Limitations in using HMMs for ASR and a survey of alternatives

Erik Ihrén
Royal Institute of Technology
eihren@kth.se

Abstract

This paper looks at some of the issues with using Hidden Markov Models for automatic speech recognition and conducts a literature study on state-of-the-art alternatives, how they work and how they overcome the limitations of Hidden Markov Models. The alternatives investigated are deep learning (including deep convolutional neural networks), long short-term memory recurrent neural networks, Hidden Semi-Markov Models and Markov Family Models.

1 Introduction

One of the difficulties involved in automatic speech recognition, is that speech is inherently variable-length. This means that the same sentence can be spoken in many different ways. Different people speak at different speeds. For example, someone who speaks very quickly might pronounce "What's up" closer to "Wassup".

Hidden Markov Models have been used for speech recognition at least since the 1980s [4], and have been shown to handle the variable-length (time-varying) part of speech recognition very well.[17] Today, they are still a very common and practical tool to use for these purposes.

However, there are problems with how Hidden Markov Models model human speech, resulting in loss of accuracy. It might be that the Hidden Markov Model is insufficient as an acoustic model when trying to reach a human-level of speech recognition accuracy.

This literature study seeks to look into some of these limitations, and investigate some of the newer, state-of-the-art approaches that have been found since Hidden Markov Models were first introduced for speech recognition.

2 Background

Hidden Markov Models model the speech process as a series of states. These states are not something we can observe. Instead we can observe "observations" from the model. More concretely, a model is defined by 5 parameters:

- A set of possible states: $S = \{s_1, s_2, \cdots, s_N\}$
- A set of possible observations: $E = \{e_1, e_2, \cdots, e_M\}$
- A probability distribution for the initial states: $\pi = \{\pi_0, \pi_1, \cdots, \pi_N\}$
- An $N \times N$ transition matrix A, where each element $a_{i,j}$ describes the probability to transition from s_i to s_j. In other words: $a_{i,j} = P(X_{t+1} = s_j | X_t = s_i)$, where X_t is the state at time t.

- An $N \times M$ emission matrix B, where each element $b_{i,j}$ describes the probability that s_i emits observation o_j. In other words: $b_{i,j} = P(o_t = e_j | X_t = s_i)$, where o_t is the observation at time t.

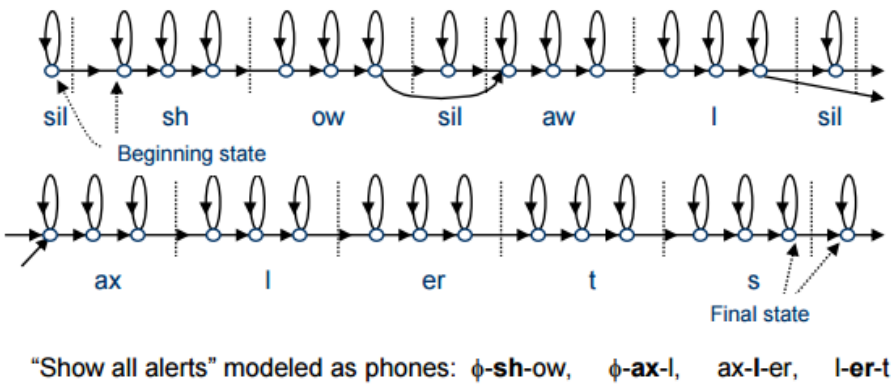

Figure 1: A Hidden Markov Model for the utterance "show all alerts."[4]

The figure above shows an example of a Hidden Markov Model that can recognize the utterance "show all alerts". In the graph, each node represents a hidden state, and an edge represents a state transition. In this example, a speech phoneme is mapped to three different states. There's a special state for silence, named "sil" in the figure. This is used at the beginning and between each word.

To discuss these different alternatives and models, we need a common dataset to compare. If we compare the accuracy of Hidden Markov Models on a very simple dataset and compare it with the accuracy of another method on a harder dataset, the result will be skewed and we wont be able to draw a correct conclusion from it. One popular dataset is the NIST 2000 Switchboard dataset[13]. This is a big dataset of telephone conversations, containing around 150 hours of conversation[14].

It's hard to find recent results on how Hidden Markov Models perform on the Switchboard dataset, since they seem to have become less popular in the recent years. However, studies seem to show that the word error rate for Hidden Markov Models on the Switchboard dataset currently lies somewhere around or above 20%. [19, 20, 21]

3 Limitations

While Hidden Markov Models perform reasonably well, they still have limitations, which affect their accuracy. This section details some of the problems that limit the accuracy of speech recognition using Hidden Markov Models.

