Intelligent Automation And Soft Computing, 2019

Vol. 25, no. 2, 351–358
https://doi.org/10.31209/2019.100000096

Modified Viterbi Scoring for HMM-Based Speech Recognition

Jihyuck Jo[a], Han-Gyu Kim[b], In-Cheol Park[a], Bang Chul Jung[c], and Hoyoung Yoo[c]

[a]School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, 34141, Republic of Korea;
[b]School of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, 34141, Republic of Korea;
[c]Department of Electronics Engineering, Chungnam National University, Daejeon, 34134, Republic of Korea

ABSTRACT

A modified Viterbi scoring procedure is presented in this paper based on Dijkstra's shortest-path algorithm. In HMM-based speech recognition systems, the Viterbi scoring plays a significant role in finding the best matching model, but its computational complexity is linearly proportional to the number of reference models and their states. Therefore, the complexity is serious in implementing a high-speed speech recognition system. In the proposed method, the Viterbi scoring is translated into the searching of a minimum path, and the shortest-path algorithm is exploited to decrease the computational complexity while preventing the recognition accuracy from deteriorating. In addition, a two-phase comparison structure is proposed to manage state probabilities efficiently. Simulation results show that the proposed method saves computational complexity and recognition time by more than 21% and 10% compared to the conventional Viterbi scoring and the previous early termination, respectively. The improvement of the proposed method becomes significant as the numbers of reference models, states, and Gaussian mixture models increase, which means that the proposed method is more desirable for recent speech recognition systems that deals with complex models.

KEYWORDS: Viterbi scoring, Dijkstra's algorithm, hidden Markov model, searching algorithms, speech recognition.

1 INTRODUCTION

AMONG a variety of speech recognition algorithms including dynamic time warping (DTW) and neural network, the hidden Markov model (HMM) based algorithm has been most widely applied to speech recognition due to its robustness to speech and speaker variations (Rabiner & Juang, 1993, Czyzewski. A., 1996, Macias-Guarasa, J. et. al. 2009, Xihao, et. al. 2013, Ting, 2013, Paramonov, 2017). In general, a speech recognition system grounded on the HMM algorithm consists of two stages (Rabiner, 1989), training and recognition, as depicted in Figure 1. The training stage is to obtain a distinct HMM for each reference model. In this stage, speech feature vectors extracted from short-segmented speech signals are trained to derive HMMs. During the speech recognition stage, the recognizer computes likelihood scores to find the best matching model by comparing the utterance with the trained HMMs. Since the reference models are generated at the off-line training stage prior to the on-line recognition stage, observation probability computation (OPC) and likelihood score computation (LSC) are the most time consuming parts of a HMM-based speech recognition system (Rabiner, 1989).

THE Viterbi algorithm has been widely employed in the likelihood score computation, as it is efficient in finding the best matching model (Rabiner & Juang, 1993, Lou, 1995, Prasad, 2018). However, the computational complexity of the Viterbi algorithm is linearly proportional to the number of reference models and their states. Due to its exhaustive procedure covering the whole reference models, the speech recognition system inevitably suffers from a large amount of computation and enormous data accesses. Moreover, each state in the Viterbi scoring demands to compute the observation probability. Such

CONTACT Hoyoung Yoo hyyoo@cnu.ac.kr

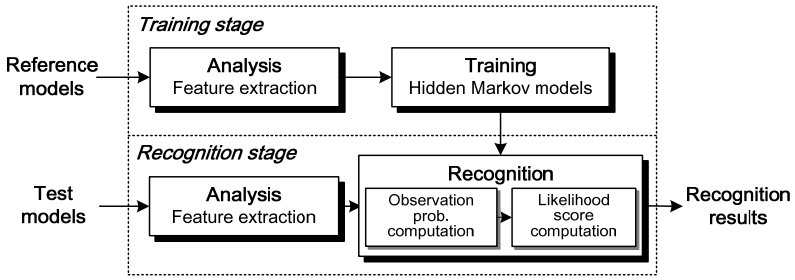

Figure 1. The typical structure of speech recognition systems.

