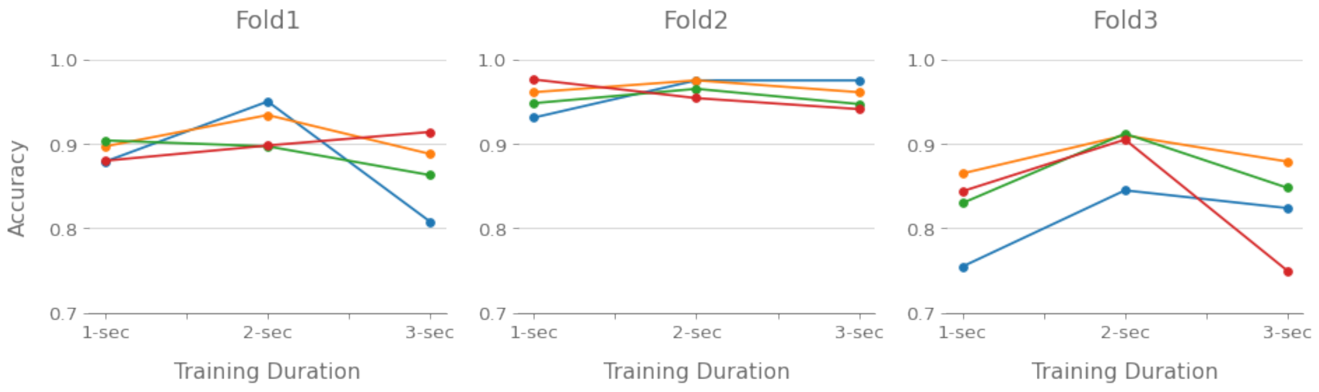


**Figure 4.3:** Effect of training window (one, two, and three-second clips) on accuracy across the three folds. The training window of two-seconds performs better than the other two. The four points and trendlines at each training window correspond to the four combinations of frame length and frame step.

## 4.4 Effect of everyday background noise on models' accuracy

Next, we tested our hypothesis of adding a random intensity of background noise to the labelled kapalbhati breathing sounds, for improving accuracy. For the test, we performed two cross-validation training experiments: one with background noise added (as in our rest of the experiments), and one without any background noise. For the training without the background noise, we simply skipped using the AudioSet dataset. After the training, we tested the two sets of models (trained with and without background noise), by passing through them the audio in the validation set with noise injected of various intensities.

The two plots in **Figure 4.4**, show one expected result: the accuracy drops as more the noise level in the test data increases, regardless of whether the model was trained with or without background noise. The comparison of the two plots also demonstrate another expected: in general, the models trained with background noise are more accurate. One finding that surprises us is that the models trained with background noise are more accurate than than their counterparts, even for audio inputs without any noise. As observed from the trend lines in both plots, we also find that a reasonable level of background noise does not hurt the accuracy significantly.

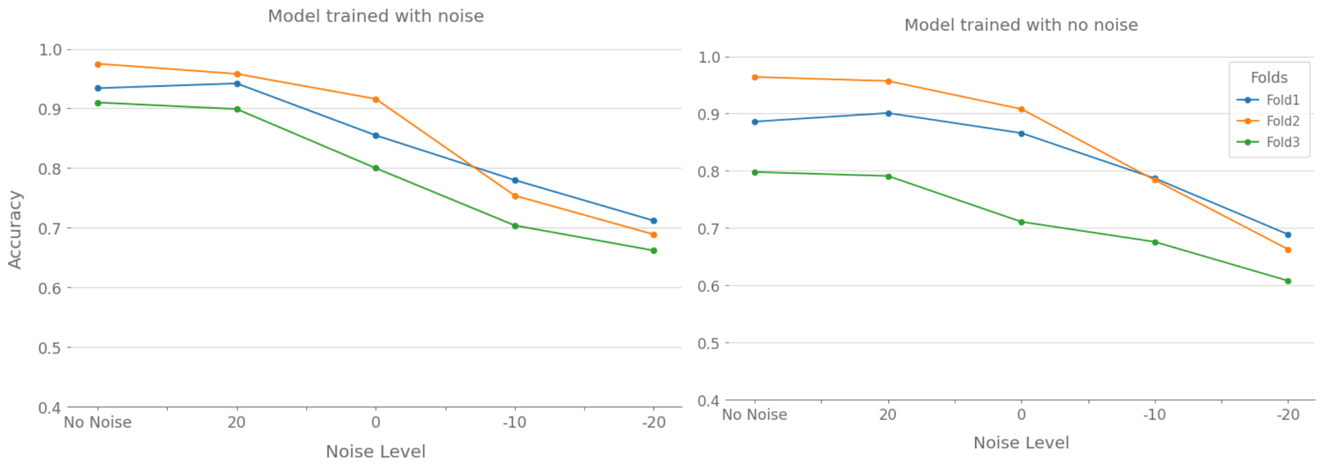


**Figure 4.4:** Effect of everyday background noise on accuracy for two sets of 3-fold cross-validated models: models trained with background noise (left plot) and the ones trained without background noise (right plot). In both plots, we observe that the accuracy drops as the background noise in the test dataset increases.

# Chapter 5

# Case studies

Our final deep learning model's strengths and limitations can be best illustrated with two case studies: one where the accuracy is high and the other where it is low. As the first case study, we discuss the accuracy of our model on a standard 32 second audio recording of one of our own kapalbhati recordings containing 37 kapalbhati breathing strokes. This audio's first four seconds' amplitudes, visualized in the top sub-figure of **Figure 5.1**, show that there are four breathing strokes of kapalbhati. The figure also shows how all four of these strokes are accurately predicted with very small residuals (prediction - truth). On this entire audio of 32 seconds, the model detected all peaks correctly and the accuracy was 100%. For our second case study, we picked another audio clip of 48 seconds having 34 kapalbhati strokes which is relatively difficult to detect breathing **Figure 5.2**. The first four seconds of this audio and the predictions visualized in the figure show that some of the predictions are false positive, true negative, and true positive with low confidence.

