A General Artificial Neural Network Extension for HTK

C. Zhang & P. C. Woodland

Cambridge University Engineering Dept., Trumpington St., Cambridge, CB2 1PZ U.K.

{cz277,pcw}@eng.cam.ac.uk

Abstract

This paper describes the recently developed artificial neural network (ANN) modules in HTK hidden Markov model toolkit, which enables ANN models with very general feed-forward architectures to be used for either acoustic modelling or feature extraction. The HTK ANN extension includes many recent ANN-based speech processing techniques, such as sequence training, model stacking, speaker adaptation, and parameterised activation functions. The implementation allows efficient training by supporting GPUs and various types of data cache. The ANN modules are fully integrated into the rest of the HTK toolkit, which allows existing GMM-HMM methods to be easily used in the ANN-HMM framework. Speech recognition results on a 300 hours DARPA BOLT conversational Mandarin task show that HTK can produce tandem and hybrid systems with state-of-the-art performance on this very challenging task. Furthermore, the flexibility of the implementation is illustrated using demo systems for a Wall Street Journal (WSJ) task. The HTK ANN extension is planned for release in HTK version 3.5.

1. Introduction

HTK [1] is a research source code toolkit designed primarily for automatic speech recognition (ASR) with more than 100,000 users around the world. The most recent version is 3.4.1 [2] which was released in 2008, and contains many commonly used hidden Markov model (HMM) based ASR techniques, such as phonetic decision tree based state-tying [3], speaker adaptation [4], and lattice-based discriminative training [5]. These have allowed the construction of state-of-the-art speech recognition systems which have been used for both research as well as commercial deployment. Beyond the official HTK release, there are a number of extensions, and the most well-known is the HTS system for parametric speech synthesis [6].

Since 2010, there has been a resurgence in the use of ANNs in the speech community [7, 8], mainly due to the recent development and good performance of systems using deep learning [9, 10] which uses many layered "deep" ANNs or deep neural networks (DNNs). Deep ANNs have successfully been used, for acoustic model likelihood computation [7, 11, 12] where they replace the Gaussian mixture models (GMMs) normally used in HMM-based speech recognition systems in a hybrid model setup; for feature extraction [13] where the generated features are modelled by GMMs in a tandem setup; and finally for language models which often include recurrent network connections [14, 15]. ASR toolkits with built-in ANN support include Kaldi [16] and RWTH-ASR [17].

Chao Zhang is supported by Cambridge International Scholarship from the Cambridge Commonwealth, European & International Trust and by the EPSRC Programme Grant EP/I031022/1 (Natural Speech Technology). Supporting data for this paper is available at the http://www.repository.cam.ac.uk/handle/1810/248385 data repository.

However, previously HTK has not directly supported ANNs, and therefore the use of ANNs in HTK-based systems has relied on external ANN tools such as QuickNet [18]. In this paper, we describe a recently developed general ANN extension to HTK (HTK-ANN), which has been actively tested at Cambridge University Engineering Department (CUED), and proved to be able to generate state-of-the-art ANN based systems on data sets ranging from 3 to 1,000 hours [19, 20].

Three principles were applied to the design of the HTK ANN extension. First, in order to accommodate new models and methods easily, without sacrificing efficiency, the design should be as generic as possible. HTK-ANN supports ANNs with flexible input feature configurations and model architectures, and relies on a universal definition of ANN layer input features. Second, the new ANN modules should be compatible with as many existing functions in HTK as possible, which minimises the effort to reuse previous HTK related source code and tools in the new framework and simplifies the transfer of many techniques, for instance, sequence training [11, 21, 22], and speaker adaptation [23, 24], from the GMM to the ANN framework. Third, it should be "research friendly" so that further extensions and modifications can be created. To promote ease of reuse and future extensions, the functions are designed to be fine-grained and loosely coupled.

The rest of the paper is organised as follows: the implementation details of HTK-ANN are described in detail in Section 2. Brief examples of a tandem and a hybrid system are given in Section 3. In Secion 4, experimental results of state-of-the-art hybrid and tandem systems, along with the example systems are presented. We conclude in Section 5.

