AUTOMATION OF SYSTEM BUILDING FOR STATE-OF-THE-ART LARGE VOCABULARY SPEECH RECOGNITION USING EVOLUTION STRATEGY

Takafumi Moriya[1], *Tomohiro Tanaka*[1], *Takahiro Shinozaki*[1], *Shinji Watanabe*[2], *Kevin Duh*[3]

[1]Tokyo Institute of Technology, Japan
[2]Mitsubishi Electric Research Laboratories (MERL), USA
[3]Nara Institute of Science and Technology, Japan

ABSTRACT

When building a state-of-the-art speech recognition system, the laborious effort required by human experts in tuning numerous parameters remains a prominent obstacle. The goal of this paper is to automate the process. We propose to tune DNN-HMM based large vocabulary speech recognition systems using the covariance matrix adaptation evolution strategy (CMA-ES) with a multi-objective Pareto optimization. This optimizes systems to achieve both high-accuracy and compact model size. An additional advantage of our approach is that it is efficiently parallelizable and easily adapted to cloud computing services. We performed experiments on the Corpus of Spontaneous Japanese (CSJ) using the TSUBAME 2.5 supercomputer. Compared with a strong manually tuned configuration borrowed from a similar system, our approach automatically discovered systems with lower WER by 0.48%, and systems with 59% smaller model size while keeping WER constant. The optimized training script is released in the Kaldi speech recognition toolkit as the first publicly available recipe for Japanese large vocabulary speech recognition.

Index Terms— large vocabulary speech recognition, evolution strategy, deep neural network, multi-objective optimization, Japanese spontaneous speech recognition

1. INTRODUCTION

Automatic speech recognition (ASR) systems consist of several statistical models that efficiently represent acoustic and linguistic patterns in speech [1]. The parameters of these models, such as the connection weights of a deep neural network (DNN) [2], the transition probabilities of a hidden Markov model (HMM) [3], and the arc weights of an weighted finite state transducer (WFST) [4], are estimated from large amounts of training data. Additionally, there are various meta-parameters, including model topology (the numbers of layers and hidden units), training configuration (e.g., the learning rate and the maximum number of iterations), and system organization (e.g., the choice of features). Meta-parameter tuning is essential for building state-of-the-art systems, but as a consequence of the increased complexity of recent ASR techniques, this process is becoming increasingly difficult and time-consuming even for human experts. Thus there is a strong demand to automate the tuning process using computers.

If we consider system performance as a function of the configuration of meta-parameters, then tuning can be formalized as an optimization problem. The function is highly complex and the analytic solution is infeasible. A grid search is also not tractable because the search space is exponential with respect to the number of meta-parameters. Solutions to this type of black-box optimization problem include evolutionary algorithms (e.g., genetic algorithm (GA) [5], evolution strategy (ES) [6]) and Bayesian optimization [7, 8]. GA imitates a biological evolutionary process that is repeated for generations in which a configuration corresponds to a gene and a system corresponds to an individual. A set of systems with different configurations are grown and evaluated, and a set of next-generation configurations is made from the current generation with the selection pressure based on fitness. ES is similar to GA but uses a real valued vector as a gene. Covariance matrix adaptation evolution strategy (CMA-ES) [9, 10, 11] is a type of ES that represents gene distribution using a multivariate Gaussian distribution. CMA-ES has demonstrated great performance in performing various tasks in a benchmarking workshop of black-box optimization [12]. Bayesian optimization defines a prior probability distribution of the black-box function and sequentially updates its posterior based on the evaluation results from the individuals investigated up to that point. The posterior distribution is used to define an acquisition function that determines the next configuration to try. Gaussian process [13] is a popular choice to represent the distribution over the function.

Notably, several researchers have applied GA to HMM acoustic modeling [14, 15, 16]. Previously, we have used CMA-ES to optimize the meta-parameters of a medium-size vocabulary DNN ASR system [17]. We have also applied CMA-ES and GA to optimize directed acyclic graph (DAG) based DNNs used as feature extractors for a keyword spotting system [18], where we found that CMA-ES and GA produced a similar final performance given sufficient generations, but CMA-ES was superior with a small number of generations. Promising results have been reported in these studies, but none of these optimization approaches have been applied to up-to-date large vocabulary speech recognition systems. In this paper, we scale up the CMA-ES approach to automated system building for complex large vocabulary DNN-HMM based speech recognition.

In the previous experiment that used CMA-ES and GA to optimize the DAG based DNN structure for keyword detection performance, we found that there were cases in which the model size became extremely large [18]. Because model size affects memory efficiency and processing speed not only for training but also for decoding, it must be controlled. In this case, we need a multi-objective optimization framework that optimizes recognition performance and model size jointly. Although multi-objective optimization is typically performed by using an weighted combination of individual objective functions, our goal is to automate the building process while avoiding the introduction of an additional weight parameter. Instead, we apply Pareto optimality [19, 20, 21], which can efficiently rank multi-objective scores without setting weights for weighted combi-

nation.

