Kaldi-based DNN Architectures for Speech Recognition in Romanian

Alexandru-Lucian Georgescu, Horia Cucu, Corneliu Burileanu
Speech and Dialogue Research Laboratory
University Politehnica of Bucharest
Bucharest, Romania
E-mail: lucian.georgescu@speed.pub.ro, horia.cucu@upb.ro, corneliu.burileanu@upb.ro

***Abstract*—Kaldi NNET3 is at the moment the leading speech recognition toolkit on many well-known tasks such as LibriSpeech, TED-LIUM or TIMIT. Several versions of the time-delay neural network (TDNN) architecture were recently proposed, implemented and evaluated for acoustic modeling with Kaldi: plain TDNN, convolutional TDNN (CNN-TDNN), long short-term memory TDNN (TDNN-LSTM) and TDNN-LSTM with attention. To the best of our knowledge, this is the first paper to describe in-depth these various flavors of TDNN, providing details regarding the speech features used at input, constituent layers and their dimensionality, regularization techniques etc. The various acoustic models were evaluated in conjunction with n-gram and recurrent language models in an automatic speech recognition (ASR) experiment for the Romanian language. We report significantly better results over the previous ASR systems for the same Romanian ASR tasks.**

Keywords—automatic speech recognition; neural networks; Kaldi

I. INTRODUCTION

In recent years, there has been an increased effervescence in all the fields which use neural networks to solve some popular tasks, such as speech processing, computer vision and so on. Automatic Speech Recognition (ASR) is one of these areas, where much progress has been made. Currently, there are various implementations, some of them borrowed from related fields, and the results are getting better and better. These implementations can be found in several toolkits, among which Kaldi [1] is the most popular one.

Kaldi was used for ASR in various languages. In [2] the authors present a Kaldi-based ASR recipe for the Arabic language. A French corpus for a home automation task was introduced in [3], along with baseline ASR results using a Kaldi acoustic model. Finally, ASR systems for Russian [4], Mandarin [5] or Vietnamese [6] were recently created using this speech recognition toolkit.

Our efforts over the last few years have focused on building a more robust ASR system for the Romanian language [7, 8]. We have had constant improvements over the time, taking advantage of better algorithms implemented in various toolkits, extending the speech and text corpora used for acoustic and language modeling, etc. Recently we focused on creating state-of-the-art, TDNN-based acoustic models for Romanian and rescoring ASR results with deep recurrent neural networks (RNNs) using the implementations available in Kaldi NNET3 library. Besides reporting the top results obtained for Romanian ASR, the paper also focuses on analyzing and describing in detail the various TDNN-based acoustic model architectures available in Kaldi.

In Section II, we offer an overview of Kaldi ASR pipeline. Section III provides an extended analysis of various Kaldi TDNN-based architectures. Section IV presents the experimental setup, while Section V reports the results. The conclusions are drawn in Section VI.

II. KALDI PIPELINE ARCHITECTURE FOR ASR

A. Kaldi general information

In the Kaldi toolkit, each ASR component is treated as an independent module. All of them are then jointly working as a pipeline system. Therefore, there is a module for each task as feature extraction, acoustic and phonetical modeling and language modeling. As a summary, the features extractor aims to obtain speech features from the acoustic signal. The acoustic model provides the probabilities for each signal sequence, represented by those speech features, given the corresponding sequence of words. The language model deals with the word sequences to match them in phrases with a logical meaning.

Nowadays, the ASR systems are divided into end-to-end and pipeline systems. Kaldi's type is pipeline and differs from the other at the level of each module. Although end-to-end systems can get as input both raw audio wave and handcrafted features, pipeline systems have specific components to extract the speech features. The acoustic model of an end-to-end architecture is capable to deliver characters as output, while in a pipeline architecture, a probability distribution over the phonetic units is provided as output. Another difference from the end-to-end networks is that the DNN-HMM hybrid implementations cannot be trained from scratch, without pre-existing frame-level phonemes alignments. From the linguistic modeling point of view, the end-to-end architectures have embedded mechanisms or network parts that perform this task, or they can plug-in an external language model. In pipeline systems, the language model is always a distinct component that is required at the decoding time, evaluating the probability of the word sequences to form a valid phrase.