2. Implementation Details

HTK includes many widely used speech processing technologies, covering the entire ASR pipeline [2]. Many of these features including front-end feature extraction, phonetic decision trees, feature transforms, and sequence discriminative training, large vocabulary decoders and lattice generation and processing are all used in the design of ANN based ASR systems. Therefore, implementing native support of ANNs in HTK can simplify the use of all of these established approaches and allow ANN-based systems to benefit from the HTK infrastructure. This is achieved by developing HTK-ANN as a number of new HTK modules (libraries) and tools, and extending other libraries and tools to be compatible with them. An overview of the newly added and extended HTK modules and tools is shown in Table 1. Here we describe how HTK-ANN is implemented.

2.1. Generic ANN Support

In HTK-ANN, an ANN model is presented as a layered structure. Each layer l has a weight matrix, a bias vector, parameter vectors for parameterised activation functions [25], and a

Name	Function Descriptions/Updates
HANNet†	ANN structures & core algorithms
HCUDA†	CUDA based math kernel functions
HFBLat	Hybrid MMI/MPE/MWE computation
HMath	ANN related math kernel functions
HModel	ANN model reading/writing interface
HNCache†	Data cache for data random access
HParm	New feature types for ANNs
HDecode	Tandem/hybrid system LVCSR decoder
HDecode.mod	Tandem/hybrid system model marking
HHEd	ANN model creation & editing
HVite	Tandem/hybrid decoder & alignment
HNForward‡	ANN evaluation & output generation
HNTrainSGD‡	SGD based ANN training

Table 1: *List of the HTK-ANN related modules and tools.* † *new HTK modules;* ‡ *new HTK tools.*

feature mixture that defines the components of its input feature vector $\mathbf{x}^l$. A feature mixture can have any number of *feature elements*, with each of which defines a fragment of $\mathbf{x}^l$, based on its source and the *context shift set*. The source of a feature element can be an ANN layer, input acoustic features, or augmented input features [26]. Meanwhile, a context shift set $\mathbf{c}$ is a set of integers indicating their time differences to the current time. For example, $\mathbf{c} = \{-1, 0, 2\}$ for the input acoustic feature vector $\mathbf{o}_t$ at time t means that the concatenated vector is formed by stacking $\mathbf{o}_{t-1}$, $\mathbf{o}_t$, and $\mathbf{o}_{t+2}$. It is known that a feedforward neural network (FNN) can have any architecture equivalent to a directed acyclic graph (DAG), while the topology of a recurrent neural network (RNN) contains directed cycles and is presented as a directed cyclic graph (DCG) [27]. HTK-ANN supports flexible model structures equivalent to any DCG. Other toolkits that support similar model architectures are Theano [28], RWTH-ASR [17], and CNTK [29].

Currently HTK-ANN only has support for training using standard error backpropagation (EBP) [27] rather than back propagation through time (BPTT) [27]. Therefore, it is able to train FNNs with any layered architecture. Furthermore, in EBP, gradients for the parameters of layer l are accumulated $n_l = \sum_e |\mathbf{c}_e^l|$ times, where e refers to the feature elements whose source are layer l, and $\mathbf{c}_e^l$ is the context shift set associated with e. Therefore, gradients of parameters of layer l need to be normalised by dividing by n_l.

Besides providing a very flexible approach to defining the model architecture, HTK-ANN also supports different kinds of activation functions, such as linear, softmax, sigmoid, ReLU [30], softplus [31], and parameterised functions [23, 25].

2.2. ANN Training

HTK-ANN supports both frame-level criteria such as cross entropy (CE) and minimum mean squared error (MMSE), and sequence discriminative criteria including maximum mutual information (MMI), minimum phone error (MPE), and minimum word error (MWE). When training FNNs with sequence discriminative criterion $\mathcal{F}$ using EBP, the first step is to compute

$$\frac{\partial \mathcal{F}}{\partial \log y_k^{l_{\text{out}}}(\mathbf{o}_t)} = \kappa \gamma_k(\mathbf{o}_t), \tag{1}$$

where κ is the acoustic scaling factor, $\gamma_k(\mathbf{o}_t)$ is the posterior probability of being in the HMM state related to ANN output target k at time t. $\gamma_k(\mathbf{o}_t)$ is acquired by lattice based forward/backward rescoring. Therefore, for different training modes, the supervision used for ANN training can come from label files with frame-to-label alignment (for CE and MMSE training), feature files (for autoencoders), and from lattice files (for MMI, MPE, and MWE).

