TIME-DOMAIN COUGH DURATION DETECTION USING DEEP NEURAL NETWORKS

Yuang Tong[*] *Matthew Whitehill*[†] *Shwetak Patel*[†]

[*]Tsinghua University
[†]University of Washington

ABSTRACT

Recent work has proven the effectiveness of automatic cough counting systems using audio [1, 2]. However, these systems only determine when a cough occurs, not the cough's duration, which is key information for monitoring patients with a chronic cough. While others have developed systems that can detect the onset and offset times of audio events [3], they are not tuned for cough detection, resulting in lower performance. Furthermore, they usually utilize a frequency transform as the feature extractor, which is inefficient when running on mobile devices. In this paper, we introduce a deep learning approach that uses 1-Dimensional (1-D) convolutions as the feature extractor and an encoder-decoder U-net architecture to determine cough duration. Our model takes the raw audio as input, eliminating the need for frequency transforms, and generates the predicted duration of cough within an audio frame. When evaluated over a 4-fold cross-validation set, our model reaches 89.72% F1-Score, a 6.91% increase over the baseline model.

Index Terms— Health sensing, Cough, Deep Neural Network, Semantic Segmentation

1. INTRODUCTION

1.1. Cough And Respiratory Disease

Cough is the most common reason for visiting a primary care physician in the United States and the prevalence is estimated to be between 10-15% of the population [4, 5]. However, determining the cause of a patient's chronic cough can be a difficult process. Currently, doctors rely on the patient's self-report of their cough symptoms to determine the cause of the cough. However, patients are notoriously poor historians. More objective measurement of their cough could provide improved diagnosis and treatment. This includes collecting data about the cough duration, since knowing whether each cough is an extended coughing fit or a small throat tickle can help doctors better determine the cough's root cause.

1.2. Cough Detection Systems

Recent work has focused on addressing this issue through the development of automatic cough detection systems. These systems operate by counting the number of coughs in an audio file by distinguishing them from background noise. The most common approach involves transforming the cough into a frequency representation, such as the Mel-frequency spectrogram, then using machine learning methods to identify the coughs [1, 2, 6]. However, these systems can only determine if a cough occurs within a frame of audio, not the cough's duration.

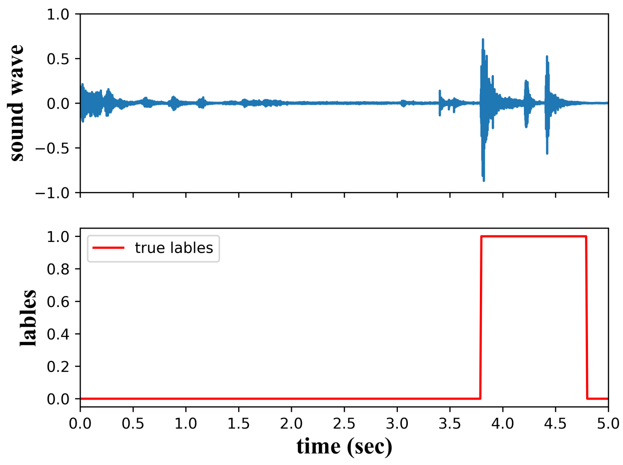

Fig. 1. (Top) The audio signal of a cough. (Bottom) Cough labels where a 1 signifies a cough, 0 signifies no cough.

1.3. Sound Classification and Duration Detection

Detecting coughs in an audio file represents a subset of the Acoustic Scene Classification (ACS) task, which attempts to determine the sound(s) present in a given audio file. Others have developed systems that have proven effective at determining the duration of sounds for ACS. Kong et al. introduced a 2-D Convolutional Neural Network (CNN) by taking the Mel-spectrum as input to classify different sounds [7]. Ren et al. proposed using an atrous Convolutional Neural Network with global attention pooling as the classification model [8]. Phan et al. introduced a 1-D multi-label, multi-task convolutional recurrent neural network (CRNN) to unify the detection of isolated and overlapping audio events [9].