B. Feature extractor

Kaldi's feature extraction component can be used to produce traditional speech features, such as (i) Mel filterbanks, (ii) Mel frequency cepstral coeficients (MFCCs) or (iii) perceptual linear features (PLP), extracted from frames of 25 ms with a shift of 10 ms. Depending on the type of acoustic model to be trained, the features extractor produces low-resolution features (e.g. 13 MFCCs per frame to train Gaussian Mixture Models) or high-

978-1-7281-0984-8/19/$31.00 ©2019 IEEE

resolution features (e.g. 40 Mel filterbanks to train TDNN models).

The feature extractor can also be used to extract 100-dimensional i-vectors, encoding speaker and channel information, from longer (1500 ms), non-overlapping frames, denoted with the term chunks. The diagonal Gaussian Mixture Models (GMMs) and Universal Background Model (UBM), created as intermediate steps while training the i-vector extractor are trained on high-resolution features.

C. Acoustic modeling

Kaldi provides support for two main acoustic modeling frameworks, both on them based on Hidden Markov Models (HMMs). The HMM models the sequential characteristic of speech, each HMM state corresponding to a sub-phonetic unit. Each phone is modeled using three HMM states: one for the central, stationary part of the phone and two for the transition between neighboring phones and the respective phone. The HMM output probability density (pdf) is modeled using a GMM or a deep neural network (DNN). Both the GMM and the DNN are used to offer posterior probabilities for all the possible sub-phonetic units that could be associated with an acoustic frame. However, they are fundamentally different models, one being a generative one and the other a discriminative one. With the recent introduction of end-to-end acoustic models, consisting of a single deep neural network accomplishing the role of both the HMM and the pdf estimation model, the HMM-GMM and the HMM-DNN architectures received the label "hybrid systems".

Acoustic model training in Kaldi is also a pipeline process. The first step consists of creating a basic HMM-GMM acoustic model (called *mono*) in which the HMM states model context-independent phones. This model is used to force-align the train dataset in order to have rough estimations of the phones boundaries needed for a more complex model.

In the second step, the forced-alignments are used to train context-independent phone models, also called triphonemes. Just as before, this newly created HMM-GMM model (called *tri1*) is used to produce better forced-alignments of the training dataset to be used in training the next, more complex acoustic model.

In the following steps, several other HMM-GMM acoustic models are created iteratively by (i) applying Linear Discriminant Analysis (LDA) [9] and Maximum Likelihood Linear Transform (MLLT) [10] transforms on the input features (*tri2* model), (ii) performing speaker adaptive training (*tri3* model) and finally (iii) applying the maximum mutual information (MMI) [11] training criteria (*tri3-mmi* model).

While the speech features used for training HMM-GMMs are traditional, 13-dimensional MFCCs along with their first and second order derivatives, from this step onwards, the acoustic models are trained using both MFCCs or filterbanks plus i-vectors. Moreover, for further training steps basic data augmentation is performed by applying volume and speed perturbations on the original speech signals.

The best HMM-GMM acoustic model (tri3-mmi) is further used to force-align the training dataset to produce high-quality alignments for the DNN-based models. These alignments are used to train the various deep networks that can be implemented using Kaldi NNET3 library, among which the most popular are the following: the pure TDNN [12], the factored TDNN (TDNN-F) [13], the CNN-TDNN and the TDNN-LSTM [14]. These DNN architectures can work in conjunction with the more traditional, cross-entropy objective function or with the newly introduced lattice-free MMI (LF-MMI) objective function [15]. The first one was the default in Kaldi NNET2 library, while second one is one of the innovations in Kaldi NNET3. The models using LF-MMI are also called chain models, as the objective function was inspired by the Connectionist Temporal Classification (CTC) [16], another objective function that allows training from scratch, without any prealigned data.

D. Language modeling and lattice rescoring

In general, in speech recognition the language model is used to assess the probability of a certain word in a larger context of several preceding or succeeding words. During the speech decoding process Kaldi only allows the usage of n-gram language models. However, after the decoding process which produces a lattice, a more complex language model can be used to rescore this lattice and finally produce better transcriptions. This process is called lattice rescoring.