3.1 Markov Assumption

The Markov Assumption states that a given state in the model only depends on the previous state, not on any earlier states beyond that. To describe it mathematically:

$$P(s_t | s_{t-1}, \cdots, s_0) = P(s_t | s_{t-1})$$

This is not necessarily a valid assumption when modeling speech [2]. Speech is complex, and it's possible to have dependencies between states that stretch further than just to the next/previous state. This in turn means that the Hidden Markov Model effectively ignores some of the relations between the states.

3.2 Conditional Independence Assumption

Another assumption that can be a huge source of word error rate when using Hidden Markov Models is the conditional independence assumption[1]. This is the assumption that the each observation is independent of every other observation, given the state that emitted it.

This assumption is very useful for Hidden Markov Models, since it greatly simplifies the computations. Usually, you train one Hidden Markov Model for each word in your corpus C. $M = \{M_x : x \in C\}$.[18] You can then take an observation sequence, $O = \{o_1, o_2, \cdots, o_n\}$, and calculate:

$$\underset{x}{\operatorname{argmax}}\, P(O|M_x)$$

In other words, which model has the highest probability that the observation sequence comes from that model.

Now if the observations are considered conditionally independent, we can calculate $P(O|m)$ for a given model m using an algorithm called The Forward algorithm.

$$\alpha_1(i) = \pi_i b_{i,1} \quad 1 \leq i \leq N \tag{1}$$

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{i,j}\right] b_{j,t} \quad 2 \leq t \leq T; 1 \leq j \leq N \tag{2}$$

$$P(O|m) = \alpha_n(s_t) \tag{3}$$

$$\tag{4}$$

Equation 1: The forward algorithm[17]

Unfortunately, in many cases the observations are not independent, and considering them as such can degrade the model performance[3].

4 Alternatives

Hidden Markov Models work pretty well, but as mentioned in the earlier section, they don't model speech perfectly. This means that there's room for improving the accuracy. To investigate how to reach a better accuracy, we have to look into some alternatives to Hidden Markov Models and see how they perform.

4.1 Deep learning

Deep learning is a machine learning technique, using deep neural networks to accomplish learning tasks. Deep neural networks are a special type of neural networks, containing several hidden layers.

4.1.1 Neural networks

Neural networks loosely trying to mimic how the human neurons works, with axons and dendrites.

