Cough detection using PCA and Deep Learning

Sunisa Khomsay
Intelligent Systems Research Units
National Electronics and
Computer Technology Center
Pathumthani, Thailand
sunisa.khomsay@nectec.or.th

Rangsarit Vanijjirattikhan
Intelligent Systems Research Units
National Electronics and
Computer Technology Center
Pathumthani, Thailand
rangsarit.vanijjirattikhan@nectec.or.th

Jittiwut Suwatthikul
Intelligent Systems Research Units
National Electronics and
Computer Technology Center
Pathumthani, Thailand
jittiwut.suwatthikul@nectec.or.th

***Abstract*—The paper presents a cough detection method by using Principal Component Analysis (PCA) and Deep Learning Networks (DLN) based on TensorFlow. PCA is used to perform feature extraction before sending the data to train a model by DLN. TensorFlow uses graph model to perform computations which are highly efficient in practice. Cough sounds from eight volunteers are collected for the experiment. The results show the cough signals can be efficiently detected by using DLN with PCA.**

***Index Terms*—Cough Detection, PCA, Deep Learning, TensorFlow.**

I. Introduction

Cough is a body's response mechanism to abnormalities in the respiratory system. It is an important protective mechanism of the body to detect and eliminate pathogens, mucus or foreign matters. The cough is the earliest symptom in almost all of the respiratory diseases. Therefore, the method for early detection of cough sounds can be a useful tool to respiratory disease screening [1], [2].

Neural networks can be defined as a model of reasoning based on the human brain which is a computing system characterized by a learning ability. The main advantage is that neural networks do not require a user to specify any problem solving algorithm but instead it learns from examples like human beings. Deep learning networks (DLN) extend the traditional neural networks by adding more hidden layers to the network architecture between the input and output layers to model more complex and nonlinear relationships. The concepts have gained researchers' interest in recent years for its good performance to become the best solution in many problems in machine learning [3]–[6].

TensorFlow developed by Google is one of the most popular deep learning libraries, that is a platform to build the machine learning model [7]. It implements mathematical model by using data flow graphs in which each node in the graph represents a mathematical operation. TensorFlow is an open source software library which does not limit developers to a specific application.

This paper presents an application of Principal Component Analysis (PCA) and DLN by using TensorFlow library for detecting cough sounds. The signal spectra are generated by using Fast Fourier Transform (FFT) technique. PCA method extracts signal features and then selects a number of most important feature components to reduce data dimensions before sending them to DLN. The experimental results show that PCA+DLN provide better performance than DLN in terms of time and accuracy, and different types of cough sounds can be efficiently detected by the trained PCA+DLN model.

II. Methodology

A. Principal Component Analysis

PCA is a useful and common statistical technique that has been found in many application fields such as finding patterns in data of high dimension. The aim of PCA is to explain the interdependence of a large number of variables by a smaller number of fundamental variables [8]. Considering a set of training data (A) size $M \times N$ matrix, the mean (μ) of variable A is defined by

$$\mu = \frac{\sum_{i=1}^{M} A_i}{M} \tag{1}$$

where $1 \leq i \leq M$, A_i is one row of A. The distance vector is as follows,

$$x_i = A_i - \mu \tag{2}$$

The covariance matrix C_X is defined by

$$C_X = XX^T \tag{3}$$

where matrix $X = [x_1, x_2, ..., x_M]$. The covariance matrix is solved by calculating the eigenvalues (Λ) and eigenvectors (λ) as follows

$$C_X U = U\Lambda \tag{4}$$

$$\Lambda = diag[\lambda_1, \lambda_1, ..., \lambda_M] \tag{5}$$

where Λ is a diagonal matrix defined by the eigenvalues of matrix C_X and U is the associated eigenvectors of λ. The eigenvectors represent the directions of the PCA space, and the corresponding eigenvalues represent the scaling factor, length, magnitude, or the robustness of the eigenvectors [9], [10].

