STRUCTURE DISCOVERY OF DEEP NEURAL NETWORK BASED ON EVOLUTIONARY ALGORITHMS

Takahiro Shinozaki

Tokyo Institute of Technology
4259 Nagatsuta-cho, Midori-ku, Yokohama,
Kanagawa, 226-8502 JAPAN

Shinji Watanabe

Mitsubishi Electric Research Laboratories (MERL)
201 Broadway, Cambridge, MA, 02139 USA

ABSTRACT

Deep neural networks (DNNs) are constructed by considering highly complicated configurations including network structure and several tuning parameters (number of hidden states and learning rate in each layer), which greatly affect the performance of speech processing applications. To reach optimal performance in such systems, deep understanding and expertise in DNNs is necessary, which limits the development of DNN systems to skilled experts. To overcome the problem, this paper proposes an efficient optimization strategy for DNN structure and parameters using evolutionary algorithms. The proposed approach parametrizes the DNN structure by a directed acyclic graph, and the DNN structure is represented by a simple binary vector. Genetic algorithm and covariance matrix adaptation evolution strategy efficiently optimize the performance jointly with respect to the above binary vector and the other tuning parameters. Experiments on phoneme recognition and spoken digit detection tasks show the effectiveness of the proposed approach by discovering the appropriate DNN structure automatically.

***Index Terms*—** Deep neural network, structure optimization, evolutionary algorithm, genetic algorithm, CMA-ES

1. INTRODUCTION

Recently, deep neural networks (DNNs) have become standard techniques in various speech and language applications including speech recognition and spoken term detection [1,2], and well-trained DNNs make it possible to build high-performance systems for such tasks. However, DNNs have very complicated configurations including network topologies and several tuning parameters (number of hidden states and learning rate in each layer) that greatly affect the performance of speech processing applications. Therefore, to optimize performance in such systems, deep understanding and expertise in DNNs is necessary, thus limiting the development of DNN systems to skilled experts. This paper aims to overcome this problem by automatically optimizing DNNs used in acoustic models in speech processing without human expert elaboration based on evolutionary algorithms.

Applications of evolutionary algorithms to acoustic modeling in speech processing have been studied by various researchers [3–7]. However, these approaches have mainly focused on model parameter optimization or several tuning parameters. DNN has another important configuration tuned by human experts that deals with the topologies of DNNs. In fact, the progress in neural network speech processing arises from the discovery of appropriate DNN topologies, e.g., deep layers, deep stacking networks, and tandem concatenation of original inputs with neural network (bottleneck layer) outputs for GMM features [8–11]. We call this problem the *structure optimization* of DNNs. There are several studies of structure optimization of Gaussian based models [12–16]; however, these studies do not address the optimization of DNNs. With this background, this paper proposes efficient structure and tuning parameter optimization algorithms based on evolutionary algorithms for DNNs used in speech processing.

Evolutionary algorithms are a generic term for nonlinear/non-convex optimization algorithms inspired by several biological evolution mechanisms, including genetic algorithms and evolution strategies, and are widely used in various applications. This paper uses a popular genetic algorithm [17] and covariance matrix adaptation evolution strategy (CMA-ES) [18, 19] as an evolutionary algorithm. They have the common functions of 1) generating various vectors encoding system configurations to be tested and 2) scoring multiple DNN systems with these prepared configurations. These steps are iteratively performed to search the optimal tuning parameters. The first generation step provides the multiple hypotheses created by a process of simulated biological gene generation in a genetic algorithm and by probabilistic random number sampling in CMA-ES. The second scoring step includes training and evaluation of DNNs to obtain their scores, which usually takes very long time, but this step can be parallelized for each configuration. Although both approaches must represent a DNN structure configuration using a vectorized form (gene), this paper also proposes to formulate the DNN structure configuration with a binary vector based on directed acyclic graph (DAG) representation, where each connection between neuron modules is represented by a binary value.

We conducted experiments on a massively parallel computing platform TSUBAME supercomputer developed at Tokyo Institute of Technology[1]. Although each evaluation of DNN is fairly simple (single-speaker phone recognition and spoken digit detection), the experiments were ran using 62-GPGPUs in parallel for both evolutionary algorithms to address DNN structure discovery problems.

