# On Speech Recognition Algorithms

Shaun V. Ault, Rene J. Perez, Chloe A. Kimble, and Jin Wang

***Abstract*—We use speech recognition algorithms daily with our phones, computers, home assistants, and more. Each of these systems use algorithms to convert the sound waves into useful data for processing which is [then interpreted by the machine. Some of these machines use older algorithms while the newer systems use neural networks to interpret this data. These systems then produce an output generated in the form of text to be used. A large amount of training data is needed to make these algorithms and neural networks function effectively.**

***Index Terms*—Speech recognition, neural network, Markov model, Gaussian mixture model, principal component analysis, K-means clustering, EM algorithm, maximum likelihood, MIXFIT.**

## I. Introduction

Speech recognition is a growing area and is engulfing the technological field. With so many new technologies ran by voice recognition (Google Home, Alexa, Siri), and many others benefitting from the rapid advancements in the field of talk-to-text (call centers, cell phones) it's no secret that speech recognition software isn't going anywhere. The truth is, almost all humans can talk faster than they can type, which is convenient, and convenience is the driving force of innovation.

Speech recognition software isn't hard to understand from a user's standpoint, for the sake of this example, let's say you're using talk-to-text on your phone. First you open the text message, and click on the microphone, you talk to your phone and can watch it type your words as you same them. The question is, however, how does your phone know what words you are saying?

When your phone accepts your speech as input, it must be translated to data first. This is done by looking at the waves of your voice, and making evenly spaced points along the wave, which will then be converted into data that will be fed into our algorithms. Since our speech recognition software includes machine learning, these machines will also learn what you are most likely to say next, the same way your phone will give you suggested words after you start typing a sentence, these machines will also learn the way you talk and the way you say certain words, not just what words to expect next.

## II. Speech Recognition Algorithms

### A. Hidden Markov Model

There are several methods for capturing speech and converting it into text. One of the more effective methods is the Hidden Markov Model (HMM) with an end point detection algorithm for pre-processing to remove unwanted noise. The HMM requires the addition of other tools to correctly interpret speech. Before the HMM method is applied to detect the speech, "the speech samples are extracted to features or coefficients by the use of Mel Frequency Cepstral Coefficient (MFCC)" [1]. This process separates the audio clip into small segments that the HMM can use as an input. The HMM uses a dictionary of word pronunciations to determine the word $W$ by choosing the maximum probability $P(W|Y)$ where $Y$ is the audio clip recorded. This maximum probability is computed using the probability $P(W)$ which is calculated using a language model. There is an HMM for each basic pronunciation in the dictionary. $P(Y|W)$ is the expected acoustic data when $W$ is constructed from the stored HMMs in the dictionary [2]. This data is used to determine the maximum probability by comparing it to the $P(W)$s that were calculated using the pronunciation dictionary. By using this technique, the HMM is able to produce the text from the given speech.

We briefly discuss the statistical framework here. Current speech recognition systems are based on the principles of statistical pattern recognition. An input speech waveform is converted by a front-end signal processor into a sequence of acoustic vectors,

$$Y = y_1, y_2, \cdots, y_T.$$

Each of these vectors is a compact representation of the short time speech spectrum covering a time period. The dictionary database consists of a sequence of words,

$$W = w_1, w_2, \cdots, w_n.$$

The speech recognition system is used to select the best word sequence $\widehat{W}$ for given the observed acoustic signal $Y$. Bayes' rule is used to implement this idea,

$$\widehat{W} = \arg\max_{W} \frac{P(W)P(Y|W)}{P(Y)}$$

$$= \arg\max_{W} P(W)P(Y|W).$$

This is an optimization problem. In order to find the best word sequence $W$ to match the observation $Y$, this equation indicates that $W$ needs to be selected to maximize the product of $P(W)$ and $P(Y|W)$. Here $P(Y)$ is fixed. In Bayes philosophy, $P(W)$ is a priori probability of observing $W$ which is independent of the observed signal $Y$. This probability is determined by a language model. $P(Y|W)$ is the probability of observing the vector sequence $Y$ given some specified word sequence $W$. This probability is determined by an acoustic model.