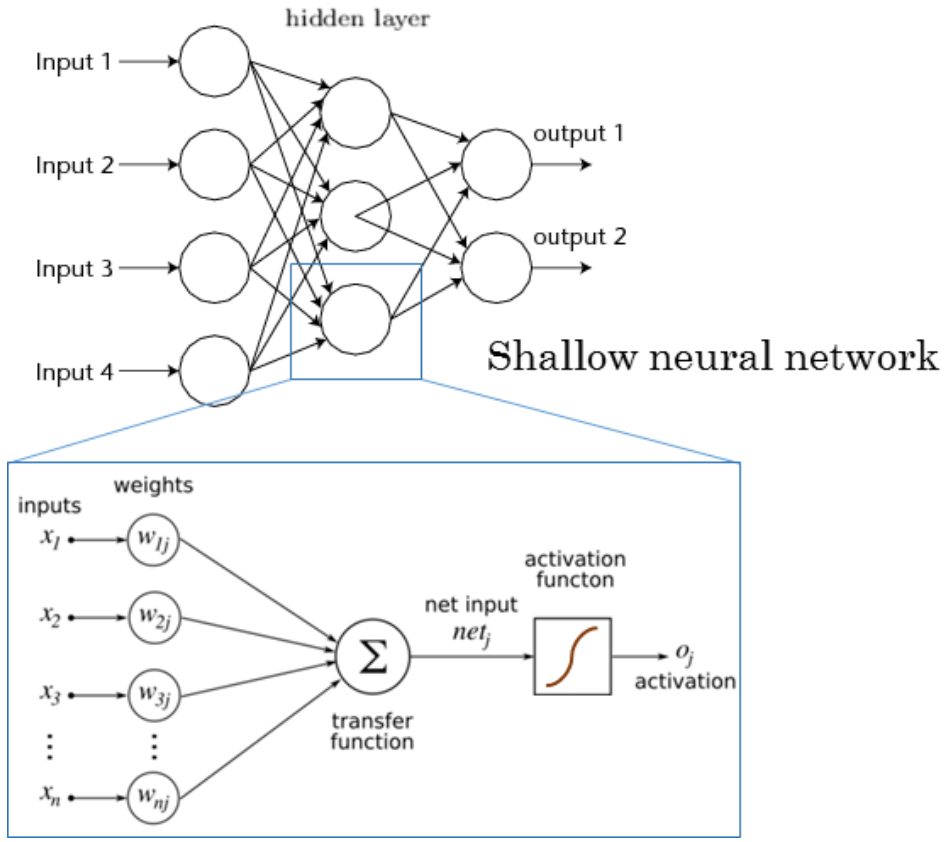

Figure 2: A detailed look at a simple feed-forward neural network[9]

As seen in the figure above, a standard neural network consists of an input layer, an output layer and a set of hidden layers in the middle. There are connection edges between each layer, each edge with its own weight. A node's value can be described with the following formula:

$$o_j = f(\sum_{i=1}^{n} w_{i,j} * x_i)$$

Where x_i is the value of node i in the previous layer, and $w_{i,j}$ is the weight between node i in the previous layer and node j in the current layer. The function $f(x)$ is called the activation function. Picking a good activation function is hard, and a subject of study in itself. The basic description of an activation function, is that it takes all its weighted inputs, and determines what the output should be. A simple activation function is a binary step function, which is defined as follows:

$$f(x) = \begin{cases} 0, & \text{if } x < 0. \\ 1, & \text{if } x \geq 0. \end{cases} \tag{5}$$

There are also some other more common activation functions, such as tanh and ReLU (Rectified Linear Unit), which are defined below.

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{6}$$

$$f(x) = \begin{cases} 0, & \text{if } x < 0. \\ x, & \text{if } x \geq 0. \end{cases} \tag{7}$$

4.1.2 Convolutional neural network

A convolutional neural network, is a neural network that takes locality it account. A normal neural network does not care about the initial order of the input data. Deep convolutional neural networks

have been shown to perform well in many tasks where the input data locality matters, such as image recognition. [10,22,23,24] The input is the pixels, which means that the pixel together with the neighboring pixels are all connected. This also true for speech recognition where the temporal aspect of the speech date is important.

Convolutional neural networks work by using a "window" that strides over the the input. Wherever the window stops, the nodes under it are combined using a convolution operation and forms the output of a node in the next layer. This is shown in figure 3, initially with a 3x3 window, and finally with a 2x2 window.

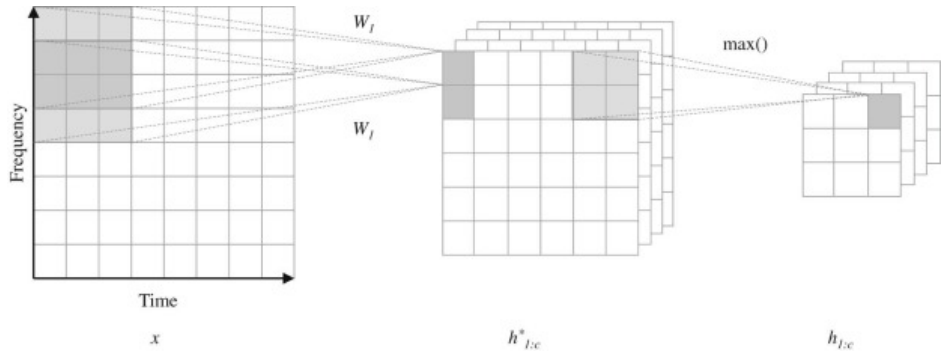

Figure 3: A multi-layer convolutional neural network[8]

Since input locality is important automatic speech recognition, it seems like convolutional neural networks can be a reasonable method to try. There has indeed been a lot of research on using deep convolutional neural networks for speech recognition[8, 10].

4.2 Long Short-term Memory Recurrent Neural Networks

In addition to normal feed-forward neural networks mentioned earlier, there is another kind of network called recurrent neural networks. These networks differ from feed-forward networks by introducing cycles in the network, creating an internal state. What's contained in this internal state is not defined. The idea is that the network learns what to save in this internal state, as part of the learning process. Since you have multiple recurrent neural nodes in the network, they can all remember different things. As a very hypothetical example, a node could remember the previous tone which might affect the recognition of the remaining sentence.

While normal neural networks "remember" what they were trained with, they won't remember anything after that. Imagine that you're using a neural network to classify something about a sentence of text. Each word is one input. By the time you send in the last word of the sentence to the network, it has forgotten everything about the previous words in the sentence.

