A KALDI-DNN-based ASR system for Italian

Experiments on Children Speech

Piero Cosi

Istituto di Scienze e Tecnologie della Cognizione
Consiglio Nazionale delle Ricerche

Unità Organizzativa di Supporto di Padova - Italy
piero.cosi@pd.istc.cnr.it

Abstract*—In this paper, the KALDI ASR engine adapted to Italian is described and the results obtained so far on some children speech ASR experiments are reported. We give a brief overview of KALDI, we describe in detail its DNN implementation, we introduce the acoustic model (AM) training procedure and we end describing some experiments on Italian children speech together with the final test procedures.

***Keywords*— DNN, Children Speech, ASR**

I. Introduction

During the last few years, many different Automatic Speech Recognition (ASR) frameworks have been developed for research purposes and, nowadays, various open-source ASR toolkits are available to research laboratories. Systems such as HTK [1], SONIC [2], [3], SPHINX [4], [5], RWTH [6], JULIUS [7], KALDI [8], the more recent ASR framework SIMON [9], and the relatively new system called BAVIECA [10] are a simple and probably not exhaustive list.

Deep Neural Networks (DNNs) are the latest hot topic in speech recognition. Since around 2010 many papers have been published in this area, and some of the largest companies (e.g. Google, Microsoft) are starting to use DNNs in their production systems.

Indeed new systems such as KALDI [8] demonstrated the effectiveness of easily incorporate "Deep Neural Network" (DNN) techniques [11] in order to improve the recognition performance in almost all recognition tasks.

In this paper, the KALDI ASR engine adapted to Italian is described and the results obtained so far on some children speech ASR experiments are reported. We give a brief overview of KALDI, and in particular of its DNN implementation, we introduce the acoustic model (AM) training procedure and we end describing some experiments on Italian children speech together with the final test procedures.

II. Kaldi

As written in his official web site (http://KALDI.sourceforge.net), the KALDI ASR environment should be mainly taken into consideration for the following simple reasons:

- it's "easy to use" (once you learn the basics, and assuming you understand the underlying science)
- it's "easy to extend and modify"
- it's "redistributable": unrestrictive license, community project
- if your stuff works or is interesting, the KALDI team is open to including it and your example scripts in our central repository: more citation, as others build on it.

In particular, even if KALDI is similar in aims and scope to HTK, and the goal is still to have modern and flexible code, written in C++, that is easy to modify and extend, the important features that represent the main reasons to use KALDI versus other toolkits include:

- code-level integration with Finite State Transducers (FSTs)
  - compiling against the OpenFst toolkit (using it as a library);
- extensive linear algebra support
  - including a matrix library that wraps standard
  - BLAS and LAPACK routines;
- extensible design
  - providing, as far as possible, algorithms in the most generic form possible; for instance, decoders are templated on an object that provides a score indexed by a (frame, fst- input-symbol) tuple, this meaning that the decoder could work from any suitable source of scores, such as a neural net;
- open license
  - the code is licensed under Apache 2.0, which is one of the least restrictive licenses available;
- complete recipes
  - making available complete recipes for building speech recognition systems, that work from widely available databases such as those provided by the ELRA or Linguistic Data Consortium (LDC).

It should be noted that the goal of releasing complete recipes is an important aspect of KALDI. Since the code is publicly available under a license that permits modifications and re-release, this encourages people to release their code, along with their script directories, in a similar format to KALDI 's own example script.

III. Deep Neural Networks in Kaldi[1]

An active area of research like Deep Neural Networks (DNNs) is difficult for a toolkit like KALDI to be well supported, because the state of the art changes constantly, which means code changes are required to keep up, and architectural decisions may need to be rethought.

KALDI currently contains two parallel implementations for DNN training. Both of these recipes are deep neural networks where the last (output) layer is a softmax layer whose output dimension equals the number of context-dependent states in the system (typically several thousand). The neural net is trained to predict the posterior probability of each context-dependent state. During decoding the output probabilities are divided by the prior probability of each state to form a "pseudo-likelihood" that is used in place of the state emission probabilities in the HMM

The first implementation is as described in [12, 13]. This implementation supports Restricted Boltzmann Machines (RBM) pre-training [14, 15, 16], stochastic gradient descent training using NVidia Graphics Processing Units (GPUs), and discriminative training such as boosted MMI [17] and state-level minimum Bayes risk (sMBR) [18, 19].