The ASR result lattice is a graph G(N,A) representation of all possible transcriptions provided by the decoder. N represents the nodes of the graph, while A denotes the arches. The nodes correspond to the words, while the arches are the transition probabilities between the words. Instead of choosing the best path through this graph, another language model can be used to rescore the probabilities in this graph. Most often, the language model used for rescoring is a more complex, higher order n-gram model, or a recurrent neural network based language model (RNN-LM) [17]. While an n-gram learns from a fixed context, an RNN-LM has the theoretical ability to learn from an infinite context: all previous or even subsequent words (if it is a bidirectional model). However, this is not feasible due to memory and computation restrictions, and, consequently, the RNN context is also limited to a few neighboring words [18].

E. Kaldi decoder

The Kaldi decoding graph, called *HCLG*, is a weighted finite state transducer (WFST) composed of four other graphs. *H* represents the HMM structure, taking as input the acoustic states and providing context-dependent phones (triphones) as output. Theoretically, the number of these context dependent phones should be equal to the cube of the total number of phones. Because that would mean too much, Kaldi uses an automatic clustering algorithm of phone states, choosing then the maximum number of classes. The weights in this graph are the HMM transition probabilities. *C* stands for the context-dependent relabeling, where the input, represented by the context-dependent phones, is linked to context-independent phones. *L* is the lexicon, which links the phones to words. The weights in L denote the pronunciation probabilities. Finally, G is the language model, which takes words as input, modeling their dependency probabilities.

These elements are created separately and then are composed together in an *offline* step, before decoding. The HCLG is obtained by composing from right to left using the formula: *H*(C*(L*G))*. This HCLG is referred to be a static model and each composition is followed by a determinization and a minimization.

III. KALDI-BASED DNNS FOR ASR

This section presents details on the component blocks of the most popular four neural networks used for acoustic modeling in Kaldi (Figure 2). They are trained on the input features presented in Figure 1. For the first network we have taken over the architecture and it dimensions from the TED-LIUM recipe, while the other three implementations are similar to those from the LibriSpeech recipe, all of them being available in Kaldi.

A. TDNN architecture

As depicted in Figure 1 (left), the input in the plain TDNN model is composed of two types of features: 40-dimensional high-resolution MFCCs extracted from frames of 25 ms length and 10 ms shift and 100-dimensional i-vectors computed from chunks of 150 consecutive frames (a chunk covers 1500 ms of speech). Three consecutive MFCC vectors and the chunk's i-vector are concatenated forming a 220-dimentional feature vector for the current frame. LDA is applied on this feature vector to decorrelate the components, while the dimensionality is preserved. Consequently, the TDNN input is a 220-dimensional feature vector (*feat type #1*).

The TDNN's trunk comprises 6 blocks, as illustrated in Figure 2 (left). Besides *TDNN Block 1* which receives at input the speech features *feat type #1*, TDNN blocks operate on the 450-dimensional output of the previous block. Each block is a succession of typical DNN operations, such as affine transforms, ReLU activations and batch normalizations.

As their name imply, each time-delay block (except the first one) performs 1-D temporal convolution, applying the operations on the current input vector as well as some previous and some future input vectors. The contexts differ from one block to another as follows:

- *TDNN Block 2* processes the input vectors at times t-1, t and t+1,
- *TDNN Block 3* processes the input vectors at times t-1, t, t+1, and t+2,
- *TDNN Blocks 4* and 5 process the input vectors at times t-3, t, and t+3,
- *TDNN Block 6* processes the input vectors at times t-6, t-3, and t.

Blocks 4, 5 and 6 use the sub-sampling technique: skipping some time-frames during the temporal convolutions, thus allowing the network to access a broader context. The receptive filed of this pure TDNN model is 14 previous frames and 12 future frames.

It was empirically found that the network performs better if it has two output blocks: the first one is chain based (LF-MMI criteria) and the second one uses the cross-entropy criteria (xent). Each one is composed by affine, ReLU, batch normalization and again the affine layers. The main difference between these two blocks is the existence or absence of the final log-softmax operation. Both blocks are used in the training process, but the inference only uses the output without softmax, because chain models are trained with sequence objective function. The output of the network is 3760-dimensional and it consists in posterior probabilities for the acoustic states. As for the other outputs, this number was automatically chosen, as a result of the phone states clustering operation.

B. CNN-TDNN architecture

As depicted in Figure 1 (right), the input in the TDNN-CNN model is composed of two types of features: 40-dimensional Mel filterbanks extracted from frames of 25 ms length and 10 ms shift and 200-dimensional i-vectors computed from chunks of 150 consecutive frames. The 40 components of the current Mel filterbank vector and the 200 components of the chunk's i-vector are organized in a 40x6 matrix of speech features (*feat type #2*).