### B. Gaussian Mixture Model

The Gaussian Mixture Model (GMM) is one of the more

Manuscript received September 3, 2018; revised October 17, 2018.

The authors are with the Valdosta State University, Valdosta, GA 31698 USA (e-mail: svault@valdosta.edu, reperez@valdosta.edu,

doi: 10.18178/ijmlc.2018.8.6.739

commonly used functions for the HMM and is a "parametric probability density function represented as a weighted sum of Gaussian component densities". The HMM uses the Gaussian component parameters as the base classifier and acquires the temporal variations while the GMM captures the special variations. This allows it to efficiently and effectively handle time-sequences. Basically, a sequence of GMMs are used to analyze the input data for the HMM which gives it the sensitivity to temporal changes. The Gaussian Mixture model does fall short in the sense that it "prevents an HMM from taking the full advantage of the correlation that exists among the frames of a phonetic segment" because it assumes a conditional independence [3]. Another drawback is that because it uses probability to determine the input, it can incorrectly choose the more common choice.

GMMs are widely used as probability distribution features, such as vocal-tract related spectral features in a speaker recognition or emotion recognition systems. GMMs having advantage that are more appropriate and efficient for speech emotion recognition using spectral feature of speech [4].

GMM is a powerful method to estimate the density of the acoustic vector $Y$. Recently in the cluster analysis, the mixture model-based approach has been widely adopted. This model is more flexible, accurate, and easy to fit into the given multivariate data set.

In computation, it is very easy to handle. The conditional probability $P(Y|W)$ requires the estimated density of $Y$ for its computation. Therefore the HMM can be implemented to find the best word sequence $\widehat{W}$ to match the given the observed acoustic signal $Y$. Recall that,

$$Y = y_1, y_2, \cdots, y_T.$$

For every vector

$$y_i = \begin{bmatrix} y_{i1} \\ \vdots \\ y_{id} \end{bmatrix}$$

is a vector of length $d$. Based on $Y$, the acoustic signal models are fitted as the follows:

The multivariate Gaussian density function [5] is given by

$$f(y;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} e^{-\frac{1}{2}(y-\mu)^T \Sigma^{-1}(y-\mu)},$$

where $\mu$ is the mean vector and Σ is the covariance matrix. A Gaussian mixture density is a weighted sum of $k$ Gaussian densities $f(y;\mu_i,\Sigma_i)$,

$$f(y,\mu_1,\cdots,\mu_k,\Sigma_1,\cdots,\Sigma_k) = \sum_{i=1}^{k} p_i f(y;\mu_i,\Sigma_i),$$

where the weights $p_i \geq 0$ and

$$\sum_{i=1}^{k} p_i = 1.$$

The GMM modeling parameters are specified by

$$\lambda = (p_i, \mu_i, \Sigma_i)_{i=1,\cdots,k}.$$

$k$ is the number of mixture components. The GMM parameters can be iteratively estimated using the expectation maximization (EM) algorithm [6].

A practical issue here is the covariance matrix estimators. In theory, covariance matrix is positive definite and invertible. For the given data set, some statistical methods may provide a noninvertible matrix estimator. There are many discussions regarding this issue in statistics community [7].

### C. K-Means Clustering Algorithm

In the Gaussian Mixture Model, the number of mixture components $k$ has to be estimated through optimization method. Then we use *K*-means clustering algorithm to partition all data points into $k$ clusters. Each data point belongs to one and only one cluster with the nearest mean in Euclidean distance. Here is the logic of the algorithm:

1) For each data point, the nearest cluster center in is identified;
2) Each cluster center is replaced by the average of all data points that are closed to it.
3) Steps 1 and 2 are alternated until convergence. Algorithm convergence to a local minimum.