After computing the gradients with the given criterion using EBP, HTK-ANN can apply commonly used modifications to the gradients, including the use of momentum, gradient clipping [14], weight decay, and max norm [32]. Then the refined graidents are scaled by learning rates and used to update ANN parameters based on stochastic gradient descent (SGD). HTK-ANN contains several different types of learning rate schedulers, such as List (pre-determined piecewise constant learning rate) [18], NewBob [18], AdaGrad (per parameter basis learning rate decay akin to 2nd-order method) [33], *etc*. In particular, we modified NewBob to be less aggressive in learning rate decay, and to enforce a minimum required number of training epochs, $N_{\min}$. When the optimisation criterion value increase is smaller than the set threshold, if the current epoch number is smaller than $N_{\min}$, the learning rate will only be reduced once; otherwise it will follow the standard NewBob schedule and reduce the learning rate in all successive epochs.

2.3. Data Cache

In SGD based speaker independent (SI) ANN training, it is important to randomise the input data to avoid the information from a particular speaker, accent, noise, or channel causing too much bias for consecutive updates. In order to minimise the I/O cost from random data access, loading data into the memory through a cache is usually crucial. To make the cache sufficiently large, HTK now includes 64bit support. Three different data rearrangements are available in the HTK-ANN data cache to support different types of models and training modes:

1. Frame level randomisation/shuffling. All frames from all utterances in the cache are shuffled, which is a commonly used approach for frame level ANN training. However, in HTK-ANN, in order to process FNNs with any architecture, context shifts for every frame in the batch should be available as well. This also paves the way for training recurrent neural network (RNN) acoustic models using an unfolding strategy [12].
2. Utterance based shuffling. This is useful in sequence training [21] and speaker adaptive training.
3. Batch of utterance level shuffling. A batch of utterances is randomly selected and is processed in parallel. Once an utterance is finished, a new utterance will be loaded to the empty position in the batch. This kind of cache has been used for RNN training [15], as well as sequence training with asynchronous SGD [34].

Efficiency is a key factor in cache design. An extra thread can be enabled to load data into the cache, while the main thread is working on forward/backward propagation.

2.4. Other Key Features

2.4.1. Math Kernels

A set of math kernel functions required by ANN processing was added. Every new kernel has standard CPU, Intel MKL, and CUDA based implementations. Both single and double precision float numbers are supported.

2.4.2. Input Transforms and Speaker Adaptation

In HTK, ANNs can directly utilise many types of SI and SD input transforms estimated by GMM-HMMs, including HLDA [35], STC [36], and CMLLR [37]. Meanwhile, since all ANN parameters are stored as either matrices or vectors, we have added a new light-weight speaker adaptation mechanism, which works for sequence training by swapping some small-grained ANN parameter units, represented by matrices and vectors, according to speaker ids.

2.4.3. Model Editing

Like previous versions of HTK, the HTK model editor tool, HHEd, is used to edit the structure of ANN models. Current edit operations allow the insertion, removal and initialisation of a layer; changing the activation functions or dimensions of a layer; modifying a feature mixture by adding or removing a feature element, or changing its associated context shift set. These operations can be used to generate ANNs with any DCG equivalent architectures, and any intermediate models for pre-training. HHEd can also associate an isolated ANN with a GMM-HMM set by assigning each ANN output target an HMM state.

2.4.4. Decoders

Both the standard HTK Viterbi decoder HVite and the LVCSR decoder HDecode were updated to support tandem and hybrid system decoding. In ANN-HMM training, HVite is also used to produce the frame-to-label alignments. Both HVite and HDecode.mod, a variant of HDecode, implement model marking on word lattices with ANN-HMMs, which is required by MPE.