However, these prior works have two main limitations. First, they are not specifically designed for detecting cough duration and we found their performance to be sub-optimal for this task. Second, these methods have slow run-times on mobile devices, primarily due to the fact they rely on the frequency transform, which is inefficient when implemented on mobile devices. To protect the user's privacy, it is crucial that cough detection algorithms run on the mobile device itself, not a server, so raw audio data does not need to be transmitted to the cloud to detect coughs or their duration.

1.4. Our Method

Our method solves both issues. First, we developed a deep learning model designed specifically for cough duration detection by leveraging a 1-D U-net encoder-decoder and LSTM output layer. This architecture is inspired by work in semantic segmentation for computer vision [10], since finding cough events within a larger frame can be viewed as a segmentation task, where the model is segmenting coughs from other sounds such as noise or speech. Second, inspired by Huang et al. [11], we use a 1-D convolutional neural network as the feature extractor rather than transforming the raw sound into the frequency-domain. This allows our model to avoid the expensive frequency transform computation when running on mobile devices. With both of these features, our model reaches an 89.72% F1-Score, a 6.91% increase over the baseline [9]. To our knowledge, this is the first model to specifically tackle the cough duration task.

2. MODEL ARCHITECTURE

Our model consists of 3 components - 1) a 1-D CNN encoder, 2) a 1-D U-Net decoder, and 3) a Long-Short Term Memory (LSTM) output layer. The Encoder is used for feature extraction, the Decoder for prediction, and the LSTM layer for leveraging information from adjacent frames when making predictions. The input to the model is an audio signal of length 5 seconds at 16khz sample rate (80,000 samples). The targets are generated by producing a binary label every 10 milliseconds (1 meaning a cough, 0 meaning not a cough) for a total of 500 targets.

2.0.1. Encoder

The Encoder is a CNN consisting of four 1-D convolutional layers. Except for the first layer, all the conv-layers are followed by a Batch-Normalization layer, ReLu activation, and a MaxPooling layer. Please see Table 1 for the complete architecture of the encoder.

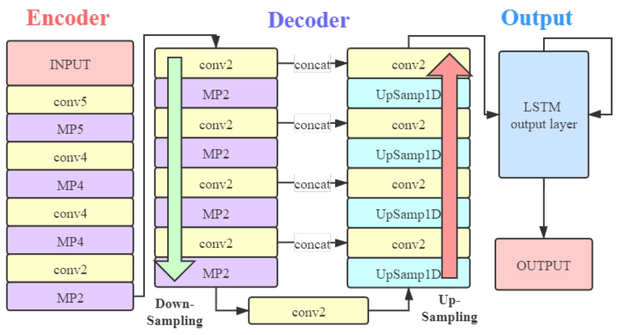

Fig. 2. The architecture of our proposed 1-D U-Net. The Encoder consists of four 1-D convolutional layers. The decoder has down-sampling, up-sampling, and short-cut layers, followed by the LSTM output layer.

Name	Layer	Output Size
Input	Input	80000×1
Conv1	Conv K5×C8-MP K5×S5	16000×8
Conv2	Conv K4×C16-MP K4×S4	4000×16
Conv3	Conv K4×C32-MP K4×S4	1000×32
Conv4	Conv K2×C32-MP K2×S2	500×32
Dropout	Dropout 0.8	500×32

Table 1. Our Encoder's architecture. All convolutional and Maxpooling layers are 1-D. K stands for kernel size, C for channels, and S for stride length, so K5 means kernel size 5. We applied Batch Normalization and ReLU activation at the end of each layer. The convolutional layers all have stride 1.

2.0.2. Decoder

Our decoder is inspired by the decoder from the U-Net architecture [10]. The Decoder consists of four 1-D down-sampling layers and four 1-D up-sampling layers. Each down-sampling layer consists of a convolutional layer and a MaxPooling layer of stride 2, which will cut the tensor in half and double the number of filters.