Two practical issues need to be addressed here. First, how to select the right number $k$ of clusters to use is often not obvious. Hamerly and Elkan [8] present an algorithm for learning $k$ while clustering. One simple rule of thumb is to use the square root of half data size as the number $k$. There are some discussions for the optimal clustering, including merging clusters and dividing a cluster into two or more clusters. In practice, around 10 mixture components will provide good performance in speech recognition system [2].

The second issue is the $k$ initial cluster centers. Typically, $k$ data points are randomly selected as the staring guesses in practice.

### D. EM Algorithm

The expectation-maximization (EM) algorithm is an iterated method for finding maximum likelihood estimates of parameters in statistical models. The convergence analysis has been done by Dempster-Laird-Rubin [6].

Multivariate data is used fit the Gaussian mixture models via this EM algorithm. This algorithm is powerful and robust. However it does require the specification of an initial estimate of these unknown parameters with respect to the components of the mixture model. We proposes the following algorithm to derive an initial estimate of unknown parameters:

1) Using the *K*-means algorithm to classify all data point into $k$ clusters;
2) Within each cluster, estimating its density mean and covariance matrix from its own cluster data points via the sample mean and sample covariance methods;
3) Using the cluster data size proportions to estimate the mixing proportions of the normal mixture model.

Here is the description of the EM algorithm ([6] and [9]). Given the observed data $Y$, a set of unobserved latent data or missing values $Z$, and an unknown parameter $\theta = (\mu, \Sigma)$ with a likelihood function $L(\theta;Y,Z) = p(Y,Z,|\theta)$. The maximum likelihood estimate (MLE) of the unknown parameters is determined by maximizing the marginal likelihood of the observed data,

$$L(\theta;Y) = p(Y|\theta) = \int p(Y,Z|\theta)dZ.$$

The EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying these two steps:

1) Expectation step (E step): Define $Q(\theta|\theta^{(t)})$ as the expected value of the log likelihood function of $\theta$, with respect to the current conditional distribution of $Z$ given $Y$ and the current estimates of the parameters $\theta^{(t)}$:

$$Q(\theta|\theta^{(t)}) = E_{Z|Y,\theta^{(t)}}[\log L(\theta; Y, Z)].$$

2) Maximization step (M step): Find the parameters that maximize this quantity:

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta|\theta^{(t)}).$$

With the maximum likelihood approach to the estimation of $\theta$, an estimate is provided by an appropriate root of the likelihood equation,

$$\frac{\partial \log L(\theta; Y, Z)}{\partial \theta} = 0.$$

Under mild conditions [10], EM converges to a maximum of $L(\theta; Y)$. The key to the efficient implementation of this algorithm is the choice of an observed/missing data structure, such that the E and M steps have simple closed-form expressions.

The main difficulties in using EM for mixture model fitting are: its critical dependence on initialization; the possibility of convergence to a point on the boundary of the parameter space with unbounded likelihood (i.e., one of the m parameters approaching zero with the corresponding covariance becoming arbitrarily close to singular) [11]. If the covariance is close to singular, a non-singular robust covariance matrix estimate is proposed by [12] to solve this popular serious problem.

MIXFIT [13] is a software program for the automatic fitting and testing of Gaussian mixture models via the EM algorithm. It is a very good option, especially for practitioners who have little statistics background. The EM algorithm does provide robust covariance matrix estimators. These matrixes should be positive definite, namely invertible. In big data mining, the issue of the matrix not being invertible does not exist here. In theory, there are more than enough data points to guarantee all covariance matrix estimators are (close enough) positive definite.

Another major option of the MIXFIT algorithm allows the user to automatically carry out a test for the smallest number of components compatible with the data.