Another more recent HDecode variant, HDecode.joint, performs joint decoding of multiple HTK acoustic models with a shared decision tree by performing a log-linear combination of the log-likelihoods. Our recent studies have found that even with pre-determined fixed combination weights, the joint decoder is able to improve both speech to text [19] and key word spotting [20] performance.

3. HTK based Hybrid/Tandem Systems

In this section, we show how a state-of-the-art hybrid SI system, a tandem SAT system, and a demo hybrid system with a complex architecture can be built using HTK.

3.1. Building Hybrid SI Systems

In HTK-ANN, the first step to build DNN-HMMs is to use HHEd to associate a DNN proto-type with a pre-generated HMM set such as a set of tied state GMM-HMMs produced by the decision tree tying approach [3]. Then the DNN parameters are trained according to the frame-to-state labels produced by Viterbi alignment using HVite with a pre-existing system (e.g. a tandem system).

Discriminative pre-training [7, 8] for DNNs with sigmoid activation functions is implemented using HHEd to insert a new hidden layer once the current model is trained for one epoch using HNTrainSGD. Once the desired DNN structure is achieved, it is further trained with EBP until convergence, which is the fine-tuning step. The frame-level training for DNN-HMMs is now complete.

Then, the resulting DNN-HMMs are used to produce the denominator word lattices using HDecode for sequence discriminative training. If the MPE criterion is used [5], phone based model marking is performed with HDecode.mod, which generates the numerator and denominator phone lattices. Finally, HNTrainSGD is used again for sequence discriminative training by computing $\gamma_k(\mathbf{o}_t)$ based on Eq. (1).

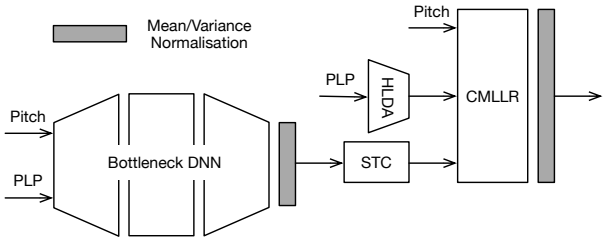

Figure 1: *A composite ANN as a Tandem SAT system front-end.*

3.2. ANN Front-ends for GMM-HMMs

In HTK, ANNs can be used as standalone models in a hybrid setup as ANN-HMMs, or for feature extraction with GMM-HMMs. To integrate ANNs into GMM-HMMs, an ANN feature mixture can be used to define the composition of the GMM-HMM input vector. Therefore, the entire HTK-ANN serves as a front-end to the GMM-HMMs, and it is possible to use any ANN for all types of GMM-HMMs supported by HTK in a seamless manner.

For example, Fig. 1 illustrates how a-state-of-the-art tandem system built with speaker adaptive training (SAT) is implemented as a single system with HTK-ANN. A composite FNN without a strictly layered structure is acquired by connecting STC, HLDA, and CMLLR transforms as ANN layers to the bottleneck DNNs. The mean and unit variance normalisation usually applied to the bottleneck features is implemented with a parametric linear activation function, $f_i^l(\cdot)$ for node i of layer l,

$$f_i^l(a_i^l(\mathbf{o}_t)) = \frac{1}{\sigma_i}\left(a_i^l(\mathbf{o}_t) - \mu_i\right),$$

where $a_i^l(\mathbf{o}_t)$ is the input activation of $f_i^l(\cdot)$ at time t, μ_i, and σ_i are the ith dimension of the mean and standard deviation vectors. The CMLLR transforms, and the pre-computed mean and standard deviation vectors are replaceable according to the speaker ids. This is achieved using the matrix/vector swap method mentioned in Section 2.4.2. This allows mean/variance normalisation at any level to be applied to the bottleneck features generated on the fly. As a result, we can construct a single system identical to a tandem SAT system usually generated by dumping tandem features or loading ANN generated features through either a pipe or a socket.

This native support for the tandem system reduces the difficulties in building even more complicated systems, and enables the joint optimisation of GMM-HMMs, transforms, and the ANN front-ends.