The up-sampling layer consists of a 1-D up-sampling layer, a concatenate layer, and a convolutional layer. The concatenate layer connects the previous convolutional layer's output and the up-sample layer's output. The decoder's structure is shown in Table 2.

2.0.3. LSTM output layer

Since knowing whether the previous convolutional output contained a cough can help determine if the current output also contains a cough, we use an LSTM layer with sigmoid activation to produce the final classification. The output of Up-sampling can be interpreted as a sequence of feature vectors, which the LSTM reads to generate the final result.

Name	Layer	Output Size
Encoder	Input	500×32
Conv1	Conv K2×C32-MP K2×S2	250×32
Conv2	Conv K2×C64-MP K2×S2	125×64
Conv3	Conv K2×C128-MP K2×S2	63×128
Conv4	Conv K2×C256-MP K2×S2	32×256
Conv5	Conv K2 ×C512	32×512
Decoder	Input	32×256
Upconv1	Upconv K2	63×256
Concat1	Concat (Conv4,Upconv1)	63×512
Conv5	Conv K2×128	C63×128
Upconv2	Upconv K2	125×128
Concat2	Concat (Conv3,Upconv2)	125×256
Conv6	Conv K2×C64	125×64
Upconv3	Upconv K2	250×64
Concat3	Concat (Conv2,Upconv3)	250×128
Conv7	Conv K2×C32	250×32
Upconv4	Upconv K2	500×64
Concat4	Concat (Conv1,Upconv4)	500×128
Conv8	Conv K2×C32	500×32
Dropout	Dropout 0.8	500×32
LSTM	LSTM	500×1

Table 2. Our decoder's architecture. The K, C, and S have the same meaning as in the encoder. We apply ReLU activation after each convolutional layer. An Upconv layer with K=2 doubles the length of the tensor but won't change the number of channels. The concatenate layer connects the output of a convolution before MP and the output of an Upconv layer.

3. EXPERIMENTAL SETUP

3.1. Datasets

Our datasets consist of both open-source datasets and proprietary datasets (see Table 3). Audioset is an open-source sound dataset collected from Youtube videos [12]. The Whosecough and South Africa datasets were collected from subjects sitting in a room next to one or more microphones for an extended period of time. The Coughsense dataset is collected by having participants carry a microphone in their pockets or around their necks for 2-6 hours [6]. The Online Survey dataset was collected by having more than 400 subjects perform up to 20 forced coughs and other sounds into their computer. We use both cough and non-cough samples (such as speech, silence, and laughter) from each dataset during training.

We combine the datasets, then divide the total dataset into 4 parts for 4-fold cross-validation. We use a batch size of 32, and select a certain number of sounds from various classes to fill out each batch. For example, for each batch, we choose 20 cough samples and 12 other sounds (such as laughter, music, speech, and respiratory sounds).

Name	Dataset Type	Number of Coughs
Audioset	Online Videos	3,727
Whosecough	Controlled	6,845
South Africa	Controlled	2,100
Coughsense	In-the-Wild	2,445
Online Survey	Voluntary	7,039
Total		22,156

Table 3. Datasets used during training and testing with the total number of coughs in each dataset.

3.2. Model Input

The audio samples from each dataset are first re-sampled to 16kHz. Each audio sample may have a different length, so we cut or zero-pad each to be 5 seconds in length. Next, all audio samples are re-labeled as ”cough” (1) or ”not cough” (0) at a frequency of 100 Hz, meaning labels occur every 10 milliseconds. The input is a 1-D array of length 80,000 while the labels array is of length 500.

3.3. Data Augmentation

During training only, we preprocess each sample before passing it to the model. Many of the samples have extended periods of silence or low-noise which are easier for the model to classify as not a cough. To expose the model to more difficult samples, we add noise from the MUSAN dataset [13] to these parts. We also add background white noise and do random gain amplification to further augment the samples. An example of all 3 augmentation techniques can be seen in Figure 3.