In the experiments, the systems are built using a massively parallel computing platform. The process is highly automated based on the multi-objective CMA-ES and implemented based on the Kaldi toolkit [22]. Compared with a strong manually tuned configuration borrowed from a similar system, our approach automatically discovered systems with lower WER by 0.48%, and systems with 59% smaller model size while keeping WER constant. The optimized training configuration is released in the Kaldi toolkit as a publicly available recipe for Japanese large vocabulary speech recognition [1].

The rest of the paper is organized as follows. In section 2, the black-box optimization algorithms are explained by focusing on CMA-ES and multi-objective optimization. Experimental conditions are described in Section 3, and the results are shown in Section 4. Finally, conclusions and future works are offered in Section 5.

2. FORMULATION

Let us represent an evaluation function $y = f(\boldsymbol{x})$ as the accuracy (or some other measure of correctness, such as negative sign of error counts) of an ASR system built from meta-parameters $\boldsymbol{x}$. The process of finding the optimal tuning parameter $\boldsymbol{x}^*$ to maximize ASR accuracy can be formulated as the following optimization problem:

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} f(\boldsymbol{x}). \tag{1}$$

As ASR systems are extremely complex, there is no analytical form for the solution. We must address this optimization problem without assuming specific knowledge for f, i.e., by considering f as a black box. Another important aspect of this problem is that evaluating the function value $f(\boldsymbol{x})$ is very expensive because training a large vocabulary model and computing its development set accuracy can take considerable time. The key point here is thus for the black box optimization to generate appropriate hypotheses $\hat{\boldsymbol{x}}$ to find the best $\boldsymbol{x}^*$ in the smallest number of ASR training and evaluation steps ($f(\boldsymbol{x})$) as possible.

2.1. CMA Evolution Strategy

CMA-ES iteratively estimates the parameters of a sample distribution for $\boldsymbol{x}$ such that the distribution is concentrated in a region with high values of $f(\boldsymbol{x})$. Hypotheses are sampled from a multivariate Gaussian distribution:

$$\hat{\boldsymbol{x}} \sim \mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{\theta}}) \;\text{ s.t. }\; \hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \underbrace{\int f(\boldsymbol{x})\mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta})d\boldsymbol{x}}_{\triangleq \mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\theta}]}. \tag{2}$$

CMA-ES iteratively re-estimates the mean vector and covariance matrix ($\boldsymbol{\theta}$) so as to optimize the expected value of $f(\boldsymbol{x})$ under the distribution. As the concrete functional form of f is unknown, it is difficult to address Eq. (2) analytically. To solve this problem, a natural gradient method [23] is used by taking a gradient of $\mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\theta}]$ with respect to $\boldsymbol{\theta}$. The expectation in the natural gradient can be approximately computed by using Monte Carlo sampling with the function evaluation $y_k = f(\boldsymbol{x}_k)$:

$$\nabla_{\boldsymbol{\theta}}\mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\theta}] \approx \frac{1}{K}\sum_{k=1}^{K} y_k \boldsymbol{F}_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\log\mathcal{N}(\boldsymbol{x}_k|\boldsymbol{\theta}), \tag{3}$$

[1] http://kaldi.sourceforge.net/index.html

where $\boldsymbol{x}_k$ is sampled from the previously estimated distribution $\mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_{n-1})$, and $\boldsymbol{F}$ is the Fisher information matrix. As CMA-ES uses a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta})$ with a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$, we can obtain the analytical forms of $\hat{\boldsymbol{\mu}}_n$ and $\hat{\boldsymbol{\Sigma}}_n$ by substituting the concrete Gaussian form into Eq. (3), leading to:

$$\begin{cases} \hat{\boldsymbol{\mu}}_n = \hat{\boldsymbol{\mu}}_{n-1} + \epsilon_{\boldsymbol{\mu}} \sum_{k=1}^{K} w(y_k)(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_{n-1}) \\ \hat{\boldsymbol{\Sigma}}_n = \hat{\boldsymbol{\Sigma}}_{n-1} + \epsilon_{\boldsymbol{\Sigma}} \sum_{k=1}^{K} w(y_k) \\ \qquad \cdot\big((\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_{n-1})(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_{n-1})^{\intercal} - \hat{\boldsymbol{\Sigma}}_{n-1}\big) \end{cases} \tag{4}$$

where $^{\intercal}$ is the matrix transpose. Note that, as in [9], y_k in Eq. (3) is approximated in Eq. (4) as a weight function $w(y_k)$, defined as:

$$w(y_k) = \frac{\max\{0, \log(K/2+1) - \log(\mathrm{R}(y_k))\}}{\sum_{k'=1}^{K}\max\{0, \log(K/2+1) - \log(\mathrm{R}(y_{k'}))\}} - \frac{1}{K}, \tag{5}$$

where $\mathrm{R}(y_k)$ is a ranking function that returns the descending order of y_k among $y_{1:K}$ (i.e., $\mathrm{R}(y_k) = 1$ for the highest y_k, $\mathrm{R}(y_k) = K$ for the smallest y_k, etc.). This equation only considers the order of y, which makes the updates less sensitive to evaluation measurements (e.g., to prevent from the different results using word accuracies and the negative sign of error counts).