3.3. Demo Hybrid System with Flexible Structures

In this subsection, an example of constructing an ANN-HMM system without a strictly layered structure using HTK-ANN is presented. The ANN architecture is similar to the deep convex network (DCN) [38]. We built the model by stacking 4 layer bottleneck FNNs, and each FNN takes the original acoustic features, as well as the bottleneck features from all previous FNNs as the input. This is achieved by using HHEd to extend the input layer feature mixture to have an additional feature element associated with the bottleneck layer of

each previous FNN, with the context shift set as $\{0\}$. Here, the input acoustic vector has 468 dimensions, and the hidden layer, bottleneck layer, and output layer sizes are 1000, 200, and 3000, respectively. The structure of the nth FNN is $(468 + (n-1) \times 200) \times 1000 \times 200 \times 3000$, $n = 1, 2, \ldots$. If the final model is stacked by N 4-layer FNNs, the last FNN has no bottleneck layer, i.e., $(468+(N-1)\times 200)\times 1000^2 \times 3000$.

Every time a new FNN is stacked, the current ANN is trained for one epoch using HNTrainSGD based on the CE criterion. This is just like the discriminative pre-training mentioned in Section 3.1. An alternative intialisation is in [38]. Once the desired ANN structure is realised, HNTrainSGD is used for fine-tuning. HNTrainSGD can automatically parse the model structure and do forward/backward propagation.

From the above procedure, we can see that with HTK-ANN, it is possible to build ANNs with complex structures using only HHEd and HNTrainSGD commands. We believe this provides a good platform to study more types of ANNs with different topologies.

4. Experiments

4.1. Experiment Setup

In this section, experimental results of the example systems trained using 300 hours of Mandarin conversational telephone speech (CTS) data from DARPA BOLT project are given. The data were spontaneous speech with multi-accent, mixed languages (Mandarin and English), and overlapping speech, which makes the task very challenging. The Hybrid SI system was built following the paradigm mentioned in Section 3.1, while the Tandem SAT system was the same as illustrated in Fig. 1. More details about the data and systems can be found in [19]. We used the development set released in 2014, *dev'14*, for testing [19].

The demo system with the structures described in Section 3.3 were trained on 15 hours Wall Street Journal (WSJ0) training set (SI-84), and tested on the 65k vocabulary 1994 H1-dev (65k dt) and Nov'94 H1-eval (65k et) testing sets [39]. The 52d acoustic feature vector consists of 13d PLP along with its Δ, $\Delta\Delta$, and $\Delta\Delta\Delta$ features. The general setup is the same as in [39].

All of the BOLT and WSJ0 DNNs had sigmoid activation functions, and their inputs were formed by concatenating the current frame with 4 frames to its left and its right context. Parameter updates were averaged over a mini-batch with 800 frames and smoothed by adding a "momentum" term of 0.5 times the previous update. In each case, 10% of the training set was randomly selected for held-out validation, and the modified NewBob learning rate scheduler presented in Section 2.2 was used for all CE training. The initial learning rate and minimum epoch number were set to 2.0×10^{-3} and 12. All sequence training was carried out with per utterance update based on MPE without the use of momentum. For the BOLT system, sequence training was performed for 2 epochs, with fixed learning rates of 4.0×10^{-5} and 2.0×10^{-5}.

4.2. BOLT Systems

Both hybrid and tandem CE DNNs were trained with alignments generated by another tandem SAT system. They used a standard DNN structure of $504 \times 2000^4 \times 1000 \times 12000$ and $504 \times 2000^4 \times 1000 \times 26 \times 12000$, respectively. The tandem system also has 12,000 GMM-HMM states. The WERs are listed in Table 2. Compared to the Tandem SAT system, the Hybrid SI systems trained using the CE and MPE criteria were respectively 3.8% worse and 4.8% better in terms of WER. Two epochs of MPE sequence training gave an 8.9% relative WER reduction, which is fairly consistent with previously reported results [21, 22]. Moreover, the overall results benefit from the complementary information from different systems, joint decoding of the MPE Hybrid SI and Tandem SAT systems [40, 19] with system dependent combination weights (1.0, 0.2) resulted in a further 1.9% relative decrease in WER.