2. EVOLUTIONARY ALGORITHM

Let us first denote the nth DNN configuration as C_n. C_n is composed of DNN specifications of tuning parameters and network topologies, which is enough information to specify the DNN training. The D-dimensional vector $\boldsymbol{x}_n$ is a code to generate the configuration C_n and can be regarded as a *gene* in the evolutionary algorithm. This problem is formulated to find the optimal configuration C_{n^*}:

$$C_{n^*} = \underset{\forall n}{\operatorname{argmax}}\, f(C_n). \tag{1}$$

[1] http://www.gsic.titech.ac.jp/en

Algorithm 1 Evolutionary algorithm

```
Initialize x
for n = 1, ··· , N_0 do
    Sample x_n
end for
while not convergence do
    for n = 1, ··· , N (N_0 for the 1st iteration) do
        Decode gene vector x_n to configuration C_n
        Evaluate configuration C_n to get score f(C_n)
    end for
    Generate child gene vectors {x_n}_{n=1}^N from current (parent)
    gene vectors {x_n}_{n=1}^N and their scores {f(C_n)}_{n=1}^N
end while
```

where function f provides the performance of the DNN (i.e., the accuracy and size of DNN). As speech processing systems are extremely complicated, there is no analytical form for f, and it is difficult to include specific knowledge in f. Such situations are best handled using an evolutionary algorithm, which is described in Algorithm 1.

The algorithm starts to provide initial gene vectors $\{\boldsymbol{x}_n\}_{n=1}^N$. Then, the gene vectors are decoded to the corresponding configurations $\{C_n\}_{n=1}^N$, and scored by function evaluations. Based on the scores, new gene vectors are generated. Genetic algorithms and CMA-ES have different strategies in this generation step, and the following section describes the gene generation step of each method in greater detail.

2.1. Genetic algorithm

The genetic algorithm is a search heuristic motivated by the natural evolution process. It is based on 1) the selection of genes according to their scores pruning inferior gene vectors for the next iteration (generation), 2) mating pairs of gene vectors to make child gene vectors that mix the properties of the parents, and 3) mutation of a part of a gene vector to produce new gene vectors.

In the selection process, this paper uses a *tournament method*, which first extracts a subset of $M(< N)$ genes ($\{\boldsymbol{x}_{n'}\}_{n'=1}^M$) from a total of N genes randomly and then selects the best gene by the score ($\boldsymbol{x}_{n^*} = \text{argmax}_{n'} f(C_{n'})$). The random subset extraction step can provide variations of genes, while the best selection step in a subset guarantees the exclusion of inferior genes.

For the mating process, the *one-point crossover* method is adopted, which first finds a pair of (parent) genes from the selected genes ($\boldsymbol{x}_{n1}^{\text{p}}$ and $\boldsymbol{x}_{n2}^{\text{p}}$) and then swaps the $d+1$ to D elements of these two vectors to obtain a new (child) gene vector pair ($\boldsymbol{x}_{n1}^{\text{c}}$ and $\boldsymbol{x}_{n2}^{\text{c}}$).

$$\begin{aligned}\boldsymbol{x}_{n1}^{\text{c}} &= \begin{bmatrix} x_{n1,1}^{\text{p}} & \cdots & x_{n1,d}^{\text{p}} & x_{n2,d+1}^{\text{p}} & \cdots x_{n2,D}^{\text{p}} \end{bmatrix}^\intercal \\ \boldsymbol{x}_{n2}^{\text{c}} &= \begin{bmatrix} x_{n2,1}^{\text{p}} & \cdots & x_{n2,d}^{\text{p}} & x_{n1,d+1}^{\text{p}} & \cdots x_{n1,D}^{\text{p}} \end{bmatrix}^\intercal \end{aligned} \tag{2}$$

where $^\intercal$ is the matrix transpose. The position d is randomly sampled. As the iteration increases, these processes provide appropriate gene vectors that encode optimal DNN configurations.