The second implementation of DNNs in KALDI [20, 21, 22] was originally written to support parallel training on multiple CPUs, although it has now been extended to support parallel GPU-based training and it does not support discriminative training.

One is located in code sub-directories nnet/[2] and nnetbin/, and is primarily maintained by Karel Vesely. The other is located in code subdirectories nnet2/ and nnet2bin/, and is primarily maintained by Daniel Povey (this code was originally based on an earlier version of Karel's code, but it has been extensively rewritten). Neither codebase is more "official" than the other. Both are still being developed in parallel.

In the example directories (referring to the HKUST Mandarin Telephone Speech, Resource Management, Switchboard, Timit, and Wall Street Journal corpora) such as egs/hkust/s5b, egs/rm/s5, egs/swbd/s5, egs/timit/s5/, and egs/wsj/s5/, neural net example scripts can be found. Karel's example scripts can be found in local/run_dnn.sh or local/run_nnet.sh, and Dan's example scripts can be found in local/run_nnet2.sh. Before running those scripts, the first stages of "run.sh" in those directories must be run in order to build the systems used for alignment.

Regarding which of the two setups you should use:

- Karel's setup (nnet1) generally gives somewhat better results but it only supports training on a single GPU card, or on a single CPU which is very slow.
- Dan's setup generally gives slightly worse results but is more flexible in how you can train: it supports using multiple GPUs, or multiple CPUs each with multiple threads. Multiple GPUs is the recommended setup. They don't have to all be on the same machine.

The reasons for the performance difference is still unclear, as there are many differences in the recipes used. For example, Karel's setup uses pre-training but Dan's setup does not; Karel's setup uses early stopping using a validation set but Dan's setup uses a fixed number of epochs and averages the parameters over the last few epochs of training. Most other details of the training (nonlinearity types, learning rate schedules, etc.) also differ.

A. *Karel's DNN training implementation*

The implementation of DNNs from Karel Vesely [12, 13] uses the following techniques:

- layer-wise pre-training based on RBMs (Restricted Boltzmann Machines)
- per-frame cross-entropy training
- sequence-discriminative training, using lattice framework, optimizing sMBR criterion (State Minimum Bayes Risk)

The systems are built on top of LDA-MLLT-fMLLR[3] features (see [23] for all acronyms references) obtained from auxiliary GMM (Gaussian Mixture Model) models. Whole DNN training is running in a single GPU using CUDA (Compute Unified Device Architecture, the parallel computing architecture created by NVidia^{TM}), however cudamatrix library is designed to also run on machines without a GPU, but this tends to be more than 10x slower.

The script for standard databases such wsj is located at: "egs/wsj/s5/local/run_dnn.sh" and it is split into several stages. At first the 40-dimensional features (MFCC, LDA, MLLT, fMLLR with CMN) are stored to disk in order to simplify the training scripts.

1) *Pre-training Phases*

The implementation of layer-wise RBM (Restricted Boltzmann Machine) pre-training is following the document http://www.cs.toronto.edu/~hinton/absps/guideTR.pdf [24]. The training algorithm is Contrastive Divergence with 1-step of Markov Chain Monte Carlo sampling (CD-1). The hyper-parameters of the recipe were tuned on the 100 hours Switchboard subset. If smaller databases are used, mainly the number of epochs N needs to be set to 100 hours/set_size. The training is unsupervised, so it is sufficient to provide single data-directory with input features.

When training the RBM with Gaussian-Bernoulli units, there is a high risk of weight-explosion, especially with larger learning rates and thousands of hidden neurons. To avoid

[1] Most of the text in this section is taken from "Deep Neural Networks in Kaldi" (http://kaldi.sourceforge.net/dnn.html) with permission from the Author (Daniel Povey).

[2] See the code at: https://svn.code.sf.net/p/kaldi/code/trunk/egs

[3] LDA: Linear Discriminant Analysis, MLTT: Maximum Likelihood Linear Transform, fMLLR: feature space Maximum Likelihood Linear Regression.

weight-explosion a mechanism, which compares the variance of training data with the variance of the reconstruction data in a minibatch has been implemented. If the variance of reconstruction is >2x larger, the weights are shrinked and the learning rate is temporarily reduced.