Log-Viterbi algorithm

for $v = 1$ **step** 1 **until** W
begin
Input: observation sequence $\boldsymbol{o}$ and HMM λ_v.
Initiation: $\delta_j(1) = \log \pi_j + \log b_j(o_1)$ for $1 \le j \le N$.
for $t = 2$ **step** 1 **until** T
begin
$\delta_j(t) = \max\{\delta_i(t-1) + \log a_{ij}\} + \log b_j(o_t)$ for $1 \le i, j \le N$.
end
Termination: $P(o \mid \lambda_v) = \max_{1 \le j \le N} \{\delta_T(j)\}$.
end
Decision: $\hat{v} = \arg\max_{1 \le v \le W} \{P(o \mid \lambda_v)\}$.

Figure 2. The conventional log-Viterbi algorithm.

computation is a serious burden in implementing a high-speed speech recognition system irrespective of whether it is realized in software or in hardware. This problem becomes more severe in recent speech recognition systems that have a large amount of reference models and states.

A modified Viterbi scoring procedure is proposed in this paper to eliminate unnecessary computations. Since the Viterbi scoring algorithm is to find the best matching model among many reference ones, it is unnecessary for the speech recognizer to calculate the remaining score probabilities after the best matching reference model is found. We apply Dijkstra's shortest-path algorithm (Nilsson, 1980) to the scoring procedure and propose an efficient structure to decide the best candidate in searching. As a result, the proposed algorithm achieves a remarkable reduction of recognition time by avoiding exhaustive computations.

2 VITERBI SCORING ALGORITHM

THIS paper focuses on the HMM-based isolated word recognition systems because the advanced search network might obstruct the clear understanding of the proposed modified Viterbi scoring process. A HMM is characterized by an initial state probability $\pi=\{\pi_j\}$, a state transition probability $A=\{a_{ij}\}$, and an observation probability $B=\{b_j(o_t)\}$ (Rabiner, 1989, Lou, 1995). In the HMM-based isolated word recognition, each word is individually represented by a HMM λ_v for $1 \le v \le W$, where W is the number of word models and λ represents a statistical set of π, A, and B. Given an observation sequence $o=[o_1, o_2, \cdots, o_T]$, where T is the number of observations, the speech recognizer calculates and compares all $P(o|\lambda_v)$ for $1 \le v \le W$ so as to find a word with the highest probability. Table 1 summarize symbols used in this paper. In general, the conventional word-based HMM system recognizes a word by using the log-Viterbi algorithm (Rabiner, 1989, Lou, 1995) described in Figure 2.

Constants in HMM
Number of states N
Length of times T
Number of candidate words W
Variables in HMM
State index $1 \le i, j \le N$
Time index $1 \le t \le T$
Word index $1 \le v \le W$
Probabilities in HMM
Initial state probability $\pi=\{\pi_j\}$,
State transition probability $A=\{a_{ij}\}$
Observation probability $B=\{b_j(o_t)\}$
State probability $\delta_j(t)$
Isolated word probability $P(o|\lambda_v)$

Table 1. Symbol list.

IN the algorithm, indices i and j range from 1 to N, and t represents the index of observation time ranging from 1 to T. Given an observation sequence $\boldsymbol{o}$ and HMM λ_v, the algorithm finds the most likely state

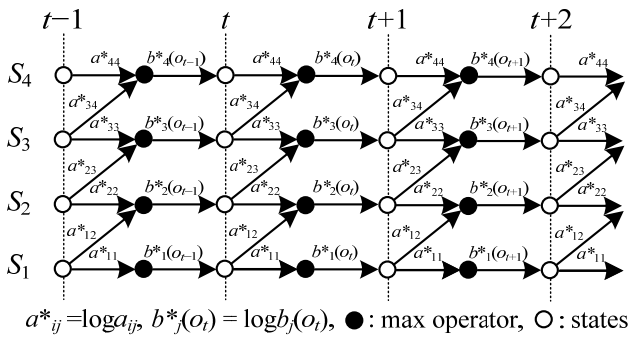

Figure 3. The trellis based on the log-Viterbi algorithm when the number of state *N* is 4.