As decribed in [13], "The MIXFIT algorithm automatically provides starting values for the application of the EM algorithm if the user does not provide any by considering a selection obtained from three sources: (a) random starts, (b) hierarchical clustering-based starts, and (c) k-means clustering based starts. Concerning (b), the user has the option of using in either standardized or unstandardized form, the results from seven hierarchical methods (nearest neighbor, farthest neighbor, group average, median, centroid, flexible sorting, and Ward's method). There are several algorithm parameters that the user can optionally specify; alternatively, default values are used. The program fits the normal mixture model for each of the initial grouping specified from the three sources (a) to (c). All these computations are automatically carried out by the program. The user only has to provide the data set the restrictions on the component covariance matrices (equal, unequal, or diagonal), the extent of the selection of the initial groupings to be used to determine starting values, and the number of components that are to be fitted. Summary information is automatically given as output for the final fit. However, it is not suggested that the clustering of a data set should be based solely on a single solution of the likelihood equation, but rather on the various solutions considered collectively. The default final fit is taken to be the one corresponding to the largest of the local maxima located. However, the summary information can be recovered for any distinct fit."

### *E. Kernel Principal Component Analysis*

We first review the standard Principal Component (PCA) algorithm ([14]-[16]) It is an efficient technique for dimensionality reduction with applications in image compression and face recognition. It represents a linear transformation where the data is expressed in a new coordinate basis that corresponds to the priority order of the maximum variance direction. It is readily performed by solving an eigenvalue problem. PCA is an orthogonal transformation of the coordinate system, so that we can best to represent the data set. The new coordinate values (arises) are called principal components.

PCA can be calculated from the covariance matrix. Let $F = (f_1, \ldots, f_N)$ be the $M \times N$ data matrix containing the entire data. The objective is to find $M$ orthogonal vectors $u_j$ (these are the principal components) that best describe the variability in the data.

Define $u = (u_1, u_2, \ldots, u_M)'$ and

$$\frac{1}{N}\sum_{i=1}^{N}(u_j \cdot f_i)^2 = \ldots = u_j'(\frac{1}{N}F * F')u_j$$

Here $C = \frac{1}{N}F * F'$ is the covariance matrix. It is positive semi-definite,

$$x'Cx = \ldots = \frac{1}{N}\sum_{j=1}^{M}\sum_{i=1}^{n}(x_j F_{ji})^2 \geq 0$$

and this implies eigenvalues are non-negative. The constrained optimization problem of maximizing:

$$\sum_{i=1}^{N}(u_j \cdot f_i)^2$$

and requiring $u_j$ is solved using Lagrange multipliers by maximization:

$$u_j'Cu_j + \lambda(1 - u_j'u_2)^2$$

The optimal vectors are eigenvectors of the covariance matrix,

$$Cu_j = \lambda u_j$$

Because the covariance matrix is symmetric, the eigenvectors $u_j$ are orthogonal. The PCA result is

$$F_{M\times N} = U_{M\times M} * V_{M\times N}$$

where $V_M = U'_{M\times M} * F_{M\times N}$. The data set can be reconstructed. If some of the PCs are judged not important, we therefore wish to reconstruct the data using only $k < M$ PCs,

$$F_{\text{reconstructed}} = U_{M\times k} * V_{k\times N}$$

The fraction of the variance explained by the $i^{\text{th}}$ PC $u_i$ is $\frac{\lambda_i}{\sum_{j=1}^{M}\lambda_j}$.

Originally in 1901, British mathematician Karl Pearson introduced principal components to use a low-dimensional summary to best describe a high- dimensional dataset [17]. Recently this idea has been widely applied to the Big Data Analysis.

However we are more interested in its application of speck recognition rather than the reduction of data dimensionality. Kernel principal component analysis (KPCA) is a generated PCA algorithm. It is an efficient method for feature extraction in speech recognition systems. KPCA technique is the result of applying the kernel function to PCA in order to obtain the representation of PCA in a higher dimensional space which can possibly generate more distinguishable speech features [18].