2.2. Multi-objective Optimization

In addition to high accuracy, objectives such as small model size and fast run-time are also important in practice. Without loss of generalization, assume that we wish to maximize J objectives $F(\boldsymbol{x}) \triangleq [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_J(\boldsymbol{x})]$ with respect to $\boldsymbol{x}$ jointly. As objectives may conflict, we adopt a notion of optimality known as Pareto optimality [24]: First, if $f_j(\boldsymbol{x}_k) \geq f_j(\boldsymbol{x}_{k'})\ \forall\ j = 1, .., J$ and $f_j(\boldsymbol{x}_k) > f_j(\boldsymbol{x}_{k'})$ for at least one objective j, then we say that $\boldsymbol{x}_k$ *dominates* $\boldsymbol{x}_{k'}$ and write $F(\boldsymbol{x}_k) \triangleright F(\boldsymbol{x}_{k'})$. Given a set of candidate solutions, $\boldsymbol{x}_k$ is *Pareto-optimal* iff no other $\boldsymbol{x}_{k'}$ exists such that $F(\boldsymbol{x}_k) \triangleright F(\boldsymbol{x}_{k'})$.

Pareto-optimality formalizes the intuition that a solution is good if no other solution outperforms (dominates) it in all objectives. Given a set of candidates, there are generally multiple Pareto-optimal solutions; this is known as the Pareto frontier. Note that an alternative approach is to combine multiple objectives into a single objective via an weighted linear combination: $\sum_j \beta_j f_j(\boldsymbol{x})$, where $\sum_j \beta_j = 1$ and $\beta_j > 0$. The advantage of the Pareto definition is that weights β_j need not be specified and it is more general, i.e., the optimal solution obtained by any setting of β_j is guaranteed to be included in the Pareto frontier.

CMA-ES can be extended to optimize multiple objectives by modifying the rank function $\mathrm{R}(y_k)$ used in Eq. (5). Given a set of solutions $\{\boldsymbol{x}_k\}$, we first assign rank = 1 to those on the Pareto frontier. Then, we exclude these rank 1 solutions and compute the Pareto frontier again for the remaining solutions, assigning them rank 2. This process is iterated until no $\{\boldsymbol{x}_k\}$ remain, and we obtain a ranking of all solutions according to multiple objectives in the end. The rest of CMA-ES remains unchanged; by this modification, future generations are drawn to optimize multiple objectives rather than a single objective. With some bookkeeping, this ranking can be computed efficiently in $O(J \cdot K^2)$ [25].

In our experiments, we jointly optimize for low word error rate and small model size. To prevent unreasonably high error rate solutions (that might otherwise have good model size) to appear on the Pareto frontier, we need an additional heuristic. Solutions with an

Algorithm 1 Multi-objective CMA-ES

Initialization of $\hat{\boldsymbol{\mu}}_0$ and $\hat{\boldsymbol{\Sigma}}_0$
for $n = 1$ to N **do**
  for $k = 1$ to K **do**
    Sample $\boldsymbol{x}_k$ from $\mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{\mu}}_{n-1}, \hat{\boldsymbol{\Sigma}}_{n-1})$
    Evaluate $F(\boldsymbol{x}_k) \triangleq [f_1(\boldsymbol{x}_k), f_2(\boldsymbol{x}_k), \ldots, f_J(\boldsymbol{x}_k)]$
  end for
  Rank $\{F(\boldsymbol{x}_k)\}_{k=1}^{K}$ according to Pareto optimality
  Update $\hat{\boldsymbol{\mu}}_n$ and $\hat{\boldsymbol{\Sigma}}_n$
end for
return subset of solutions $\{\boldsymbol{x}, F(\boldsymbol{x})\}$ that lie on the Pareto front (rank 1) of all stored $N \times K$ samples

error rate above a manually set threshold are penalized to not appear as rank 1.