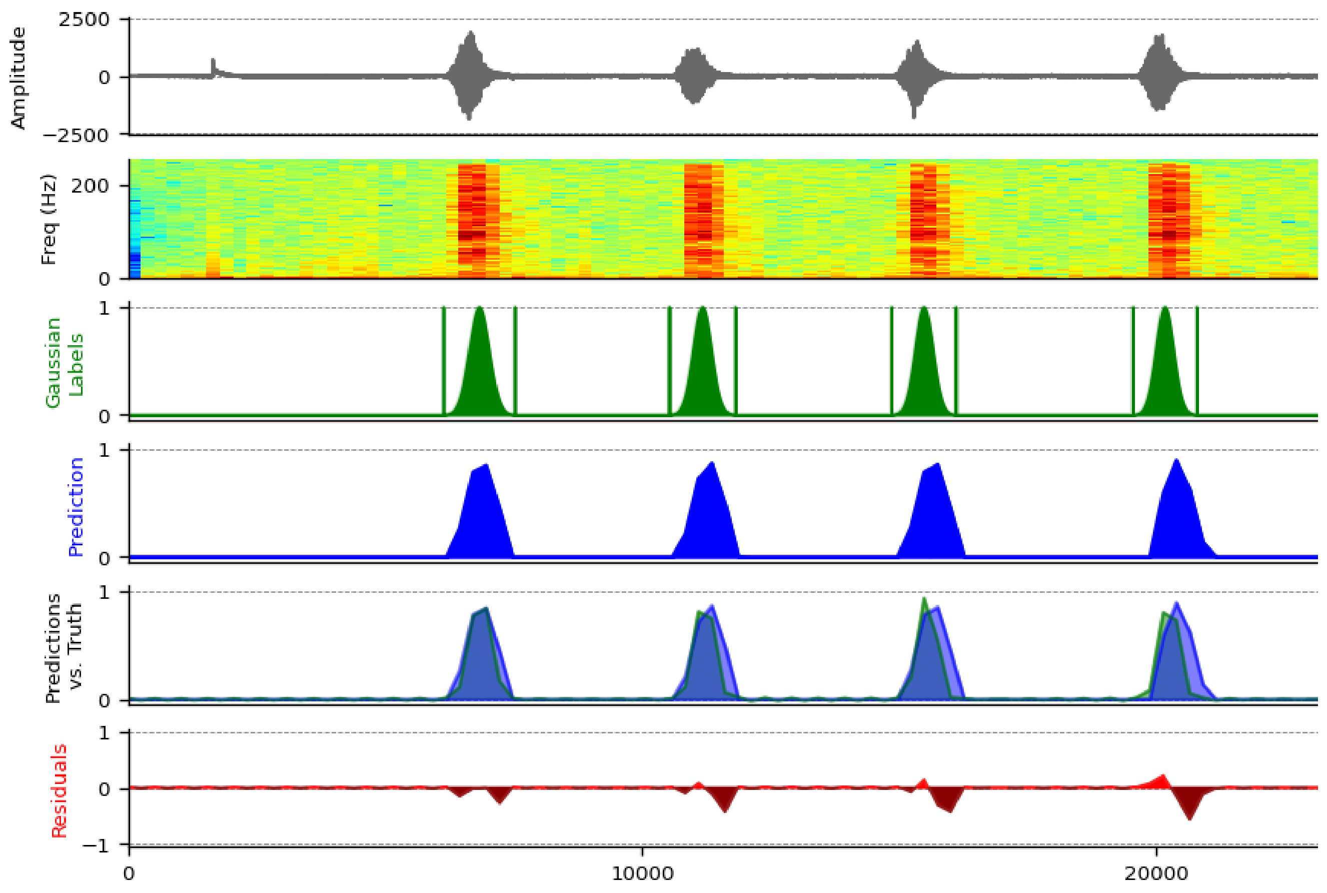


**Figure 5.1:** Screenshot of the first four-seconds of an audio demonstrating the case of an accurate prediction. The top subplot shows an amplitude plot containing four breathing strokes of kapalbhati, followed by the corresponding spectrogram and the training labels hand crafted (in green). In this third subplot, the green vertical lines show the label start and end markers, and the green gaussians in between show how we translate our markers to continuous gaussian values. The fourth, fifth, and last plots show predicted probabilities, overlap of true and predictions, and residuals (predictions - true), respectively.

