Neural Architecture Search
For LF-MMI Trained Time Delay Neural Networks

Shoukang Hu, Xurong Xie, Mingyu Cui*, Jiajun Deng*, Shansong Liu, Jianwei Yu, Mengzhe Geng,
Xunying Liu, Member, IEEE, Helen Meng, Fellow, IEEE

Abstract—State-of-the-art automatic speech recognition (ASR) system development is data and computation intensive. The optimal design of deep neural networks (DNNs) for these systems often require expert knowledge and empirical evaluation. In this paper, a range of neural architecture search (NAS) techniques are used to automatically learn two types of hyper-parameters of factored time delay neural networks (TDNN-Fs): i) the left and right splicing context offsets; and ii) the dimensionality of the bottleneck linear projection at each hidden layer. These techniques include the differentiable neural architecture search (DARTS) method integrating architecture learning with lattice-free MMI training; Gumbel-Softmax and pipelined DARTS methods reducing the confusion over candidate architectures and improving the generalization of architecture selection; and Penalized DARTS incorporating resource constraints to balance the trade-off between performance and system complexity. Parameter sharing among TDNN-F architectures allows an efficient search across up to 72 different systems. Statistically significant word error rate (WER) reductions of up to 1.2% absolute and relative model size reduction of 31% were obtained over a state-of-the-art 300-hour Switchboard corpus trained baseline LF-MMI TDNN-F system featuring speed perturbation, i-Vector and learning hidden unit contribution (LHUC) based speaker adaptation as well as RNNLM rescoring. Performance contrasts on the same task against recent end-to-end systems reported in the literature suggest the best NAS auto-configured system achieves state-of-the-art WERs of 9.9% and 11.1% on the NIST Hub5’00 and RT03s test sets respectively with up to 96% model size reduction. Further analysis using Bayesian learning shows that the proposed NAS approaches can effectively minimize the structural redundancy in the TDNN-F systems and reduce their model parameter uncertainty. Consistent performance improvements were also obtained on a UASpeech dysarthric speech recognition task. Our code is available at https://github.com/shhu101/TDNN-F_NAS.

Index Terms—Neural Architecture Search, Time Delay Neural Network, Speech Recognition

I. INTRODUCTION

State-of-the-art automatic speech recognition (ASR) systems are becoming increasingly complex. Deep learning techniques play a key role in these systems [1]–[7] and have in recent years evolved into a large set of advanced deep neural network (DNN) models. These include the traditional hybrid HMM-DNN architecture [7]–[12] featuring convolutional neural networks (CNNs) [8], [9], time delay neural networks (TDNNs) [1], [5]–[7] or recurrent neural networks (RNNs) [10]–[12] and their long short-term memory variants [13], [14]; and the recently emerging all neural end-to-end (E2E) modelling paradigm represented by listen, attend and spell (LAS) [15], connectionist temporal classification (CTC) [16], RNN transducers (RNN-T) [17] and neural transformer models [18]–[21].

The development of these systems is data and computation intensive. The optimal design of neural architectures in these systems often requires a large set of hyper-parameters encoding varying structural configurations to be set, for example, the hidden layer dimensionality and connectivity between different layers. To date these are largely determined based on expert knowledge or empirical choice. As explicitly training and evaluating the performance of all possible neural structural configurations is highly expensive, the need of deriving suitable automated neural network architecture learning techniques [22], [23] for speech recognition systems becomes particularly salient.

To this end, neural architecture search (NAS) approaches [24] have gained increasingly interest in recent years in both the computer vision [25]–[44] and speech [45]–[58] communities. The key objectives of NAS methods can be formulated as three fold. First, to allow the best system to be selected, it is crucial for NAS methods to produce an accurate performance rank ordering over different candidate neural architectures. Second, when designing practical systems operating on a given ASR accuracy performance target, preference should be given to neural architectures with fewer parameters in order to reduce the parameter uncertainty and minimize the risk of overfitting to limited training data. Furthermore, to ensure the NAS algorithms’ scalability and efficiency on large data sets, a compact search space containing all candidate neural architectures of interest and exploiting the structural commonalities among them needs to be defined.

Earlier forms of NAS techniques were based on neural evolution [59]–[63], where genetic algorithms were employed during mutation and crossover rounds by randomly selecting architecture choices. Performance and efficiency of these evolution based NAS methods heavily depends on the precise choice over parent model structures, mutation population groups and off-springs. For example, a pairwise competition based tournament selection approach [64] was used to sample parents in [27], [28], while a Pareto optimality based multi-objective approach was used in [29] to select neural architectures and adjust the trade-off between the predicted

*Two authors contributed equally. Shoukang Hu (e-mail: shhu@se.cuhk.edu.hk), Mingyu Cui, Jiajun Deng, Shansong Liu, Jianwei Yu, Mengzhe Geng, Xunying Liu (e-mail: xyliu@se.cuhk.edu.hk), Helen Meng (e-mail: hmmeng@se.cuhk.edu.hk) are with the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong SAR, China. Xurong Xie (e-mail: xurong@iscas.ac.cn) is with Institute of Software, Chinese Academy of Sciences, Beijing, China. (Corresponding author: Xunying Liu.)
performance and number of parameters. NAS methods based on Bayesian Optimization (BO) [65] were also explored to predict the performance rank ordering. Performance ranking prediction over candidate architectures was learned in a non-parametric fashion using Gaussian Process (GP) models [66] in [30], [67], where the similarity metric among candidate architectures was efficiently computed using kernel functions.

Reinforcement learning (RL) based NAS approaches [25], [26] have also been investigated. In these methods, the search space is formulated as a discrete state space and the generation of neural architectures is regarded as the agent’s actions in response to the reward based on the performance of the sampled models. Explicit system training and evaluation of candidate architectures are required in the above existing NAS techniques. In addition, as the architecture hyper-parameters and actual DNN parameters are separately learned, for example, within the RL controller and candidate neural network systems, a tighter integration of both is preferred.

Alternatively, differentiable neural architectural search (DARTS) techniques [31]–[38] can be used. Neural architecture search is efficiently performed over an over-parameterized super-network model that contains all possible candidate architectures to be considered. Within such a super-network, the underlying NAS problem is transformed into the estimation of the weight parameters assigned to each candidate neural architecture. After the super-network model containing both architecture weights and normal DNN parameters has been trained to convergence, the optimal architecture is obtained by pruning lower weighted paths. One prominent advantage of DARTS based NAS methods is that they allow both the architecture selection and candidate DNN parameters to be consistently optimized within the same super-network model.

In this paper, a range of DARTS based NAS techniques are used to automatically learn two architecture hyper-parameters that heavily affect the performance and model complexity of state-of-the-art lattice-free Maximum Mutual Information (LF-MMI) trained factored time delay neural network (TDNN-F) [1], [5]–[7] acoustic models: i) the left and right splicing context offsets; and ii) the dimensionality of the bottleneck linear projection at each hidden layer. These include the standard DARTS method that fully integrates the estimation of architecture weights and TDNN-F parameters in LF-MMI training; the Gumbel-Softmax DARTS technique producing approximately one-hot architecture distributions to reduce the confusion during model search; the pipelined DARTS method that circumvents the overfitting of architecture weights using validation data; and the penalized DARTS approach that further incorporates a resource penalty to flexibly adjust the trade-off between performance and system complexity. Parameter sharing among candidate architectures allows efficient search over a large number (up to $2^{28}$) of different TDNN-F systems to be performed.

Experiments conducted on a state-of-the-art 300-hour Switchboard corpus trained baseline LF-MMI TDNN-F system featuring speed perturbation, i-Vector and learning hidden unit contribution (LHUC) [68] based speaker adaptation as well as RNNLM rescoring suggest the NAS configured TDNN-F models consistently outperform the baseline systems using manually designed configurations or random architecture search. Significant absolute word error rate (WER) reductions up to 1.2% and model size reduction of 31% relative were obtained. Performance contrasts on the same task against most recent hybrid and end-to-end attention and transformer based systems reported in the literature [69]–[72] suggest our best NAS auto-configured system achieve state-of-the-art WERs of 9.9% and 11.1% on the NIST Hub5’00 and R03s test sets respectively with up to 96% model size reduction. Further analysis using Bayesian learning shows the proposed NAS approaches can effectively minimize the structural redundancy in the TDNN-F systems and reduce their model parameter uncertainty. Consistent performance improvements were also obtained on a UASpeech [73] dysarthric speech recognition task.

