Structure Growth for Small-Footprint Speech Recognition

Jiayao Wu[†‡], Zhiyuan Tang[†] and Dong Wang[†*]
[†] Center for Speech and Language Technologies, Tsinghua University, Beijing, China
[‡] School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China
Corresponding email: wangdong99@mails.tsinghua.edu.cn

***Abstract*—Modern speech recognition (ASR) is based on large-scale deep neural nets (DNNs) with various architectures. For small-footprint applications running on low-power chips, however, the size of the DNNs must be extremely constrained. In this case, training a generalizable acoustic model is not feasible, especially when the acoustic conditions are diverse.**

Most of existing approaches to small-footprint networks start from a large net and reduce its scale by pruning. In this paper, we investigate a reverse idea: starting from a small net and increasing it gradually. This structure-growth approach follows a 'general to specific' principle and grows the net gradually. We start from the AdaBoost algorithm that builds specific nets for error-prone data, and then propose a new ConBoost that builds specific nets for specific conditions. Our experiments on a small-footprint ASR task demonstrated that both AdaBoost and ConBoost outperform the baseline and other comparative methods including bagging and double-net retraining. Furthermore, ConBoost performs better than AdaBoost.

I. Introduction

Automatic speech recognition (ASR) has achieved significant performance in recent years, mostly attributed to the emerging of large data sets and the models that can consume this large data – the deep neural nets (DNNs) [1]. With a large and deep neural net, it is possible to learn the complex patterns of speech signals in multiple conditions, hence very promising performance [2]. Recently, end-to-end system that involves both acoustic and language models in a unified DNN architecture has been proposed [3].

In spite of the great progress in DNN-based architecture, most of existing techniques require a large network, which is not suitable for small-footprint applications running on resource-limited devices, e.g., chips. On these devices, only a very limited memory and computational power are available. For example, the MVSILICON 8224 that we used in this experiment offers only 100k SDRAM and the MCU is only 240 MHz. With these stringent limitations, it is impossible to use large-scale networks and so the benefit of DNNs is not easy to harvest.

Various approaches have been proposed to improve the performance of DNNs with limited-resource devices. For example, factorization methods by SVD [4], tensor training decomposition [5], or structured matrix [6]. These methods, however, cannot obtain high compression rate and limited performance loss simultaneously. Structure pruning methods were also proposed [7], [8], but sparse matrix multiplication is slow without hardware support. Another approach is dark knowledge transfer [9], [10], where a large net is trained firstly, and then a small net is trained by transferring the knowledge from the large net. This approach requires sufficient data to train the large net, hence not suitable for cases where the training data is limited as well.

All the above methods start from a large-scale network. In this paper, we present an opposite approach that starts from a small net that describes the general data and then grow the net incrementally by considering more difficult data. This novel structure-growth approach follows the **general to specific** principle of problem solving, and so it is suitable to solve special data that are usually difficult to solve by the general model. Moreover, this structure-growth approach can increase the model capacity gradually when new data is available, hence offering a way of life-long learning. Finally, this approach produces modular networks, by which a particular task can be easily addressed by net composition. Note that the structure-growth approach is orthogonal to the traditional methods such as pruning and dark-knowledge transfer, and their combination shall lead to further improvement.

In this study, we start from the AdaBoost algorithm [11] that builds a general model first using the whole training data and then builds specific models for error-prone data. Moreover, we propose a new ConBoost that divides the training data into different conditions and builds specific models for specific conditions. As a preliminary study, we consider only two conditions: clean and noisy recordings. Our experiments on a small-footprint ASR task demonstrated that both AdaBoost and ConBoost outperform the baseline and other comparative methods including bagging and double-net retraining. Furthermore, ConBoost performs better than AdaBoost.

II. Related work

The structure-growth approach is related to bagging [12], [13]. In bagging, each basic classifier is trained by a subset randomly selected from the training data, and the main goal is to reduce the variance of the model. The structure-growth approach (AdaBoost and ConBoost) follows a very different philosophy: it tries to solve the general problem, and then pays more attention to special (and usually more difficult) problems.

This work was supported by the National Natural Science Foundation of China No. 61633013.

The structure-growth idea was also explored by Xu et al. [14] and Moriya et al. [15] in the name of progressive learning. However, their focus is to learn new conditions with the help of old conditions, rather than a model that works better on both new and old conditions.