2.2. Covariance matrix adaptation evolution strategy

CMA-ES assumes that a gene vector $\boldsymbol{x}$ is generated from a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean vector and covariance matrix, respectively. After the function evaluation in each epoch, CMA-ES updates $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ using the following equation:

$$\begin{cases} \boldsymbol{\mu} & \leftarrow \boldsymbol{\mu} + \varepsilon_\mu \sum_{n=1}^N w(f(C_n))(\boldsymbol{x}_n - \boldsymbol{\mu}) \\ \boldsymbol{\Sigma} & \leftarrow \boldsymbol{\Sigma} + \varepsilon_\Sigma \sum_{n=1}^N w(f(C_n)) \left((\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^\intercal - \boldsymbol{\Sigma}\right), \end{cases} \tag{3}$$

where ε_μ and ε_Σ are step sizes, and $w()$ is a weighting function for a score $f(C_n)$. This procedure is based on a well-known method of parameter update based on a gradient. These parameters are explained in detail in [20]. Once the parameter is updated, new gene vectors are sampled from the Gaussian distribution, i.e., $\{\boldsymbol{x}_n\}_{n=1}^N \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Compared with the genetic algorithm, CMA-ES assumes the continuous gene vector and provides a simple (well-known) gene generation algorithm. However, the actual DNN configuration is represented by binary and discrete values in addition to continuous values, and the genetic algorithm is more flexible for directly considering these variables.

3. STRUCTURE OPTIMIZATION

3.1. Representation of structured DNN

When applying structure optimization to DNN, one issue is how to maintain the computational efficiency of the DNN while allowing flexible configurations. For a DNN formed by a simple cascade of multiple layers, the computation required for training and evaluation is based on layer-wise coarse grain matrix operation. Therefore, it is efficiently computed on GPGPU, which is a very important property for practical speech applications. However, if arbitrary connections are allowed between neurons, the computation is small grained, and efficient computation becomes difficult. To overcome this problem as well as to make compact gene expression possible, we consider DNNs formed as a directed acyclic graph (DAG) of neuron modules. It is an extension of the standard multi layer-structured DNNs by allowing arbitrary connections between layers with a constraint that it does not have directed cycles. By analogy to neuroanatomy, we refer to the neuron module as the nucleus. Right hand side of Figure 1 shows an example. In the figure, it is referred to as a "phenotype" in contrast to the "genotype" that is explained in the next subsection.

Arbitrary feed-forward structures are represented without limitation if we choose the nucleus to be equal to a single neuron. However, if there are a large number of small nuclei, it is difficult to efficiently perform computations on GPGPU. Therefore, we consider nuclei with a moderate number of neurons. A DAG of nucleus allows efficient application of RBM-based pre-training. That is, nuclei are pre-trained simply one by one from the input side to the output side of the DAG structure. If a nucleus has multiple inputs, they are concatenated to form a single input vector. The transformations to obtain the inputs are always already computed because there is no directed cycle. Fine tuning by back-propagation is also easily applied.

3.2. Genetic representation

To efficiently represent arbitrary DAG structures as genes, we utilize the fact that the nodes of a DAG can be numbered so that all the directed connections face the same direction from a lower numbered node to a higher numbered node. This property is easily confirmed by considering the lemmas that there exists at least one node that does not have a parent, and a graph obtained by removing such a node is also a DAG.

Once all the nodes of a DAG are numbered, then the connections can be represented by a lower triangular binary matrix $\boldsymbol{C}$ where an

element $c_{i,j}$ is one if nucleus i receives a connection from nucleus j and zero otherwise. A gene representation of the graph structure is obtained by arranging the elements of $\boldsymbol{C}$ into a vector in a predefined order. If a real number is used as a gene element, as in CMA-ES, binary values are obtained, for example, by first rounding the number to an integer and then obtaining a residue divided by two. We assumed the maximum number of nuclei to be pre-determined. However, genes are allowed to have defunct nuclei that lack input and/or output connection and the effective number of nuclei can change during the optimization process. The matrix in Figure 1 is the genotype of the DNN structure shown in the same figure.

Without loss of generality, it can be assumed that there is only one input nucleus that directly connects to the input, which makes implementation a bit easier. For this purpose, we introduced a special "input nucleus" that simply outputs the same vector as the input. Additionally, we introduced an "extraction nucleus" that extracts a part of an input vector to allow separate treatment of sub-dimensions. In the example in Figure 1, the DNN consists of one input nucleus (nucleus id = 0), seven extraction nuclei (id = 1,2,..,7), four sigmoid nuclei (id = 8,9,10,11), and one output softmax nucleus (id = 12). The input to the network is a concatenation of seven frames of speech features, and the output of the seven extraction nuclei corresponds to each of the original frames.

4. EXPERIMENTS

We performed two types of experiments. The first is a preliminary experiment based on frame-wise phone recognition. An in-house phone corpus was used where utterances were given by a single male speaker. The second is the main experiment, where continuous DP matching based spoken digit detections was performed using DNN-based features. The AURORA2 [21] corpus was used for this task.