## III. Neural Networks

### A. Brief Introduction to Neural Networks

Neural network is mathematical model which is used to perform a particular function. It works like a human brain and involves using different estimation functions (functions based on probabilities) to calculate probable answers (estimates), these functions are then called neurons [19]. Next, we will use these neurons in another function with different probabilities that will model our functions by linking all the neurons together to make a web, or network [19]. This makes a foundation of building on other outputs to produce a better, more average output. Neural networks are the method of machines learning to do something based of training examples [20]. For example, to learn what the word "Hello" sounds like, the machine will be given training data of many different people using different accents, pitches, varying levels of background noise, and other variable sound/word influencers to help the machine to extrapolate the foundation of the word.

As discussed in [21], computers are trained through machine learning algorithms to perform tasks by their own. Networks are trained to form association between input layer and output layer with certain weights. It trains the network how to classify speech patterns into phoneme categories. Networks learn the underlying patterns and generalize from the training data to new examples. This is essential in speech recognition. Speech is a highly nonlinear process. Networks can compute any nonlinear functions of their input, enabling them to perform arbitrarily complex transformations of data. Networks offer a uniform computational paradigm which can easily integrate constraints from different types of inputs. This makes it easy to use both basic and differential speech inputs.

### B. Deep Neural Network

By definition, Deep learning, sometimes referred as representation learning or unsupervised feature learning, is a new area of machine learning [22]. Deep Neural Networks (DNNs) come from the recent revolution of the neural networks used in the gaming industry. "The complex imagery and rapid pace of today's video games require hardware that can keep up, and the result has been the graphics processing unit (GPU), which packs thousands of relatively simple processing cores on a single chip," which was found to be very similar to a neural net [20].

As pointed by [22], HMM speech recognition systems can be trained automatically and are simple to use. In comparison, one of the main drawbacks of GMMs is that they are statistically inefficient for modeling data. Neural networks trained by back-propagation error derivatives emerged as an attractive acoustic modeling approach for speech recognition. They make no assumptions about feature statistical properties. When used to estimate the probabilities of a speech feature segment, neural networks allow discriminative training in a natural and efficient manner. The new term "deep" learning comes from the depth of the network's layer enabled by the modern GPUs [20]. Because of the effectiveness of Deep Learning, it "is becoming a mainstream technology for speech recognition" and has become the replacement for the Gaussian mixture model [22]. Using DNNs effectively has been attempted for years but recently, with the help of several breakthroughs, it has become possible [23]. The reduction in costs and increase in computing power has made it reasonable to train these DNNs through brute force using large datasets [23]. Another reason that the concept of DNNs has increased in popularity is that there is much more data that has been collected to teach these systems and improved algorithms designed for large amounts of data are better able to train the DNNs.

### C. Deep Belief Network

Deep Belief Networks (DBNs) are neural networks consisting of a stack of restricted Boltzmann machine (RBM) layers that are trained one at a time, in an unsupervised fashion to induce increasingly abstract representations of the inputs in subsequent layers [22]. The design of a DLN is basically a stack of RBM where each machine has two layers to it [24]. Each node contains "one layer of (typically Bernoulli) stochastic hidden units and one layer of (typically Bernoulli or Gaussian) stochastic visible units," such that each hidden layer is connected to every visible layer and each visible layer is connected to every hidden layer [24]. However, no layer has connections within itself. This layered structure uses a greedy learning strategy called unsupervised pre-training [24]. This strategy can theoretically "improve a variation lower bound on the likelihood of the training data under the composite model" [24]. Once the unsupervised training has been completed, the supervised training begins which adjusts "resulting weights using gradient descent learning to improve the performances of DBN" [24]. The overall purpose of pre-training, which is unsupervised in the DBN, is to initialize the weights in the network to have a slightly better approximation than to have a random initialization [24]. This method helps "[avoid] local minima when using supervised gradient descent" [24].