III. Methods

In this section, we first describe the two structure-growth approaches: AdaBoost and ConBoost, and then describe two methods to combine the general and specific models in ConBoost: score fusion and log-linear combination.

A. AdaBoost

AdaBoost is a very popular structure-growth approach [11]. The main idea of AdaBoost is to build a sequence of basic models where each new model is trained on data that tends to be mis-classified by the previous classifiers. These basic models are finally combined with appropriate weights that are derived from the performance of each individual model. The combination is simply voting with non-probabilistic models (e.g., decision tree, SVM), but can be also a score fusion method if the basic models are probabilistic.

Although most AdaBoost implementations use decision trees as the basic classifier [16], Schwenk et al. [17] demonstrated that it also applies to shallow MLPs. However, applying AdaBoost to DNNs seems not successful yet, especially in speech recognition. A possible reason is that DNNs are very powerful classifiers so the *weak classifier* assumption presumed by AdaBoost is not well suited. In this study, due to the limited resource in our targeted small-footprint applications, the DNN models are not so strong and so AdaBoost could be helpful.

However, applying AdaBoost to speech recognition is not performable. Note that DNN[1] is trained with samples at the frame level, so AdaBoost-compliant data weighting should be on frames too. However, the frame-level weighting incurs high computation, and more importantly, it is inconsistent with the evaluation metric of the ASR task, i.e., word error rate (WER). We therefore choose a modified version of Adaboost that treats each sentence as a training sample and weights every sentence according to its WER or averaged frame accuracy (FA). More specifically, we firstly train a general DNN using the whole training data, and then transcribe the training data using the general DNN and a language model (LM). For each sentence, the WER and FA are calculated and based on which the sentence is weighted. Finally, the weighted data is used to train the specific model.

In fact, this approach has deviated from the generic AdaBoost, and so it can be called *pseudo AdaBoost*. The deviation is from three aspects: (1) the sentence-level weighting is inconsistent with the frame-level DNN training; (2) the objective function for DNN training, i.e., cross entropy (CE) is inconsistent with the *pseudo loss* required by AdaBoost [17]; (3) the sentence-level WER and FA are dependent on the LM used for transcribing the training data, so the error-prone sentences selected according to these criteria may be not hard for the DNN, but for the LM.

Since pseudo AdaBoost deviates from true AdaBoost, the weights for the general and the specific models computed according to the AdaBoost algorithm are not optimal anymore. We therefore choose a more empirical solution that uses a development set to determine the weights.

B. ConBoost

AdaBoost follows the *general to specific* principle, where the specific data are those that are hard to recognize by the general model. In another way, we can define specific data as with a particular condition. This condition can be any property of the data, e.g., noisy background, reverberated channel, a particular accent, a particular gender. For each condition, the data exhibits a particular property and so is ideally modeled by a specific DNN. These specific DNNs can be integrated with the general DNN as in the AdaBoost algorithm, leading to a novel boosting algorithm that we call *condition Boosting*, or *ConBoost* in short.

In this paper, we treat noisy data as specific and use them to train a specific DNN. More specifically, we start from a clean data set (WSJ) and augment half of the data with a particular type of noise at a particular SNR level. All the noise-augmented data are treated as specific to train the specific DNN, and all existed data to train the general DNN. Fig. 2 (b) illustrates the ConBoost approach.

Note that ConBoost is a general algorithm and can deal with multiple conditions, i.e., train multiple specific DNNs. However, in this study we target for concept proof and so focus on the simplest single specific condition case.

C. Model integration

Once we have trained the general model and specific model, we need to combine them appropriately. A simple way is to average their output, which is the posterior of senones in our experiment, i.e.,

$$p(q|x_t) = \alpha p^G(q|x_t) + (1-\alpha)p^S(q|x_t),$$

where q denotes senones, x_t the frame segment centered at time t. The factor α is introduced to balance p^G and p^S , which denote the output of the general model and the specific model respectively. In practice, a dev set was used to optimize the value of α. This score fusion approach is shown in Fig. 1.

Another more sophisticated approach is to combine the logit output (output of the last hidden layer before the softmax), and then retrain the final layer. This is shown in the bottom picture of Fig. 1 and formulated as follows:

$$p(q|x_t) = \sigma(W \begin{bmatrix} F^G(x_t) \\ F^S(x_t) \end{bmatrix}),$$

where σ is the softmax function, F^G and F^S are the output of the penultimate layer of the general and specific DNNs, respectively. W denotes the last hidden layer and is trainable.