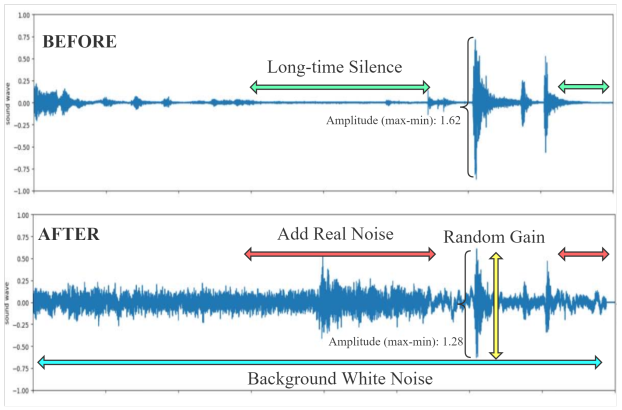

Fig. 3. An example of data augmentation. The green arrow indicates extended periods of silence while the red arrow shows the added noise. The turquoise arrow shows the added background white noise and the yellow arrow shows the random amplitude gain.

3.4. Training Parameters

We use TensorFlow 2.2.0 and train the model for 5000 epochs on an NVIDIA Titan Xp GPU. We use the Adam optimizer with initial learning rate of 0.001, exponentially decayed after the first 1500 epochs.

3.5. Baseline Model

The most similar prior work to our model is the CRNN model proposed by Phan et al [9], which we use as a baseline for comparison with a couple of modifications. In order to match the same size output as our model, we change the kernel of the last pooling layer from 2 to 8. Furthermore, we use the 16khz sample rate data from our datasets even though the paper reported using a 44.1khz sample rate.

4. RESULTS

We test the baseline model and our model's performance on the 4-fold cross-validation dataset. We use precision, recall, and F1-Score to evaluate the model. An example of visualized predictions can be seen in Figure 4 and the results are shown in Table 4. As can be seen, our model achieves an 89.72% F1-score, a 6.91% improvement over the baseline.

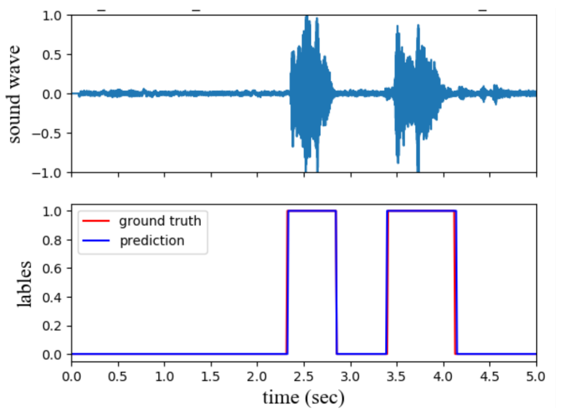

Fig. 4. Visualized prediction results. (Top) The audio signal of a cough. (Bottom) Ground truth (red) and predictions (blue), where 1 signifies a cough, 0 signifies no cough.

Model	prec(%)	recall(%)	F1-Score(%)
Baseline	**93.96**	75.82	83.92
Our model	92.40	**87.20**	**89.72**

Table 4. The performance of our model and the baseline.

4.1. Duration Error

To quantify the error in cough duration for each sample, we calculate the Median Absolute Error (MedAE) of the predictions generated from our model and the baseline model for each dataset. For each audio sample with a cough in it, we sum the 100 Hz cough labels in the prediction and compare it with the sum of the cough labels in the ground truth. The difference indicates the duration error. For example, since each label represents 10 ms, a difference of 2 in the sum of labels indicates an error of 20 ms. Then we calculate the median of the absolute duration error.

Please see Table 5 for a summary of the results. We have also included the median duration of ground truth cough labels in each sample across each dataset. As can be seen in Table 5, our model's MedAE is lower than the baseline for each dataset and reaches just 50 ms of error for a median duration of 700 ms of cough across all datasets.