Algorithm 1 summarizes the CMA-ES optimization procedure with Pareto optimality, which is used to rank the multi objectives $F(\boldsymbol{x}_k)$. The rank obtained is used to update the mean vector and covariance matrix of the CMA-ES. The CMA-ES gradually samples neighboring tuning parameters from the initial values, and finally provides a subset of solutions, $\{\boldsymbol{x}, F(\boldsymbol{x})\}$, that lie on the Pareto front (rank 1) of all stored $N \times K$ samples. Note that, because CMA-ES is a gradient method, initial values must be set. As CMA-ES assumes a multivariate Gaussian for $\boldsymbol{x}$, it is originally suitable for tuning parameters that take continuous values and requires some extra discretization for discrete value optimization. Finally, the evaluation of $F(\boldsymbol{x}_k)$ can be performed independently for each k. Therefore, it is easily adapted to parallel computing environments – such as cloud computing services – for shorter turnaround time. The number of samples, K, is automatically determined from the number of dimensions of $\boldsymbol{x}$ [9] or we can set it manually by considering computer resources.

2.3. Bayesian Optimization

While CMA-ES involves a distribution over the tuning parameter $\boldsymbol{x}$ and takes the expectation over $\boldsymbol{x}$, Bayesian optimization uses a probabilistic model of the output y and considers the expectation defined over y called an acquisition function. Several acquisition functions have been proposed [26]. Here we use expected improvement, which is suggested as a practical choice [8]. The improvement is defined as $\max\{0, y - y^*_{m-1}\}$ where $y^*_{m-1} = \max_{1 \le m' \le m-1} y_{m'}$ is the best score among $m-1$ previous scores. Bayesian optimization then performs a deterministic search for the next candidate $\hat{\boldsymbol{x}}_m$ by maximizing the expected improvement over y, $a^{EI}(\boldsymbol{x}_m)$:

$$\hat{\boldsymbol{x}}_m = \arg\max_{\boldsymbol{x}_m} \underbrace{\int \max\{0, y - y^*_{m-1}\} p(y|D_{1:m-1}, \boldsymbol{x}_m) dy}_{\triangleq a^{EI}(\boldsymbol{x}_m)} \quad (6)$$

where $p(y|D_{1:m-1}, \boldsymbol{x}_m)$ is a predictive distribution of y given observed data $D_{1:m-1} \triangleq \{\boldsymbol{x}_{1:m-1}, y_{1:m-1}\}$ and $\boldsymbol{x}_m$, modeled as a Gaussian process. Eq. (6) selects $\boldsymbol{x}_m$ that is likely to lead to a high score y_m with high confidence given the predictive distribution.

3. EXPERIMENTAL SETUP

Experiments were performed using the Kaldi speech recognition toolkit with speech data from the Corpus of Spontaneous Japanese (CSJ) [27]. We ran two separate experiments with training sets having different amount of data: one consists of 240 hours of academic presentations, whereas the other is a 100-hour subset. A common development set consisting of 10 academic presentations was used for performance computation in CMA-ES and Bayesian optimization. The official evaluation set defined in CSJ, which has 10 academic presentations that total 110 minutes, was used as the evaluation set.

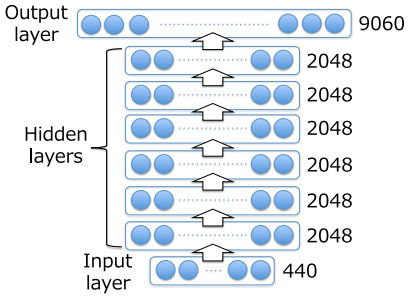

Fig. 1. Structure of the initial DNN.

Acoustic models were trained by first making a GMM-HMM by maximum likelihood estimation and then building a DNN-HMM by pre-training and fine-tuning using alignments generated by the GMM-HMM. For the performance evaluation of the system, the DNN-HMM was used as the final model. The language model was a 3-gram model trained on CSJ with academic and other types of presentations, which amounted 7.5 million words in total. The vocabulary size was 72k. Speech recognition was performed using the openfst WFST decoder [28]. As an initial configuration, we used a manually optimized Kaldi recipe for the Switchboard corpus (i.e. egs/swbd/s5b). We chose the recipe because the task was similar while the language was different, and because it was well tuned and publicly available.

In the evolution experiments, feature types, DNN structures, and learning parameters were optimized. These meta-parameters were implemented as configuration variables for training scripts. The first and second columns of Table 1 describe these variables. We specify three base feature types (feat_type) for GMM-HMM and DNN-HMM models: mel-frequency cepstrum coefficients (MFCC) [29], perceptual linear prediction (PLP) [30], and filter bank (FBANK). The dimensions of these features were 13, 13, and 36, respectively. The GMM-HMMs were trained with specified features and their delta and delta-delta. The DNN-HMMs were trained with the features with delta and delta-delta that were first transformed by fMLLR and then expanded to # splice pre and post context frames before inputting to the first DNN layer. Other settings were the same as those used in the Kaldi recipe. Because CMA-ES uses genes represented as real-valued vectors, mappings from a real scalar value to a required type are necessary, depending on the parameters. As for the mapping, we used $ceil(10^x)$ for positives integers (e.g. splice), 10^x for positive real values (e.g. learning rates), and $\mathrm{mod}\ (\lfloor ceil(abs(x) * 3) \rfloor, 3)$ for a multiple choice (feature type). The third column of the tables presents the initial meta-parameter configurations that were also used as baselines. Figure 1 shows the structure of the initial DNN.

