CoughNet: A Flexible Low Power CNN-LSTM Processor for Cough Sound Detection

Hasib-Al Rashid, Arnab Neelim Mazumder, Utteja Panchakshara Kallakuri Niyogi, Tinoosh Mohsenin
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, USA
Email: {hrashid1, arnabm1, ukalla1, tinoosh}@umbc.edu

Abstract—**The continuing effect of COVID-19 pulmonary infection has highlighted the importance of machine-aided diagnosis for its initial symptoms such as fever, dry cough, fatigue, and dyspnea. This paper attempts to address the respiratory-related symptoms, using a low power scalable software and hardware framework. We propose CoughNet, a flexible low power CNN-LSTM processor that can take audio recordings as input to detect cough sounds in audio recordings. We analyze the three different publicly available datasets and use those as part of our evaluation to detect cough sound in audio recordings. We perform windowing and hyperparameter optimization on the software side with regard to fitting the network architecture to the hardware system. A scalable hardware prototype is designed to handle different numbers of processing engines and flexible bitwidth using Verilog HDL on Xilinx Kintex-7 160t FPGA. The proposed implementation of hardware has a low power consumption of o 290 mW and energy consumption of 2 mJ which is about 99 $\times$ less compared to the state-of-the-art implementation.**

I. Introduction

Coughing is one of the most common symptoms that is reported among patients, and it is usually the first symptom of most respiratory illnesses. In fact, coughing is one of the early symptoms of the recent infectious disease, COVID-19. Traditionally, when patients feel symptoms, they either contact a doctor or have themselves examined by medical professionals at walk-in facilities where extensive use of vital signs, visual and auditory input is used to make diagnostic decisions. During a pandemic, in-person clinical visits are restricted to minimize the virus transmission and to help the health care system. As a result, machine aided remote diagnosis is getting more attention due to this highly contagious COVID-19 outbreak. Machine learning and deep learning models for various early symptoms detection would be a solution which can be implemented on low-powered mobile devices to replace the initial screening by the health practitioners to reduce the risk of infection to be spread.

When the patients record their cough sounds through a web based or an app based recorder, noise from the surroundings might change the early detection results. As a result, detecting cough sounds from recorded audio is another important research direction to make the early symptoms detection work as per the plan. Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) Networks are two important deep learning algorithms which have shown impressive performance in image and time-series classification tasks which make them good contenders for audio recognition tasks as well. Researchers in [1] have suggested end-to-end CNN models for audio recognition tasks. However, because of their large model sizes and computations, these models are not ideally adapted for integrated, low-power applications. To address this research question, this paper presents a combined CNN-LSTM based low powered energy efficient cough detection architecture which can be implemented on resource constrained, small processors of the cell-phones and tablets and/or FPGAs. The main contributions of this paper include:

- Propose CoughNet, a flexible software hardware CNN-LSTM framework that can take audio recordings and be configured for detecting cough sounds in it.
- Design a parameterized and flexible hardware in verilog HDL for different input modalities, numbers of processing engines (PE) and flexible data bitwidth that replicate the CoughNet for low power deployment.

II. Related Work

With the advent of numerous machine learning and deep learning technologies [2]–[6], audio based medical diagnosis has recently become an active field of study. Authors in [7]–[9] used deep CNN and RNN to classify cough and lung sounds. Authors in [8] proposed Log quantized deep CNN-RNN based model for respiratory sound classification for memory limited wearable devices. Authors in [9] presented an end-to-end CNN model to detect cough sounds from directly the recorded audio. However, their models still require large number of operations due to large input size and model architectures which requires large power consumption that is limited in the resource constrained wearable devices. In this paper, we tried to address the minimization of the number of computations by leveraging the mel-spectrogram of the input audio recordings and making an CNN-LSTM based cough detection model. We also emphasize on the energy efficient implementation on FPGA hardware.