The CNN-TDNN architecture is illustrated in Figure 2 (center). *Conv. Block 1* receives at input three matrices of speech features (*feat type #2*): the features for the current, previous and next acoustic frames, or, equivalently, a feature volume of 6 x 40 x 3. It uses 64 filters of size 3x3 to perform time and feature space convolutions and outputs a 64 x 40 x 1 volume.

Three time-consecutive volumes as the one output by *Conv. Block 1* are spliced together to form the 64 x 40 x 3 feature volume representing the input in *Conv. Block 2*. This convolutional block applies another 64 filters of size 3x3 to perform time and feature space convolutions and outputs a 64 x 40 x 1 volume.

The procedure continues with *Conv. Blocks 3 – 6* which apply more filters (128 and finally 256), but preserve the size of the feature volume by decreasing the height from 40 to 20 and finally to 10. The output of the convolutional blocks and the input into the succeeding time-delay blocks is a 2560-dimensional activation vector.

The CNN front-end is followed by 12 blocks of factored TDNN (TDNN-F). There are two key differences between a TDNN-F block and a TDNN block. First, the TDDN-F block comprises a linear-affine sequence of operations that act like a bottleneck transforming the 1536-dimensional input vector into an 160-dimensional intermediary vector and then back into an 1536-dimensional output vector. Second, the TDDN-F block ends with a summation operation that adds the output of the current processing block to the down-scaled (75%) output of the previous block: this acts like a residual connection.

Except *TDNN-F Block 1*, which processes only the input from the current time-frame, all TDNN-F blocks perform 1-D temporal convolution, processing the input vectors at times t-3, t and t+3. The input vectors at times t-3 and t are spliced together into the linear layer, while the input vectors at times t and t+3 are spliced together into the affine layer.

The output blocks of the CNN-TDNN have a similar structure to those from the pure TDNN architecture. A new linear layer and a second batch normalization layer are added to these blocks. The network output is represented by 6016-dimensional acoustic states posterior probabilities.

C. TDNN-LSTM architecture

The input for the TDNN-LSTM architecture is similar to that on the pure TDNN architecture. The difference consists in the fact that 5 instead of 3 40-dimensional MFCC vectors are concatenated with the 100-dimensional i-vector, forming a 300-dimensional speech feature vector. This architecture comprises 6 TDNN blocks interleaved with 3 LSTM blocks (Figure 2, right).

Each TDNN block is composed by an affine layer, followed by ReLU and batch normalization (batch renorm) layers. The affine layer of the first block performs a linear transform from 300-dimensional input to 512-dimensional output. The second

block operates over a context with indexes {-1,0,1}, transforming the 1536-dimensional input to 512-dimensional output. The first LSTM is a unidirectional recurrent layer, which takes a 640-dimensional input, consisting into the output of the second TDNN and the recurrent connection input, which represents the output of the LSTM cell, being equal to 128. The third TDNN block is applied over a context with indexes {-3,0,3}. It takes a gaped output of the LSTM cells, summing a 768-dimensional input and it provides a 512-dimensional output. The 4th TDNN block has the same dimensions as the second block, but it operates on the output of the third block with the context {-3,0,3}. The following layers are similar to the previous ones. The next block is the second LSTM block, which is similar to the first LSTM. This is succeeded by the 5th and 6th TDNN blocks, which are similar to the second, respectively third blocks. The last recurrent layer, LSTM block 3 is linked on top of the 6th TDNN block, being similar to the previous LSTMs.

The final layer of the network takes a 256-dimensional input, corresponding to the forward and backward outputs of the third LSTM. This last block performs an affine transform, a log and a softmax operation, providing an 3760-dimensional output, which represents the posterior probabilities of the acoustic states.

D. TDNN-LSTM-Attention architecture

This architecture is almost identical to the simple TDNN-LSTM, excepting the third LSTM block, which is replaced by a different layer, characterized by the attention mechanism. This feature is specific for the encoder-decoder networks. An encoder is a component designed to encode an entire input sequence over time into a fixed-length context vector representation. The decoder role is to create the output sequence while reading from the context/encoding vector. More briefly, both input and output have a variable length, while the length of the context vector is fixed. The attention model aims to bypass the limitation of fixed length encoding vector. Working with long sequences could be an issue because the neural network must be able to compress all the information into a fixed-length vector. The attention approach encodes the input into a sequence of vectors and choose a subset of them during the decoding. Each time the network tries to generate a new output symbol, it looks at a set of positions in the source sequence where the relevant information is concentrated.