[1]For simplicity, we used the simple feed-forward architecture

This equals to concatenating the features output from the two DNNs and then training a log-linear model to perform the classification. In our experiments, the log-linear model is trained using the entire training data.

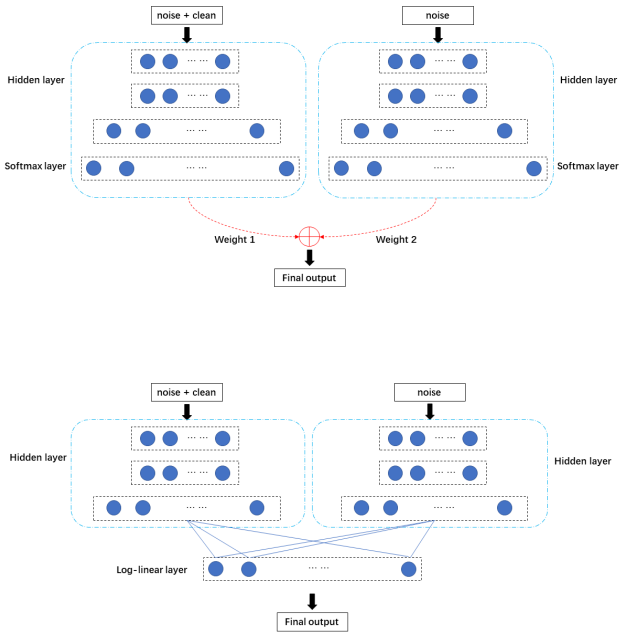

Fig. 1. Top: score fusion; Bottom: log-linear combination.

D. Comparison with other methods

It is interesting to compare AdaBoost and ConBoost with several related methods, including random bagging, conditional bagging, and double net. The five approaches are illustrated in Fig. 2.

- **Random bagging**. Both AdaBoost and ConBoost belong to the ensemble approach that constructs multiple experts to make group decisions. Random bagging is another ensemble approach, where each expert is trained with a subset of data independently sampled from the original training data. Boosting and bagging follow different principles. Random bagging aims to reduce the variance of DNN models caused by the randomness of both the initial parameters and the training data. It therefore targets for reducing the generalization error, i.e., error on test data. The boosting approach, in contrast, follows the general to specific principle and aims to improve the representation capability of the model [18].
- **Conditional bagging**. Conditional bagging is another bagging approach, which trains models for individual conditions and then bags them together. This is similar to random bagging in which all the expert models are independent, but the models in conditional bagging are trained on data of specific conditions, not via random sampling as in random bagging.
- **Double Net**. The final architecture we need to compare is a double-net architecture, where the network is the same as the general model except that the number of units per layer is doubled. This means that the total number of parameters is the same as in AdaBoost, ConBoost and bagging. Therefore, the comparison between them will shed light on the true contribution of the boosting and the bagging approaches.