2) Frame-level cross-entropy training.

In this phase a DNN which classifies frames into triphone-states is trained. This is done by mini-batch Stochastic Gradient Descent. The default is to use Sigmoid hidden units, Softmax output units and fully connected layers AffineTransform. The learning rate by default is 0.008, size of mini-batch 256; no momentum or regularization is used. The optimal learning-rate differs with type of hidden units, the value for sigmoid is 0.008, for tanh 0.00001.

The input_transform and the pre-trained DBN (i.e. Deep Belief Network, stack of RBMs) is passed into to the script using the options '–input-transform' and '–dbn', only the output layer is initialized randomly. An early stopping criterium is used to prevent over-fitting, for this the objective function on the cross-validation set (i.e. held-out set) is measured, therefore two pairs of feature-alignment dirs are needed to perform the supervised training.

A good summary paper on DNN training is “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups” by Geoffrey Hinton et al. [13].

3) Sequence-discriminative training.

In this phase, the neural networks is trained to classify correctly the whole sentences, which is closer to the general ASR objective than frame-level training. The objective of sequence-discriminative training is to maximize the expected accuracy of state labels derived from reference transcriptions, and lattice framework to represent competing hypothesis is used. The training is done by Stochastic Gradient Descent (SGD) with per-utterance updates, low learning rate 1e-5 which is kept constant is used and 3-5 epochs are done. Faster convergence when re-generating lattices after 1st epoch are observed.

MMI, BMMI, MPE and sMBR[4] training are all supported (see [23] for all acronyms references). In sMBR optimization, silence frames are excluded from accumulating approximate accuracies[5] [25].

B. Dan's DNN training implementation

For the full documentation that covers Dan Povey's version of the deep neural network code in KALDI one could refer to the following web link [20] and to the following papers [21, 22].

In its last implementation stage, as indicated in [22] where the Dan’s DNN is used on a speech recognition setup called Fisher English, which is English language conversational telephone speech, sampled at 8 kHz, for a total amount of training data of 1600 hours, the “… DNN system uses speaker adapted features from a GMM system, so it requires a first pass of GMM decoding and adaptation. …”.

“… The GMM system is based on MFCC features, spliced across ±3 frames and processed with LDA+MLLT to 40- dimensional features, then adapted with feature-space MLLR (fMLLR) in both training and test time. See [8] for an explanation of these terms and the normal system build steps. All these systems used the same phonetic context decision tree with 7880 context-dependent states; the GMM system had 300000 Gaussians in total. ...”

“… The 40-dimensional features from GMM are spliced across ±4 frames of context and used as input to the DNN. DNN is a p-norm DNN [21] with 5 hidden layers and p-norm (input, output) dimensions of (5000, 500) respectively, i.e. the nonlinearity reduces the dimension tenfold. In this framework 15000 “sub-classes” are used (see Section C.3 of [22] for explanation), and the number of parameters is 19.3 million. The system is trained for 12 epochs with learning rate varying from 0.08 to 0.008, trained with 8 parallel jobs with online natural gradient SGD (NG-SGD) and, for this DNN system, K=400000 samples per outer iteration for each machine are used for training. …”.

As for the TIMIT recipe (s5) the Dan’s DNN is much more simpler and adopts a classic Hybrid Training and Decoding framework using a simple deep network with tanh nonlinearities. Moreover, also system combination using minimum Bayes risk decoding is used, and in this case a lattice combination is used to create a union of lattices normalized by removing the total forward cost from them and using the resulting lattice as input the last decoding step.

IV. KALDI ON ITALIAN

In this section, the adaptation of KALDI to Italian, starting from the TIMIT recipe, is described and the results obtained so far on some children speech ASR experiments are reported (see Table 1).

In the experiments here described the Italian FBK ChildIt Corpus [26] was taken into consideration. This is a corpus that counts almost 10 hours of speech from 171 children; each child has read about 60 children literature sentences; the audio was sampled at 16 kHz, 16 bit linear, using a Shure SM10A head- worn mic.

Various experiments have been carried out in many configurations and in all cases, training and test materials have been kept separate. In all the experiments here described, the standard MFCC features were considered, setting reasonable defaults, but other options could be exploited in the future. The results, shown in Table 1, are the best obtained with KALDI on the CHILDIT corpus and they are the best obtained so far in comparison with those obtained by similar experiments reported in the literature [26, 27, 28, 29, 30, 31, 32].