System	Criterion	%WER
Hybrid SI	CE	34.5
Hybrid SI	MPE	31.6
Tandem SAT	MPE	33.2
Hybrid SI $\otimes$ Tandem SAT	MPE	31.0

Table 2: *Performance of BOLT tandem and hybrid systems with standard configurations evaluated on dev'14.*

4.3. WSJ Hybrid Systems

The held-out set frame classification accuracies along with the WERs are shown in Table 3.

FNN	%Accuracy		%WER	
Num	Train	Held-out	65k dt	65k et
1	69.9	58.1	9.3	10.9
2	72.8	59.1	9.0	10.4
3	73.9	59.1	8.8	10.7

Table 3: *Performance of the WSJ0 Demo Systems.*

In Table 3, the stacking procedure was stopped because over-fitting was observed from the results. The system stacked with 3 modules results in a complex structure. However, this demo system shows that HTK-ANN can perform forward/backward propagation through complex architectures.

5. Conclusions

We present our recently developed ANN extension, which integrates native support of ANNs into HTK and enables HTK based GMM technologies to directly apply to the ANN-based systems. HTK-ANN can build FNNs with very flexible topologies, different activation functions, various input features, using frame or sequence level discriminative training. An ANN based speaker dependent parameter swap method and parameterised linear activation functions are also proposed, and help to accommodate the tandem SAT system architecture. Experiments on 300 hour CTS task showed HTK can generate standard state-of-the-art tandem and hybrid systems. The other WSJ0 hybrid training task showed that HTK can build systems with very flexible structures with a simple standard feedforward architecture. A version of HTK-ANN will be made available with the release of HTK 3.5 in 2015. It is intended to continue to develop HTK in future and to integrate backpropagation through time (BPTT), convolutional layers, and 2nd-order optimisation methods.

6. References

[1] http://htk.eng.cam.ac.uk

[2] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain., D. Kershaw, X.-Y. Liu, G. Moore, J. J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland, *The HTK book (for HTK version 3.4)*. Cambridge University Engineering Department, 2006.

[3] S. J. Young, J. J. Odell, and P. C. Woodland, "Tree-based state tying for high accuracy acoustic modelling," *Proc. Human Language Technology Workshop*, Plainsboro, NJ, USA: Morgan Kaufman Publishers Inc, 1994.

[4] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Computer Speech & Language*, Vol. 9, No. 2, pp. 171–185, 1995.

[5] D. Povey and P. C. Woodland, "Minimum phone error and I-smoothing for improved discriminative training," *Proc. ICASSP'02*, Orlando, FL, USA, 2002.

[6] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. W. Black, and K. Tokuda, "The HMM-based speech synthesis system (HTS) version 2.0," *Proc. 6th ISCA Workshop on Speech Synthesis*, Bonn, Germany, 2007.

[7] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," *Proc. ASRU'11*, Waikoloa, HI, USA, 2011.

[8] G. E. Hinton, L. Deng, D. Yu *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, pp. 2–17, Nov. 2012.

[9] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul 2006.

[10] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends® in Machine Learning*, Vol. 2, No. 1, pp. 1–127, 2009.

[11] B. Kingsbury, T. N. Sainath, and H. Soltau, "Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization," *Proc. Interspeech'12*, Portland, OR, USA, 2012.

[12] G. Saon, H. Soltau, A. Emami, and M. Picheny, "Unfolded recurrent neural networks for speech recognition," *Proc. Interspeech'14*, Singapore, 2014.

[13] Z.-J. Yan, Q. Huo, and J. Xu, "A scalable approach to using DNN-derived features in GMM-HMM based acoustic modeling for LVCSR," *Proc. Interspeech'13*, Lyon, France, 2013.

[14] T. Mikolov, "Statistical language models based on neural networks," Ph.D. dissertation, Ph. D. thesis, Brno University of Technology, 2012.