III. CoughNet Framework

The high level overview of the proposed CoughNet framework and the more detailed architecture of the

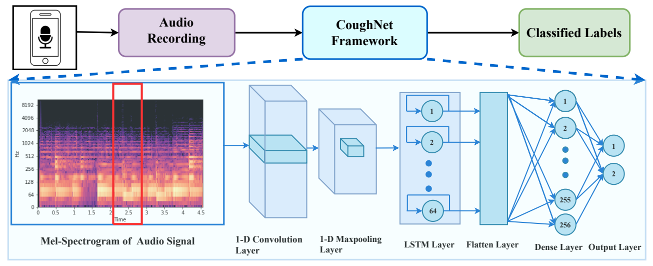

Fig. 1: The detailed architecture of the proposed flexible *CoughNet* in which CNN-LSTM based deep neural network is implemented that can be used for cough detection. The input of the model is Mel-Spectrogram 2d image of size (101, 40) which is converted from audio recordings with sampling rate of 44.1 KHz.

framework is presented in Figure 1. CoughNet can take an audio recordings of the cough from the user and detect accordingly. As the input is in the form of audio recordings, we converted the audio recordings into mel-spectrogram 2D images, where rows correspond to frequencies in Mel scale and columns correspond to time (window) and each value represents log amplitude value of the signal corresponding to that frequency and time window. Then it is divided into window frames to extract features since the right windows to distinguish between static and continuous signals are crucial. Windowing involves first standardizing the independent variables and then creating sliding T windows. Then the window frames are forwarded to the one dimensional CNN layer with 40 filters and kernel size 8 for necessary feature extraction. Then the output is passed to a maxpool layer with pool size 8 which selects the maximum values from a pixel neighborhood to minimize the total parameters of the network. Then to learn temporal relations from the audio signals, the output from the maxpool layer is passed to an LSTM layer of 64 units. Then the output is flattened and then forwarded to a fully connected layers to isolate sufficient window frame information with interconnections between nodes. At the end, the output is seen in the form of the probability distribution of the last fully connected layer with the Softmax activation function. We trained our model with categorical cross-entropy loss and SGD optimizer with 0.6 as momentum.

IV. Experimental Results and Analysis

In this section, CoughNet is evaluated with in-depth analysis using three different datasets for cough detection along with the respective experimental results. Moreover, a brief comparison with some state-of-the-art models is also presented here.

A. *Data Sets*

We evaluated CoughNet for cough detection on three different datasets: ESC-50 [10], FSDKaggle2018 [11], and Coughvid [12]. The ESC-50 dataset contains a total of 2,000 audio recordings of normal environmental sounds. It has 50 equally distributed classes including “coughing”, so that each class has 40 audio recordings. All the audio recordings are 5 seconds in length and are stored as single-channel audio waveform files at 44.1 kHz sampling

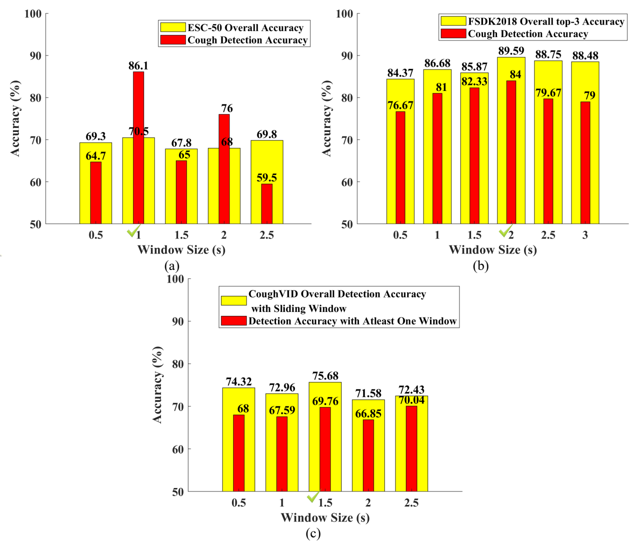

Fig. 2: Detection Accuracy with different window sizes for (a)ESC-50 cough detection, (b) FSDK2018 Cough Detection (c) CoughVID Cough Detection