The main contributions of this paper are summarized below:

1) This paper presents the first use of DARTS based NAS techniques to automatically learn architecture hyper-parameters that directly affect the performance and model complexity of state-of-the-art LF-MMI trained TDNN-F acoustic models. In contrast, previous NAS researches conducted on similar systems either used a) evolutionary algorithms requiring expert setting of initial genes and long evaluation time for each individual candidate architecture [49] (up to 4 days even with a manual early-stopping mechanism) while in our NAS approaches the entire architecture search is performed over all possible $2^{28}$ candidate systems and model training cycle is limited to approximately 6.6 GPU days; or b) architecture sampling based straight-through gradient approach [53] on a TDNN-CTC end-to-end system producing much higher WERs (12.6% and 23.2%) on the swbd and callhm subsets of Hub5’00 test set than our NAS auto-configured TDNN-F systems on the same data (6.9% and 13.0%) presented in this paper.

2) To facilitate efficient search over a very large number of TDNN-F systems, this paper presents the first use of a flexible model parameter sharing scheme that is tailor-designed for specific hyper-parameters contained in TDNN-F systems, to the best of our knowledge. The generic nature of the proposed NAS methods accompanying parameter sharing technique also allows them to be employed to improve the efficiency and scalability of similar neural architecture design issues encountered during system development for end-to-end approaches including transformers [54], [55].

3) This paper presents the best published NAS based LF-MMI TDNN-F system performance reported in the literature on the 300-hour Switchboard task to the best of our knowledge. Performance contrasts on the same task against most recent hybrid and end-to-end attention and transformer based systems reported in the literature [69]–[72] suggest our best NAS auto-configured system achieves state-of-the-art WERs of 9.9% and 11.1% on the NIST Hub5’00 and R03s test sets respectively with model size reduction of up to 96% relative.
4) This paper further presents the earliest work on analysing the efficacy of NAS approaches when being used to minimize the structural redundancy in the TDNN-F systems and reduce their model parameter uncertainty when given limited training data. In contrast, only speech recognition accuracy performance and model size reduction are investigated in previous researches [49]–[56], [72].

A. Related works

With the rapid development of NAS techniques in the machine learning and computer vision communities, there has been also increasingly interest in applying these to speech recognition systems.

- Evolutionary algorithms were used in [49] to automatically learn a series of architecture hyper-parameters in time delay neural networks (TDNNs) [1], [5], and achieved 0.9% absolute WER reduction and 36% relative model size reduction over the Kaldi recipe trained baseline system on a spontaneous Japanese speech recognition task [74].

- Genetic algorithms were also used in [50] to automatically configure transformer model architectures. The NAS configured transformer achieves 12.9% WER on the Wall Street Journal (WSJ) task, outperforming the baseline transformer system by 0.6% absolute WER reduction.

- DARTS based NAS method have been investigated to automatically configure hyper-parameters for CNNs [51], [52], ST-NAS [53] and transformer models [54], [55] respectively. In particular, the concurrent work ST-NAS [53] applied an architecture sampling based straight-through gradient approach on a TDNN-CTC end-to-end system. WERs of 12.6% and 23.2% were reported on the swbd and callhm subset of the Hub5’00 test set on the Switchboard conversational telephone speech recognition task. The ST-NAS configured system outperformed the baseline system by 2.0% and 2.3% absolute WER reduction.

- Performance of various NAS techniques were also evaluated in [56] on the TIMIT task for CNN models.

The rest of this paper is organized as follows. In section 2, a set of differentiable neural architecture search (DARTS) techniques are first introduced. These include the standard DARTS, Gumbel-Softmax DARTS, pipelined DARTS and penalized DARTS methods. Time delay neural networks are reviewed in Section 3. The search spaces for automatically learning the architecture weights and normal DNN parameters is used by each candidate system. For example, when selecting the TDNN-F hidden layer context offsets, the linear transformation is a binary-valued matrix for each candidate architecture, and the architecture weights are computed as, the indicator function. When selecting the dimensionality of the bottleneck linear projection at each hidden layer, the linear transformation \( \mathbf{W}_l = \mathbf{W}_l^{(1)} \mathbf{W}_l^{(2)} \) is a decomposed matrix, while \( \phi_l(\cdot) \) is also an identity matrix.

A. Softmax DARTS

In the conventional DARTS [31] system, a Softmax function is used to model the architecture selection weights \( \lambda_i^l \):

$$
\lambda_i^l = \frac{\exp(\log \alpha_i^l)}{\sum_{j=0}^{N_l-1} \exp(\log \alpha_j^l)}
$$

where \( \log \alpha_i^l \) is the architecture dependent parameter determining their contribution during NAS search.

When using the standard back-propagation algorithm to update the architecture weights parameter \( \lambda_i^l \), the loss function (including LF-MMI criterion [6] considered in this paper) is used by each candidate system. For example, when selecting the TDNN-F hidden layer context offsets, the linear transformation is a binary-valued matrix for each candidate architecture, and the architecture weights are computed as, the indicator function. When selecting the dimensionality of the bottleneck linear projection at each hidden layer, the linear transformation \( \mathbf{W}_l = \mathbf{W}_l^{(1)} \mathbf{W}_l^{(2)} \) is a decomposed matrix, while \( \phi_l(\cdot) \) is also an identity matrix.

B. Gumbel-Softmax DARTS

In order to address the above issue, a Gumbel-Softmax distribution [32], [33], [35], [75] is used to sharpen the architecture weights to produce approximately a one-hot vector. This allows the confusion between different architectures to be minimised. The architecture weights are computed as,

$$
\lambda_i^l = \frac{\exp((\log \alpha_i^l + G_i^l)/T)}{\sum_{j=0}^{N_l-1} \exp((\log \alpha_j^l + G_j^l)/T)}
$$

where \( G_i^l = -\log(-\log(U_i^l)) \) is the Gumbel variable, and \( U_i^l \) is a uniform random variable. When the temperature parameter is large, the Gumbel-Softmax distribution is close to being a one-hot vector.
porating the underlined neural network size is used, given as follows:

$$L = L_{LF-MM1} + \eta \sum_{l,i} \lambda_i^L C_i^l$$

where $L_{LF-MM1}$ is the lattice-free MMI criterion considered in this paper. $C_i^l$ is the term related with the model complexity of the $i$-th candidate considered at the $l$-th layer, which can be expressed in different forms, for example, the number of model parameters, floating-point operations (FLOPs) or the latency computed given the specified hardware. Unless otherwise stated in this paper, we treat $C_i^l$ as the number of parameters of the $i$-th candidate considered at the $l$-th layer. $\eta$ is the penalty scaling factor empirically set for different tasks.

### III. TIME DELAY NEURAL NETWORK

Time delay neural networks (TDNNs) [1], [5]–[7], [77], [78] based hybrid HMM-DNN acoustic models in recent years defined state-of-the-art speech recognition performance over a wide range of tasks, due to their strong power in modelling long range temporal dependencies in speech. In particular, the recently proposed factored TDNN systems [7] featuring lattice-free MMI sequence discriminative training [6] remain highly competitive against all neural end-to-end approaches to date [70]–[72], [78]–[81].

**Fig. 2.** An example TDNN architecture with splicing context offsets {-1,0} {0,1} {-3,0} {0,3} in turn from the bottom to the top layer. 

TDNN can be considered as a special form of one-dimensional convolutional neural networks (CNNs) [82] when parameters are tied across different time steps. An example TDNN model is shown in Fig. 2. The bottom layers of TDNNs are designed to learn a narrower temporal context span, while the higher layers to learn wider, longer range temporal contexts. One important type of hyper-parameters in TDNN models controlling its temporal modelling ability is the left and right splicing context offsets. These alter the temporal context ranges learned in each TDNN hidden layer. The splicing context offsets used in the example of Fig. 2 are {-1,0} {0,1} {-3,0} {0,3} from the bottom to the top layer. To further reduce the risk of overfitting to limited training data and the number of parameters, a factored TDNN (TDNN-F) model structure was proposed in [7], which compresses the weight matrix by using semi-orthogonal matrix decomposition. In this TDNN-F model, the hidden layer specific bottleneck.
projection dimensionality settings present another group of hyper-parameter that needs to be determined.

If we assume 7 possible context offset choices to the left and another 7 context offset choices to the right to be learned at each of the 14 TDNN-F layers, there are up to \(7^{28}\) context offset choices being considered in the search space. When determining the number of bottleneck dimensions, for example, out of a total of 8 possible settings, at each of the 14 hidden TDNN-F layers, again a very large number of possible system configurations of \(8^{14}\) need to be considered during NAS. In this paper, we adopt the DARTS based NAS methods presented in Sec. II to automatically learn the above discussed two sets of hyper-parameters of TDNN-F models. The over-parameterized super-network containing all possible left and right splicing context offsets and bottleneck projection dimensionality choices and the associated parameter sharing technique will be introduced in the next section.