The system training and evaluation were performed using the TSUBAME 2.5 supercomputer [2]. Population sizes for CMA-ES

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

Table 1. Meta-parameters subject for optimization and their automatically tuned results for the system using 240-hour training data. CMA-ES with Pareto was used for the automatic tuning.

Name of meta-parameters	Description	baseline	Values obtained by evolution using 240h training data					
			gen1	gen2	gen3	gen4	gen5	gen6
feat_type	MFCC, FBANK, or PLP	MFCC	MFCC	MFCC	MFCC	MFCC	MFCC	MFCC
splice	segment length for DNN	5	6	9	10	17	21	18
nn_depth	number of hidden layers	6	7	6	6	6	5	7
hid_dim	units per layer	2048	1755	1907	2575	1905	2904	3304
param_stddev_first	init parameters in 1st RBM	1.0E-1	1.1E-1	1.3E-1	1.1E-1	1.2E-1	0.7E-1	0.6E-1
param_stddev	init parameters in other RBMs	1.0E-1	1.0E-1	1.3E-1	1.0E-1	2.3E-1	1.9E-1	1.6E-1
rbm_lrate	RBM learning rate	4.0E-1	5.2E-1	5.7E-1	4.1E-1	4.7E-1	3.6E-1	3.6E-1
rbm_lrate_low	lower RBM learning rate	1.0E-2	1.3E-2	1.1E-2	0.8E-2	0.7E-2	0.8E-2	1.1E-2
rbm_l2penalty	RBM Lasso regularization	2.0E-4	2.1E-4	2.2E-4	1.2E-4	1.6E-4	1.9E-4	1.5E-4
learn_rate	learning rate for fine tuning	8.0E-3	7.3E-3	6.5E-3	7.8E-3	4.4E-3	5.3E-3	3.7E-3
momentum	momentum for fine tuning	1.0E-5	0.9E-5	0.9E-5	0.4E-5	0.9E-5	0.4E-5	0.7E-5

Table 2. Word error rate and DNN size of base systems.

Training data	Dev set	Eval set
MFCC 100h	14.4	13.1
PLP 100h	14.5	13.1
FBANK 100h	15.1	13.8
MFCC 240h	13.5	12.5
PLP 240h	13.6	12.5
FBANK 240h	14.1	13.0

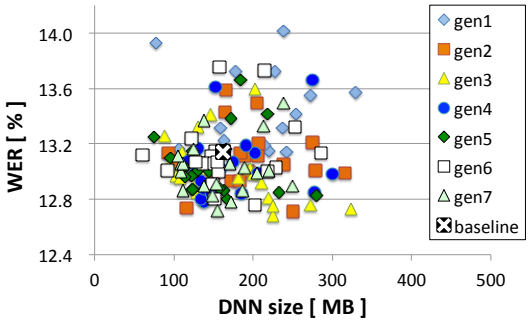

Fig. 2. CMA-ES based optimization when using 100-hour training data. The horizontal axis is the DNN size and the vertical axis is the word error rate of the evaluation set. Results of n-th generation are denoted as "gen n".

were 20 for the 100-hour training set and 44 for the 240-hour training set. Multiple NVIDIA K20X GPGPUs were used in parallel through the message-passing interface (MPI). For the CMA-ES optimization, we used the Python version of the implementation [3]. For the Bayesian optimization, we used the Spearmint package [4]. As the initial condition of the Bayesian optimization, we specified the range of the meta-parameters between 20% and 600% of the baseline configuration. The MFCC based baseline system with the 240-hour training set took 12 hours for the RBM pre-training and 70 hours for fine-tuning. The maximum word error rate thresholds for multi-objective optimization were set to include the top 1/2 and 1/3 of the populations at each generation, respectively, for the trainings using the 100 and 240-hour data sets.

4. RESULTS

Table 2 shows word error rates and DNN sizes for systems with the default configuration using the 100- and 240-hour training sets with one of the three types of features. Among the features, MFCC was the default in the Switchboard recipe, and it yielded the lowest word error rates for the development set for both of the training sets. The corresponding word error rates for the evaluation set were 13.1% and 12.5% for the 100- and 240-hour training sets, respectively.