rate. The FSDKaggle2018 dataset also contains 41 sound classes and cough is one of them. There are 11,073 audio recording samples, where each of the audio recordings is an uncompressed Pulse Code Modulation (PCM) 16 bit, 44.1 kHz, mono audio file. We consider a window with a stride always 0.25s and its label as one instance of model input for all cases. However, since the sound of an audio recording may only exist in some of the extracted windows, we evaluate the predictions at audio recording level by probability voting [13]. Moreover, Coughvid is a crowdsourced dataset for machine learning researchers aiming to find the connections between COVID-19 diagnosis and cough sound features. It provides over 20,000 cough recordings donated by participants. As an initial step of taking fully advantage of this dataset for COVID-19 research, we evaluate cough detection with this dataset.

B. *Results*

Figure 2(a) shows the accuracy results for the ESC-50 dataset with respect to window size. As evident in Figure 2 (a), all the experiments show similar performances on the overall accuracy metric. As for the performance on cough detection, 1s windows show the good and balanced performance of extracting distinctive features. Thus, a window size of 1s is chosen for our implementation scenario.

We considered the overall top-3 accuracy and recall score of the cough class as our metrics to assess the proposed architecture on cough detection. Figure 2 (b) shows the overall top-3 accuracy and recall score results for FSDKaggle2018 dataset with respect to window size. As evident in Figure 2 (b), all the experiments show similar performances on overall top-3 accuracy metric. As for the performance on cough detection, 2s windows show good and balanced performance of extracting distinctive features. A window size of 2s is chosen for our implementation.

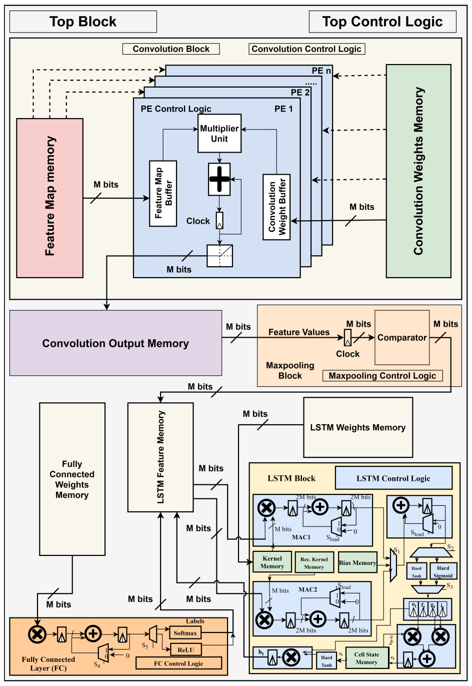

Fig. 3: CoughNet The hardware architecture designed for CoughNet consists of convolution, maxpooling, LSTM and fully connected modules and can be configured for M-bit precision where M ranges from 8 to 64 bits.

We used models trained on the ESC-50 dataset and used transfer learning to predict cough existence into CoughVID dataset, and compared with an assumed ground truth based on the affiliated probability. We considered two cough existence prediction schemes here. For the first one, we predict the audio recording contains cough if cough class is among the top-5 predictions of the sliding-window probability-voting results. For the second one, if at least one window gives a cough prediction among the top-5 predictions, we consider the audio recording has cough. This approach is taken because CoughVID dataset contains some silent audio recordings as this is a volunteer crowd-sourced dataset. Figure 2 (c) shows the results for both schemes by different input window sizes, we chose 1.5 sec window size as it gives better performance.

V. Hardware Architecture Design

The hardware accelerator for the CoughNet framework illustrated in Figure 2, was designed with the primary implementation requirements of low utilization overhead, low power consumption and infrequent memory accesses for the hardware RTL (Register-Transfer Level) configuration. The individual modules and the final pipelined architecture have been elaborated in the following paragraphs.