Recurrent neural networks get around this by introducing nodes that link to themselves. The output of each node is defined as:

$$h_t = f(W * x_t + U * h_{t-1})$$

In the definition above, h_t is the output of state h at timestep t. This means that a state node can give a different output depending on the time. $W * x_t$ is the same as in a normal network. We multiply the inputs from this timestep with our weight W. The $U * h_{t-1}$ is the output of this node, from the previous timestep, multiplied by a weight U. Finally, f is the transition function used.

This allows the network to retain some information about the earlier inputs. However, it's vulnerable to a problem called exploding/vanishing gradients, which is when the gradient of the error function grows to infinity or shrinks to 0. This affects the common ways of training the networks, which rely on calculating the gradient. [15]

One popular recurrent neural network which avoids this issue is the Long short-term memory (LSTM)[15], which was designed to fix this issue. LSTM networks can initially look complex, but the basics of how they work is reasonably simple.

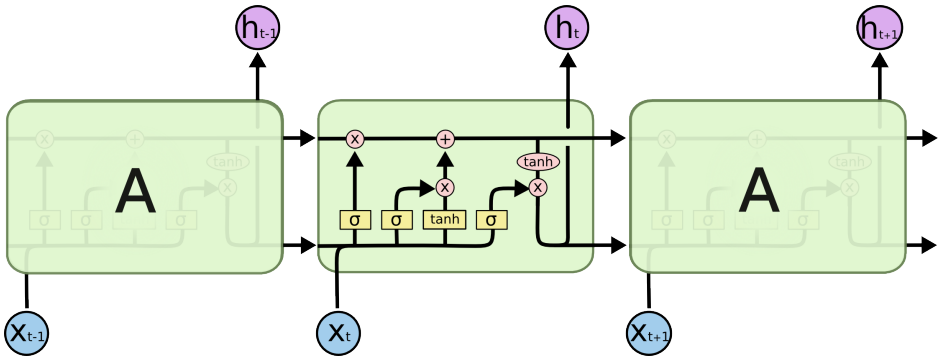

Figure 4: A long short-term memory cell[16]

To start dissecting the figure, let's first consider the green boxes, marked with "A". All three boxes represent the same cell but at different timesteps. After receiving X_{t-1} as input, the cell will output h_{t-1}. It will also pass along two pieces of information: C_{t-1} and h_{t-1}. C can be seen as the internal state of the cell. It's a vector of information. As we will see later on, this can pass along unchanged, or be changed in cell. This allows the cells to remember things for a long time. The reason h_{t-1} is passed to the cell in the next iteration is to give it some insight of what the previous output was.

Next, lets look at the internal mechanisms in the cell. There are three parts to it. The "forget" gate, the "input" gate and the "output" gate. The yellow boxes are neural network layers containing weights and a bias, which need to be learned. The σ-gates are neural layers that maps the input vector to values between 0 and 1, which can be used to multiply with another vector to determine how much of the vector to keep. The forget gate is the yellow σ-box on the left. It takes the input from this timestep and the output from the previous timestep, and decides how much of the cell state to keep. This is then multiplied with the cell state from the previous step. This way, if the forget gate returns all 1's, none of the cell state is forgotten, while if it returns all 0's, it forgets everything about the previous cell state, since it will all be multiplied by 0.

Next up is the input gate, which can add to the cell state. This is the middle part of the figure above, including one σ-box and one tanh-box. The tanh-box is a neural network layer that decides what to add to the cell state. The sigma box decides how much of it to add. It's multiplied with the tanh-output to get the scaled input. This is then added to the cell state.

Finally, there's the output gate. The cell state continues to the next timestep. For the output h_t, it's a combination of the previous output (h_{t-1}) and the cell state (C_t). The cell state is transformed in a tanh-box, which transforms it through a neural network layer, and constrains it to (-1,1). The previous output goes through another sigma-box, deciding how much to keep of the old output. Finally, we calculate $h_t = \sigma(W_o[h_{t-1}, x_t] + b_o) * tanh(C_t)$. [16] $[H_{t-1}, x_t]$ is a concatenated vector containing both h_{t-1} and x_t.

Since in every timestep, the cell state is only multiplied by the output of the forget gate, the gradient won't decrease exponentially. This allows LSTMs to avoid this issue, leading to faster training and learning. [15]

If we look at some of the latest records of achieved accuracy[11,12] on the NIST 2000 Switchboard dataset, they're not based on Hidden Markov Models. Instead they both use long short-term memory components in one way or another. It seems like, at least for now, the state-of-the-art approaches are based on deep learning, with long short-term memory components to handle the "memory".