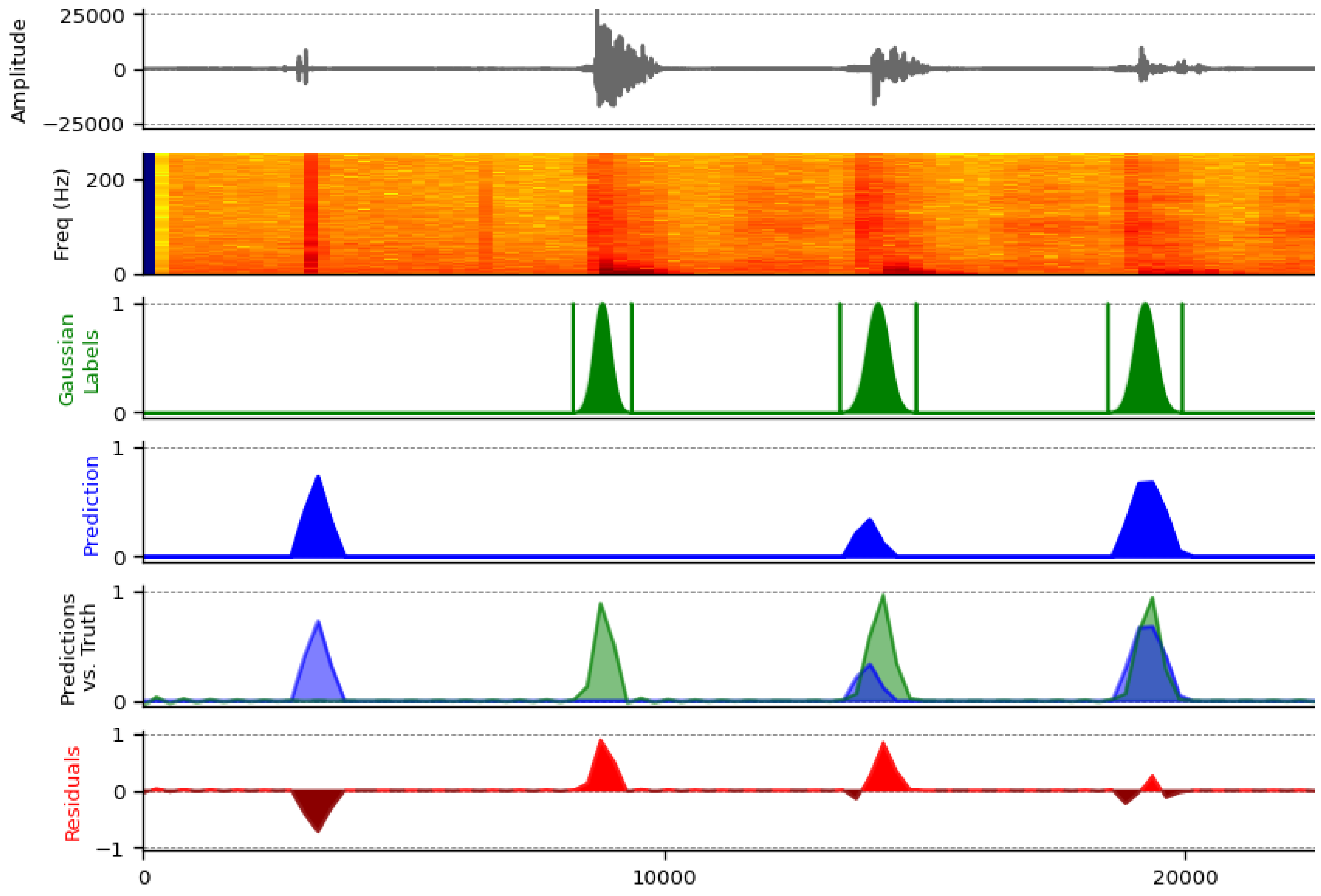


**Figure 5.2:** Screenshot of the first four-seconds of an audio demonstrating the case of a poor prediction. The top subplot shows an amplitude plot containing four breathing strokes of kapalbhati, followed by the corresponding spectrogram and the training labels hand crafted (in green). In this third subplot, the green vertical lines show the label start and end markers, and the green gaussians in between show how we translate our markers to continuous gaussian values. The fourth, fifth, and last plots show predicted probabilities, overlap of true and predictions, and residuals (predictions - true), respectively.

# Chapter 6

# Limitations, conclusion and future work

Our current method, despite being reasonably accurate, has several limitations. First, pranayama is usually practiced by middle-aged and elderly people, but the age distribution of the individuals in the dataset we have prepared does not accurately match this practitioners' age group. Addressing this by collecting more representative data could resolve this. Second, people do not always practice pranayama when they are perfectly healthy. We have not tested our method with breathing from sick/unhealthy group. Finally, pranayama is often practiced in group, either at home or in various yoga camps/sessions. Our method does not work as expected if audio with multiple practitioners practicing at the same time is fed to it.

Using machine learning to detect kapalbhati pranayama sounds is an uninvestigated problem. To the best of our knowledge, this work is the first effort. Despite being the first work, our method's high accuracy of more than 90%, even in the presence of background noise, is highly encouraging.

We believe that the accuracy of our models will further improve significantly if trained using a larger dataset and with a more advanced deep learning model. These results also open a set of possibilities for others to create a variety of mobile applications to aid pranayama practitioners, and more importantly, to minimize side effects and to avoid injuries.

# Appendices

## Appendix A

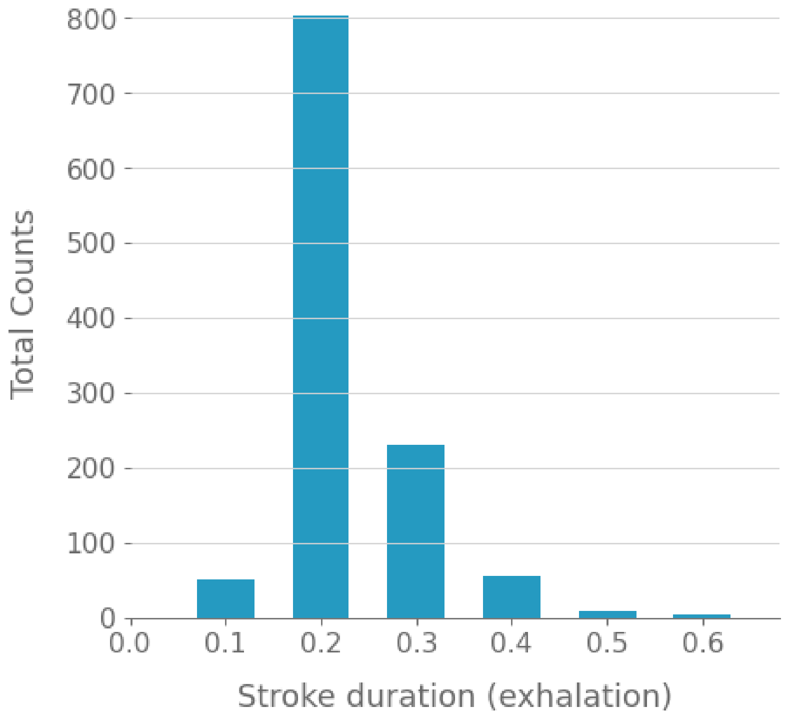


**Figure 1:** Distribution of kapalbhati stroke duration (only exhalation). 0.2 seconds is the most common stroke duration for kapalbhati exhalation.