### D. Recurrent Neural Network

To make neural networks even more effective, we can add states. The basic idea of Recurrent Neural Networks is to make the neural network save its state every time it

encounters input [22]. Recurrent neural networks (RNNs) are a powerful model for sequential data. This will benefit the network in its abilities to make predictions based off previous input [22]. In speech recognition algorithms, this will help the machine to predict what the user will say next based on what the user has said previously. For example, let's say the primary user of an iPhone uses Siri to text their Dad when they are on their way home from school every day. When the user says, "Siri text Dad I am on my --," Siri will automatically know the rest of the sentence is "way home" because it has this saved as a state and because this is what the user usually says, that will be the most reasonable assumption of what the user will say this time, given the start of the sentence was the same as usual.

### E. Time Delay Neural Network

Using the Time Delay Neural Network (TDNN) for speech recognition provides a better result for "longer temporal context than the classical DNN" [25]. With the multi-splice sub-sampling architecture that is uses for its speech recognition, the different layers can model "the wide term temporal context dependencies at a different resolution[s]" [25]. The deeper layers focus on processing the "hidden activations from a wider temporal context" while the first layer focuses of the narrower context [25]. This sub-sampling method which the TDNN is composed of greatly reduces the time spent for model training by requiring less computation to achieve the same result as the classical DNN. These advantages make it a more desirable method for automatic speech recognition than the other methods discussed previously.

## IV. Current Implementations

### A. Google Home

The Google Home is part of a new trend where people use a home assistance via automatic speech recognition. This is just one of the several main systems in the market that use such technology. This system uses a neural network model that does "multichannel processing jointly with acoustic modelling, and Grid-LSTMs to model frequency variations" [26]. In addition, on the system level, Google Home adapts the model using its own specific data. This grid long short term memory allows it "to better model frequency variations, particularly in noisy conditions, compared to a convolutional layer" [26]. The neural networks and their algorithms have become effective enough to be a reliable source for the population. However, it does require an internet connection because the current technology is not powerful enough to computer the data on its own and requires the cloud to perform that computation.

### B. Alexa

The Alexa, similar in functionality as the Google Home, uses cloud-based deep neural networks to compute the data retrieved from the system at the user's home and requires an internet connection to function due to the constraints of the technology. Alexa uses a two-step, scalable, and efficient neural shortlisting-reranking approach to find the most relevant skill for a given utterance [27]. Alexa was released in 2014, and since has become the top seller in the U.S. market for voice-powered AI devices, [because] Amazon is believed to ring up about 70 percent of all unit sales" [28]. Alexa and other voiced-based AI machines are changing our everyday lives [28]. While everyone has somehow seemed to manage their everyday lives without voice-controlled machines before Alexa was released, it has been a very popular gift. Last year, Jeff Wilke, Amazon's CEO of Worldwide Consumers said, "despite our best efforts and ramped-up production, we still had trouble keeping them in stock," when speaking of the Amazon Echo Dot [29]. Now, users who have Alexa, find life hard to go without. As there become fewer limitations on technology, people continually find new ways to incorporate technology into everyday lives. While there are hiccups along the way, such as children ordering toys through Alexa, the technological giants come back with solutions, such as parental controls.

### C. Siri

Apple first added Siri to its OS in 2011. It was one of the first major voice recognition systems to be widely accepted by the market. "Hey Siri" is what Apple users will say to trigger the voice-controlled command center inside their devices. "A very small speech recognizer runs all the time and listens for just those two words" and will listen for the rest of the user's command [30]. This "Hey Siri" detector uses a DNN to detect if what the user said and the device then "uses a temporal integration process to compute a confidence score" to determine how confident it is that what you said was "Hey Siri" and not something like "Hey Susan" when greeting your friend, Susan [30]. If the confidence scores is high enough, the device will listen for a command.