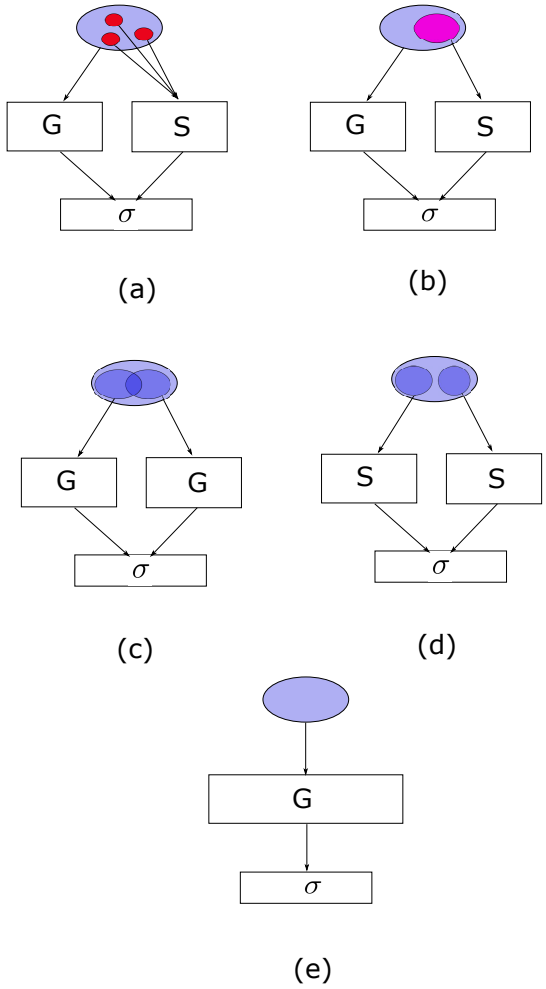

Fig. 2. Architecture of (a) AdaBoost; (b) ConBoost; (c) Random bagging; (d) Conditional bagging; (e) Double net. The blue oval denotes the entire training data; the pink oval is a subset of the data with a particular condition; the red circles denote *hard data* for the general DNN.

IV. Experiments

A. Database and configurations

The experiments were conducted with the Wall Street Journal (WSJ) Corpus [19]. We randomly selected 3500 utterances from 83 different speakers in SI-84, about 7 hours in total. And half of the data was augmented by adding noise signals sampled from the MUSAN dataset. The dev93 dataset was used for development and the eval92 dataset was used for

evaluation. They were augmented by noisy data in the same way as the training set.

The ASR system was based on the HMM-DNN hybrid architecture [20], where the DNN plays the role of acoustic model that maps acoustic features to posterior probabilities of senones, i.e., pdfs that correspond to the clustered states of the HMMs. We used the kaldi [21] toolkit to perform the experiments. The training and inference follow the wsj s5 nnet3 chain recipe [22]. The input feature involves 40-dimensional Fbanks, and the DNN contains 3 hidden layers that involve 64,64,512 nodes, respectively. This small size ensures the net is computable in a low-power chip. We choose the chain model because this model is less impacted by the prior of the pdfs, which helps the experiment as prior may change when we sample data from the training set. The language model is the *tgpr* model of the wsj recipe, which involves 3-grams of 123k words.

B. Performance

The performance in terms of WER of various training methods and model structures are reported in Table I. The systems are:

- **Baseline**: DNN baseline trained with clean + noisy data.
- **Random Bag.** : Random bagging approach, where the expert DNNs are trained by dataset independently sampled from the clean + noisy data. These two DNNs are combined by score fusion.
- **Conditional Bag.** : Condition bagging approach, where one expert DNN is trained by the clean data, and the other expert DNN is trained by the noisy data. These DNNs are combined by score fusion.
- **Double Net**: DNN trained using clean + noisy data, but the model size is doubled.
- **AdaBoost (WER)**: AdaBoost method with clean + noisy data to train the general DNN and the sentences weighted according to WER are used to train the specific DNN. These two DNNs are combined by score fusion. The tgpr LM is used to evaluate the weight of each sentence and perform decoding on the training data.
- **AdaBoost (FA)**: AdaBoost method with clean + noisy data to train the general DNN and the sentences weighted according to average frame accuracy are used to train the specific DNN. These two DNNs are combined by score fusion. The tgpr LM is used to evaluate the weight of each sentence and perform decoding on the training data.
- **ConBoost (Fusion)**: Conditional structure growth, and the general and specific DNNs are combined by score fusion.
- **ConBoost (Log-linear)**: Conditional structure growth, and the general and specific DNNs are combined by log-linear model.

In Table I, it can be seen that almost all the ensemble methods (boosting and bagging) can achieve performance gains than the baseline, except the conditional bagging. However, since all the ensemble methods leverage double-sized parameters, in order to discover the true value of these methods, we should compare them with the double-net performance (18.55%). This comparison reveals three effective methods: Random bagging (18.11%), AdaBoost (FA) (18.13%) and ConBoost (Log-Linear) (17.54%). These results demonstrated that both the variance-reduction principle (in Random Bagging) and the general-to-specific principle (AdaBoost and ConBoost) work in small-footprint scenarios. The superior performance of ConBoost plus log-linear combination is interesting. On one hand, it demonstrates that training specific models for specific conditions is reasonable, and on the other hand, it indicates that the value of this conditional training is more effective on the feature level, so the classifier needs to be retrained.

TABLE I
PERFORMANCE OF VARIOUS SYSTEMS IN WER.