sequence and its probability. At the beginning of the search, each state is initialized to the initial state probability $\boldsymbol{\pi}^*=\{\log\pi_j\}$ and the first observation probability $\log b_j(o_1)$. To update the state probability $\delta_j(t)$, we need to find the most likely transition coming into each state. This is achieved by adding log transition probabilities $\boldsymbol{A}^*=\{\log a_{ij}\}$ to their previous state probability $\delta_i(t-1)$. As the HMM structure normally simplifies its transitions based on the left-to-right model, the state transition probability $a_{ij}=0$ if $i \neq j$ or $i \neq j-1$. This sum is then added to the log observation probability $\boldsymbol{B}^*=\{\log b_j(o_t)\}$ assigned to the given state of the t-th observation. This process is performed recursively until the observation sequence is completed. At the end of the recursion, the highest state probability becomes $P(\boldsymbol{o}|\lambda_v)$ representing the probability of the HMM λ_v for the given observation. This log-Viterbi algorithm computes all $P(\boldsymbol{o}|\lambda_v)$ for each word model. At last, a word v with the highest $P(\boldsymbol{o}|\lambda_v)$ is selected as the recognized word for the given observation sequence $\boldsymbol{o}$. Figure 3 shows a trellis of the Viterbi scoring procedure that uses the left-to-right state transition model. For the purpose of clarification, only a part of trellis ranging from $t-1$ to $t+2$ is illustrated for a word. Though the Viterbi algorithm efficiently finds the best matching model, the algorithm necessitates a tremendous amount of computation and enormous data accesses, since it takes into account every state in all the reference words for the length of the test utterance. More precisely, the number of states that should be considered to recognize a word is $N\times T\times W$. Moreover, each state is associated with the most complex computation of the speech recognizer, the observation probability computation, which is performed under multi-dimensional Gaussian mixture model (GMM).

TO lessen the computational complexity, efficient structures have been proposed for hardware implementation (Yoshizawa, Wada, Hayakawa, & Miyanaga, 2006, Nakamura, Shimazaki, Yamamoto, K. Takagi, & N. Takagi, 2012), most of which employ parallel and pipelining techniques to achieve high-speed realizations. Although those approaches made a success in achieving high-speed speech recognizers, the amount of computation is still proportional to $N\times T\times W$. Unlike the advanced implementation techniques, (Park, K. Cho, & J. Cho, 2002, Paramonov et. al., 2014) presented an early termination technique to skip a redundant computation of the state probabilities. As reference words in the Viterbi algorithm is searched one by one, (Park et. al., 2002) terminates the scoring computation for a word and move on to the next testing word when the state probability of the word is less likely than $P(\boldsymbol{o}|\lambda_v)$ of the previously compared words. The early termination method can reduce unnecessary state probability scoring, however, each state probability is demanded to compare with the most likely $P(\boldsymbol{o}|\lambda_v)$ of the previously tested words, and the improvement depends on the order of reference models. For instance, when the last reference model is the desired word, the computational saving is not as significant as the case that the first reference model is the word.

3 PROPOSED ALGORITHM

A new modified Viterbi scoring method is proposed for HMM-based speech recognition systems. Dijkstra's shortest-path algorithm (Nilsson, 1980) is employed so as to eliminate unnecessary state computations while preventing the recognition accuracy from deteriorating. We first analyze the trellis to translate the Viterbi scoring into the shortest-path searching, and then describe the proposed Viterbi scoring algorithm.

THE conventional log-Viterbi algorithm described in Figure 2 decides the reference model v with the highest $P(\boldsymbol{o}|\lambda_v)$ as the recognized word. For this, it computes the intermediate state probability $\delta_j(t)$ by summing the most likely transition probability and the observation probability $\log b_j(o_t)$. To decide the most likely transition probability for state j at time t, the comparison is performed to select the maximum sum of the previous state probability $\delta_i(t-1)$ and transition probability $\log a_{ij}$. The logarithm is applied to convert multiplication into addition and to avoid the underflow of state probability. Probability values ranging from 0 to 1 are all negative in the logarithm domain. If all the log-probabilities associated with π_j, a_{ij}, $b_j(o_t)$, and $\delta_j(t)$