IV. SEARCH SPACE AND PARAMETER SHARING

This section describes the search space and its implementation when NAS methods of Sec. II are used to automatically learn two sets of hyper-parameters of TDNN-F models: i) the left and right splicing context offsets; and ii) the dimensionality of the bottleneck linear projection at each hidden layer. Parameter sharing among candidate architectures used to facilitate efficient search over a large number of TDNN-F systems is also presented. Finally, NAS lattice used to extract top-ranked candidate models is introduced.

A. TDNN-F Context Offset Search Space

![Fig. 3. Example part of a super-network containing all the context offsets for a TDNN-F layer. Dashed lines with different colors represent different context offsets in each linear (left context) and affine (right context) transforms. The blue integers denote the super-network system using all the context offsets, while the red integers represent a candidate offset choice of \pm 1.](image)

Context offset settings play an important role in modelling the long temporal information in TDNN-F models. However, manually selecting context offsets is time-consuming for different applications. Inspired by the parameter-sharing used in earlier NAS research [26], we design a TDNN-F super-network (Fig. 3) to contain all possible context offsets to the left \((-d,0),\cdots,\{-1,0\},\{0,0\}\) and right \((\{0,0\},\{0,1\},\cdots,\{0,d\})\) at each layer during search. For the super-network system, it requires the sparse context connection weights to be densely set as 1 for all context offsets. Any candidate TDNN-F model with particular context offsets out of the total \((d+1)^2\) possible choices contained in the super-network is represented by setting the corresponding connection weights to be 1, while setting the others to be 0.

B. TDNN-F Bottleneck Dimensionality Search Space

![Fig. 4. Example part of a super-network containing different bottleneck projection dimensionality choices in the TDNN-F hidden layer. The full weight matrix is factored into one semi-orthogonal linear weight matrix \(W_{0,n-1}\) and one affine weight matrix \(W_{0,n-1}\). Architectures with different projection dimensions are represented by the corresponding submatrices starting form the first column.](image)

Similarly, a TDNN-F super-network containing all the candidate architectures with different projection dimensions is designed, as shown in Fig. 4 for one hidden layer. When applying the NAS methods in Sec.2, the \(\phi_i(\cdots)\) of Eq. 1 is set as an identity matrix. In common with the standard TDNN-F model, the weight matrix \(W_i\) of the \(i\)-th architecture choice in the \(l\)-th layer is factored into one semi-orthogonal weight matrix \(W_{0,n-1}\) and one affine weight matrix \(W_{0,n-1}\) as shown in Fig. 4. \(n_i\) is the dimensionality of the \(i\)-th architecture. Parameter sharing among different candidate architectures' linear matrices \(W_{0,k}\) (left to right from the first column) and affine matrices \(W_{0,k}\) (bottom to up from the first row) \((0\leq k\leq n-1)\) is implemented by taking the corresponding submatrices extracted from the largest matrix \(W_{0,n-1}\). Such sharing allows a large number of TDNN-F projection dimensionality choices at each of the 14 layers, e.g., selected from 8 values \(\{25, 50, 80, 100, 120, 160, 200, 240\}\) as considered in this paper, to be compared for selection during search. This leads to a total of \(8^{14}\) candidate TDNN-F systems to be selected from.

C. NAS lattice

![Fig. 5. Part of an example NAS lattice containing architecture weights. Blue integers denote different TDNN-F left or right context offset choices, while red integers are their associated weights. Among all 9 possible context choices shown in the figure, the brown colored path with \(+/-2\) offsets is chosen with the highest probability \(0.7\times0.8=0.56\).](image)

When the architecture weights are learned using various NAS methods presented in Sec. II, all the candidate architectures contained in the super-network (as introduced in Sec. IV-A and Sec. IV-B) can be represented by a NAS...
lattice carrying their associated weights as the measure of their ranking order. Top 1-best or top N-best candidate architectures can then be extracted from the resulting NAS lattice. An example part of a NAS lattice for determining the TDNN-F left and right context offsets is shown in Fig. 5.

V. EXPERIMENTS

This section is organized as follows. Firstly, the performance of learning two architecture hyper-parameters that heavily affect the performance and model complexity of state-of-the-art factored time delay neural network (TDNN-F) [1], [5]–[7] acoustic models trained on the benchmark 300-hour Switchboard corpus and featuring speed perturbation, i-Vector and learning hidden unit contribution (LHUC) [68] based speaker adaptation as well as RNNLM rescoring is presented in Sec. V-A. Further analysis using Bayesian learning are conducted to show that the proposed NAS approaches can effectively minimize the structural redundancy in the TDNN-F systems and reduce their model parameter uncertainty. Secondly, to further evaluate the performance of the proposed NAS techniques, they were applied to automatically configure the same two sets of hyper-parameters of a state-of-the-art dysarthric speech recognition task based on the UASpeech corpus [73], [83] in Sec. V-B.

All of our models were trained with one thread on a single NVIDIA Tesla V100 Volta GPU card. In the searching stage, TDNN-F super-network models are trained on the training set for 3 epochs, while architecture parameters of PipeSoftmax and PipeGumbel systems are updated for additional 3 epochs using a held-out data set by fixing the normal DNN parameters. Note that we randomly select 5% of the original training set as the held-out data set and T in the Gumbel-Softmax distribution is linearly annealed from 1 to 0.03 in our experiments. Once candidate TDNN-F models are derived from the searching stage, they are trained for 3 epochs from scratch. The matched pairs sentence-segment word error (MAPSSWE) based statistical significance test [84] was performed at a significance level α = 0.05.

A. NAS Experiments on 300-Hour Switchboard Task

This section presents our NAS experiments of learning either the left and right splicing context offsets or bottleneck linear projection dimensionality or both of them at each layer on the 300-hour (900-hour after speed perturbation) Switchboard telephone speech recognition task using the Kaldi toolkit [85]. In all our experiments, we follow the Kaldi chain model setup1, except that we used 40-dimension filterbank features for neural network training instead of the 40-dimensional Mel-frequency cepstral coefficients (MFCCs).

Task Description: The Switchboard I telephone speech corpus consists of approximately 300 hours audio data released by LDC (LDC97S62). The baseline GMM-HMM system was trained based on 40-dimension Mel-frequency cepstral coefficients (MFCCs) to generate alignments for the neural network training. For performance evaluation, a four-gram language model (LM) trained on the Switchboard and Fisher transcripts (LDC2004T19, LDC2005T19) was used to evaluate NIST HUB5’00 (LDC2002S09, LDC2002T43), RT03 (LDC2007S10) and RT02 (LDC2004S11) test sets. In addition, the Kaldi recipe LSTM recurrent neural network language model (RNNLM) trained on the Switchboard and Fisher transcripts (LDC2004T19, LDC2005T19) was used to rescore the nbest lists produced by the LF-MMI trained systems with a four-gram language model (LM). The performance of LF-MMI trained TDNN baseline system incorporated with i-Vector [86] and speed perturbation is shown in line 1 of Tab. I, Tab. II and Tab. III. Furthermore, the effects of LHUC [68] based speaker adaptation were investigated.

1) TDNN-F Context Offset Search: In this section, we describe the experimental results of searching context offsets at each factored TDNN layer by using various NAS methods of Sec. II. In Tab. I, system (1) is the baseline Kaldi recipe trained factored TDNN system. As a direct and exhaustive search over all possible context offset settings is infeasible, a heuristic based two stage manual search is adopted. First, the position of the single hidden layer where the left and right context offsets are both restricted to 0 (default position 4th hidden layer in Sys (1)), was relocated to the 2nd, 3rd, 5th and 8th hidden layers respectively. These changes produced manual systems (2)-(5) shown in Tab. I, and were designed to intuitively vary the ratio between the hidden layers in the lower section of the system modelling shorter range of temporal contexts, and those in the higher section capturing longer span information. No significant performance difference can be obtained over the baseline recipe setting (Sys (1)). Second, this is then followed by fixing the L/R contexts as 0 in the 4th layer while further enlarging those used by the higher positioned hidden layers (from layer 5 to 14) incrementally to ±6, ±9, ±12. Among the resulting manually configured systems (6)-(8), the setting of ±9 produced absolute WER reduction up to 0.6% (Sys (7) in Tab. I) across three test sets over the baseline recipe system (1). Based on these results, the following NAS experiments were conducted to perform the search over $7^{28}$ TDNN-F configurations with maximum context offsets set as ±62 at each layer. The remaining part of Tab. I shows the performance and corresponding hidden layer context offsets learned using the four DARTS based NAS methods of Sec. II, Softmax, Gumble-softmax (Gumble), Pipelined Softmax (PipeSoftmax) and Pipelined Gumbel-softmax (PipeGumbel). For each of these four techniques, the associated super-network system that averaging over all possible system configurations, and the top 1, 2 and 3 architecture after system retraining are presented in turn, for example, shown in system (9)-(12) for the Softmax DARTS method.