B. Deep Learning Networks

1) Network Architecture: Deep learning is a subfield of machine learning. It is a typical feedforward network in which the inputs flow from the input layer to the output layer through a number of hidden layers. The structure of DLN is illustrated in Fig. 1

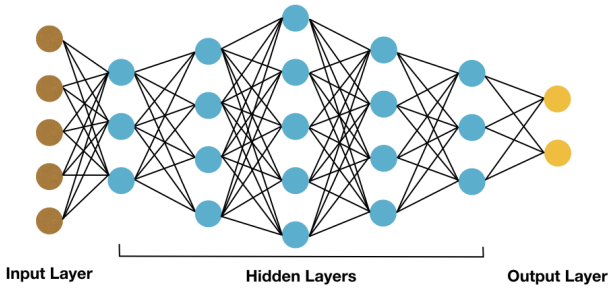


Fig. 1. The structure of DLN.

The DLN structure consists of several sets of neurons connected together. Typically, the network consists of an input layer, hidden layers and an output layer. Each layer has its own specific function. The input layer accepts input signals from the outside world and redistributes these signals to all neurons in the hidden layers. The output layer accepts output signals from the hidden layers and establishes the output pattern of the entire network [11]. Output of a neuron is calculated by,

$$y = f(n) \tag{6}$$

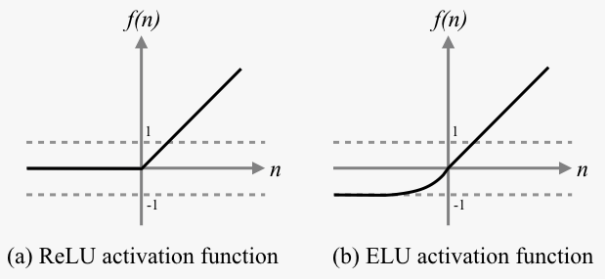
where n is the net weighted input that is computed as follows,

$$n = \sum_{i=1}^{m} p_i w_i + b \tag{7}$$

where m is the number of inputs p, weight and bias of neuron are w and b respectively. These input values are passed through an activation function.

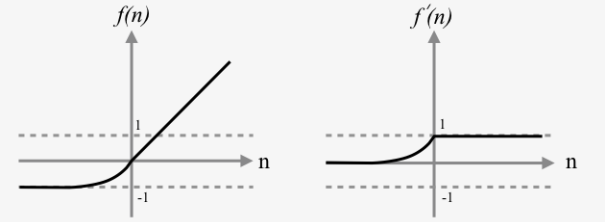
2) Activation Function: An activation function is an extremely important feature of the neural networks. This function is used for transforming activation level to output signals. One activation function is associated with one neural unit.

Recently, a new activation function named Exponential Linear Unit or ELU has been introduced, which speeds up learning on deep neural networks and leads to higher accuracy [12]. ELU is another type of activation function based on Rectified Linear Unit (ReLU). Both functions are similar except the negative part of the inputs as seen in Fig. 2. The positive parts of both functions are constant gradient. This enables learning on that side not to be saturated. The negative part of ELU slowly goes to α whereas ReLU is zero for $n \leq 0$. ELU rectifies the vanishing gradient problem and speeds up the learning. ELU activation graph is shown in Fig. 3.



(a) ReLU activation function (b) ELU activation function

Fig. 2. ReLU and ELU functions.



(a) *f(n)* in forward pass (b) *f′(n)* in backward gradients

Fig. 3. ELU function.

ELU function helps cost function converge to zero faster and produces more accurate results. The equation can be computed as,

$$f(n) = \begin{cases} n & \text{if} \quad n > 0 \\ \alpha(e^n - 1) & \text{if} \quad n \leq 0 \end{cases} \tag{8}$$

$$f'(n) = \begin{cases} 1 & \text{if} \quad n > 0 \\ f(n) + \alpha & \text{if} \quad n \leq 0 \end{cases} \tag{9}$$

where α is a constant value initialized to 1. $f(n)$ is the forward pass and its derivative pass is $f'(n)$ for calculating its backward gradient.