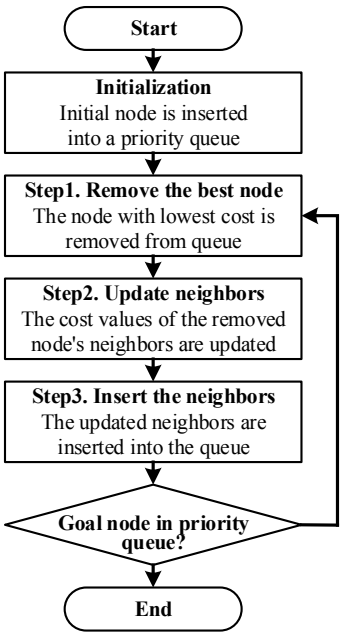

Figure 4. Dijkstra's algorithm

are negated, they are converted to positive values, and the comparison performed to select the maximum, $\max\{\delta_i(t-1)+\log a_{ij}\}$, is changed to $\min\{\delta_i(t-1)-\log a_{ij}\}$. As the trellis of Viterbi scoring becomes monotonically increasing in that case, the Viterbi scoring can be considered as a kind of shortest-path search that deals with positive costs of $-\log a_{ij}$ and $-\log b_j(o_t)$. A word with the lowest $P(\boldsymbol{o}|\lambda_v)$ is determined as the recognized word.

DIJKSTRA's algorithm is widely used to find the shortest path in a graph (Nilsson, 1980). It follows the best-first search and always finds the lowest-cost path based on the principle of optimality (Dreyfus & Law, 1977). Let us assume that a graph consists of nodes with given cost values. To find the shortest path, Dijkstra's algorithm always finds the best path first and move further until reach to the goal node. More precisely, the algorithm runs as shown in Figure 4. As Dijkstra's algorithm uses the best-first search, it first searches for a node that appears to be close to the goal. Furthermore, unvisited nodes do not need to be searched after arriving to the goal node. Given the monotonically increasing trellis, the algorithm can be employed so as to decrease state computations. The trellis is easily considered as a tree structure with a common initial node that is virtually assumed, and every state in all the reference words is regarded as the node. The node with the smallest state probability is selected as the most likely path.

FURTHERMORE, the algorithm is slightly revised to manage the most likely path efficiently. Although Dijkstra's algorithm takes into account all $N\times W$ nodes at a time t, the speech-recognition system decides a word based on the reference model rather than individual paths. Thus, we employ two-phase comparison to efficiently manage the minimum values. In the proposed algorithm, the first phase chooses a local minimum (LM) associated with the minimum state probability for each reference model, and the second phase compares W local minima to decide the global minimum (GM). To expedite a selection of the GM, a min-heap structure (Tarjan, 1983) is used to implement the priority queue. The GM is effectively managed with a min-heap of size W to avoid a large amount of comparisons to be performed for $N\times W$ nodes. All the N nodes of the reference model corresponding to the GM are searched to find a new LM for the reference model.

THE proposed algorithm is described in Figure 5. At the beginning of searching, each state is initialized to the sum of $-\log\pi_j$ and $-\log b_j(o_1)$, and for each reference model v, the $LM(v)$ is selected among N state probabilities $\delta_j(1)$, $1\le j\le N$. The most likely model is the reference model indicated by the GM. To find the lowest-cost path, N paths in the reference model $\tilde{v}$ corresponding to the GM is examined. Based on the updated state probabilities $\delta_j(t_{\tilde{v}})$, the LM of $\tilde{v}$ is newly selected, and the min-heap and the GM are accordingly updated. When we arrive at the end of a reference model at time T, the scoring procedure is terminated and the reference model becomes the recognized word. Note that the remaining uncalculated scores are guaranteed to be higher than the score of the recognized word, since the most likely reference model is selected as the GM every time. Figure 6 exemplifies a graphical representation of the proposed algorithm that uses the left-to-right state transition model. For the sake of simplicity, the numbers of words and states are both fixed to 4. Word 3 is

Proposed algorithm

Input: observation sequence $\boldsymbol{o}$ and HMM λ_v.

Initiation:

for $v = 1$ **step 1 until** W

begin

$\delta_j(1) = -\log \pi_j - \log b_j(o_1)$ for $1 \le j \le N$.

$LM^*(v) = \min\{\delta_j(1)\}$ for $1 \le j \le N$.

$t_v = 1$.

end

Best-first scoring:

Step1: $H^* = \{LM(v)\}$ for $1 \le v \le W$.

Step2: $GM^* = \min\{LM(v)\}$,
$\overline{v} = \underset{v}{\operatorname{argmin}}\{LM(v)\}$ for $1 \le v \le W$.
$H = GM$ is deleted.