Dataset		OS	CS	SA	WC	AS	All
Med cough duration (ms)		460	640	570	780	930	700
MedAE (ms)	**Baseline**	10	130	80	130	530	100
	Our Model	0	110	70	70	320	50

Table 5. The MedAE of our model and the baseline model on different datasets. OS stands for the online survey dataset, CS for coughsense, SA for southafrica, WC for whosecough and AS for audioset. All means all of the datasets.

4.2. Ablation study

To show the importance of different components in our network, we run an ablation study using models with different components removed. Please see Table 6 for the results. As can be seen, each component is crucial to the overall performance of our model.

Model	prec(%)	recall(%)	F1-Score(%)
simple-CNN	74.92	29.55	42.38
simple-CNN + LSTM	85.59	69.64	68.73
1D-Unet (no LSTM)	90.63	71.01	76.63
Our model	97.00	87.20	89.72

Table 6. Results of the ablation study for different model components. The simple-CNN model has only four convolutional layers with no decoder or LSTM layer. The simple-CNN+LSTM model is the simple-CNN model with an LSTM output layer. The 1-D U-Net is our model without an LSTM output layer.

5. CONCLUSION

In this paper, we introduce a deep learning model to detect the duration of coughs in a given audio clip. We introduce a 1-D U-net encoder-decoder architecture that outperforms prior work in this space. In total, our model achieves an 89.72% F1-Score and a MedAE of 50ms on the 4-fold validation set.

6. REFERENCES

[1] Justice Amoh and Kofi Odame, "Deep neural networks for identifying cough sounds," *IEEE transactions on biomedical circuits and systems*, vol. 10, no. 5, pp. 1003–1011, 2016.

[2] Forsad Al Hossain, Andrew A Lover, George A Corey, Nicholas G Reich, and Tauhidur Rahman, "Flusense: a contactless syndromic surveillance platform for influenza-like illness in hospital waiting areas," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 1, pp. 1–28, 2020.

[3] Simon Dixon, "Onset detection revisited," in *Proceedings of the 9th International Conference on Digital Audio Effects*. Citeseer, 2006, vol. 120, pp. 133–137.

[4] Richard S Irwin, Louis-Philipe Boulet, Michelle M Cloutier, Richard Fuller, Philip M Gold, Victor Hoffstein, Alvin J Ing, F Dennis McCool, Paul O'Byrne, Robert H Poe, et al., "Managing cough as a defense mechanism and as a symptom: a consensus panel report of the american college of chest physicians," *Chest*, vol. 114, no. 2, pp. 133S–181S, 1998.

[5] Kian Fan Chung and Ian D Pavord, "Prevalence, pathogenesis, and causes of chronic cough," *The Lancet*, vol. 371, no. 9621, pp. 1364–1374, 2008.

[6] Eric C Larson, TienJui Lee, Sean Liu, Margaret Rosenfeld, and Shwetak N Patel, "Accurate and privacy preserving cough sensing using a low-cost microphone," in *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 2011, pp. 375–384.

[7] Qiuqiang Kong, Yong Xu, Iwona Sobieraj, Wenwu Wang, and Mark D Plumbley, "Sound event detection and time–frequency segmentation from weakly labelled data," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 4, pp. 777–787, 2019.

[8] Zhao Ren, Qiuqiang Kong, Jing Han, Mark D Plumbley, and Björn W Schuller, "Attention-based atrous convolutional neural networks: Visualisation and understanding perspectives of acoustic scenes," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 56–60.

[9] Huy Phan, Oliver Y Chén, Philipp Koch, Lam Pham, Ian McLoughlin, Alfred Mertins, and Maarten De Vos, "Unifying isolated and overlapping audio event detection with multi-label multi-task convolutional recurrent neural networks," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 51–55.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[11] Jonathan J Huang and Juan Jose Alvarado Leanos, "Aclnet: efficient end-to-end audio classification cnn," *arXiv preprint arXiv:1811.06669*, 2018.

[12] 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)*. IEEE, 2017, pp. 776–780.

[13] David Snyder, Guoguo Chen, and Daniel Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.