For the training and evaluation of DNNs utilizing GPGPU, we implemented software using the Theano python library [22]. Experiments were performed using the TSUBAME2.5 supercomputer. In the following experiments, 62 NVIDIA K20X GPGPUs were used in parallel through the message passing interface (MPI).

4.1. Preliminary experiment using single-speaker phone recognition

4.1.1. Experimental setup

The training and development sets were speech utterances given by a male speaker that totaled 12 minutes and 3.5 minutes, respectively. The speech features were 12 MFCC, energy, and their deltas. Segment features were used as an input to DNNs having 130 (=26x5) dimensions in total concatenating five frames. The genes specified a DNN structure with at most four plastic nuclei that were subject to the optimization in terms of sizes, learning rates, and connections. Extraction nuclei were used to represent the subvectors of a composite feature vector. The input and extraction nuclei were fixed, and their connections were not changed during the optimization. Five ($= N_0$) initial genes were manually prepared. The DNNs were trained to predict 19 subsets of Japanese phonemes, and frame accuracy was evaluated. Both CMA-ES and GA used real number sequences as genes in this experiment. The mapping from the real value to the nucleus sizes was given by Equation (4), and the mapping to the learning rates was given by Equation (5). For the mutation process of GA, Gaussian noise with zero mean and 0.05 standard deviation was uniformly added to the gene.

$$Size = ceil\left(10^{x}\right), \quad (4)$$

$$learnrate = ceil\left(10^{5x}/100000\right). \quad (5)$$

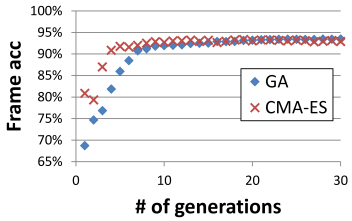


Fig. 2. Preliminary experimental results.

CMA-ES used the frame accuracy of the development set as the fitness function. GA used a weighted sum of frame accuracy and network size, normalized in each generation so that a gene with 95% normalized accuracy performance was evaluated as equal to the top one if it had a 50% normalized network size. The network size was the sum of the sizes of affine transformation matrices in a DNN. The population size N was set to 62 for both algorithms. The tournament size M was four for the gene selection in GA.

4.1.2. Results

Figure 2 shows the best frame accuracy among each generation obtained during the optimizations. The horizontal axis is the number of generations, and the vertical axis is the frame accuracy. In the early generations, CMA-ES gave better accuracy than GA showing its efficiency in the search. After approximately 10 iterations, both of them gave similar performance, showing good improvement from the initial generation. Thus, the experiment verified the effectiveness of both evolutionary algorithms for the structure optimization.

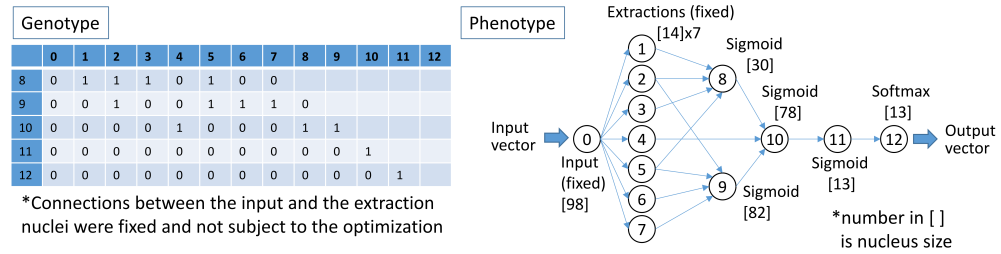
4.2. Spoken digit detection

4.2.1. Experimental setup

Spoken digit detection was performed based on continuous DP matching using DNN extracted digit posterior features. The digit posterior features had 13 dimensions to represent 0 to 9, oh, and two types of silences. The evaluation was based on frame-wise mean average precision (MAP) [23]. In DP hypotheses, the same frame may be included in multiple candidate segments starting and ending at different time frames with different distance scores. For the MAP evaluation, the minimum of the matching distance is adopted at each frame.

DNN structure optimizations by evolutionary algorithms were performed using the training set of the AURORA2 corpus. Among the training set, utterances from five male and five female speakers were used to provide keywords to detect. Another five male and five female speakers were used to provide target utterances within which to search for the keywords and were used to evaluate the MAP score during the optimizations, where 50 keywords and 100 utterances per keyword were randomly selected to evaluate the MAP score. The rest of the training data from 90 speakers were used to pretrain and fine-tune the DNN parameters. The speech features were 13 MFCCs and an energy. Seven frames were concatenated to form a composite feature vector with 98 dimensions in total.