Step3: $\delta_j(t_{\overline{v}}) = \min\{\delta_i(t_{\overline{v}} - 1) - \log a_{ij}\} -$
$\log b_j(o_{t_{\overline{v}}})$ for $1 \le i, j \le N$.
$t_{\overline{v}} = t_{\overline{v}} + 1$.

Step4: $LM(\overline{v}) = \min\{\delta_j(t_{\overline{v}})\}$ for $1 \le j \le N$.
$H = LM(\overline{v})$ is inserted.

Step5: if $t_{\overline{v}} = \mathrm{T}$, quit. Otherwise, go to Step2.

Decision: $\hat{v} = \overline{v}$.

**H: min-heap, LM: local min., GM: global min.*

Figure 5. The proposed modified Viterbi scoring algorithm.

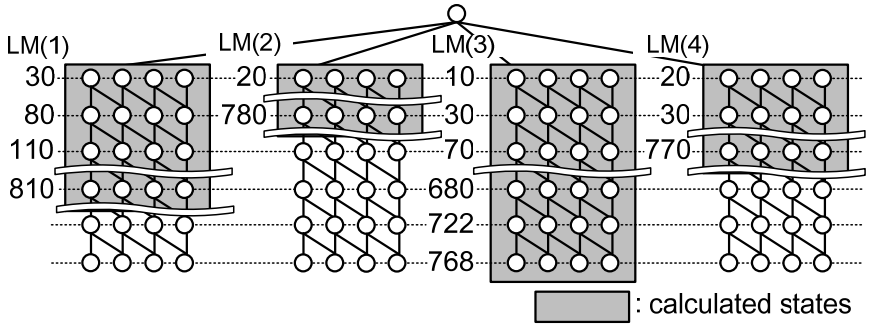

Figure 6. The tree structure based on the proposed algorithm when the numbers of words and states are 4.

selected as the recognized word since it is arrived first, and its final LM is smaller than those of the other words. Consequently, the overall computational complexity of speech recognition can be decreased due to the skipped OPCs and LSCs. Note that the proposed method always need less computations without any recognition degradation. Unlike the previous early termination (Park et. al., 2002), in addition, the computational reduction does not depend on the testing order of reference models.

4 SIMULATION RESULTS

TO compare the computational complexities of different scoring algorithms, we have simulated three different ones: the conventional Viterbi algorithm (Nilsson, 1980), the early termination method (Park et. al., 2002), and the proposed modified Viterbi algorithm with various configurations of words, states, and Gaussian mixture models. In general, the pruning technique (Huang, Alejandro, & Hon, 2001) that removes unlikely paths is one of fascinating methods to decrease the computational complexity, but it degrades the recognition accuracy inherently due to the finite beam width and threshold. Note that the three scoring algorithms do not induce any searching errors degrading the recognition accuracy. For fair comparison, the typical speech recognition systems shown in Figure 1 is assumed and the pruning technique is not considered in the comparison.

FOR speech-recognition experiments, the speech signal is sampled at 16 KHz with 16-bit quantization, and 39 feature vectors are extracted for every 32ms overlapped frame. In a speech corpus collected from 32 female and 38 male speakers from Korean

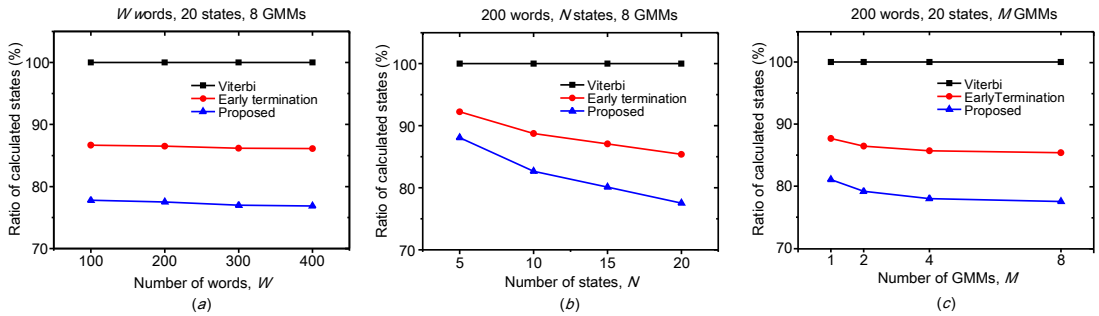

Figure 7. Ratio of calculated states versus numbers of (a) words *W*, (b) states *N*, and (c) GMMs *M*.