[4] MI: Maximum Mutual Information; BMMI: Boosted MMI; MPE: Minimum Phone Error; sMBR: State-level Minimum Bayes Risk.

[5] More detailed description is at: http://www.danielpovey.com/files/2013_interspeech_dnn.pdf

Phone Error Rate (PER) was considered for computing the score of the recognition process. The PER is defined as the sum of the deletion (DEL), substitution (SUB) and insertion (INS) percentage of phonemes in the ASR outcome text with respect to a reference transcription. PCR and SER refers to the Phone Correct Rate and Sentence Error Rate respectively.

Ideally, a hand-labelled reference would have been preferred, because it would have been corrected at the phonetic level to take into account of children's speech pronunciation mistakes. Since this was not available for the CHILDIT corpus, the automatic phonetic sequences obtained by a Viterbi alignment of the word-level orthographic transcription have been used. The reference test transcriptions were created using a SONIC-based aligner using a previously trained children speech Italian acoustic model [28]. This method was chosen because it allowed for automatically selecting the best pronunciation for each word in the training data among the alternative choices available in the 400,000-word Italian lexicon available.

TABLE 1. Preliminary results obtained in the experiments executed on the CHILDIT corpus in various configurations adapting the KALDI's TIMIT-recipe scripts (see text for the definition of all keywords).

	Training & Decoding	%PCR	%SUB	%DEL	%INS	%PER	%SER
MonoPhone	mono	81.6	10.5	7.9	2.5	20.8	99.6
Delta + Delta-Deltas	tri1	87.5	7.1	5.4	1.6	14.1	97.6
LDS + MLLT	tri2	88.7	6.2	5.1	1.4	12.7	96.7
LDA + MLLT + SAT	tri3	91.0	4.9	4.1	1.2	10.2	93.4
sgmm2_4: SGMM2	sgmm2_4	92.1	4.1	3.8	1.0	8.9	91.0
MMI + SGMM2 (iteration n.1)	sgmm2_4_mmi_b0.1	92.8	4.0	3.2	1.5	8.7	90.8
MMI + SGMM2 (iteration n.2)	sgmm2_4_mmi_b0.2	92.9	4.0	3.1	1.6	8.7	89.9
MMI + SGMM2 (iteration n.3)	sgmm2_4_mmi_b0.3	92.9	4.0	3.1	1.6	8.7	90.1
MMI + SGMM2 (iteration n.4)	sgmm2_4_mmi_b0.4	92.9	4.0	3.1	1.6	8.7	90.2
DNN Hybrid (Dan's)	tri4-nnet	90.6	4.8	4.5	1.8	11.1	94.1
SGMM2 + DNN Hybrid (Dan's) (it. 1)	combine_2 (1)	92.9	4.0	3.1	1.3	8.4	89.7
SGMM2 + DNN Hybrid (Dan's) (it. 2)	combine_2 (2)	92.9	4.0	3.1	1.3	8.4	89.9
SGMM2 + DNN Hybrid (Dan's) (it. 3)	combine_2 (3)	92.8	4.0	3.3	1.1	8.3	89.0
SGMM2 + DNN Hybrid (Dan's) (it. 4)	combine_2 (4)	93.0	4.0	3.0	1.3	8.4	89.5
DNN Hybrid (Karel's)	dnn4_pretrain-dbn_dnn	92.7	3.9	3.4	1.4	8.6	90.6
DNN Hybrid (Karel's), sMBR training (it. 1)	dnn4_pretrain-dbn_dnn_smbr (1)	92.9	3.8	3.3	1.2	8.3	89.8
DNN Hybrid (Karel's), sMBR training (it. 6)	dnn4_pretrain-dbn_dnn_smbr (6)	93.2	3.7	3.0	1.4	8.1	88.6

The results shown in Table 1 refer to the various training and decoding experiments (see [23] for all acronyms references):

- MonoPhone (mono);
- Deltas + Delta-Deltas (tri1);
- LDA + MLLT (tri2);
- LDA + MLLT + SAT (tri3);
- SGMM2 (sgmm2_4);
- MMI + SGMM2 [33] (sgmm2_4_mmi_b0.1-4);
- Dan's Hybrid DNN (tri4-nnet),
- system combination, that is Dan's DNN + SGMM (combine_2_1-4);
- Karel's Hybrid DNN (dnn4_pretrain-dbn_dnn);
- system combination that is Karel's DNN + sMBR (dnn4_pretrain-dbn_dnn_1-6).