Apple has published a fascinating new entry in its Machine Learning Journal this month that explains how the voice-activated 'Hey Siri' detector works in detail. As stated in [31], Apple explains that the iPhone and Apple Watch microphone "turns your voice into a stream of instantaneous waveform samples, at a rate of 16000 per second" before the detector on device decides if you intended to invoke Siri with your voice: A spectrum analysis stage converts the waveform sample stream to a sequence of frames, each describing the sound spectrum of approximately 0.01 sec. About twenty of these frames at a time (0.2 sec of audio) are fed to the acoustic model, a DNN which converts each of these acoustic patterns into a probability distribution over a set of speech sound classes: those used in the "Hey Siri" phrase, plus silence and other speech, for a total of about 20 sound classes.

## V. Conclusions

When comparing the functionality of different methods for converting speech to text, it can be concluded that neural networks are replacing the Gaussian Mixture Model as the main method. They're more effective at giving the correct output and are able to compute more difficult sentences by comparing the words and using its memory to understand the input better. These methods are being implemented in today's technology, which allows the neural networks to grow even faster and understand speech better by using the data that people feed it. Neural networks provide a promising future for speech recognition.

## REFERENCES


[1] S. M. Mon and H. M. Tun, "Speech-to-text conversion (STT) system using hidden Markov model (HMM)," *International Journal of Scientific & Technology Research*, vol. 4, no. 6, pp 349-352, Jun. 2015.

[2] S. Young, "A review of large-vocabulary continuous-speech," *IEEE Signal Processing Magazine*, vol. 13, no. 5, pp. 45-67, Sep. 1996.

[3] F. Fauziya and G. Nijhawan, "A comparative study of phoneme recognition using GMM-HMM and ANN based acoustic modeling," *International Journal of Computer Applications*, vol. 98, no. 6, pp. 12-16, July 2014.

[4] A. S. Utane, "Emotion recognition through speech using gaussian mixture model and hidden Markov model," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 4, April 2013.

[5] G. J. McLachlan, Finite Mixture Models, New York, Wiley, 1999.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm (with discussion)," *Journal of the Royal Statistical Society*, B 39, pp. 1-38, 1997.

[7] J. Wang, and C. Liu, "Generating multivariate mixture of normal distributions using a modified cholesky decomposition," in *Proc. the 2006 Winter Simulation Conference*, 2006, pp. 342-347.

[8] G. Hamerly and C. Elkan, " Learning the K in K-Means," in *Proc. NIPS'03 the 16th International Conference on Neural Information Processing System*, Whistler, British Columbia, Canada, December 09-11, 2003 s, pp. 281-288

[9] Wikipedia. Expectation–Maximization Algorithm. [Online]. Available: https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm

[10] G. McLachlan and T. Krishnan. The EM Algorithm and Extensions, John Wiley & Sons, New York, 1997.

[11] M. A. T. Figueiredo, J. M. N. Leitão, and A. K. Jain, "On fitting mixture models," *Energy Minimization Methods in Computer Vision and Pattern Recognition*, vol. 1654, Springer, Berlin, Heidelberg.

[12] M. Wang, R. Martin, and G. Mao, "Nonsingular robust covariance estimation in multivariate outlier detection," in *Proc. International Conference on Scientific Computing*, 2015, pp. 192-195.

[13] G. J. McLachlan, "MIXFIT: an algorithm for the automatic fitting and testing of normal mixture models," in *Proc. the Fourteenth International Conference on Pattern Recognition*, 1998, pp. 553-557.

[14] G. Strang, *Linear Algebra and Its Applications*, 4th ed. Belmont, CA: Thomson, Brooks/Cole, 2006.

[15] M. Lovric, *Principal Component Analysis*, 2nd ed. Springer-Verlag Berlin Heidelberg, 2011.

[16] E. Tziperman, "Applied Mathematics 120: Applied linear algebra and big data," *Lecture Notes*, Harvard University, Spring 2018.