Phonetically Balanced Words (KPBW) (ETRI, 1995), 60% utterances are used for training and the remaining 40% are used for testing. More precisely, Hidden Markov Model Toolkit (HTK) developed from Cambridge University (Yong et. al., 2002) is used for obtain trained HMM sets. For the three scoring algorithms, Figure 7 shows how the ratio of the calculated states to the overall $N \times T \times W$ states depends on the numbers of reference models, states, and GMMs. Since the effectiveness of the early termination method (Park et. al., 2002) is dependent on the testing order of reference models, the correct word model is tested at the middle of the reference word sequences. On the average, the proposed modified Viterbi algorithm saves the computational complexity by more than 21% and 10%, compared to the conventional Viterbi scoring and early termination algorithms, respectively. In order to bring a practical contribution, the overall recognition time including comparison computations is measured in a 2.4 GHz computer system. The results are plotted in Figure 8, where it is clear that the less computational complexity leads to the faster recognition. From Figure 7 and Figure 8, we can see that the improvement of the proposed algorithm becomes more significant as the numbers of the words, states, and GMMs increase. Furthermore, an advanced tree searching algorithm including A-Star algorithm is currently investigated so as to achieve a further improvement (Nilsson, 1980).

5 CONCLUSION

THE Viterbi scoring that compares test utterances with reference models to find the best matching model suffers from huge computational complexity. This paper has presented a modified Viterbi scoring algorithm so as to effectively eliminate unnecessary computations. In the proposed method, the Viterbi scoring is translated into the shortest-path search, and Dijkstra's shortest-path algorithm is applied to decrease the computational complexity without sacrificing recognition accuracy. Moreover, a two-phase comparison method has been proposed to manage the state probabilities efficiently. The proposed method reduces the computational complexity and recognition time by more than 21% and 10% compared to the conventional Viterbi scoring and the early termination (Park et. al., 2002) algorithms, respectively. The complexity reduction becomes more significant if the numbers of words, states, and GMMs increase. Furthermore, the proposed method can be applied to the connected word recognition and continuous speech recognition when the Viterbi searching is employed.

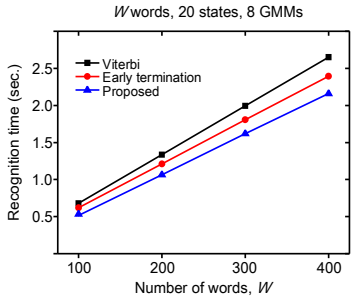

Figure 8. Recognition time versus number of words W.

ACKNOWLEDGMENT

THIS work was supported by research fund of Chungnam National University in 2017.

REFERENCES

Czyzewski, A. (1996). Speaker-Independent Recognition of Digitsmdash Experiments with Neural Networks, Fuzzy Logic and Rough Set. *Intelligent Automation & Soft Computing*. Pp. 133-145: TSI Press. doi: 10.1080/10798587.1996.10750662

Dreyfus, S.E., & Law, A.M. (1977). *The art and theory of dynamic programming*. New York, NY: Academic Press Inc.

Electronics and Telecommunications Research Institute (ETRI). (1995). *Research on the Korean Speech Synthesis Technology (V)*: 5KT21000481420F.

Huang, X., Alejandro, A., & Hon, H.W. (2001). *Spoken language processing*. Upper Saddle River, NJ: Prentice Hall.

Lou, H.-L. (1995). Implementing the Viterbi algorithm. *IEEE Signal Processing Magazine, 12*, 42-52. doi: 10.1109/79.410439.

Macias-Guarasa, J. et. al. (2009). Novel Applications of Neural Networks in Speech Technology Systems: Search Space Reduction and Prosodic Modeling. *Intelligent Automation & Soft Computing*. pp. 631-646: TSI Press. doi: 10.1080/10798587.2009.10643054

Nakamura, K., Shimazaki, R., Yamamoto, M., Takagi, K., & Takagi, N. (2012). A VLSI architecture with multiple fast store-based block parallel processing for output probability and likelihood score computations in HMM-based isolated word recognition. *IEICE Transactions on Electronics, E95-C*, 456-467. doi: 10.1587/transele.E95.C.456.

Nilsson, N.J. (1980). *Principles of artificial intelligence*. Palo Alto, CA: Tioga Publishing Co.