C. TensorFlow

TensorFlow is a deep learning library developed by Google that implements a model by using data flow graph. The basic units consist of nodes which are mathematical operations, and edges which represent tensors (multi-dimensional arrays). TensorFlow uses sessions for graph computation. The outputs are calculated by inputs that are passed from one node to another node. The data flow graph of linear regression model is illustrated in Fig.4. The model is a relatively simplistic type of mathematical method that, when used properly, can help predict model behavior.

TensorFlow implements model by the graph that is uncomplicated and high performance. GPU support is available for training purposes for faster development [13]. During a long running training process, TensorFlow can save the state of variable of a computation graph to disk and restore the model from checkpoint. Tensor has an ability to visualize the computation graph itself by using TensorBoard, a web

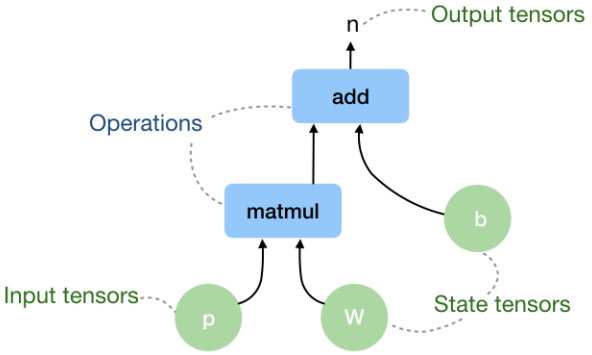


Fig. 4. Data flow graph of linear regression model.

interface that helps visualize scalars, histograms, audio and distribution of tensors.

III. Experiment

A. Data Acquisition

The experiment data was obtained from eight volunteers. The cough sounds from these volunteers were labeled as productive cough and non-productive cough. These labels were used for training and testing processes. The recording system consists of Raspberry Pi and a microphone recording at 44.1 kHz sampling frequency. The details of the volunteers are shown in Table I.

TABLE I
VOLUNTEERS DETAILS

No	Age	Sex	Label
1	88	F	non-productive cough
2	34	F	non-productive cough
3	42	M	non-productive cough
4	45	F	non-productive cough
5	43	F	non-productive cough
6	39	M	non-productive cough
7	41	F	productive cough
8	45	M	productive cough

The total dataset consists of 810 events: 229 non-productive cough, 74 productive cough and 507 other sounds (speech, knock and other ambient sounds). Labels 0, 1 and 2 represent ambient sound, non-productive cough and productive cough, respectively. Examples of the cough sounds are shown in Fig. 5 and Fig. 6. The signal characteristics between productive cough and non-productive cough are different.

B. System Structure

The system structure for cough detection is summarized in Fig. 7. The acoustic signals measured by the microphone are processed through FFT and separated into training and testing datasets. DLN is trained by the training dataset in order to get model which is then used in the detection process. The total datasets are randomized and divided into 2 groups. 70% of them are used in the training process. The rest are used for testing performance of the system. In this experiment, training and testing performances of DLN and PCA+DLN methods are evaluated. For PCA+DLN, the datasets are extracted by using PCA technique to reduce dimensions from 1024 to 100 before sending to DLN. The main workflow of the training process is illustrated in Fig. 8. The training is computed by using a momentum optimizer, with a learning rate of 0.1 and a momentum parameter of 0.8. The layer architecture consists of 3 hidden layers and 1 output layer. The optimal hidden layer nodes are 80, 20 and 10, respectively. ELU activation function is used in all layers.