The genes specified a DNN structure with at most five plastic nuclei. GA used binary values to represent network connections, which was more convenient to directly specify the mutation flip rate than using real numbers. CMA-ES requires the use of real numbers. However, because the variance is automatically adjusted in the optimization process, it does not require manually setting the flipping



	0	1	2	3	4	5	6	7	8	9	10	11	12
8	0	1	1	1	0	1	0	0					
9	0	0	1	0	0	1	1	1	0				
10	0	0	0	0	1	0	0	0	1	1			
11	0	0	0	0	0	0	0	0	0	0	1		
12	0	0	0	0	0	0	0	0	0	0	0	1	

Fig. 1. Example of DAG of nucleus DNN and a part of a gene corresponding to the structure. This structure was obtained by genetic algorithm at the 30th generation using the GA(MAP, Size, Sim) strategy described in Section 4.2.2. Compared to one of the most similar initial DNNs, connections from nuclei 2 to 9 and 5 to 8 were added.

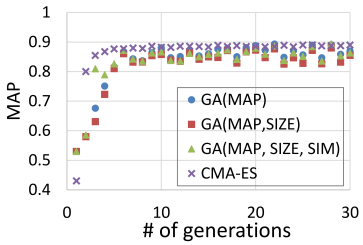


Fig. 3. Spoken digit detection results using AURORA2 corpus. Clean condition training was used.

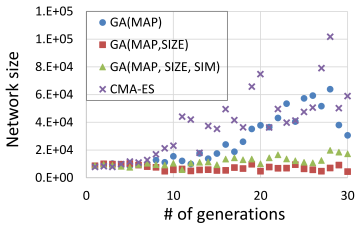


Fig. 4. Network sizes of DNNs that gave the best MAP score in each generation.

probability. The population size was set to 62. As the fitness function, CMA-ES used the MAP score. For GA-based optimization, the following three strategies were explored.

GA(MAP) Uses MAP score to compare two genes.

GA(MAP,Size) Uses MAP score. However, gene a wins over gene b that has a higher MAP score if the difference of the scores is less than 0.02 and the network size of gene a is less than 90% of gene b.

GA(MAP,Size, Sim) In addition to strategy 2, structure similarity is considered when mating two genes by selecting the best partner among 5 candidates.

This flexibility of selecting candidates with the above advanced criteria is an advantage of GA on CMA-ES. Training and evaluating a DNN took around 30 to 60 minutes depending on individuals.

4.2.2. Results

Figure 3 shows the best MAP score among a generation using the clean training condition, and Figure 4 shows the sizes of the DNNs corresponding to Figure 3. As in the preliminary experiment, CMA-ES gave better results than the GA-based optimizations in the early generations. After approximately 15 iterations, the difference between CMA-ES and GA(MAP) was small, and both of them gave improvements from the initial generation, although the scores by CMA-ES looked more stable. When the three GA strategies were compared, generally GA(MAP) gave the best MAP score and GA(MAP, Size) the worst. However, the differences were not large. There were, in contrast, large differences in their network sizes. GA(MAP, Size, Sim) and GA(MAP, Size), which consider network size in the selection process, gave much smaller network sizes than GA(MAP) and CMA-ES. GA(MAP, Size, Sim) and GA(MAP, Size) are advantageous when running spoken digit detection on resource-limited hardware. The structure shown in Figure 1 is the 30th generation DNN with the best MAP score obtained by GA(MAP, Size, Sim).

Table 1. *MAP scores on test sets. Optimizations were performed using the multicondition training set. DNN(ini) is the result of the first generation, DNN(GA) is the result of GA at the 30th generation. DNN(expt) is the result of tuning by a human expert.*

Features	setA(clean)	setA(SNR15)	setB(SNR15)
MFCC	0.38	0.22	0.21
DNN(ini)	0.44	0.36	0.33
DNN(GA)	0.81	0.75	0.74
DNN(expt)	0.80	0.73	0.73

Table1 shows the test set MAP scores of DNNs optimized/trained using the multicondition training set of the AURORA2 corpus. For the structure optimization, GA(MAP, SCORE, SIM) was used. Test set A was noise closed, and test set B was open. As can be seen, DNN-based digit posterior features were already better than MFCC in the first generation, and further gains were obtained by GA. The MAP scores given by DNN at the 30th generation were 0.81 for the clean test data and 0.75 and 0.74 for the noisy test data of set A and B, respectively, which were comparable or slightly better than tuned results by a human expert.