[15] X. Chen, Y.-Q. Wang, X.-Y. Liu, M. J. F. Gales, and P. C. Woodland, "Efficient GPU-based training of recurrent neural network language models using spliced sentence bunch," *Proc. Interspeech'14*, Singapore, 2014.

[16] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y.-M. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Veselý, "The Kaldi speech recognition toolkit," *Proc. ASRU'11*, Waikoloa, HI, USA, 2011.

[17] S. Wiesler, A. Richard, P. Golik, R. Schlüter, and H. Ney, "RASR/NN: The RWTH neural network toolkit for speech recognition," *Proc. ICASSP'14*, Florence, Italy, 2014.

[18] D. Johnson, "Quicknet," http://www1.icsi.berkeley.edu/speech/qn.html.

[19] X.-Y. Liu, F. Flego, L.-L. Wang, C. Zhang, M. J. F. Gales, and P. C. Woodland, "The Cambridge University 2014 BOLT conversational telephone Mandarin Chinese LVCSR system for speech translation," *Proc. Interspeech'15*, Dresden, Germany, 2015.

[20] H.-P. Wang, A. Ragni, M. J. F. Gales, K. M. Knill, P. C. Woodland, and C. Zhang, "Joint decoding of tandem and hybrid systems for improved keyword spotting on low resource languages," *Proc. Interspeech'15*, Dresden, Germany, 2015.

[21] H. Su, G. Li, D. Yu, and F. Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription," *Proc. ICASSP'13*, Vancouver, Canada, 2013.

[22] K. Veselý, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," *Proc. Interspeech'13*, Lyon, France, 2013.

[23] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," *Proc. IWSLT'14*, Lake Tahoe, USA, Dec. 2014.

[24] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," *Proc. ICASSP'13*, Vancouver, Canada, 2013.

[25] C. Zhang and P. C. Woodland, "Parameterised sigmoid and ReLU hidden activation functions for DNN acoustic modelling," *Proc. Interspeech'15*, Dresden, Germany, 2015.

[26] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," *Proc. ASRU'13*, Olomouc, Czech Republic, 2013.

[27] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[28] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, Lake Tahoe, USA, 2012.

[29] D. Yu, A. Eversole, M. Seltzer, K. Yao, Z. Huang, B. Guenter, O. Kuchaiev, Y. Zhang, F. Seide, H. Wang, J. Droppo, G. Zweig, C. Rossbach, J. Currey, J. Gao, A. May, B. Peng, A. Stolcke, and M. Slaney, "An introduction to computational networks and the computational network toolkit," Microsoft, Tech. Rep. MSR-TR-2014-112, 2014.

[30] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, "On rectified linear units for speech processing," *Proc. ICASSP'13*, Vancouver, Canada, 2013.

[31] A. Senior and X. Lei, "Fine context, low-rank, softplus deep neural networks for mobile speech recognition," *Proc. ICASSP'14*, Florence, Italy, 2014.

[32] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[33] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[34] E. McDermott, G. Heigold, P. J. Moreno, A. Senior, and M. Bacchiani, "Asynchronous stochastic optimization for sequence training of deep neural networks: Towards big data," *Proc. Interspeech'14*, Singapore, 2014.

[35] X.-Y. Liu, M. J. F. Gales, and P. C. Woodland, "Automatic complexity control for HLDA systems," *Proc. ICASSP'03*, Hong Kong, 2003.

[36] M. J. F. Gales, "Semi-tied covariance matrices for hidden Markov models," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 272–281, 1999.

[37] M. J. F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[38] L. Deng, D. Yu, and J. Platt, "Scalable stacking and learning for building deep architectures," *Proc. ICASSP'12*, Kyoto, Japan, 2012.

[39] C. Zhang and P. C. Woodland, "Standalone training of context-dependent deep neural network acoustic models," *Proc. ICASSP'14*, Florence, Italy, 2014.

[40] P. Swietojanski, A. Ghoshal, and S. Renals, "Revisiting hybrid and GMM-HMM system combination techniques," *Proc. ICASSP'13*, Vancouver, Canada, 2013.