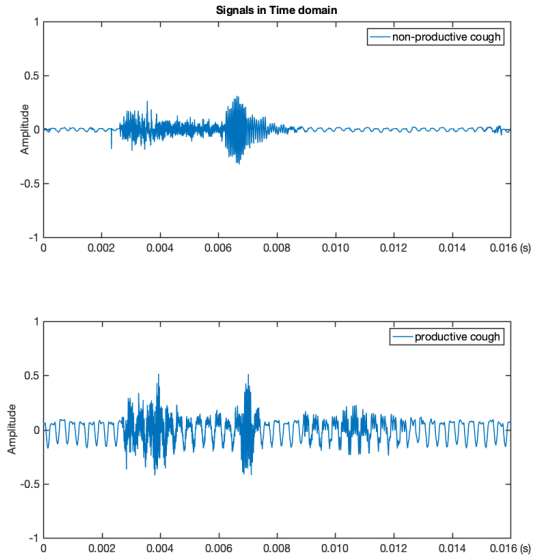


Fig. 5. Cough signals in time domain

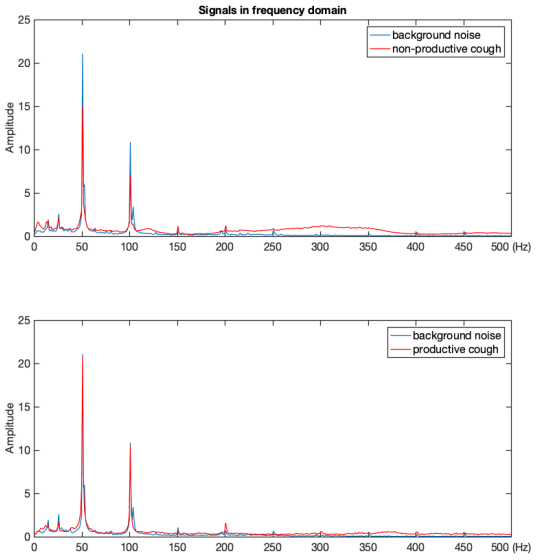


Fig. 6. Cough signals in frequency domain.

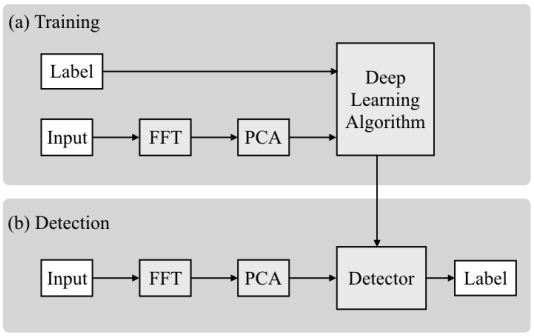


Fig. 7. Detection method using deep learning.

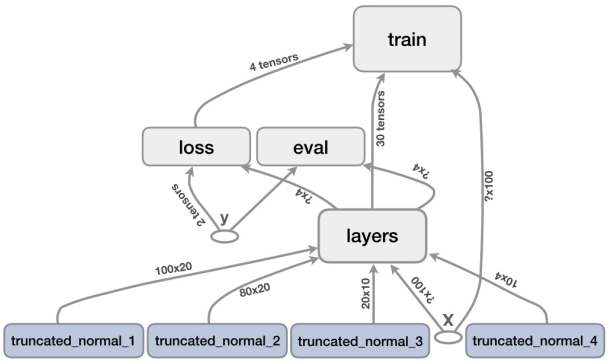


Fig. 8. Data flow graph of TensorFlow.

C. Experiment Results

Fig. 9 shows a training performance comparison between DLN and PCA+DLN models. DLN took about 50 seconds to reach accuracy of 0.983 while PCA+DLN took about 15 seconds to achieve accuracy of 0.989. The results show that the PCA+DLN model performed better than the DLN model in terms of time usage and accuracy. Convergence curves of DLN and PCA+DLN models are illustrated in Fig. 10 where the MSE of PCA+DLN cost function goes down more faster than that of DLN.