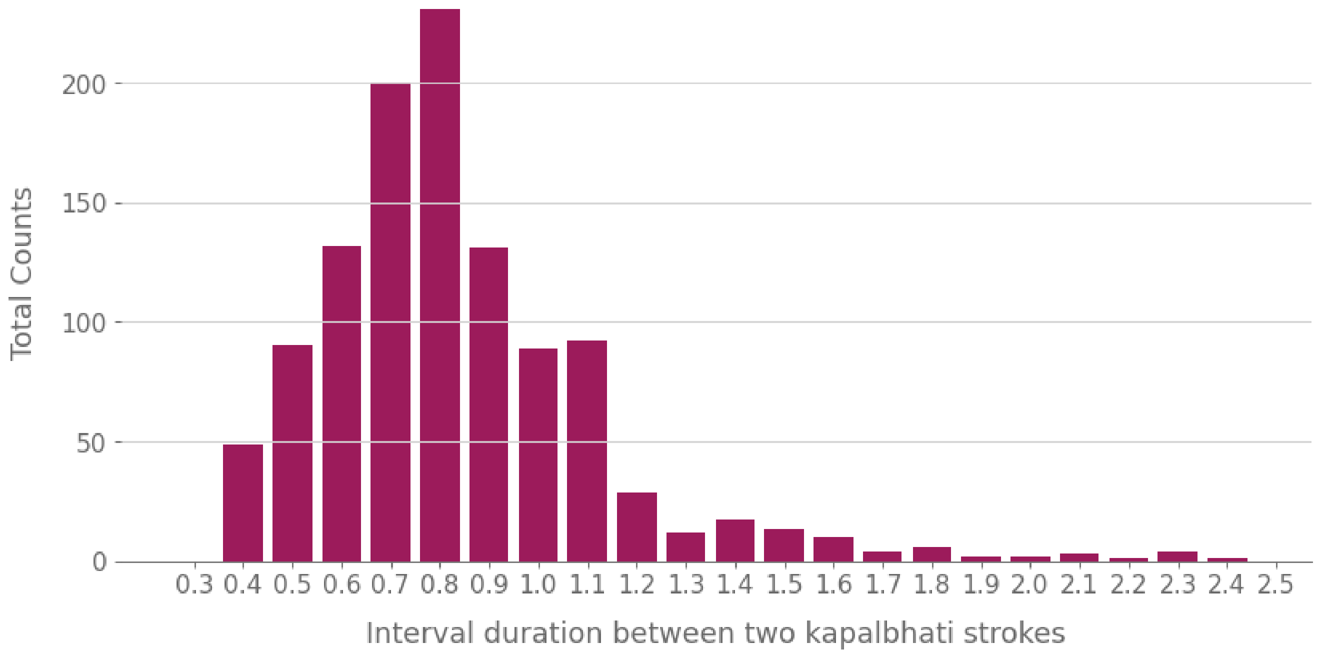


**Figure 2:** Distribution of interval duration between two kapalbhati strokes. The most common interval between two kapalbhati strokes is 0.8 seconds and then 0.7 seconds.

# Appendix B

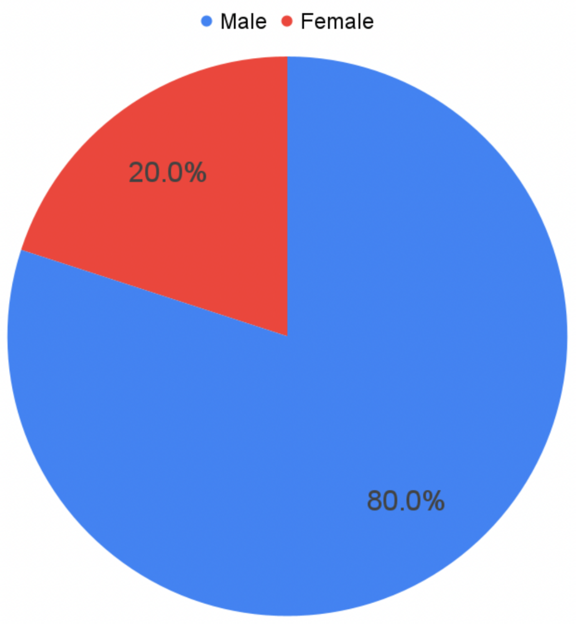
### .0.1 Faster labeling using Audacity

We developed a new approach to prepare the output labels for the datasets. The output label is a signal which has a normal Gaussian distribution. There are multiple steps involved in the process of making the labels which are described below.

1. First we analyzed each of the positive samples in a software called Audacity where we select the starting and ending timestamp of each breathing stroke that were saved as discrete labels. These discrete labels were then used to generate a continuous one-dimensional signal of length L, where L depends on the length of the spectrogram generated using the audio signal.

2. We plotted a normal gaussian distribution for each breathing stroke using the starting and ending timestamps and added them into the output signal (i.e. output label). These gaussian distributions in the signal were considered as a peak which indicates the presence of a breathing stroke which can be seen in the **Figure 4**.