Convolution module is constructed to perform both 1D and 2D convolution of the feature space with the help of sliding filters. The module takes in image pixels as feature input and filter weights as filter input in the form of block RAM (BRAMs) memories. It mimics the convolution operation by moving the filters all over the image space to isolate spatial features. The module considers the edge cases of the feature space and performs valid padding. Also, the capacity of the module is further extended by parallelizing the filter movement with multiple processing engines. Each processing engine contains a multiplier along with an adder to replicate the ReLu activation logic of the convolutional layers.

Maxpooling module reduces the feature space by selecting the maximum feature value within a space bounded by the filter shape. Even though this module does not perform any computation, it still implements the logic to extract the maximum feature value by using a comparator in the design flow. Along with this, the design for this block also takes care of uniform and non-uniform striding of the filters to accommodate maxpooling for any given window shape.

LSTM module performs the time-space processing of long data sequences and to this extent, the hardware architecture implements matrix-vector multiplications showed in [14]. To reduce frequent memory accesses and high-power overhead, the kernel weights and recurrent kernel weights use two different memories. Furthermore, the hardware performs the sigmoid and tanh activation of these equations to exactly mimic the computational complexity.

Fully Connected block mirrors the multilayer perceptron functionality where every input node is connected to all output nodes. In the hardware RTL, this block is implemented with ReLu activation logic for specifically addressed MAC operations.

The complete pipeline of the hardware architectures includes all these modules discussed above and is illustrated in Figure 3. The top state machine flow regulates the access of different memories. First, the convolution layer takes in M-bit feature and filter data and performs 1D or 2D convolution. Then, the output of the convolution is written into the convolution output memory which serves as the input memory of the maxpooling block to reduce the feature space size. Output from the maxpooling block in the vein of a reduced shape is written to the LSTM feature memory. With the help of the LSTM weights memory, the LSTM module performs all necessary matrix multiplications to generate output features and then saves these features in the LSTM feature memory again. Thus, the LSTM feature memory becomes the input feature map for the concurrent fully connected blocks and with weights data coming from a separate fully connected weights memory, it performs the relevant MAC operations and generates M-bit output. The bulk of the computation, in this case, is dedicated to the first fully connected layer operation. Hence, we reuse the LSTM feature memory to store the output values from the LSTM module itself. This allows the design to bypass the use of a separate memory to store LSTM output values and thus, the memory overhead is decreased.

TABLE I: Hardware implementation results and comparison with a previous CNN hardware architecture [9] for cough detection.

Architecture	[9]	This Work		
Application	Cough Detection	Cough Detection		
FPGA Platform	Artix-7 100t	Kintex-7 160t		
Input Dimension	44100×1	101×40		
Model Size (Kb)	960	933		
Computations (MOP)	75.2	0.51		
Fixed Point Precision	32-bit	32 bit		
Frequency (MHz)	47.6	95		
#PE used	8	2	4	8
Latency (ms)	1000	7.52	7.38	7.31
Total Power (mW)	211	284	287	290
Energy (mJ)	211	2.14	2.12	2.12
Performance (GOPS)	0.08	0.07	0.07	0.07
Efficiency (GOPS/W)	0.38	0.25	0.24	0.24

VI. FPGA Implementation and Results

Our hardware RTL was written in Verilog HDL and implemented on the Xilinx Kintex-7 160t device using the Xilinx Vivado 2018.3 tool for synthesis and implementation. The Kintex-7 160t device was chosen as the target device since it has 107 mW of static power dissipation which is suitable for low power embedded device implementation. Besides, the device also accommodates 325 BRAM slices of 36 Kb which totals to a memory capacity of 1462 KB. The hardware RTL includes parallelization only for the convolution layer to take care of the large feature input. Corresponding maxpooling, LSTM, and dense layers are implemented sequentially. The results in Table I indicate that as the number of processing engines increases, the total power increases, and the latency goes down. The latency reduction is a direct result of the parallelization of the convolution operation. However, the decrease in latency is not significant for this particular architecture as there is only one convolution layer and a major portion of the hardware computation is dedicated to the first fully connected layer only. With multiple layers of convolution and a small feature input to the fully connected layers, the latency numbers would scale logarithmically.