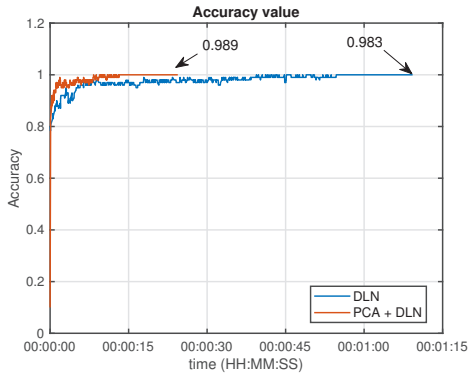


Fig. 9. Accuracy of the models.

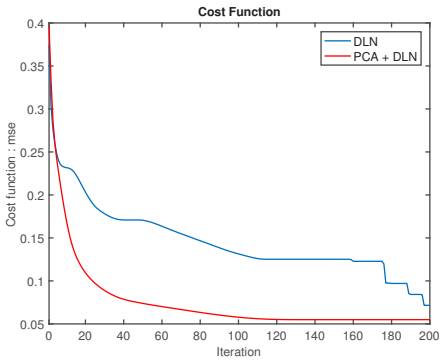


Fig. 10. Convergence curves.

The training and testing accuracy of DLN and PCA+DLN are illustrated by confusion graphs in Fig 11 and Fig 12. The performance of PCA+DLN is almost 99.91% while DLN is about 98.45%. Table II and Table III summarize the accuracy, errors and time usage of the training and testing processes.

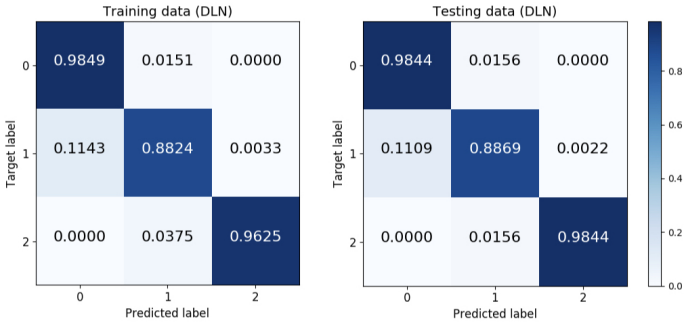


Fig. 11. Confusion graph of DLN algorithm.

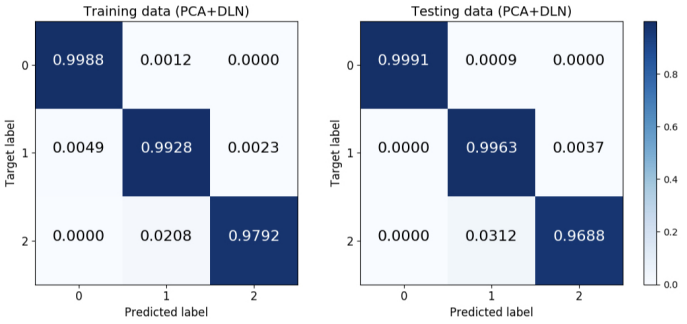


Fig. 12. Confusion graph of PCA+DLN algorithm.

Fig 13 shows the test results of cough detection for continuous data in a short duration. For the ambient sound at the top, the results show performance of 100% accuracy. In the middle, the system identifies all non-productive cough events correctly. For productive cough mixed with some ambient sound at the bottom, 1 false alarm is detected, resulting in the total performance of 94.4%.