4.3 Tuning the model

Another alternative to the Hidden Markov Model is to try and tune and adjust the model to avoid the degraded performance from the incorrect assumptions.

There has been a lot of research done on how to accomplish this[3,5,6,7]. Some of these methods have been summarized below.

4.3.1 Second-order Hidden Markov Models

One method that researchers have investigated to improve the Hidden Markov Model performance, is to use a second-order Hidden Markov Model (instead of the first-order model we've talked about previously). [25]

As mentioned in the section about limitations for Hidden Markov Models, the first-order Hidden Markov Model's state transition probability only depends on the previous state. In a second-order Hidden Markov Model, it depends on the two previous states.

While this does not take the entire history of previous states into account, it at least alleviates a bit of the problems tied to the Markov Assumption.

Research also seems to show that using a second-order Hidden Markov Model does improve accuracy for speech recognition. Unfortunately, the method is not evaluated on the switchboard dataset, and is therefore hard to compare with the rest of the alternatives investigated in this report. It seems to only improve the relative accuracy by around 2-3% [25]

4.3.2 Hidden Semi-Markov Models

Hidden Semi-Markov Models are a modified type of model, where the model fulfills the semi-markov property, instead of the full markov property. Since a Hidden Markov Model has transitions between any two states, it's natural for there to be loops, where a state keeps transitioning to itself. However, since in a normal Hidden Markov Model the markov property defines that a state only depends on the most recent state, there's no concept of how long it's been in a state.

Hidden Semi-Markov Models instead only fulfill the markov property in transitions between two different states. However, the probability of a state looping back to itself depends on how long it's been in this state, breaking the markov property.

This breaks the markov assumption, meaning it's possible that it can be used to model speech better than a normal Hidden Markov Model. There's been some research into this[5], but it requires the development of new, more complex algorithms. The commonly used algorithms can't be used, since they rely on the markov property[3].

4.3.3 Markov Family Model

Yuan defines the concept of an m-dimensional Markov Family Model[7] as a vector of m markov chains of order n_i. The value of each markov chain at a point in time t is only dependent of the previous values of that variable and the values of the other chains at the same point of time t.

Just like in the Hidden Semi-Markov Models, the idea is to break the markov assumption to overcome modeling inaccuracies. These inaccuracies lead to problems modeling difference in speaker rate, where different speakers might speak faster or slower than average. A modified version of the Markov Family Model is proposed, called the Duration Distribution Based Markov Family Model. This model takes the duration distribution into account[7].

Using this model, Yuan was able to use these Markov Family Models to obtain a 12% im-

provement in word error rate on the SI-84 training material, getting the error rate down to 10.4%. [7]

Unfortunately, since this was not done using the Switchboard dataset, it's a bit hard to compare it to the rest of the alternatives. The best we can do is estimate it based on the Markov Family Model's relative improvement of 12% over Hidden Markov Models.

Since Hidden Markov Models reached around 20% accuracy on the Switchboard data, it seems possible that Markov Family Models could improve on this. Possibly to around 17%. However, this estimate is not scientific and should be taken with a huge grain of salt. Ideally, the experiment should be repeated on the switchboard dataset to get a conclusive result.

5 Discussion and Conclusions

It seems like Hidden Markov Models are reaching their limits. While it's possible to try and circumvent the invalid assumptions that plague the HMM model, it seems like the improvements aren't good enough to get close to parity with human accuracy. The studies cited in the background section show an accuracy for Hidden Markov Models around 20%. This is far from the human accuracy on the switchboard which has been estimated to be somewhere around 5.1%. [11]

It seems like Long Short-term Memory methods are getting close to parity with humans. Looking at IBM's results[11], they were able to achieve a word error rate of 5.5%. According to their investigation, humans were able to achieve around 5.1% word error rate on that same dataset.

Back in May 2017, Google mentioned during their Google I/O keynote that they've reached a word-error rate of 4.9%. Unfortunately, except for the brief comment, there don't seem to be any other source for this claim, which means we don't really know for what dataset they achieved this. Hopefully they'll release a more comprehensive paper on this in the near future.

6 References

[1] Cohen, J., Krishnan, S.H., Morgan, N., & Wegmann, S. (2013). Final Report: OUCH Project (Outing Unfortunate Characteristics of HMMs).