Figure 2 shows a scatter plot of the CMA-ES based single-objective optimization using the 100-hour training data. The evolution was performed so as to minimize the development set word error rate. The horizontal axis is the DNN size and the vertical axis is the word error rate of the evaluation set. The initial mean vector of the multivariate Gaussian for CMA-ES was set equal to the baseline settings. The baseline marked on the figure is the MFCC based system. Ideally, we want systems on the lower-left side of the plot. The distribution is shown to shift with the progress of generations toward lower word error rates and lower DNN file sizes from the baseline. Similarly, Figure 3 shows the results of the multi-objective optimization using CMA-ES with the Pareto rank. Using the Pareto rank, systems were ranked based on the development set word error rate and the DNN model size. The result is similar to that produced using CMA-ES, but the distribution is oriented more to the lower-left side of the plot. Figure 4 presents the results using the Bayesian optimization for the optimization. In this case, the initial configuration is not directly specified but ranges of the meta-parameters are specified. We found that specifying a proper range was actually not straightforward and required knowledge of the problem. On one hand, if the ranges are too wide, the initial samples are coarsely distributed in the space, and it is likely that the systems have lower performance. On the other hand, if the ranges are too narrow, it is likely that the optimal configuration is not included in the search space. As a result, the improvement with the Bayesian optimization was smaller than that with the CMA-ES. Careful setting of the ranges might solve the problem but would again assume expert human knowledge.

Figures 5 and 6 are plots of the DNN model size and word error rate of the best system in each generation. Figure 5 is the result of the development set and Figure 6 is the result of the evaluation set. In

[3] https://www.lri.fr/~hansen/cmaes_inmatlab.html
[4] https://github.com/JasperSnoek/spearmint

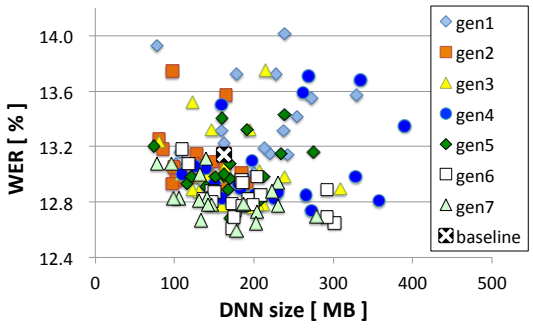

Fig. 3. CMA-ES and Pareto based multi-objective optimization when 100 hour training data was used. The horizontal axis is the DNN size and the vertical axis is the word error rate of evaluation set. Results of n-th generation are denoted as "gen n".

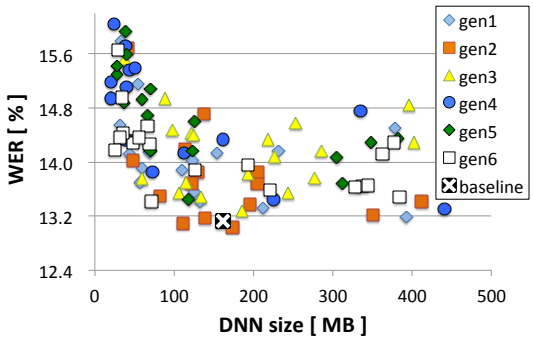

Fig. 4. Results when the Bayesian optimization was used with 100-hour training data. The horizontal axis is the DNN size and the vertical axis is the word error rate of the evaluation set. Results of n-th generation are denoted as "gen n".

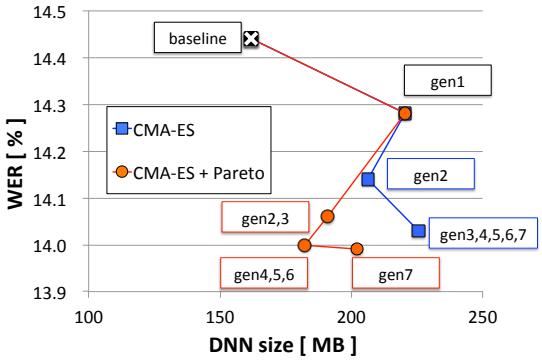

Fig. 5. The DNN model size and the development set word error rate of the best system. At each generation, the best system was selected that yielded the lowest word error rate among the initial to the current generation.

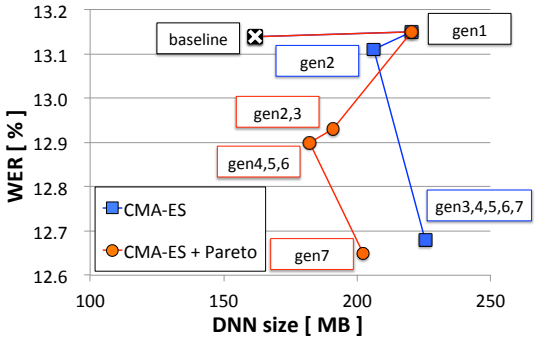

Fig. 6. The DNN model size and the evaluation set word error rate of the best system selected based on development set. At each generation, the best system was selected that yielded the lowest development set word error rate among the initial to the current generation.

both cases, the best system was selected based on the development set word error rate among the systems from the initial to the current generation. The evaluation set word error rate obtained by the best system after the 7th iterations was 12.7%, both when CMA-ES was used alone and when CMA-ES was used with Pareto. However, a smaller DNN model size was obtained by using CMA-ES with Pareto. The DNN model size when CMA-ES was used alone was 225.5 Mbytes, whereas it was 202.4 Mbytes when CMA-ES was used with Pareto, which was 89.8% of the former. The selected feature type was all MFCC except for the seventh generation, which was PLP.