5. SUMMARY

A DAG of nucleus DNNs has been proposed that is suited for GPGPU computation and for compact gene expression. Joint structure and parameter optimizations have been explored based on evolutionary algorithms. The experimental results showed that both CMA-ES and GA gave large improvements from the initial generation without tuning by human experts. Future work will include applying the framework to large vocabulary speech recognition.

6. ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Numbers 26280055, 25280058. Part of this work was also supported by the Kayamori Foundation of Information Science Advancement.

7. REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] B. Kingsbury, J. Cui, X. Cui, M.J.F. Gales, K. Knill, J. Mamou, L. Mangu, D. Nolden, M. Picheny, B. Ramabhadran, et al., "A high-performance Cantonese keyword search system," in *Proc. ICASSP*. IEEE, 2013, pp. 8277–8281.

[3] C. W. Chau, S. Kwong, C. K. Diu, and W. R. Fahrner, "Optimization of HMM by a genetic algorithm," in *Proc. ICASSP*. IEEE, 1997, vol. 3, pp. 1727–1730.

[4] S. Kwong, C. W. Chau, K. F. Man, and K. S. Tang, "Optimisation of HMM topology and its model parameters by genetic algorithms," *Pattern Recognition*, vol. 34, no. 2, pp. 509–522, 2001.

[5] F. Yang, C. Zhang, and T. Sun, "Comparison of particle swarm optimization and genetic algorithm for HMM training," in *Proc. ICPR*, 2008, pp. 1–4.

[6] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Proc. ICASSP*. IEEE, 2013, pp. 8609–8613.

[7] S. Watanabe and J. Le Roux, "Black box optimization for automatic speech recognition," in *Proc. ICASSP*. IEEE, 2014, pp. 3256–3260.

[8] A. R. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[9] L. Deng, D. Yu, and J. Platt, "Scalable stacking and learning for building deep architectures," in *Proc. ICASSP*. IEEE, 2012, pp. 2133–2136.

[10] H. Hermansky, DPW Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *Proc. ICASSP*. IEEE, 2000, vol. 3, pp. 1635–1638.

[11] F. Grezl, M. Karafiát, S. Kontár, and J. Cernockỳ, "Probabilistic and bottle-neck features for LVCSR of meetings.," in *Proc. ICASSP*. IEEE, 2007, pp. 757–760.

[12] K. Shinoda and T. Watanabe, "Acoustic modeling based on the MDL criterion for speech recognition," in *Proc. Eurospeech*, 1997, vol. 1, pp. 99–102.

[13] S. S. Chen and P. S. Gopalakrishnan, "Clustering via the Bayesian information criterion with applications in speech recognition," in *Proc. ICASSP*. IEEE, 1998, vol. 2, pp. 645–648.

[14] S. Watanabe, Y. Minami, A. Nakamura, and N. Ueda, "Variational Bayesian estimation and clustering for speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 12, pp. 365–381, 2004.

[15] X. Liu and M. Gales, "Automatic model complexity control using marginalized discriminative growth functions," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1414–1424, 2007.

[16] T. Shinozaki, S. Furui, and T. Kawahara, "Gaussian mixture optimization based on efficient cross-validation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 3, pp. 540–547, 2010.

[17] L. Davis et al., *Handbook of genetic algorithms*, vol. 115, Van Nostrand Reinhold New York, 1991.

[18] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[19] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík, "Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009," in *Proc. the 12th annual conference companion on Genetic and evolutionary computation (GECCO)*, 2010, pp. 1689–1696.

[20] N. Hansen, "The CMA evolution strategy: a comparing review," in *Towards a new evolutionary computation*, pp. 75–102. Springer, 2006.

[21] D. Pearce and H. Hirsch, "The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *ISCA ITRW ASR2000*, 2000, pp. 29–32.

[22] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *Proc. the Python for Scientific Computing Conference (SciPy)*, 2010.

[23] K. Kishida, "Property of average precision and its generalization: an examination of evaluation indicator for information retrieval," Tech. Rep., National Institute of Informatics, 2005.