3. In case of negative samples, the whole output signal was assigned with zeros with no peaks.



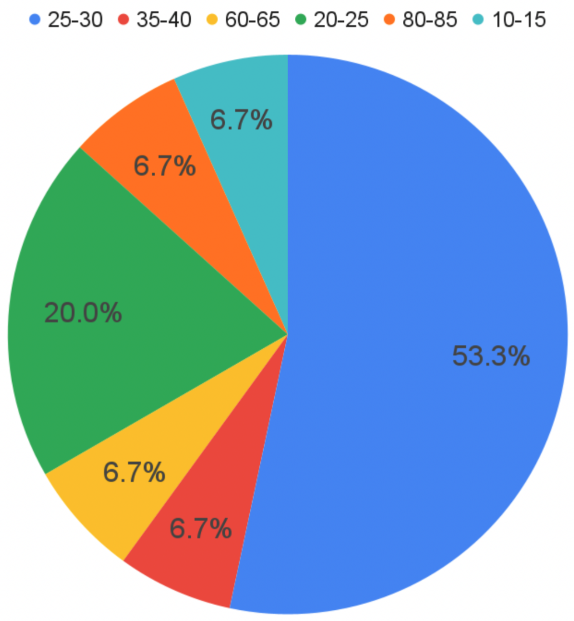


**Figure 3:** Left figure shows the distribution of gender and right figure shows the distribution of age of all the individuals. 80% of the kapalbhati sound samples were collected from male individuals while 20% were collected from female. Similarly, 53.3% of the individuals were of age group 25-30, 20% were of age group 20-25, and rest of the age groups cover 6.7% each.

In other words, the output label for a negative signal is a one dimensional vector of zeros with length L.

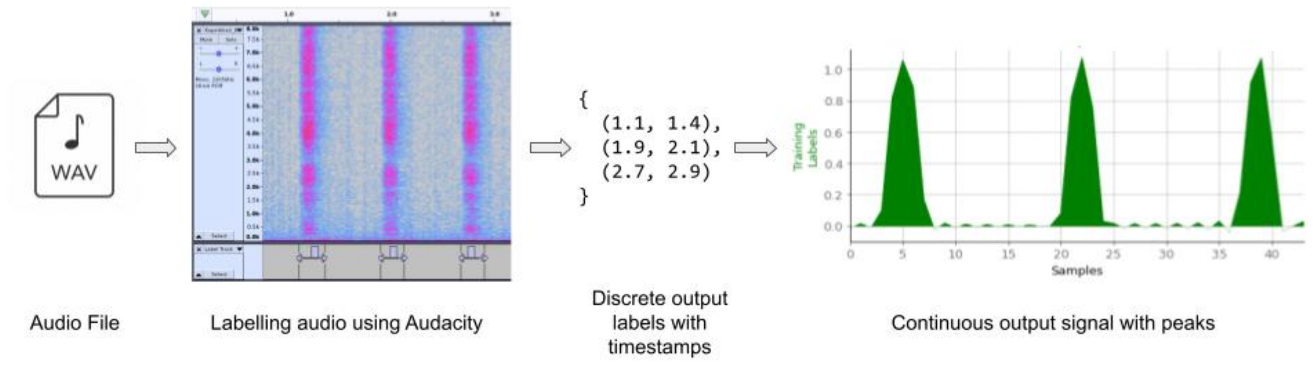

WAV
Audio File
Labelling audio using Audacity
{
(1.1, 1.4),
(1.9, 2.1),
(2.7, 2.9)
}
Discrete output labels with timestamps
Training Labels
Samples
Continuous output signal with peaks

**Figure 4:** Audacity was used to mark the starting and ending time of the kapalbhati breathing strokes. The markings were then exported to a text file that contains the discrete labels which was used to generate the continuous label using Gaussian distribution.

# Appendix C

### .0.2 Calculating signal-to-noise ratio (SNR)

Before adding noise to the positive signal, we need to generate a noise signal using the random SNR. To do so, the following steps were taken:

1. Firs root mean square (RMS) of the positive signal was calculated:

$$rms\_signal = \sqrt{\mu(positive\_signal^2)}$$

2. RMS of the given noise signal was calculated:

$$rms\_noise = \sqrt{\mu(noise^2)} + \epsilon$$

   where $\epsilon$ is a very small number.

3. Then using the rms_signal and the SNR, required RMS for the noise signal was calculated which determines the intensity of the noise.

$$req\_rms\_noise = \sqrt{\frac{rms\_signal^2}{(\frac{SNR}{10})^{10}}}$$

4. At last, using the req_rms_noise and rms_noise, the noise signal was generated:

$$new\_noise\_signal = noise \times \frac{req\_rms\_noise}{rms\_noise}$$

## Appendix D

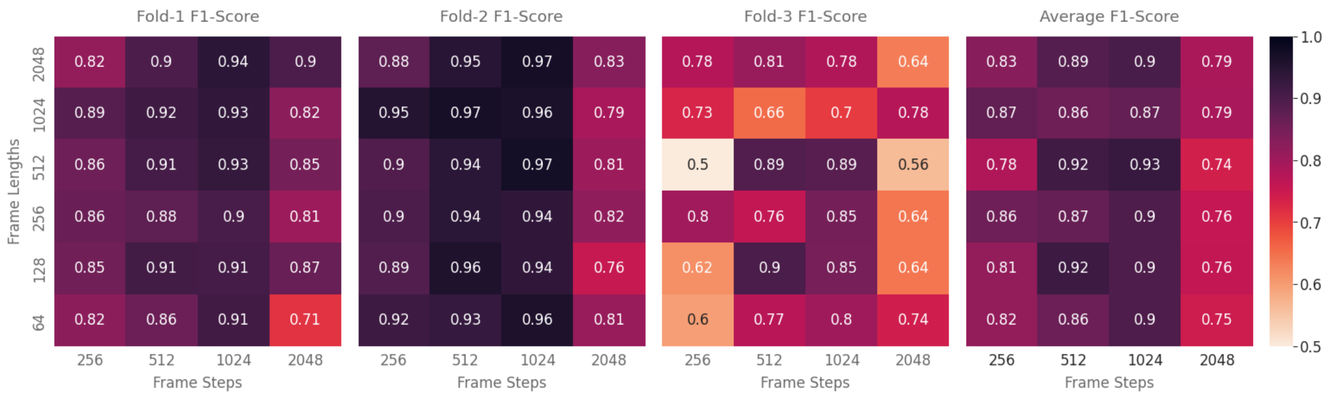


**Figure 5:** Effect of frame length and step on the accuracy across the three folds (first three columns) and the average of the three folds (last column) using F1-Scores for all the 24 combinations of cross validation experiments. Each cell in the first three heat maps represents the F1-Score for the fold and the cells in the last heatmap show averages. The average F1-Score of fold one, two, and three are 0.87, 0.9, and 0.74, respectively, suggesting fold three is the most difficult one followed by fold one.