Paramonov, P. & Sutula, N. (2014). Simplified Scoring Methods in HMM Based Speech Recognition. *International Conference on Soft Computing and Machine Intelligence*. (pp. 154-156) New Delhi, India: IEEE. doi: 10.1109/ISCMI. 2014.32

Paramonov, P. (2017). Fast algorithm for isolated words recognition based on Hidden Markov model stationary distribution. *International Conference on Soft Computing & Machine Intelligence* (pp. 128-132). Port Louis, Mauritius: IEEE. doi: 10.1109/ISCMI.2017.8279612

Park, B.-G., Cho, K.-S., & Cho, J.-D. (2002, March). Low power VLSI architecture of Viterbi scorer for HMM-based isolated word recognition. *International Symposium on Quality Electronic Design* (pp. 235-239). San Jose, California, USA: IEEE. doi: 10.1109/ISQED.2002.996739.

Prasad, N., Chakrabarti, I., & Chattopadhyay, S. (2018). An Energy-Efficient Network-on-Chip-Based Reconfigurable Viterbi Decoder Architecture. *Transactions on Circuits and Systems I: Regular Papers*. Early Access. IEEE: doi: 10.1109/ TCSI.2018.2825362

Rabiner, L.R., & Juang, B.H. (1993). *Fundamentals of speech recognition*. Englewood Cliffs, NJ: Prentice Hall Inc.

Rabiner, L.R. (1989). *A tutorial on hidden Markov models and selected applications in speech recognition*. *Proceedings of the IEEE, 77*, 257-286. doi: 10.1109/5.18626.

Tarjan, R.E. (1983). *Data structures and network algorithms*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Ting, H.-N., Yong, B.-F., & Mirhassani, S. M. (2013). Self-Adjustable Neural Network for Speech Recognition. *Engineering Applications of Artificial Intelligence*. pp. 2022-2027: Elsevier. doi.org/10.1016/j.engappai.2013.06.004

Xihao, S. & Miyanaga, Y. (2013). Dynamic time warping for speech recognition with training part to reduce the computation. *International Symposium on Signals, Circuits and Systems*. (pp. 1-4). Iasi, Romania: IEEE. doi: 10.1109/ISSCS. 2013.6651269.

Yoshizawa, S., Wada, N., Hayakawa, N., & Miyanaga, Y. (2006). Scalable architecture for word HMM-based speech recognition and VLSI implementation in complete system. *IEEE Transactions on Circuits and Systems I: Regular Papers, 53*, 70-77. doi: 10.1109/TCSI.2005.854408.

Young, S. J., Evermann, G., & Gales, M. J. F., et. al. (2009). *The HTK book (version3.4)*: Cambrige University Engineering Department.

DISCLOSURE STATEMENT

NO potential conflict of interest was reported by the authors.

NOTES ON CONTRIBUTORS

Jihyuck Jo received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2012, 2014, and 2018, respectively. Since 2018, he has been a Senior Engineer at Samsung Electronics. His current research interests include VLSI architectures for general-purpose microprocessors and neural network processors.

Han-Gyu Kim received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2009 and the M.S. and the Ph.D. degree in School of Computing, KAIST, Daejeon, South Korea, in 2011 and 2018. He is currently working as a researcher in Clova Speech, Naver Corp. His research interests include speech recognition, source separation and deep learning.

In-Cheol Park received the B.S. degree in electronic engineering from Seoul National University, and the M.S. and Ph.D. degrees in electrical engineering from KAIST. He is currently a Professor with the School of Electrical Engineering, KAIST. His research interests include computer-aided design algorithms for high-level synthesis and very large scale integration architectures for general-purpose microprocessors.

Bang Chul Jung received the Ph.D. degrees in Electrical & Computer Engineering from KAIST, Daejeon, Korea, in 2008. He is currently an Associate Professor of the Department of Electronics Engineering, Chungnam National University, Daejeon, Korea. His research interests include wireless communication systems, statistical signal processing, and machine learning.

Hoyoung Yoo received Ph.D. degrees in Electronics Engineering from KAIST, Daejeon, Korea, in 2016. Since September 2016, he has been an assistant professor in the department of Electronics Engineering at Chungnam National University. Specific areas of current interest include VLSI design for Error Correction Codes and 5G communication systems.