1Systems performing the search over $10^{28}$ TDNN-F choices with the maximum context offsets of ±9 can not produce better results in practice than those with the maximum context offsets of ±6. Hence, all the following NAS experiments perform the search over $7^{28}$ TDNN-F choices with the maximum context offsets of ±6.
TABLE I
PERFORMANCE (WER%) COMPARISON OF TDNN-F MODELS CONFIGURED WITH CONTEXT OFFSETS PRODUCED BY THE BASELINE SYSTEM, MANUAL-DESIGNED SYSTEMS, SOFTMAX DARTS (SOFTMAX), GUMBEL-SOFTMAX DARTS (GUMBEL), PIPELINE SOFTMAX DARTS (PIPE SOFTMAX), PIPELINED GUMBEL-SOFTMAX DARTS (PIPE GUMBEL) SYSTEMS DESCRIBED IN SEC. II. \{a, b\} \{c, d\} denotes context offsets \{c, 0\} to the left and \{0, d\} to the right used from a-tth layer to b-th layer inclusive. \dagger denotes a statistically significant difference is obtained over the baseline system (SYS (1)). (SWB1 and CHM denote the Switchboard and Callhome subsets of the HuB5'00 test set; FSH and SWB2 denote the FISHER and Switchboard subsets of the RT03S test set; SWB3, SWB4 and SWB5 denote three Switchboard subsets in the R020 test set.)

| Sys   | Method       | I-Vec +SP | Context Offsets               | Hub5'00 | RT03S | SWB3 | SWB4 | SWB5 | Avg | #param | Time |
|-------|--------------|-----------|--------------------------------|---------|-------|------|------|------|-----|-------|------|
| 1     | Baseline     | ✓         | \{1,1\}; \{1,2\}; \{1,3\}; \{1,4\}; \{1,5\}; \{1,6\}; \{1,7\}; \{1,8\}; \{1,9\}; \{1,10\}; \{1,11\}; \{1,12\} | 9.7     | 18.0  | 12.6 | 19.5 | 11.5 | 15.3 | 20.0 | 15.5 | 18.6M | 30h  |
| 2     | Softmax      |           | \{1,1\}; \{1,2\}; \{1,3\}; \{1,4\}; \{1,5\}; \{1,6\}; \{1,7\}; \{1,8\}; \{1,9\}; \{1,10\}; \{1,11\}; \{1,12\} | 9.4 \dagger 17.9 | 12.5 | 19.4 | 11.4 | 15.3 | 19.4 \dagger | 15.4 |       |      |
| 3     | PipeSoftmax  |           | \{1,1\}; \{1,2\}; \{1,3\}; \{1,4\}; \{1,5\}; \{1,6\}; \{1,7\}; \{1,8\}; \{1,9\}; \{1,10\}; \{1,11\}; \{1,12\} | 9.4 \dagger 17.8 | 12.3 \dagger 19.6 | 11.4 | 15.0 \dagger | 19.5 \dagger | 15.3 |       |      |
| 4     | Gumbel       |           | \{1,1\}; \{1,2\}; \{1,3\}; \{1,4\}; \{1,5\}; \{1,6\}; \{1,7\}; \{1,8\}; \{1,9\}; \{1,10\}; \{1,11\}; \{1,12\} | 9.6 \dagger 17.6 \dagger | 12.4 | 19.4 | 11.3 | 15.5 | 19.4 \dagger | 15.3 |       | 18.6M | 30h  |
| 5     | PipeGumbel   |           | \{1,1\}; \{1,2\}; \{1,3\}; \{1,4\}; \{1,5\}; \{1,6\}; \{1,7\}; \{1,8\}; \{1,9\}; \{1,10\}; \{1,11\}; \{1,12\} | 9.4 \dagger 17.9 | 12.3 \dagger 19.6 | 11.4 | 15.4 | 19.4 \dagger | 15.4 |       |      |
| 6     | PipeGumbel   |           | \{1,1\}; \{1,2\}; \{1,3\}; \{1,4\}; \{1,5\}; \{1,6\}; \{1,7\}; \{1,8\}; \{1,9\}; \{1,10\}; \{1,11\}; \{1,12\} | 9.4 \dagger 17.5 \dagger | 12.1 \dagger 19.0 \dagger | 11.4 | 15.1 | 18.9 \dagger | 15.1 \dagger | 14.9 \dagger | 30h  |
| 7     | PipeGumbel   |           | \{1,1\}; \{1,2\}; \{1,3\}; \{1,4\}; \{1,5\}; \{1,6\}; \{1,7\}; \{1,8\}; \{1,9\}; \{1,10\}; \{1,11\}; \{1,12\} | 9.6 \dagger 18.1 | 12.1 \dagger 18.9 \dagger | 11.3 | 15.0 \dagger | 18.8 \dagger | 15.1 \dagger |       |      |

results in Tab. I.

1) NAS configured TDNN-F systems (Sys (10), (14), (18), (22)) consistently outperform the baseline Kaldi recipe [85] TDNN-F system (Sys (1)). For example, PipeGumbel DARTS system (Sys (22)) significantly outperforms the baseline system (Sys (1)) by 0.7% and 1.0% absolute WER reductions on the CHM subset of Hub5'00 test set and SWB5 subset of RT02 test set.

2) Compared with the manually designed systems (Sys (2)-(8)) constructed using a total of 210 GPU hours (30*7 hours for 7 systems), the NAS configured TDNN-F systems (Sys (18) & (22)) produce comparable performance to the best manually crafted system (Sys (7)) and a much smaller number of GPU hours of 110 (reduced by 48% relative) during both architecture search and subsequent model retraining.

3) No significant WER difference averaged over all test sets is obtained among the top 1-best Softmax (Sys (10)), Gumbel-Softmax DARTS (Sys (14)), PipeSoftmax (Sys (18)) and PipeGumbel DARTS (Sys (22)) systems, while
the super-network training for pipelined DARTS systems (Sys (17) & (21)) were in practice found to converge faster than those of the non-pipelined counterparts (Sys (9) & (13)).

2) TDNN-F Bottleneck Projection Dimensionality Search: To further evaluate the performance of DARTS based NAS methods of Sec. II, a set of experiments comparable to those in Tab. I are conducted to learn the suitable bottleneck projection dimensionality choices at each factored TDNN layer. In Tab. II, system (1) is shown as the baseline Kaldi recipe trained TDNN-F system. Systems (2)-(8) are the manually designed factored TDNN-F systems by uniformly setting all the layer specific bottleneck projection dimensions to be one of the following values: 25, 50, 80, 100, 120, 160, 200 or 240.

Tab. II shows the performance of NAS based auto-configuration of bottleneck projection dimensions at each factored TDNN layer from the following choices: 25, 50, 80, 100, 120, 160, 200 or 240, using the four methods of Sec. II. This set leads to a total of 814 factored TDNN systems to be selected from. Again in common with the context offset NAS experiments of Tab. I, for each of these four techniques, Softmax, Gumble-softmax (Gumble), Pipelined Softmax (PipeSoftmax) and Pipelined Gumble-softmax (PipeGumbel), the performance of the associated super-network system