Figure 7 shows a plot of the CMA-ES with Pareto based optimization using the 240-hour training data. The horizontal axis is the DNN model size and the vertical axis is the development set word error rate. Due to the increased data size and time constraints, a time limit was introduced for the training at each generation. If a system did not finish the training within four days, it was interrupted and the last model in the iterative back-propagation training at that timing was used as the final model. Depending on the generation, approximately 70% of the individuals completed the training within the limit. The figure shows that the distributions shifted toward lower word error rates and lower DNN file sizes with the progress of generations.

Figure 8 shows the word error rates of the best system selected within each generation based on the development set word error rate when the 240-hour training set was used. Although the development set error rate monotonically decreased with the number of the generation, the evaluation set error rate appeared to be saturated after the fourth generation, which might have resulted from over fitting to the development set because we used the same development set for all the generations. The lowest word error rate of the development set was obtained at the 6th generation. The corresponding evaluation set error rate was 12.1%. The difference in the evaluation set word error rates between the baseline (12.5%) and the optimized system (12.1%) was 0.48% and this was statistically significant under the MAPSSWE significance test [31]. The relative word error reduction was 3.8%.

If desired, we can choose a system from the Pareto frontier that best matches to the required balance of the word error rate and the model size. Figure 9 shows the Pareto frontier derived from the results from the initial to the 6th generation using the 240-hour training data. The figure thus shows that if we choose a system with approx-

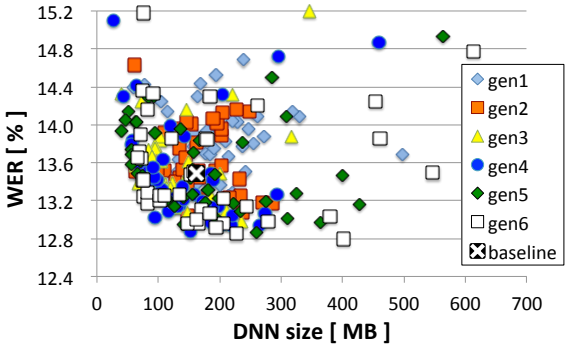

Fig. 7. The DNN model size and the development set word error rate when the 240-hour training set was used. Results of n-th generation are denoted as "gen n".

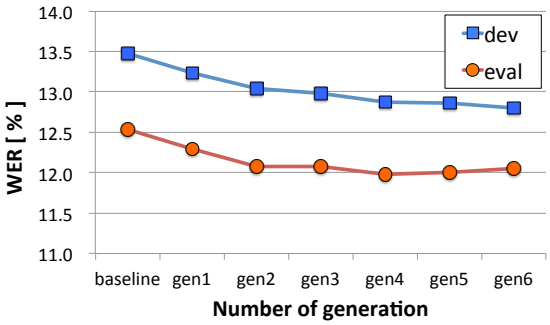

Fig. 8. The development and evaluation set word error rates of the best system at each generation when the 240h training set was used. In the figure "dev" and "eval" indicate the results of the development and the evaluation sets, respectively.

imately the same word error rate as the initial model, we can obtain a reduced model size that is only 41% of the baseline. That is, the model size was reduced by 59%. The decoding time of the evaluation set by the reduced model was 15.9 minutes, which was 85.4% of 18.7 minutes by the baseline. Similarly, the training time of the reduce model was 54.3% of the baseline model.

Columns 4 to 9 of the Table 1 show meta-parameter configurations obtained as the result of evolution using the 240-hour training set. These are the configurations that yielded the lowest development set word error rates at each generation. When we analyze the obtained meta-parameters, it is notable that MFCC was selected at all the generations. There were systems using PLP or FBANK in the population, but they did not produce the best result. Although the changes were not monotonic for most of the meta-parameters, we found that splice size was increased more than three times from the initial model. We also note that the learning rate decreased more than half from the initial condition.

As a supplementary experiment, sequential training [32] was performed using the best model at the 4th generation as an initial model. As the sequential training is computationally intensive, it took an extra 7 days. After the training, the word error rate was further reduced and 10.9% was obtained for the evaluation set. This

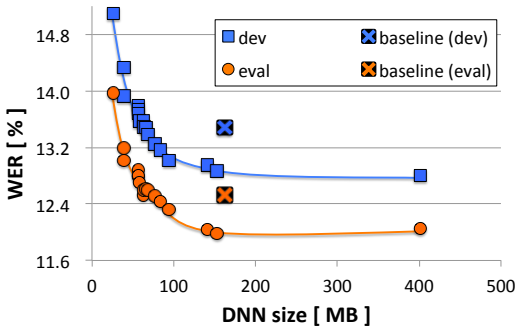

Fig. 9. Pareto frontier derived from the results from the initial to the 6th generation using 240-hour training data. In the figure "dev" and "eval" indicate the results of the development and the evaluation sets, respectively.

was lower than the word error rate of 11.2% obtained with sequential training using the baseline as the initial model. The difference was statistically significant, which affirms the effectiveness of the proposed method.