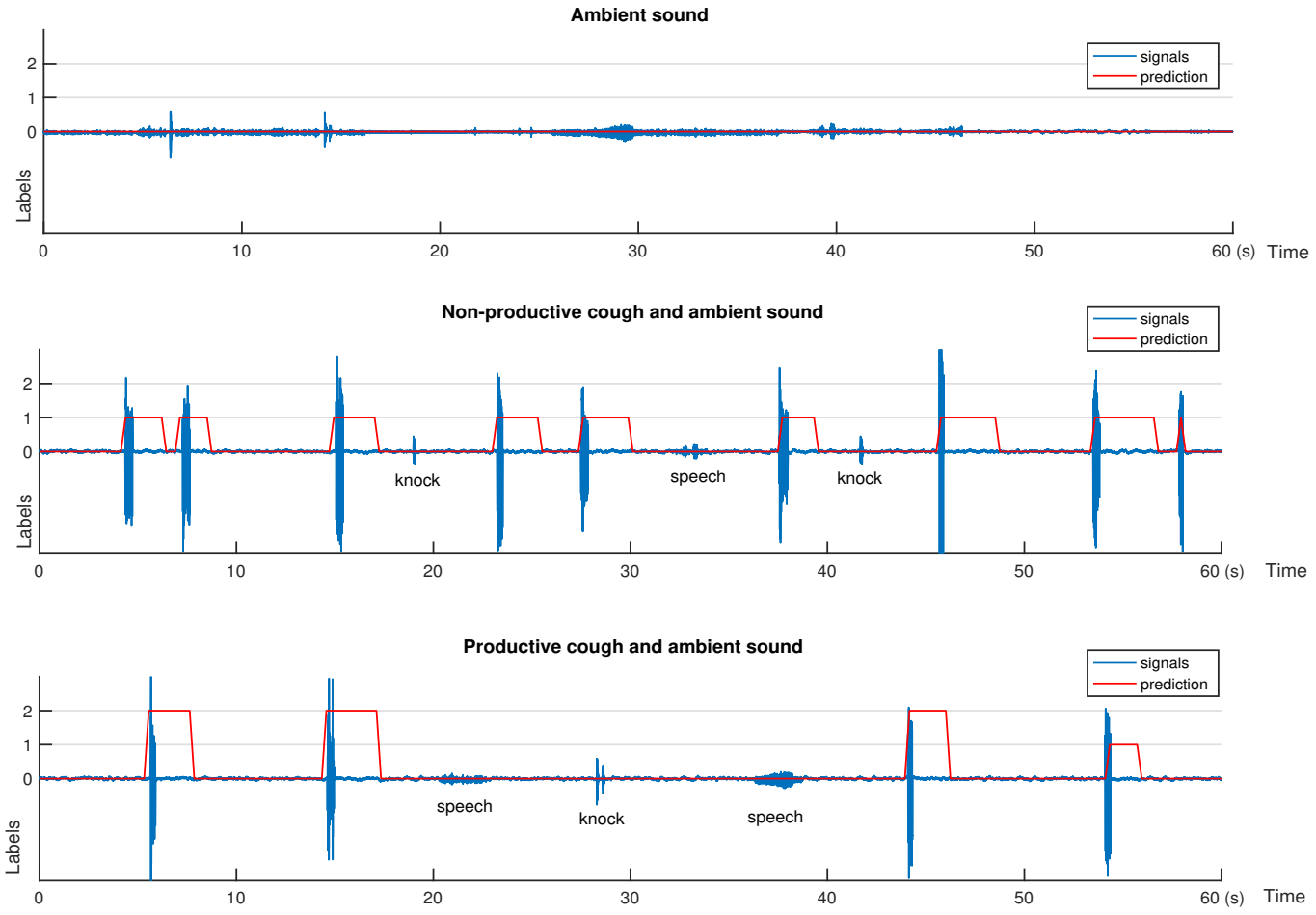


Fig. 13. The detection results of continuous data for a short duration.