[17] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, series 6, vol. 2, no. 11, pp. 559-572, 1901.

[18] M. A. M. Saleh, N. S. Ibrahim, and D. A. Ramli, "Data reduction on MFCC features based on kernel PCA for speaker verification system," *WALIA Journal*, vol. 30, pp. 56-62, 2014.

[19] A. Geitgey, "Machine learning is fun! Part 2," Medium, Feb. 13, 2018.

[20] L. Hardesty, "Explained: Neural networks," MIT News, Feb. 13, 2018.

[21] D. Dhanashri and S. B. Dhonde, "Speech recognition using neural networks: A review," *International Journal of Multidisciplinary Research and Development*, vol. 2, no. 6, pp. 226-229, June 2015

[22] A. Ng and Y. Zhang. Speech recognition using deep learning algorithms. [Online]. Available: http://cs229.stanford.edu/proj2013/zhang_Speech%20Recognition%20Using%20Deep%20Learning%20Algorithms.pdf

[23] K. Noda, Y. Yamaguchi, K. Nakadai, H. G. Okuno, and T. Ogata, "Audio-visual speech recognition using deep learning," *Applied Intelligence*, vol. 42, no. 4, pp. 722-737, Jun. 2015.

[24] Y. Wang, J. Yang, L. Jun, H. Liu, and L. Wang, "Hierarchical deep belief networks based point process model for keywords spotting in continuous speech," *International Journal of Communication Systems*, vol. 28, no. 3, pp. 483-496, Feb. 2015.

[25] Y. Long, Y. Li, H. Ye, and H. Mao, "Domain adaptation of lattice-free MMI based TDNN models for speech recognition," *International Journal of Speech Technology*, vol. 20, no. 1, pp. 171-178, Mar. 2017.

[26] B. Li, T. Sainath, A. Narayanan et al., "Acoustic modeling for Google Home," *INTERSPEECH*, Feb. 14, 2018.

[27] Alexa Blogs. The scalable neural architecture behind Alexa's ability to select skills. [Online]. Available: https://developer.amazon.com/blogs/alexa/post/4e6db03f-6048-4b62-ba4b-6544da9ac440/the-scalable-neural-architecture-behind-alexa-s-ability-to-arbitrate-skills

[28] G. Anders. Amazon's Alexa is a bet that in the future we will be talking to our computers. [Online]. Available: https://www.reddit.com/r/TechnologyReview/comments/6wyinb/amazons_alexa_is_a_bet_that_in_the_future_we_will/

[29] A. Ng. Amazon's best holiday season shipped over 1 billion items. [Online]. Available: https://www.cnet.com/news/amazons-prime-best-holiday-season-1-billion-echo-alexa/

[30] Apple.com. Hey Siri: An on-device DNN-powered voice trigger for Apple's personal assistant. [Online]. Available: https://news.ycombinator.com/item?id=15499136

[31] Z. Hall, "Apple explains how 'Hey Siri' works using a Deep Neural Network and machine learning," *9TO5Mac*, September 18, 2018.


**Shaun Ault** is an associate professor and the Interim Department head of Mathematics at Valdosta State University. He received his Ph.D. in algebraic topology from The Ohio State University. His current research interests include general areas of mathematics, including topology, combinatorics, computational mathematics, and mathematics education.

**Rene J. Perez** is currently a student at Valdosta State University with a math minor. His major is in computer science. His research interests include deep learning, AI, algorithms, neural network, Markov model, and applied statistics.

**Chloe A. Kimble** is currently a student at Valdosta State University. Her major is in computer science. Her research interests include data mining, big data analysis, AI, algorithms, neural network, and numerical optimization.

**Jin Wang** is a professor of mathematics at Valdosta State University. He received his Ph.D of operations research from the School of industrial engineering at Purdue University. His research interests include data mining, big data analysis, stochastic modeling and optimization, supply chain management, and financial engineering.