5. CONCLUSIONS

To automate the process of building a state-of-the-art large vocabulary speech recognition systems, we propose to use covariance matrix adaptation evolution strategy (CMA-ES). Further, we applied the combination of CMA-ES and Pareto rank to perform multi-objective optimization that considered both word error rate and model size while minimizing the tuning factors that require human experts. The proposed automation method was applied to build DNN-HMM based systems using data from the Corpus of Spontaneous Japanese (CSJ). In the experiments using 100-hour training set, CMA-ES, CMA-ES with Pareto and Bayesian optimization were compared. Both of the CMA-ES methods yielded lower word error rates than the baseline. By using CMA-ES with Pareto to jointly minimize the word error rate and the DNN model size, a system was obtained that has a similar word error rate with a smaller DNN model size compared to using only CMA-ES with the word error rate as its single objective. CMA-ES was more convenient to optimize speech recognition systems than Bayesian optimization, which requires ranges of the meta-parameters to be specified. For the 240-hour training set, a 3.8% relative word error rate reduction from the baseline was obtained. If a system was chosen that had approximately the same word error rate as the initial model instead of minimizing the word error rate, a reduced model size was obtained that was only 41% of the baseline. Future work includes applying the proposed method for more complex systems and improving the efficiency of the evolution strategy.

6. ACKNOWLEDGEMENTS

The work of T. Moriya, T. Tanaka, T. Shinozaki and K. Duh was supported by JSPS KAKENHI Grant Number 26280055. The work of S. Watanabe was supported by MERL.

7. REFERENCES

[1] S. Furui, "Recent progress in corpus-based spontaneous speech recognition," *IEICE Transactions on Information and Systems*, vol. E88-D, pp. 366–375, 2005.

[2] 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.

[3] M. Gales and S. Young, "The application of hidden Markov models in speech recognition," *Signal Processing*, vol. 1, no. 3, pp. 195–304, 2007.

[4] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16, pp. 69–88, 2002.

[5] L. Davis, Ed., *Handbook of genetic algorithms*, vol. 115, Van Nostrand Reinhold New York, 1991.

[6] H. G. Beyer and H. P. Schwefel, "Evolution strategies – a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[7] J. Mockus, "On Bayesian methods for seeking the extremum," in *Proceedings of the IFIP Technical Conference*, London, UK, UK, 1974, pp. 400–404, Springer-Verlag.

[8] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems 25*, 2012.

[9] 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.

[10] Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi, "Bidirectional relation between CMA evolution strategies and natural evolution strategies," in *Proc. Parallel Problem Solving from Nature (PPSN)*, 2010, pp. 154–163.

[11] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 949–980, 2014.

[12] 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.

[13] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*, MIT Press, 2006.

[14] 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.

[15] 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.

[16] 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.

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

[18] T. Shinozaki and S. Watanabe, "Structure discovery of deep neural network based on evolutionary algorithms," in *Proc. ICASSP*, 2015, pp. 4979–4983.

[19] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *Evol. Comput.*, vol. 15, no. 1, pp. 1–28, Mar. 2007.

[20] M. Laumanns and J. Ocenasek, "Bayesian optimization algorithms for multi-objective optimization," in *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, London, UK, UK, 2002, PPSN VII, pp. 298–307, Springer-Verlag.

[21] B. Sankaran, A. Sarkar, and K. Duh, "Multi-metric optimization using ensemble tuning.," in *HLT-NAACL*, 2013, pp. 947–957.

[22] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlıcek, Y. Qian, P. Schwarz, J. Silovskı, G. Stemmer, and K. Veselı, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[23] S. Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[24] K. Miettinen, *Nonlinear Multiobjective Optimization*, Springer, 1998.

[25] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, 2002.

[26] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[27] S. Furui, K. Maekawa, and H. Isahara, "A Japanese national project on spontaneous speech corpus and processing technology," in *Proc. ASR'00*, 2000, pp. 244–248.

[28] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "Openfst: A general and efficient weighted finite-state transducer library," *Implementation and Application of Automata*, pp. 11–23, 2007.

[29] S.B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transaction on Acoustic Speech and Singal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[30] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc. Am*, vol. 87, no. 4, pp. 1738–1752, 1990.

[31] L. Gillick and S. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *Proc. ICASSP*, 1989, pp. 532–535.

[32] K. Vesely, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proc. Interspeech*, 2013, pp. 2345–2349.