	WER%	
	Dev.	Eva.
Baseline	29.69	19.10
Double Net	28.13	18.55
Random Bag.	28.89	18.11
Conditional Bag.	33.01	22.47
AdaBoost (WER)	29.35	18.66
AdaBoost (FA)	28.48	18.13
ConBoost (Fusion)	28.72	18.55
ConBoost (Log-Linear)	**26.67**	**17.54**

The bad performance of conditional bagging indicates that the idea of training conditional dependent DNNs and then bagging them together does not work. This is understandable as this approach does not learn any common patterns in the clean and noisy data, so the combination simply averages the performance of all the models on each test sentence.

V. CONCLUSIONS

We studied two structure growth algorithms, AdaBoost and ConBoost, to deal with multi-conditional data with a small net. These two algorithms follow the same general-to-specific principle that grows the net gradually. This growth is on demand, which makes controlling the net volume easier. Our experiments on a small-footprint ASR task shows that both AdaBoost and ConBoost outperformed the baseline model, as well as a double-sized model. Comparing these two algorithms, ConBoost is more superior. These results indicate that the general-to-specific principle works for DNN-based ASR, and dividing the training data into different conditions and training specific models for difficult conditions improves the overall performance. Since the model is grown incrementally, we hope the structure growth idea may inspire a new way of life-long learning.

The future work will investigate combining specific DNNs of more conditions, e.g., different genders and different channels. Another work will investigate other suitable ways for DNN combination.

REFERENCES

[1] Dong Yu and Li Deng, *AUTOMATIC SPEECH RECOGNITION.*, Springer, 2016.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436, 2015.

[3] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[4] Dan Kalman, "A singularly valuable decomposition: the svd of a matrix," *The college mathematics journal*, vol. 27, no. 1, pp. 2–23, 1996.

[5] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov, "Tensorizing neural networks," in *Advances in neural information processing systems*, 2015, pp. 442–450.

[6] Tyson R Browning, "Applying the design structure matrix to system decomposition and integration problems: a review and new directions," *IEEE Transactions on Engineering management*, vol. 48, no. 3, pp. 292–306, 2001.

[7] Tianxing He, Yuchen Fan, Yanmin Qian, Tian Tan, and Kai Yu, "Reshaping deep neural network for fast decoding by node-pruning," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 245–249.

[8] Chao Liu, Zhiyong Zhang, and Dong Wang, "Pruning deep neural networks by optimal brain damage," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[10] Zhiyuan Tang, Dong Wang, and Zhiyong Zhang, "Recurrent neural network training with dark knowledge transfer," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5900–5904.

[11] Yoav Freund, Robert E Schapire, et al., "Experiments with a new boosting algorithm," in *icml*. Citeseer, 1996, vol. 96, pp. 148–156.

[12] Leo Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[13] J Ross Quinlan et al., "Bagging, boosting, and c4. 5," in *AAAI/IAAI, Vol. 1*, 1996, pp. 725–730.

[14] Sirui Xu and Eric Fosler-Lussier, "Application of progressive neural networks for multi-stream wfst combination in one-pass decoding," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5914–5918.

[15] Takafumi Moriya, Ryo Masumura, Taichi Asami, Yusuke Shinohara, Marc Delcroix, Yoshikazu Yamaguchi, and Yushi Aono, "Progressive neural network-based knowledge transfer in acoustic models," in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2018, pp. 998–1002.

[16] Harris Drucker and Corinna Cortes, "Boosting decision trees," in *Advances in neural information processing systems*, 1996, pp. 479–485.

[17] Holger Schwenk and Yoshua Bengio, "Boosting neural networks," *Neural computation*, vol. 12, no. 8, pp. 1869–1887, 2000.

[18] Md Monirul Islam, Xin Yao, SM Shahriar Nirjon, Muhammad Asiful Islam, and Kazuyuki Murase, "Bagging and boosting negatively correlated neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 3, pp. 771–784, 2008.

[19] Douglas B Paul and Janet M Baker, "The design for the wall street journal-based csr corpus," in *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, 1992, pp. 357–362.

[20] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al., "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine*, vol. 29, 2012.

[21] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," Tech. Rep., IEEE Signal Processing Society, 2011.

[22] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi.," in *Interspeech*, 2016, pp. 2751–2755.