| Sys | Method     | I-Vec +SP | η (%) | Bottleneck Dim Index | Hab5** 00 | RB5** 00 | R65S | R62S | R62O | SWB3 | SWB4 | SWB5 | Avg | #param | Time |
|-----|------------|-----------|-------|----------------------|---------|---------|------|------|------|------|------|------|-----|--------|------|
| 1   | Baseline   | ✓         | -     |                      | 9.7     | 18.0    | 12.6 | 19.5 | 11.3 | 15.3 | 20.0 | 15.5 | 18.6M| 30h  |
| 2   | Manual     | ✓         | -     |                      | 1.0     | 18.8    | 13.1 | 20.7 | 11.9 | 15.0 | 20.9 | 15.3 | 16.3 | 7.5M | 22h  |
| 3   | PipeGumbel | 12.3      | 9.8   | 18.2                | 9.9     | 18.1    | 12.3 | 20.0 | 11.6 | 15.6 | 21.0 | 15.8 | 9.6M | 24h  |
| 4   | PipeGumbel | 15.0      | 5.5   | 6.2                 | 9.5     | 17.8    | 12.0 | 19.6 | 11.5 | 15.0 | 20.3 | 15.4 | 12.1M| 26h  |
| 5   | PipeGumbel | 15.0      | 5.5   | 6.2                 | 9.5     | 17.6†   | 12.0†| 19.4 | 11.3 | 14.8†| 20.4 | 15.3 | 13.7M| 27h  |
| 6   | PipeGumbel | 14.8      | 5.5   | 6.2                 | 9.6     | 17.8†   | 12.1†| 19.3 | 11.5 | 15.2 | 20.3 | 15.0↑| 15.4M| 28h  |
| 7   | PipeGumbel | 14.8      | 5.5   | 6.2                 | 9.8     | 18.2    | 12.7 | 19.9 | 12.1 | 16.0 | 20.5 | 15.9 | 22.0M| 32h  |
| 8   | PipeGumbel | 14.8      | 5.5   | 6.2                 | 10.0    | 18.6    | 12.6 | 19.8 | 12.1 | 15.8 | 20.5 | 15.9 | 25.3M| 34h  |

| Sys | Method       | I-Vec +SP | η (%) | Bottleneck Dim Index | Hab5** 00 | RB5** 00 | R65S | R62S | R62O | SWB3 | SWB4 | SWB5 | Avg | #param | Time |
|-----|--------------|-----------|-------|----------------------|---------|---------|------|------|------|------|------|------|-----|--------|------|
| 9   | Softmax Supernet | ✓         | 0     |                      | 9.6     | 18.0    | 12.2 | 19.3 | 11.5 | 15.2 | 20.0 | 15.4 | 25.3M| 45h  |
| 10  | Softmax Top1  | ✓         | 0     |                      | 0       | 17.7†   | 12.2 | 19.4 | 11.6 | 15.3 | 20.2 | 15.4 | 19.4M| 36h  |
| 11  | Softmax Top2  | ✓         | 0     |                      | 9.6     | 17.5†   | 12.2 | 19.7 | 11.1†| 15.1 | 19.9 | 15.4 | 18.1M| 24h  |
| 12  | Softmax Top3  | ✓         | 0     |                      | 9.4†    | 17.6†   | 12.0†| 19.5 | 11.3 | 15.2 | 19.9 | 15.3 | 20.7M| 24h  |
| 13  | Gumbel Supernet | ✓         | 0     |                      | 10.0    | 18.5    | 13.1 | 20.3 | 11.9 | 15.8 | 20.4 | 16.1 | 25.3M| 45h  |
| 14  | Gumbel Top1†  | ✓         | 0     |                      | 2       | 17.3†   | 12.0†| 19.1†| 10.7†| 14.8†| 19.3†| 15.0†| 17.2M| 30h  |
| 15  | Gumbel Top2†  | ✓         | 0     |                      | 2       | 17.4†   | 11.8†| 19.3 | 10.8†| 14.6†| 19.1†| 15.0†| 17.4M| 30h  |
| 16  | Gumbel Top3†  | ✓         | 0     |                      | 2       | 17.4†   | 11.9†| 19.1†| 10.8†| 14.7†| 19.3†| 15.0†| 17.4M| 30h  |
| 17  | PipeSoftmax Supernet | ✓         | 0     |                      | 9.7     | 18.2    | 12.6 | 20.4 | 11.4 | 14.6 | 20.0 | 15.7 | 25.3M| 45h  |
| 18  | PipeSoftmax Top1 | ✓         | 0     |                      | 9.5     | 18.3    | 12.5 | 19.7 | 11.5 | 15.4 | 19.8 | 15.6 | 15.1M| 27h  |
| 19  | PipeSoftmax Top2 | ✓         | 0     |                      | 9.7     | 18.1    | 12.6 | 20.0 | 11.7 | 15.4 | 20.1 | 15.7 | 15.6M| 26h  |
| 20  | PipeSoftmax Top3 | ✓         | 0     |                      | 9.8     | 18.5    | 12.8 | 20.1 | 11.7 | 15.5 | 20.6 | 15.9 | 15.3M| 27h  |
| 21  | PipeSoftmax Top1† | ✓         | 0     |                      | 9.5     | 17.8    | 12.2†| 19.6 | 11.4 | 14.9†| 20.3 | 15.4 | 11.0M| 24h  |
| 22  | PipeSoftmax Top1† | ✓         | 0.5   |                      | 9.8     | 17.8    | 12.3†| 20.1 | 11.3 | 15.0†| 20.3 | 15.6 | 9.6M | 23h  |

Tab. II shows the performance of NAS based auto-configuration of bottleneck projection dimensions at each factored TDNN layer from the following choices: 25, 50, 80, 100, 120, 160, 200 or 240, using the four methods of Sec. II. This set leads to a total of 814 factored TDNN systems to be selected from. Again in common with the context offset NAS experiments of Tab. I, for each of these four techniques, Softmax, Gumble-softmax (Gumble), Pipelined Softmax (PipeSoftmax) and Pipelined Gumble-softmax (PipeGumbel), the performance of the associated super-network system
When compared with the baseline Kaldi recipe [85] system (Sys (1)), the Gumbel and PipeGumbel systems (Sys (14), (24)) with a similar model size of approximately 17 million parameters achieve comparable or better performance. For example, the Gumbel Top 1 system (Sys (14)) produces 0.5% absolute significant WER reduction on average across all test sets. When compared with the best-performing manually designed system (Sys (5)), Gumbel Top 1 system (Sys (14)) also produces 0.3% statistically significant absolute WER reduction on average across all three test sets. If we further add the resource penalty to the objective loss function as presented in the penalized DARTS method of Sec. II-D, the PipeGumbel system (Sys (27), with penalty coefficient $\eta = 0.1$) can produce statistically significant absolute WER reductions ranging from 0.3% (SWB2 subset of Rt03S test set) to 0.9% (SWB4 subset of Rt02 test set) and a relative model size reduction of 33% over the baseline Kaldi recipe trained TDNN-F system (Sys (1)), by selecting fewer bottleneck projection dimensions at higher layers than the PipeGumbel system (Sys (24)) using no model size penalty.

The PipeGumbel system (Sys (28) with penalty coefficient $\eta = 1.0$) achieves the largest relative model size reduction of 46% with a marginal WER reduction of 0.1% when compared with the baseline recipe TDNN-F system (Sys (1)). This serves an alternative penalized PipeGumbel system setting to system (27), which favours the most aggressively compressed model architecture incurring no accuracy performance degradation.

Note that the resource penalty was not added to the architecture search stage of Softmax and Gumbel-Softmax DARTS systems due to the requirement of retraining super-network parameters for different resource penalty scaling factors. For efficiency, in the following NAS experiments where both the hidden layer L/R context offsets and bottleneck projection dimensions are automatically determined, only Pipelined Softmax and Gumbel-Softmax DARTS systems are used.

3) TDNN-F Context Offsets and Bottleneck Projection Dimensionality Search: The performance of Pipelined Gumbel-Softmax and Pipelined Softmax DARTS methods are further evaluated by searching both context offsets and bottleneck projection dimensionality at each factored TDNN layer. In Tab. III, their performance are again contrasted with those of the baseline Kaldi recipe system (Sys (1)), the best manually designed system (Sys (2)) and a random search configured system (Sys (3)). The manually designed system (2) is configured using the best-performing context offset and dimensionality settings selected from those manually designed systems in Tab. I and Tab. II. The Random Search system (3) is obtained by selecting the best performing model from 6 randomly sampled models with different context offsets or dimensionality choices during search. Systems (4)-(5) in Tab. III are produced by using the PipeSoftmax or PipeGumbel method to learn the context offsets and bottleneck projection dimensionality settings in a one by one, “two-stage” fashion. This means the architecture search over these two attributes are separately performed in each stage. In contrast, systems (6)-(7) are produced by using the PipeSoftmax or PipeGumbel method to learn these two attributes jointly in the same super-network during one single round of architecture search. For all systems in Tab. III, their performance prior to and after further applying additional LHUC [68] speaker adaptation and Kaldi recipe LSTM RNNLM rescoring are also shown.