TABLE II
PERFORMANCE OF TRAINING WITH DIFFERENCE ALGORITHMS

Method	Accuracy (%)			Error (total)	Time (s)
	Ambient	*Non-productive cough*	*Productive cough*		
DLN	98.45	88.24	96.25	1.62	54.10
PCA+DLN	99.88	99.28	97.92	1.07	24.81

TABLE III
PERFORMANCE OF TESTING WITH DIFFERENCE ALGORITHMS

Method	Accuracy (%)			Error (total)
	Ambient	*Non-productive cough*	*Productive cough*	
DLN	98.44	88.69	98.44	1.65
PCA+DLN	99.91	99.63	96.88	1.19

IV. CONCLUSION

The paper has proposed cough detection by using PCA and DLN via TensorFlow. This method focuses on feature extraction of cough signals in frequency domain by using PCA and model training for signal classification by using DLN. The model is trained with cough datasets collected from eight volunteers. The results show that the PCA+DLN model performed better than the DLN model in terms of time usage and accuracy. The system provides good detection accuracy overall. Future work is to implement and test this method on an elderly care robot to be able to detect and record cough history. This can be useful for doctors to have accurate patient cough data recorded at home.

REFERENCES

[1] S. Ranjani, V. Santhiya and A. Jayapreetha, "A Real time Cough Monitor for Classification of Various Pulmonary Diseases", *Third International Conference on Emerging Applications of Information Technology*, pp. 102–105, 2012.

[2] S. Vinayak, A. Udantha, A. Yusuf, H. Craig, T. Rina and S. Amalia, "Neural network based algorithm for automatic identification of cough sounds", *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1764–1767, 2013.

[3] H. Mohsen, E.A. E1-Dahshan, E.M. E1-Horbaty and A.M. Salem, "Classification using deep learning neural networks for brain tumors", *Future Computing and Information Journal 3*, pp. 68–71, 2018.

[4] J. Amoh and K. Odame, "Deep Neural Networks for Identifying Cough Sound", *IEEE Transactions on Biomedical Circuits and Systems*, Vol. 10, No. 5, pp. 1003–1011, 2016.

[5] P.A. Rao, B.N. Kumar, S. Cadabam, P.T. Jaikanthan, "Distributed Deep Reinforcement Learning using TensorFlow", *International Conference on Current Trends in Computer, Electrical, Electronics and Communication*, pp. 171–174, 2017.

[6] V. Swarnkar, U.R. Abeyratne, Y. Amrulloh, C. Hukins, R. Triasih, A. Setyati, "Neural Network based algorithm for automatic identification of cough sounds", *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1764–1767, 2013.

[7] M. Abadi, P. Barham, J. Chen, X. Zheng, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P. Tucker, V. Vasudeven, P. Warden, M. Wicke, Y. Yu and X. Zheng, "TensorFlow: a system for large scale machine learning", *Operating System Design and Implementation*, pp. 265-283, 2016.

[8] M. Z. Susac, N. Sarlija and S.Pfeifer, "Combining PCA Analysis and Artificial Neural networks in Modelling Entrepreneurial Intentions of Students", *Croatian Operational Research Review*, pp. 306–317, 2013.

[9] R. Sekhar, A. Kaul, R. Nath, A.S. Arora and S. Chauhan, "Comparison of PCA and 2D-PCA on Indian Faces", *International Conference on Signal Propagation and Computer Technology*, pp. 561–566, 2014.

[10] H. M. Ebied, "Feature Extraction Using PCA and Kernel-PCA for Face Recognition", *International Conference on INFOrmatics and Systems*, pp. 74–80, 2012.

[11] M. Negnevitsky, *Artificial Intelligence A Guide to Intelligent systems*, Addison Wesley, England, 2005.

[12] D.A. Clevert, T. Unterthiner and S. Hochreiter, "Fast and accuracy deep network learning by Exponential Linear Units (ELUs)", *International Conference for Learning Representations*, pp. 1–14, 2016.

[13] Y.J. Mo, J. Kim, J.K Kim, A. Mohaisen and W. Lee, "Performance of Deep Learning Computation with TensorFlow Software Library in GPU-Capable Muli-Core Computing Platforms", *International Conference on Ubiquitous and Future Networks*, pp. 240–242, 2017.