## Web Application

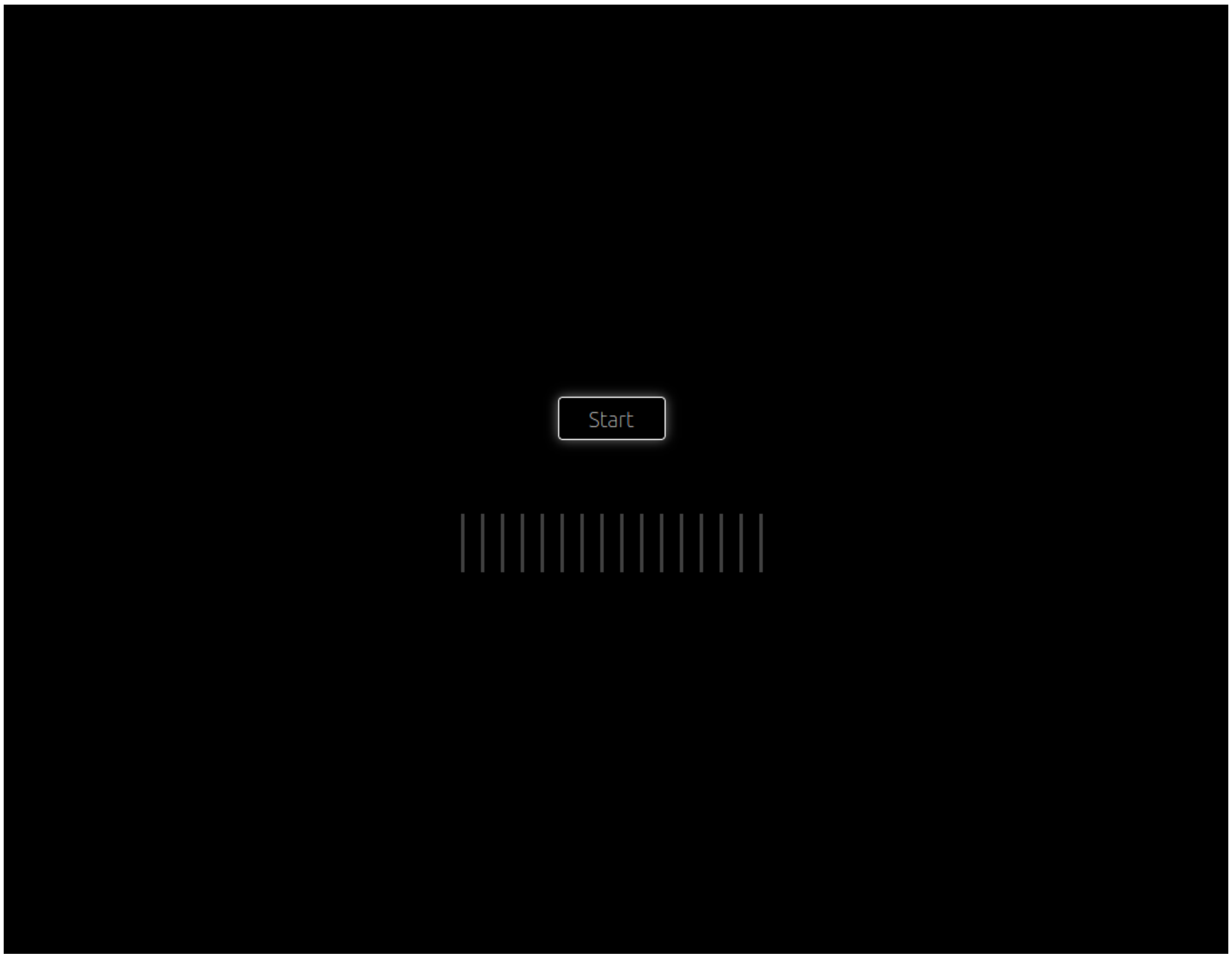


**Figure 6:** Home page of the app showing the start button which starts the recording when clicked.

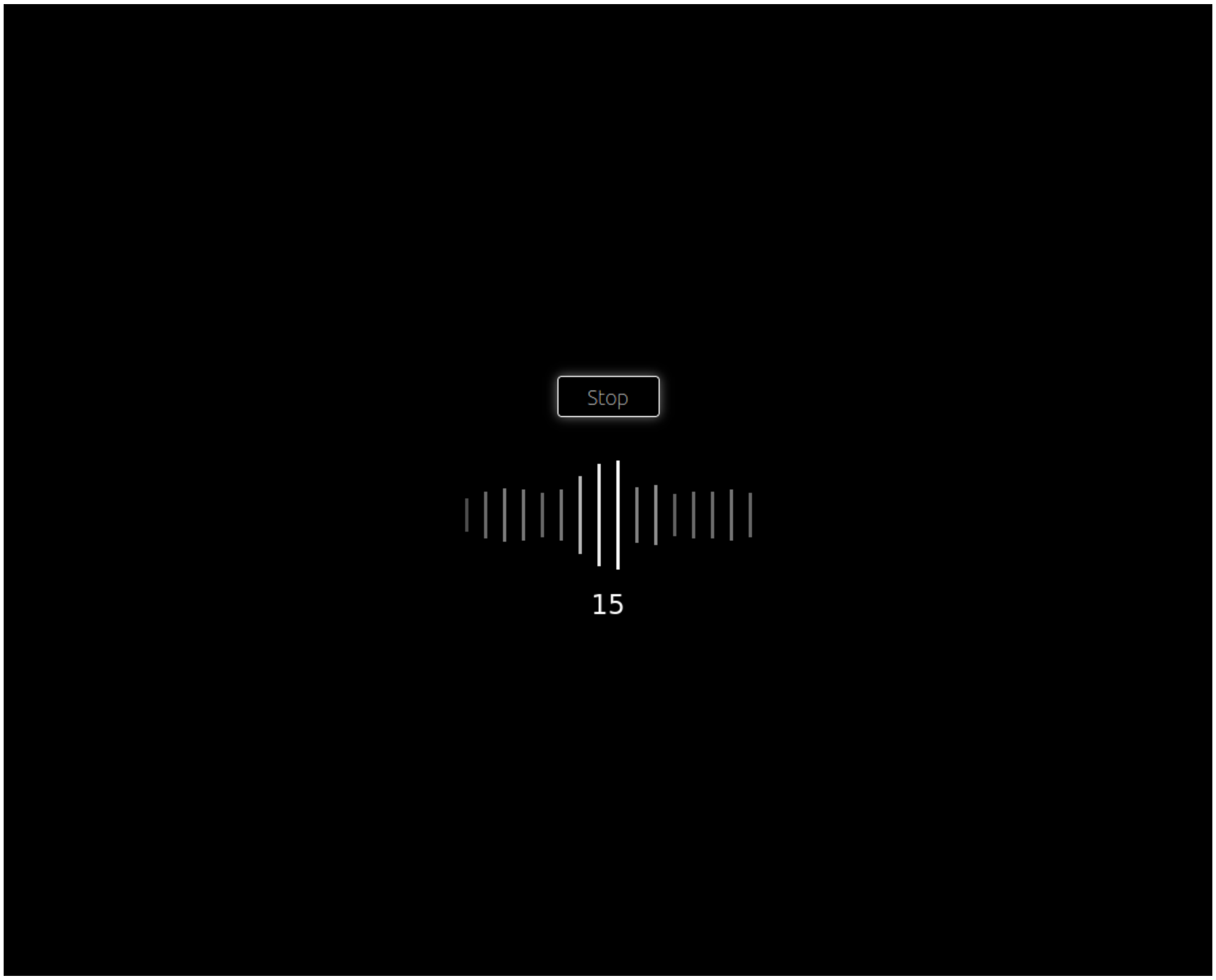


**Figure 7:** The figure shows the running app where it is recording the audio and at the same time showing the count of the breathing cycle of the user's kapalbhati breathing.

# Bibliography

[1] Mohanad Alkhodari and Ahsan Khandoker. Detection of covid-19 in smartphone-based breathing recordings using cnn-bilstm: a pre-screening deep learning tool. *medRxiv*, 2021.

[2] Abdul Malik Badshah, Jamil Ahmad, Nasir Rahim, and Sung Wook Baik. Speech emotion recognition from spectrograms with deep convolutional neural network. In *2017 international conference on platform technology and service (PlatCon)*, pages 1–5. IEEE, 2017.

[3] Ananda Balayogi Bhavanani and Zeena Sanjay. Immediate effect of sukha pranayama on cardiovascular variables in patients of hypertension. *International journal of yoga therapy*, 21(1):73–76, 2011.

[4] Daniel Chamberlain, Rahul Kodgule, Daniela Ganelin, Vivek Miglani, and Richard Ribón Fletcher. Application of semi-supervised deep learning to lung sound analysis. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 804–807. IEEE, 2016.

[5] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE, 2016.

[6] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778. IEEE, 2018.

[7] S Cooper, J Oborne, S Newton, V Harrison, J Thompson Coon, S Lewis, and A Tattersfield. Effect of two breathing exercises (buteyko and pranayama) in asthma: a randomised controlled trial. *Thorax*, 58(8):674–679, 2003.