[2] Chandralika Chakraborty and P H Talukdar. Issues and Limitations of HMM in Speech Processing: A Survey. International Journal of Computer Applications 141(7):13-17, May 2016.

[3] Marta Casar and José A. R. Fonollosa (2008). Overcoming HMM Time and Parameter Independence Assumptions for ASR, Speech Recognition, France Mihelic and Janez Zibert (Ed.), InTech

[4] Lawrence R. Rabiner. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proceedings of the IEEE 77, no. 2 (February 1989), p. 257-286.

[5] Bonafonte, A.; Ros, X. & Mariño, J.B. (1993). An efficient algorithm to find the best state sequence in HSMM, Proceedings of European Conf. On Speech Technology (EUROSPEECH)

[6] Z. P. Hu and S. Imai, "Modeling improvement of the continuous hidden Markov model for speech recognition," ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, San Francisco, CA, 1992, pp. 373-376 vol.1.

[7] Lichi Yuan, "An improved HMM speech recognition model," 2008 International Conference on Audio, Language and Image Processing, Shanghai, 2008, pp. 1311-1315.

[8] Andrew L. Maas, Peng Qi, Ziang Xie, Awni Y. Hannun, Christopher T. Lengerich, Daniel Jurafsky, Andrew Y. Ng, "Building DNN acoustic models for large vocabulary speech recognition", Computer Speech & Language, Volume 41, January 2017, Pages 195-213

[9] Exploring Deep Learning & CNNs, http://www.rsipvision.com/exploring-deep-learning/

[10] Qian, Yanmin ; Woodland, Philip C, "Very Deep Convolutional Neural Networks for Robust Speech Recognition", Computation and Language

[11] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dim-itriadis, X. Cui, B. Ramabhadran, M. Picheny, L.-L. Lim,B. Roomi, and P. Hall, "English Conversational Telephone Speech Recognition by Humans and Machines,"

[12] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, G. Zweig, "The Microsoft 2016 Conversational Speech Recognition System", CoRR abs/1609.03528 2016

[13] The 2000 NIST Evaluation Plan for Recognition of Conversational Speech over the Telephone, http://www.itl.nist.gov/iad/mig/tests/ctr/2000/h5_2000_v1.3.html

[14] 2000 NIST Speaker Recognition Evaluation, https://catalog.ldc.upenn.edu/LDC2001S97

[15] Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780

[16] Understanding LSTMs, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[17] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon. 2001. Spoken Language Processing: A Guide to Theory, Algorithm, and System Development (1st ed.). Prentice Hall PTR, Upper Saddle River, NJ, USA.

[18] Peinado, A. M., et al. "Improvements in HMM-based isolated word recognition system." IEE Proceedings I (Communications, Speech and Vision) 138.3 (1991): 201-206.

[19] Hain, Woodland, Evermann, Gales, Xunying Liu, Moore, . . . Lan Wang. (2005). Automatic transcription of conversational telephone speech. Speech and Audio Processing, IEEE Transactions on, 13(6), 1173-1185.

[20] Hain, T., P.C. Woodland, T.R. Niesler, and E.W.D. Whittaker. "The 1998 HTK System for Transcription of Conversational Telephone Speech." Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on 1 (1999): 57-60.

[21] Ma, and Deng. "A Path-stack Algorithm for Optimizing Dynamic Regimes in a Statistical Hidden Dynamic Model of Speech." Computer Speech & Language 14, no. 2 (2000): 101-14.

[22] Wang, Ruochen, and Zhe Xu. "A Pedestrian and Vehicle Rapid Identification Model Based on Convolutional Neural Network." Proceedings of the 7th International Conference on Internet Multimedia Computing and Service, 2015, 1-4.

[23] Tong, T., Mu, Zhang, Yi, and Hu. "MBVCNN: Joint Convolutional Neural Networks Method for Image Recognition." AIP Conference Proceedings 1839, no. 1 (2017): Materials Science, Energy Technology, And Power Engineering I: 1st International Conference on Materials Science, Energy Technology, Power Engineering (MEP 2017), Hangzhou, China (15–16 April 2017):.

[24] Zhang, Yuanyuan, Dong Zhao, Jiande Sun, Guofeng Zou, and Wentao Li. "Adaptive Convolutional Neural Network and Its Application in Face Recognition." Neural Processing Letters 43, no. 2 (2016): 389-99.

[25] Mari, J.-F., D. Fohr, and J.-C. Junqua. "A Second-order HMM for High Performance Word and Phoneme-based Continuous Speech Recognition." Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on 1 (1996): 435-38.