The details of Kaldi recipe LSTM RNNLM rescoring are as follows: two unidirectional (forward/backward) context based long short-term memory recurrent neural network language models (LSTM LMs) [87] were built following the standard Kaldi recipe. The transcripts of both the Switchboard training data and the Fisher English corpora were encoded as 1024 dimensional embedding vectors to train the LSTM LMs. Each LSTM LM consisted of 2 unidirectional LSTM layers with 1024 cells. Projections were used inside the LSTMs to reduce the output dimensions to 512. A context splicing layer with ReLU activation was exploited before and after each LSTM layers. The splicing indices of the three context splicing layers were {-1, 0}, {-3, 0}, and {-3, 0}. Finally, an affine transformation was used to generate the output embedding vectors. For performance evaluation, the two LSTM LMs were employed in turn to rescore the lattices generated by the 4-gram LM.

Several trends can be observed from Tab. III.

1) By searching the context offsets and bottleneck projection dimensionality in a sequential manner, the PipeGumbel Top 1 system (Sys (5)) produces 0.5% (SWB1 subset of Hub5'00 test set) to 1.2% (SWB5 subset of Rt02 test set) absolute WER reductions on three test sets and a relative model size reduction of 31% over the baseline Kaldi recipe factored TDNN system (Sys (1)).

2) When compared with the manually designed system (2) in Tab. III, the PipeGumbel Top 1 system (Sys (5)) achieves comparable performance and reduces the overall system design and training time from 463 to 160 GPU hours. The PipeGumbel Top 1 system (Sys (5)) also outperforms the Random Search system (3) in Tab. III.

3) The two stage search over the above two groups of hyper-parameters (PipeGumbel Sys (5)) in practice outperforms performing a joint search over the two (PipeGumbel Sys (7)) by statistically significant absolute WER reduction of 0.3% on average across three test sets. One possible explanation is that increased modelling confusion and search errors are encountered in the latter

\[ \text{NAS selected projection dimensions at each layer:}\{80, 80, 80, 50, 80, 50, 80, 80, 100, 120, 160\} \text{ and context configurations}\{-2,2\},\{-2,4\},\{-5,5\},\{-6,6\},\{-6,5\},\{-6,6\},\{-6,6\},\{-6,6\},\{-6,6\},\{-6,6\},\{-6,6\},\{-6,6\},\{-6,6\},\{-6,6\},\{-6,6\}\]
TABLE III

| Sys  | Method       | I-Vec + SP | LHUC | LM | Context & Dim | η | Hub5* 00 | R03S | R02 | Avg | #param | Time |
|------|--------------|------------|------|----|---------------|---|----------|------|-----|-----|--------|------|
| 1    | Baseline     |            | -    |    |               |   | 9.7 18.0 | 12.6 19.5 | 11.5 15.5 | 20.0 | 15.5 18.6M | 30h  |
| 2    | Manual       |            | -    |    |               | 9.1 17.3 | 11.9 18.7 | 10.6 14.6 | 19.0 | 14.8 | 13.7M  | 463h |
| 3    | Random Search|            | -    |    |               |   | 9.5 18.2 | 12.3 19.7 | 11.0 15.0 | 20.1 | 15.3 13.8M | 162h |
| 4    | PipeSoftmax  (Top1) | ✓      | ✓    |    | 4-gram        | 0.1 | 9.5 17.4 | 12.2 19.5 | 11.2 15.0 | 19.5 | 15.2 | 13.7M  | 160h |
| 5    | PipeGumbel   (Top1) |         |      |    |               | 0.3 | 9.2 17.3 | 11.8 18.8 | 10.7 14.7 | 18.8 | 14.8 | 12.9M  | 162h |
| 6    | PipeSoftmax  (Top1) |         |      |    |               | 0.1 | 9.4 17.5 | 12.1 19.4 | 11.2 14.8 | 19.5 | 15.2 | 11.7M  | 162h |
| 7    | PipeGumbel   (Top1) |         |      |    |               | 0.3 | 9.3 17.6 | 12.1 19.2 | 11.2 15.2 | 19.2 | 15.1 | 11.7M  | 162h |

TABLE IV

| System | I-Vec + SP | LHUC | LM | Context Offset & Bottleneck Dim | η | Hub5* 00 | R03S | R02 | Avg | #Param | Parameter Variance |
|--------|------------|------|----|---------------------------------|---|----------|------|-----|-----|--------|---------------------|
| 1      | TDNN       | ✓    |    | 4-gram                          |   | 9.7 18.0 | 12.6 19.5 | 11.5 15.5 | 20.0 | 15.5 18.6M | -                  |
| 2      | NAS        |      |    | Manual                          | 9.2 17.3 | 11.8 18.8 | 10.7 14.7 | 18.8 | 14.8 | 12.9M  | 0.037               |
| 3      | Bayes      | ✓    |    | 4-gram                          | 9.4 17.3 | 12.1 19.2 | 11.4 14.7 | 19.1 | 15.1 | 18.6M  | 0.025               |
| 4      | TDNN       | ✓    |    | Manual                          | 9.3 17.5 | 11.8 18.8 | 11.1 14.7 | 18.6 | 14.8 | 12.9M  | -                  |

TABLE V

| Sys  | Method | Search | Retrain | LHUC | LM | Hub5* 00 | R03S | R02 | Avg | #param | Time |
|------|--------|--------|---------|------|----|----------|------|-----|-----|--------|------|
| 1    | NAS    | 300h   | 300h    |      | 4g | 9.6 19.8 | 13.2 20.9 | 11.6 15.6 | 21.8 | 16.4 | 13M    |
| 2    | NAS    | 300h   | 900h    |      |    | 9.5 17.5 | 12.2 19.7 | 11.4 15.1 | 19.6 | 15.3 | 13M    |
| 3    | NAS    | 900h   | 900h    |      |    | 9.2 17.3 | 11.8 18.8 | 10.7 14.7 | 18.8 | 14.8 | 13M    |

case when optimizing both attributes in the same super-network for the PipeGumbel Top 1 system (Sys (7)).
4) We further hypothesize that by reducing the model size and structural redundancy of TDNN-F systems, the auto-configured NAS systems, for example, using the best performing penalized PipeGumbel system (Sys (5) in Tab. III), with 12.9M parameters, the uncertainty associated with the resulting TDNN-F model parameters can also be reduced. In order to verify this hypothesis, we apply Bayesian estimation [88]–[93] to both the baseline Kaldi recipe configured TDNN-F system and the PipeGumbel NAS configured TDNN-F system (Sys (1) & Sys (5) in Tab. III, shown again as Sys (1) & (2) in Tab. IV). Efficient variational inference and parameter sampling based training approaches [93] were used to estimate the latent parameter posterior distributions in these Bayesian estimated systems. The performance of the corresponding two Bayesian TDNN-F systems are shown as system (3) and (4) in Tab. IV. It is found that the NAS auto-configured TDNN-F system (Sys
TABLE VI
Performance contrasts of TDNN-F/CNN-TDNN-F models configured with both context offsets and projection dimensionality produced by baseline system or Pipelined Gumbel (PipeGumbel) DARTS (after applying Bayesian LHUC and large RNNLMs) against other state-of-the-art systems conducted on the 300-hour Switchboard task. The overall WERs in "( )" are not reported by the original papers and are recalculated using the subset WERs.

| System | #Param | Hub5'00 Avg. | RT03S Avg. |
|--------|--------|--------------|-------------|
|        |        | CHM          | SWB1        | SWB2        |
| 1 RWTH SMBR BLSTM [69] | 13M | 6.7 14.7 10.7 | - - - |
| 2 + Affine transform based environment adaptation | - | 6.7 13.5 10.2 | - - - |
| 3 JHU ESPNet Transformer [70] | 19M | 9.0 18.1 13.6 | - - - |
| 4 Google Listen, Attend and Spell network + SpecAugment [71] | - | 6.8 14.1 (10.5) | - - - |
| 5 IBM LSTM based Attention encoder-decoder | 29M | 7.4 14.6 (11.0) | - - - |
| 6 + SpecAugment + weight noise [72] | 75M | 6.8 13.4 (10.1) | - - - |
| 7 | 280M | 6.4 12.5 (9.5) | 8.4 14.8 (11.7) |
| 8 LF-MMI TDNN + BLHUC + Large RNNLM | 19M | 7.6 14.4 11.0 | 8.9 14.3 11.7 |
| 9 LF-MMI NAS TDNN + BLHUC + Large RNNLM | 13M | 7.1 13.5 10.2 | 8.5 13.9 11.3 |
| 10 LF-MMI CNN-TDNN + BLHUC + Large RNNLM | 15.2M | 6.9 12.9 9.9 | 8.3 13.9 11.2 |
| 11 LF-MMI NAS CNN-TDNN + BLHUC + Large RNNLM | 10.8M | 6.9 13.0 9.9 | 8.3 13.7 11.1 |

(4) in Tab. IV) has a smaller average parameter latent distribution’s variance (0.025) compared with a larger one (0.033) associated with the baseline system using the baseline recipe configuration (Sys (3) in Tab. IV). Furthermore, no performance is obtained by applying Bayesian estimation to the NAS auto-configured factored TDNN-F system (Sys (2) vs. (4) in Tab. IV), in contrast to the significant WER reduction of 0.4% found on the baseline recipe systems without using NAS (Sys (3) vs. Sys (1)). These results suggest the proposed NAS approaches can effectively minimize the structural redundancy in the TDNN-F systems and reduce their model parameter uncertainty.