[8] Jonathan Dennis, Huy Dat Tran, and Haizhou Li. Spectrogram image feature for sound event classification in mismatched conditions. *IEEE signal processing letters*, 18(2):130–133, 2010.

[9] Ahmet Elbir and Nizamettin Aydin. Music genre classification and music recommendation by using deep learning. *Electronics Letters*, 56(12):627–629, 2020.

[10] Ferdos Fessahaye, Luis Perez, Tiffany Zhan, Raymond Zhang, Calais Fossier, Robyn Markarian, Carter Chiu, Justin Zhan, Laxmi Gewali, and Paul Oh. T-recsys: A novel music recom-

mendation system using deep learning. In *2019 IEEE international conference on consumer electronics (ICCE)*, pages 1–6. IEEE, 2019.

[11] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017.

[12] Joe G Greener, Shaun M Kandathil, and David T Jones. Deep learning extends de novo protein modelling coverage of genomes using iteratively predicted structural constraints. *Nature communications*, 10(1):1–13, 2019.

[13] Yue Gu, Shuhong Chen, and Ivan Marsic. Deep mul timodal learning for emotion recognition in spoken language. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5079–5083. IEEE, 2018.

[14] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[15] Rajeev Gupta. Acute effect of kapalbhati yoga on cardiac autonomic control using heart rate variability analysis in healthy male individuals. *Journal of Human Physiology— Volume*, 2(01), 2020.

[16] Takaaki Hori, Jaejin Cho, and Shinji Watanabe. End-to-end speech recognition with word-based rnn language models. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 389–396. IEEE, 2018.