The CNN hardware implementation in [9] uses the same datasets to detect cough but with an end-to-end CNN structure. The hardware takes in raw audio waveforms as feature input in this case which restricts it to have very high latency. In this work, the raw input waveform is processed through mel-spectrogram to represent the short-term power spectrum of the input audios. This allows the input dimension to go down significantly which is almost 11× smaller than the input of [9]. As a result, our framework can run at twice the frequency with a latency improvement of almost 138×. Finally, our implementation consumes 2.12 mJ energy which is approximately 100× the energy consumed in [9]. Our hardware RTL performs at lower but moderate efficiency when compared to [9]. This stems from the fact that in this work our primary objective was to reduce computation to ensure low energy consumption and to utilize the LSTM module for time-space feature extraction.

VII. Conclusion

In this paper, to identify various respiratory symptoms, we proposed CoughNet, a flexible low power CNN-LSTM processor that can take audio recordings and detect cough sounds out of it. We evaluate and use three distinct publicly accessible databases to detect cough sounds as part of our experiment. The hardware prototype for CoughNet is also scalable and reconfigurable to accommodate different data bitwidth precisions and parallel processing engine numbers. The proposed implementation of hardware has a low power consumption of 290 mW and energy consumption of 2.12 mJ.

VIII. Acknowledgement

This research is based upon work supported by the National Science Foundation CAREER Award under Grant No. 1652703.

References

[1] Y. Tokozume and T. Harada, “Learning environmental sounds with end-to-end convolutional neural network,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2721–2725.

[2] H.-A. Rashid *et al.*, “A low-power lstm processor for multi-channel brain eeg artifact detection,” in *2020 21th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2020.

[3] A.Mazumder *et al.*, “An Energy-Efficient low power LSTM processor for human activity monitoring,” in *2020 IEEE 33rd International System-on-Chip Conference (SOCC 2020)*, 2020, in press.

[4] M. Hosseini *et al.*, “A fast method to fine-tune neural networks for the least energy consumption on fpgas,” in *Proceedings of the Hardware Aware Efficient Training workshop of ICLR 2021*, 2021.

[5] M. Khatwani, H.-A. Rashid *et al.*, “A flexible multichannel eeg artifact identification processor using depthwise-separable convolutional neural networks,” *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2020.

[6] S. Islam *et al.*, “Fire frontline monitoring by enabling uav-based virtual reality with adaptive imaging rate,” in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, 2019, pp. 368–372.

[7] M.Hosseini *et al.*, “Neural networks for pulmonary disease diagnosis using auditory and demographic information,” in *epiDAMIK 2020: 3rd epiDAMIK ACM SIGKDD International Workshop on Epidemiology meets Data Mining and Knowledge Discovery*. ACM, 2020, pp. 1–5, in press.

[8] J. Acharya and A. Basu, “Deep neural network for respiratory sound classification in wearable devices enabled by patient specific model tuning,” *IEEE transactions on biomedical circuits and systems*, vol. 14, no. 3, pp. 535–544, 2020.

[9] H.Ren *et al.*, “End-to-end scalable and low power multi-modal CNN for respiratory-related symptoms detection,” in *2020 IEEE 33rd International System-on-Chip Conference (SOCC 2020)*, 2020, in press.

[10] K. J. Piczak, “Esc: Dataset for environmental sound classification,” in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 1015–1018.

[11] E. Fonseca *et al.*, “General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline,” *arXiv preprint arXiv:1807.09902*, 2018.

[12] L. Orlandic, T. Teijeiro, and D. Atienza, “The coughvid crowdsourcing dataset: A corpus for the study of large-scale cough analysis algorithms,” *arXiv preprint arXiv:2009.11644*, 2020.

[13] K. J. Piczak, “Environmental sound classification with convolutional neural networks,” in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2015, pp. 1–6.

[14] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.