In the Table:

- SAT refers to the Speaker Adapted Training (SAT), i.e. train on fMLLR-adapted features. It can be done on top of either LDA+MLLT, or delta and delta-delta features. If there are no transforms supplied in the alignment directory, it will estimate transforms itself before building the tree (and in any case, it estimates transforms a number of times during training).
- SGMM2 refers to Semi-supervised training of Gaussian Mixture Models [33] with speaker vectors. This training would normally be called on top of fMLLR features obtained from a conventional system, but it also works on top of any type of speaker-independent features (based on deltas+delta-deltas or LDA+MLLT).

On this difficult phonetic ASR task, Karel's DNN looks slightly better than Dan's DNN and it outperforms all other non-DNN configurations.

V. CONCLUSIONS

At its early stage, the KALDI toolkit supported modeling of context-dependent phones of arbitrary context lengths, all commonly used techniques that can be estimated using maximum likelihood, and almost all adaptation techniques available in the ASR literature. At present, it also supports the recently proposed Semi-supervised Gaussian Mixture Model (SGMMs) [33, 34], discriminative training, and the very promising DNN hybrid training and decoding [12, 13, 20, 21, 22]. Moreover, developers are working on using large language models in the FST framework, and the development of KALDI is continuing.

The adaptation of KALDI to Italian, and in particular to the ChildIt corpus, was indeed quite straightforward, and results are truly exceptional with respect to those previously obtained on the same data. Indeed, this is mainly because all the very last ASR techniques, including DNNs, could be easily implemented by adapting to Italian the already available downloadable scripts written by a quite large number of various developers around the world on a similar task for English.

Acknowledgments

I would like to acknowledge and thank Daniel Povey for his invaluable work on KALDI, indeed a great ASR Toolkit, and for his great help and kindness in letting me use some of the text of KALDI's web pages describing the software. Moreover, I would like to acknowledge Karel Vesely for his great work on his very effective version of DNN.

References

[1] Young S., Evermann G., Gales M., Hain T., Kershaw D., Liu X., Moore G., Odell J., Ollason D., Povey D., Valtchev V., and Woodland P., The HTK Book (for version 3.4). Cambridge Univ. Eng. Dept., 2009.

[2] Pellom B. 2001. "SONIC: The University of Colorado Continuous Speech Recognizer", Technical Report TR-CSLR-2001-01, Center for Spoken Language Research, University of Colorado, USA, 2001.

[3] Pellom B. and Hacioglu K. 2003. "Recent Improvements in the CU SONIC ASR System for Noisy Speech: The SPINE Task", Proc. ICASSP 2003.

[4] Lee K.F., Hon H.W., and Reddy R. (1990), "An overview of the SPHINX speech recognition system", in IEEE Transactions on Acoustics, Speech and Signal Processing 38.1 (1990), 35-45.

[5] Walker W., Lamere P., Kwok P., Raj B., Singh R., Gouvea E., Wolf P., and Woelfel J., "Sphinx-4: A flexible Open Source Framework for Speech Recognition," Sun Microsystems Inc., Technical Report SML1 TR2004-0811, 2004.

[6] Rybach D., Gollan C., Heigold G., Hoffmeister B., Lööf J., Schlüter R., and Ney H., "The RWTH Aachen University Open Source Speech Recognition System," in Proc. Interspeech, 2009, 2111-2114, 2009.

[7] Lee A., Kawahara T., and Shikano K, "JULIUS - an open source real-time large vocabulary recognition engine", in Proc. of Interspeech 2001, 1691-1694.S, 2001.

[8] Povey D., Ghoshal A. et al., "The KALDI Speech Recognition Toolkit", in Proc. of ASRU, 2011.

[9] Simon. WEB site: http://www.simon-listens.com.

[10] Bolaños D., "The Bavieca Open-Source Speech Recognition Toolkit", in Proc. of IEEE Workshop on Spoken Language Technology (SLT), December 2-5, 2012, Miami, FL, USA, 2012.