[17] Miao Jiang, Ziyi Yang, and Chen Zhao. What to play next? a rnn-based music recommendation system. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pages 356–358. IEEE, 2017.

[18] Dinkar R Kekan. Effect of kapalbhati pranayama on body mass index and abdominal skinfold thickness. *Ind Med Gaz*, 431:421–5, 2013.

[19] Ruhul Amin Khalil, Edward Jones, Mohammad Inayatullah Babar, Tariqullah Jan, Mohammad Haseeb Zafar, and Thamer Alhussain. Speech emotion recognition using deep learning techniques: A review. *IEEE Access*, 7:117327–117345, 2019.

[20] Seonhoon Kim, Daesik Kim, and Bongwon Suh. Music genre classification using multimodal deep learning. In *Proceedings of HCI Korea*, pages 389–395. 2016.

[21] Jordi Laguarta, Ferran Hueto, and Brian Subirana. Covid-19 artificial intelligence diagnosis using only cough recordings. *IEEE Open Journal of Engineering in Medicine and Biology*, 1:275–281, 2020.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[23] Juncheng Li, Wei Dai, Florian Metze, Shuhui Qu, and Samarjit Das. A comparison of deep learning methods for environmental sound detection. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 126–130. IEEE, 2017.

[24] Xingguang Li, Wenjun Song, and Zonglin Liang. Emotion recognition from speech using deep learning on spectrograms. *Journal of Intelligent & Fuzzy Systems*, 39(3):2791–2796, 2020.

[25] Wootaek Lim, Daeyoung Jang, and Taejin Lee. Speech emotion recognition using convolutional and recurrent neural networks. In *2016 Asia-Pacific signal and information processing association annual summit and conference (APSIPA)*, pages 1–4. IEEE, 2016.

[26] Kristin McClure, Brett Erdreich, Jason HT Bates, Ryan S McGinnis, Axel Masquelin, and Safwan Wshah. Classification and detection of breathing patterns with wearable sensors and deep learning. *Sensors*, 20(22):6481, 2020.

[27] Monika Mourya, Aarti Sood Mahajan, Narinder Pal Singh, and Ajay K Jain. Effect of slow- and fast-breathing exercises on autonomic functions in patients with essential hypertension. *The journal of alternative and complementary medicine*, 15(7):711–717, 2009.

[28] Venkata Srikanth Nallanthighal and H Strik. Deep sensing of breathing signal during conversational speech. 2019.

[29] L Nivethitha, A Mooventhan, NK Manjunath, Lokesh Bathala, and Vijay K Sharma. Cerebrovascular hemodynamics during the practice of bhramari pranayama, kapalbhati and bahir-kumbhaka: An exploratory study. *Applied psychophysiology and biofeedback*, 43(1):87–92, 2018.

[30] Sergio Oramas, Oriol Nieto, Francesco Barbieri, and Xavier Serra. Multi-label music genre classification from audio, text, and images using deep features. *arXiv preprint arXiv:1707.04916*, 2017.

[31] Karol J Piczak. Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th international workshop on machine learning for signal processing (MLSP)*, pages 1–6. IEEE, 2015.

[32] Tapas Pramanik, Hari Om Sharma, Suchita Mishra, Anurag Mishra, Rajesh Prajapati, and Smriti Singh. Immediate effect of slow pace bhastrika pranayama on blood pressure and heart rate. *The Journal of Alternative and Complementary Medicine*, 15(3):293–295, 2009.

[33] Bharathi Raja, S Preetha, and Jothi Priya. Effect of kapalbhati pranayama in the blood sugar level in diabetic patients. *Drug Invention Today*, 10(11), 2018.

[34] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar. Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 193–199. IEEE, 2017.

[35] Unais Sait, Gokul Lal KV, Sanjana Shivakumar, Tarun Kumar, Rahul Bhaumik, Sunny Prajapati, Kriti Bhalla, and Anaghaa Chakrapani. A deep-learning based multimodal system for covid-19 diagnosis using breathing sounds and chest x-ray images. *Applied Soft Computing*, page 107522, 2021.

[36] Apar Avinash Saoji, BR Raghavendra, and Nandi K Manjunath. Effects of yogic breath regulation: A narrative review of scientific evidence. *Journal of Ayurveda and integrative medicine*, 10(1):50–58, 2019.

[37] Melissa Stolar, Margaret Lech, Robert S Bolia, and Michael Skinner. Acoustic characteristics of emotional speech using spectrogram image classification. In *2018 12th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–5. IEEE, 2018.

[38] Melissa N Stolar, Margaret Lech, Robert S Bolia, and Michael Skinner. Real time speech emotion recognition using rgb image classification and transfer learning. In *2017 11th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–8. IEEE, 2017.

[39] Manoranjan Tripathy. The effect of anuloma viloma pranayama and kapalbhati on resting pulse rate and stress of school going children in bhubaneswar. *Group*, 71(312):71–029, 2018.

[40] Yisen Wang, Xuejiao Deng, Songbai Pu, and Zhiheng Huang. Residual convolutional ctc networks for automatic speech recognition. *arXiv preprint arXiv:1702.07793*, 2017.

[41] Jie Xie and Mingying Zhu. Handcrafted features and late fusion with deep learning for bird sound classification. *Ecological Informatics*, 52:74–81, 2019.

[42] Promod Yenigalla, Abhay Kumar, Suraj Tripathi, Chirag Singh, Sibsambhu Kar, and Jithendra Vepa. Speech emotion recognition using spectrogram & phoneme embedding. In *Interspeech*, pages 3688–3692, 2018.

[43] Yeonguk Yu and Yoon-Joong Kim. Attention-lstm-attention model for speech emotion recognition and analysis of iemocap database. *Electronics*, 9(5):713, 2020.