5) We also report an ablation study on the transferability of NAS configured hyper-parameters across data sets of different sizes, where we contrast the performance of transferring the 300-hour subset set (without speed perturbation) NAS learned TDNN-F context offsets and projection dimensions to the 900-hour full set (with speed perturbation) during system training (Sys (2) in Tab. V), against performing both NAS and system training exclusively using the 900-hour full set (Sys (3) in Tab. V). The significant WER differences observed between these two systems suggest further research is required to improve the transferability and generalization of NAS methods across different data sets and quantities.

6) By further incorporating both LHUC speaker adaptation [68] and the Kaldi recipe LSTM RNNLM rescoring, similar performance improvements can still be maintained. Statistically significant absolute WER reductions of 0.5% (on SWB1 subset of Hub5'00 test set and SWB4 subset of RT02 test set) to 1.0% (on CHM subset of Hub5'00 test set and SWB5 subset of RT02 test set) are obtained by the PipeGumbel Top 1 system (12) over the Kaldi recipe configured baseline system (8).

In the following experiments, the performance of the best PipeGumbel Top 1 system (Sys (12) in Tab. III) was further refined by incorporating the more powerful Bayesian LHUC adaptation [94] in place of the conventional deterministic LHUC adaptation, and a larger LSTM RNNLM with a doubled number of LSTM cells (2048) and projection dimensionality (1024) compared with the smaller LSTM RNNLM used in Tab. III. This is shown as system (9) in Tab. VI, together with a comparable PipeGumbel configured CNN TDNN-F system (11) whose TDNN layer context offsets and projection dimensionality settings are automatically determined. The baseline Kaldi configured TDNN-F and CNN TDNN-F systems are shown as system (8) and (10) respectively. The PipeGumbel NAS configured TDNN-F and CNN TDNN-F systems (Sys (9) & (11)) are further compared with the state-of-the-art performance obtained using a series of recent hybrid and end-to-end systems reported in the literature (Sys (1)-(7) in Tab. VI). Among these, system (1) and (2) in Tab. VI are the RWTH BLSTM hybrid systems without and with affine transformation for environment adaptation [69]. System (3) is the JHU ESPNet based transformer end-to-end system [70]. System (4) is the Google Listen, Attend and Spell end-to-end system built with SpecAugment [71]. System (5)-(7) are the IBM LSTM based attention encoder-decoder end-to-end systems constructed using SpecAugment and weight noise [72].

Competitive performance is obtained by the Pipelined Gumbel-Softmax searched TDNN-F and CNN TDNN-F systems (Sys (9) & (11)) on the CHM subset of Hub5'00 test set and RT03S test sets when compared with the other hybrid and end-to-end systems (Sys (1)-(7) in Tab. VI), while retaining more compact system sizes. In particular, the NAS configured CNN-TDNN-F system (Sys (11) with 10.8M parameters) achieves a state-of-the-art WER of 11.1% on the RT03S test set, outperforming the most complex and best performing IBM system (Sys (7) with 280M parameters) by a WER reduction of 0.6% absolute as well as a 96.1% relative model size reduction.

B. NAS Experiments on Dysarthric UASpeech Task

In order to further evaluate the performance of the proposed NAS techniques, they were applied to automatically configure the above discussed two sets of hyper-parameters of a state-of-the-art dysarthric speech recognition task based on the UASpeech corpus [73].
### Task Description

The UASpeech corpus is the largest publicly available dysarthric speech corpus that is designed as an isolated word recognition task [73]. Approximately 103 hours of speech was recorded from 29 speakers among which 16 are dysarthric speakers while the remaining 13 are healthy control speakers. For speech recognition system development, the entire corpus is further divided into 3 subset blocks per speaker, with each block containing different speech contents based on a mix of common and uncommon words. Among these, the same set of common words contents are used in all three blocks, while the uncommon words in each block are different. The data from Block 1 (B1) and Block 3 (B3) of all the 29 speakers are used as the training set (69.1 hours of audio, 99195 utterances in total), while the data of Block 2 (B2) collected from all the 16 dysarthric speakers (excluding speech from healthy control speakers) serve as the evaluation data set (22.6 hours of audio, 26520 utterances in total). After removing excessive silence at the start and end of speech audio segments [95], a combined total of 30.6 hours of audio data from Block 1 and 3 (99195 utterances) is used as the training set, while 9 hours of speech from Block 2 (26520 utterances) is used for performance evaluation. Similar as Sec. V-A, a GMM-HMM system was trained with 39-dimensional PLP features to generate alignments for the neural network training. The factored TDNN baseline featuring speed perturbation was trained following the Kaldi chain setup, except that i-Vector was not incorporated. Following the configurations specified in [95], [97], recognition is performed using a uniform language model with a word grammar network constructed using 255 text set words.

The performance comparison among the baseline TDNN-F system, a random search configured system and the PipeGumbel auto-configure TDNN-F system, before and after LHUC speaker adaptive training (SAT) and test time unsupervised adaptation were applied, are shown in Tab. VII. The same trend previously observed on the Switchboard task of Sec. V-A can be found again. First, the PipeGumbel auto-configure TDNN-F system (3) consistently outperforms the baseline TDNN-F system (1) and random search configured system (2). For example, statistically significant WER reduction of 1.0% on average and 33% relative model size reduction were obtained by PipeGumbel auto-configure TDNN-F system (3) when compared with the baseline system (1). Second, after further incorporating LHUC SAT and test time unsupervised adaptation, similar performance improvements can still be retained.

### VI. Conclusion

This paper presents a range of DARTS based neural architecture search techniques to automatically learn two groups of architecture hyper-parameters that heavily affect the performance and model complexity of state-of-the-art lattice-free MMI trained factored time delay neural network acoustic models: i) the left and right splicing context offsets; and ii) the dimensionality of the bottleneck linear projection at each hidden layer. Parameter sharing among candidate neural architectures was used to facilitate efficient search over a very large number of (up to $2^{26}$) different TDNN-F system configurations.

Experiments conducted on both the benchmark 300-hour Switchboard corpus and the 103-hour UASpeech dysarthric speech recognition task suggest the resulting NAS auto-configure TDNN-F models consistently outperform the baseline systems using manually designed configurations or random search by significant absolute word error rate (WER) reductions up to 1.2% and model size reduction of 31% relative. State-of-the-art recognition accuracy performance was obtained on the NIST Hub5’ 00 and Rt03s test sets and

---

**TABLE VII**

Performance (WER%, number of parameters) of TDNN-F systems configured with both context offsets and projection dimensionality produced by baseline setting, PipeGumbel-Softmax DARTS (PipeGumbel) or Random Search before and after applying LHUC SAT and test time unsupervised adaptation. Random Search selects the best performed model from 6 randomly sampled models (with maximum context offsets $\pm 3$). $\eta$ is the penalty factor in Eqn. (6). $\{(a, b)\}: \{-c, d\}$ denotes context offsets $\{-c, 0\}$ to the left and $(0, d)$ to the right used from $a$-th layer to $b$-th layer inclusive. The dimensionality index denotes the index of 8 dimensionality choices: $[25, 50, 80, 100, 120, 160, 200, 240]$. “Very Low”, “Low”, “Mid” and “High” denote the group of speakers with different intelligibility. $^†$ Denotes a statistically significant difference is obtained over the baseline Kaldi recipe systems (Sys (1) & (4)).