[11] Bengio Y., "Learning Deep Architectures for AI", in Foundations and Trends in Machine Learning, Vol. 2, No. 1 (2009) 1-127.

[12] WEB - Karel's DNN: http://KALDI.sourceforge.net/dnn1.html

[13] Vesely K., Ghoshal A., Burget L., and Povey D., "Sequence-Discriminative Training of Deep Neural Networks," in Proc. Interspeech 2013.

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

[15] Dahl G.E., Dong Yu D., Deng L, and Acero A., "Context-dependent pre- trained deep neural networks for large-vocabulary speech recognition," Audio, Speech, and Language Processing, IEEE Transactions on , vol. 20, no. 1, pp. 30–42, 2012.

[16] Seide F., Li G., and Yu D., "Conversational speech transcription using context-dependent deep neural networks," in Proc. Interspeech, 2011, pp. 437–440.

[17] Povey D., Kanevsky D., Kingsbury B., Ramabhadran B., Saon G., Visweswariah K., "Boosted MMI for Feature and Model Space Discriminative Training," in Proc. ICASSP, 2008.

[18] Gibson M. and Hain T., "Hypothesis Spaces For Minimum Bayes Risk Training In Large Vocabulary Speech Recognition," in Proc. Interspeech, 2006.

[19] Povey D. and Kingsbury B., "Evaluation of proposed modifications to MPE for large scale discriminative training," in Proc. ICASSP, 2007.

[20] WEB - Dan's-DNN: http://KALDI.sourceforge.net/dnn2.html.

[21] Zhang X, Jan Trmal J., Povey D., Khudanpur S., "Improving Deep Neural Network Acoustic Models Using Generalized Maxout Networks," Proc. ICASSP 2014.

[22] Povey D., Zhang H. and Khudanpur S., Parallel Training of DNNs with Natural Gradient and Parameter Averaging, (under review as a conference paper at ICLR 2015).

[23] Rath S.P., Povey D., Vesely K. and Cernocky J., Improved feature processing for Deep Neural Networks, Interspeech 2013, 109-113.

[24] Hinton G., A Practical Guide to Training Restricted Boltzmann Machines, UTML TR 2010:003, Momentum, 9:1, 2010.

[25] Vesely K., Ghoshal A., Burget L. and Povey D., Sequence-Discriminative Training of Deep Neural Networks, Interspeech 2013, 2345-2349.

[26] Gerosa M., Giuliani D. and Brugnara F., "Acoustic Variability and automatic recognition of children's speech", Speech Communication, Vol. 49, 2007.

[27] Giuliani D. and Gerosa M. 2003. "Investigating Recognition of Children's Speech", Proc. ICASSP, Hong Kong, 2003.

[28] Cosi P. & Pellom B., "Italian Children's Speech Recognition For Advanced Interactive Literacy Tutors", in Proc. of Interspeech 2005, Lisbon, Portugal, 2201-2204, 2005.

[29] Cosi, P., "Recent Advances in Sonic Italian Children's Speech Recognition for Interactive Literacy Tutors", in Proc. of 1st Workshop On Child, Computer and Interaction (WOCCI), Chania, Greece, 2008/.

[30] Cosi, P., "On the Development of Matched and Mismatched Italian Children's Speech Recognition Systems", in Proc. of INTERSPEECH 2009, Brighton, UK, 540-543, 2009.

[31] Cosi, P., Hosom, J.P., "High Performance General Purpose Phonetic Recognition for Italian", in Proc. Of ICSLP 2000, Beijing, 527-530, 2000.

[32] Cosi, P., Nicolao, M., Paci, G., Sommavilla, G., Tesser, F., "Comparing Open Source ASR Toolkits on Italian Children Speech" in Proc. of WOCCI 2014 – Workshop on Child Computer Interaction, Satellite Event of INTERSPEECH 2014, Singapore, September 19, 2014.

[33] Huang, J.-T., Hasegawa-Johnson, M., "Semi-Supervised Ttraining of Gaussian Mixture Models by Conditional Entropy Minimization," in Proceedings. of INTERSPEECH 2010, 1353–1356.

[34] Povey D., Burget L. et al., "The subspace Gaussian mixture model-A structured model for speech recognition," Computer Speech & Language, vol. 25, no. 2, pp. 404–439, April 2011.