| Sys | Method | SP | LHUC SAT | Context offsets & Bottleneck Dim Index | WER% | #Param | Time |
|-----|--------|----|----------|----------------------------------------|-------|--------|------|
| 1   | Baseline (Manual) | ✓ | ✓ | - | 56.1 | 9.8M | PipeGumbel | 29.4 | 19.7 | Mid | 31.8 | Very Low | 28.4 | Time | 13.3 | 23.6 | 9.8M | 17h |
| 2   | Random Search (1 best) | ✓ | ✓ | - | 56.1 | 9.8M | PipeGumbel (Top1) | 0 | (1,3) | (9), (4,7) | (2,2) & 3 4 7 2 3 3 0 | 57.7 | 31.8 | 22.8† | 14.2 | 29.6 | 9.8M | 17h |
| 3   | Baseline (Manual) | ✓ | ✓ | - | 56.1 | 9.8M | PipeGumbel (Top1) | 0 | (1,3) | (9), (4,7) | (2,2) & 3 4 7 2 3 3 0 | 55.8† | 56.3 | 30.1 | 19.9 | 27.9 | 9.8M | 21h |
| 4   | Random Search (1 best) | ✓ | ✓ | - | 56.1 | 9.8M | PipeGumbel (Top1) | 0 | (1,3) | (9), (4,7) | (2,2) & 3 4 7 2 3 3 0 | 55.8† | 58.6 | 30.6† | 21.5† | 11.7† | 28.4† | 6.4M | 13h |

---

5The frames_per_eg variable was set to be 150,110,90,60,30.
6Following our previous work [96], data augmentation featuring speed perturbation of both control and dysarthric speech was conducted, which leads to an augmented training set of 130h.
compared with those of the most recent hybrid and end-to-end attention and transformer based systems reported in the literature [69]–[72]. Further analysis using Bayesian learning demonstrates that the proposed set of NAS approaches can effectively minimize the structural redundancy in the TDNN-F systems and reduce their model parameter uncertainty. The generic nature of these techniques also allows their wider application to other speech task domains as well as similar automatic neural architecture configuration problems when developing speech recognition systems using end-to-end approaches.

ACKNOWLEDGMENT

This research is supported by Hong Kong Research Grants Council GRF grant No. 14200218, 14200220, 14200021, Theme-based Research Scheme T45-407/19N, Innovation & Technology Fund grant No. ITS/254/19, PiH/350/20, InP/275/20 and Shun Hing Institute of Advanced Engineering grant No. MMT-p1-19.

REFERENCES

[1] A. Waibel, “Consonant recognition by modular construction of large phonemic time-delay neural networks,” in Advances in neural information processing systems, 1989, pp. 215–223.
[2] B. Kingsbury, “Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling,” in ICASSP, 2009, pp. 3761–3764.
[3] K. Veselić, A. Ghoshal, and al et al., “Sequence-discriminative training of deep neural networks,” in INTERSPEECH, 2013, pp. 2345–2349.
[4] H. Su, G. Li, D. Yu, and F. Seide, “Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription,” in ICASSP, 2013, pp. 6664–6668.
[5] V. Puddinti, D. Povey, and S. Khudanpur, “A time delay neural network architecture for efficient modeling of long temporal contexts,” in INTERSPEECH, 2015, pp. 3214–3218.
[6] D. Povey and et al., “Purely sequence-trained neural networks for ASR based on lattice-free mini.” in INTERSPEECH, 2016, pp. 2751–2755.
[7] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, “Semi-orthogonal low-rank matrix factorization for deep neural networks,” in INTERSPEECH, 2018, pp. 3743–3747.
[8] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, “Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition,” in ICASSP, 2012, pp. 4277–4280.
[9] O. Abdel-Hamid, L. Deng, and D. Yu, “Exploring convolutional neural network structures and optimization techniques for speech recognition,” in INTERSPEECH, 2013, pp. 73–5.
[10] L. R. Medsker and L. Jain, “Recurrent neural networks,” Design and Applications, vol. 5, pp. 64–69, 2001.
[11] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in ICASSP, 2013, pp. 6645–6649.
[12] A. Amberkar, P. Awasarmal, G. Deshmukh, and P. Dave, “Speech recognition using recurrent neural networks,” in International Conference on Current Trends towards Converging Technologies, 2018, pp. 1–4.
[13] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.
[14] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition,” in INTERSPEECH, 2014, pp. 338–342.
[15] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” in ICASSP, 2016, pp. 4960–4964.
[16] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in ICML. ACM, 2006, pp. 369–376.
[17] A. JGraves, “Sequence transduction with recurrent neural networks,” in Proc. ICML Workshop on Representation Learning, 2012.
[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in neural information processing systems, 2017, pp. 5998–6008.
P. J. Angeline, G. M. Saunders, and J. B. Pollack, “An evolutionary approach to speech recognition,” *INTERSPEECH*, pp. 1803–1807, 2020.

L. He, D. Su, and D. Yu, “Learned transferable architectures can surpass hand-designed architectures for large scale speech recognition,” *ICASSP*, 2021, pp. 6788–6792.

H. Zheng and et al., “Efficient neural architecture search for end-to-end speech recognition via straight-through gradients,” in *IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 60–67.

Y. Liu, T. Li, P. Zhang, and Y. Yan, “Improved conformer-based end-to-end speech recognition using neural architecture search,” arXiv preprint arXiv:2104.05390, 2021.

X. Shi, P. Zhou, W. Chen, and L. Xie, “Efficient gradient-based neural architecture search for end-to-end asr,” in *Companion Publication of the 2021 International Conference on Multimodal Interaction*, 2021, pp. 91–96.

A. Mehrotra and et al., “Nas-bench-ask: Reproducible neural architecture search for speech recognition,” in *ICLR*, 2021.

S. Hu, X. Xie, S. Liu, M. Cui, M. Geng, X. Liu, and H. Meng, “Neural architecture search for lf-mmi trained time delay neural networks,” in *ICASSP*, 2021, pp. 6758–6762.

J. Deng, F. R. Gutierrez, S. Hu, M. Geng, X. Xie, Z. Ye, S. Liu, J. Yu, X. Liu, and H. Meng, “Bayesian parametric and architectural domain adaptation of lf-mmi trained tdnn for elderly and dysarthric speech recognition,” in *INTERSPEECH*, 2021, pp. 4818–4822.

P. J. Angeline, G. M. Saunders, and J. B. Pollack, “An evolutionary algorithm that constructs recurrent neural networks,” *IEEE transactions on Neural Networks*, vol. 5, no. 1, pp. 54–65, 1994.

K. O. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

D. Floreano, P. Dürr, and C. Mattiussi, “Neuroevolution: from architectures to learning,” *Evolutionary intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

K. O. Stanley, D. B. D’Ambrosio, and J. Gauci, “A hypercube-based encoding for evolving large-scale neural networks,” *Artificial life*, vol. 15, no. 2, pp. 185–212, 2009.

R. Jozefewicz, W. Zaremba, and I. Sutskever, “An empirical exploration of recurrent network architectures,” in *ICML*, 2015, pp. 2342–2350.

D. E. Goldberg and K. Deb, “A comparative analysis of selection schemes used in genetic algorithms,” in *Proceedings of the First Workshop on Foundations of Genetic Algorithms*. Bloomington, Indiana, USA, July 15-18 1990, G. J. E. Rawlins, Ed. Morgan Kaufmann, 1990, pp. 69–93.

B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, “Taking the human out of the loop: A review of bayesian optimization,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006, vol. 1.

K. Swersky, D. Duvenaud, J. Snoek, F. Hutter, and M. A. Osborne, “Raiders of the lost architecture: Kernels for bayesian optimization in conditional parameter spaces,” in *Proc. Neural Information Processing Systems Workshop on Bayesian optimization*, 2013.

P. Swietojanski, J. Li, and S. Renals, “Learning hidden unit contributions for unsupervised acoustic model adaptation,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 8, pp. 1450–1463, 2016.

M. Kitza, P. Golik, R. Schlüter, and H. Ney, “Cumulative adaptation for bilstm acoustic models,” *INTERSPEECH*, pp. 754–758, 2019.

S. Karita, N. Chen, and et al., “A comparative study on transformer vs rnn in speech applications,” in *ASRU*, 2019, pp. 449–456.

D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “Specaugment: A simple data augmentation method for automatic speech recognition,” *INTERSPEECH*, pp. 2613–2617, 2019.

Z. Tüske, G. Saon, K. Audhkhasi, and B. Kingsbury, “Single headed attention based sequence-to-sequence model for state-of-the-art results on switchboard-300,” *INTERSPEECH*, pp. 551–555, 2020.

H. Kim, M. Hasegawa-Johnson, A. Perlman, and et al., “Dysarthric speech database for universal access research,” in *INTERSPEECH*, 2008, pp. 1741–1744.

S. Furui, K. Maekawa, and H. Isahara, “A japanese national project on spontaneous speech corpus and processing technology,” in *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW)*, 2000.

C. J. Maddison, A. Mnih, and Y. W. Teh, “The concrete distribution: A continuous relaxation of discrete random variables,” *ICLR*, 2017.