Consequently, in our case, the attention mechanism encodes a context equal to 5 inputs at the left and 2 inputs at the right of the current frame, using a stride equal to 3. These inputs represent the output of the 6th TDNN block. The attention layer is followed by ReLU and normalization layers.

IV. Experimental Setup

From the acoustic modeling point of view, all the experiments are based on Kaldi DNN architectures: pure TDNN, CNN-TDNN, TDNN-LSTM, TDNN-LSTM-Attention. The values of the parameters, the network input, the number and the type of the layers are exactly those which were previously described in Section III.

From the language modeling point of view, we performed first pass decoding using a 2-gram LM and lattice rescoring using a 4-gram LM, in a similar manner as we did in our past experiments [7] and using the same models. These n-gram models were obtained by applying an interpolation operation with a 0.5 factor, over two text corpora. The first one contains 315M words collected from news websites and another one is composed by 40M words from talkshow transcripts. All the LMs have vocabularies of 200k words. As an update for the current work, we performed lattice rescoring using RNN-LM with a 5 words context. This network is composed by 3 TDNN blocks, composed by affine, ReLU and renorm layers. Between these blocks, two LSTM layers are interleaved. The embedding size is 800.

The speech corpora are the same as described in our previous experiments [7]. The RSC corpus contains read speech utterances recorded in a silent environment. The SSC corpus contains spontaneous utterances, gathered from the Romanian media broadcasts. Some of them are clean, coming from a silent environment, while the others are polluted by background noise. Both RSC and SSC were split into *train* and *eval* sets, used to train and to evaluate the ASR systems. Details about the datasets duration are provided in Table 1.

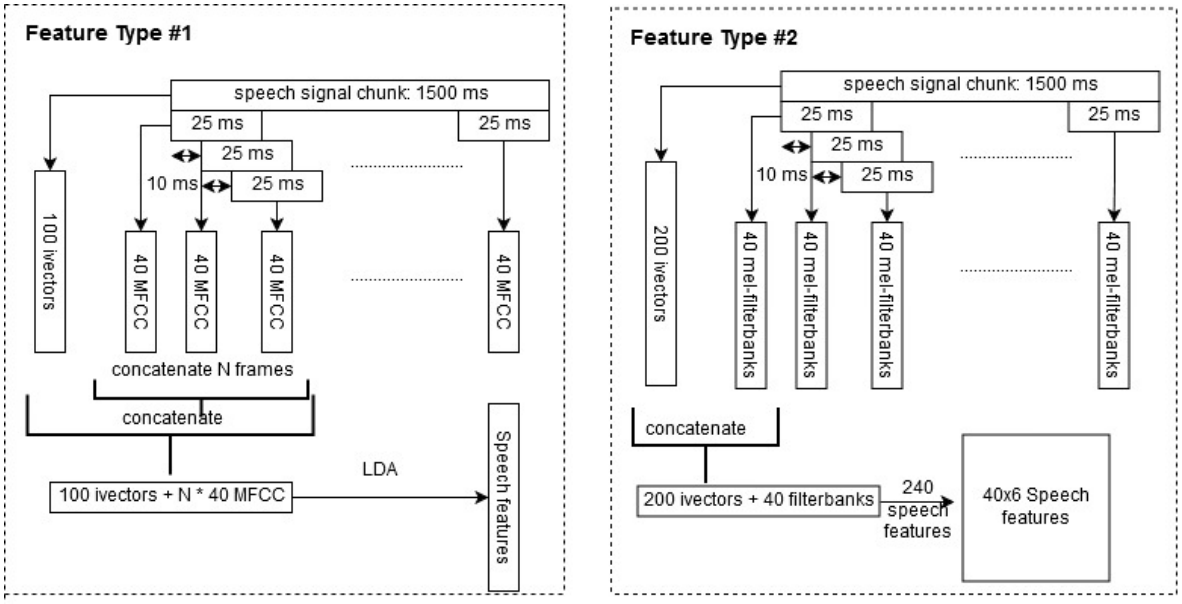

Figure 1. Feature types

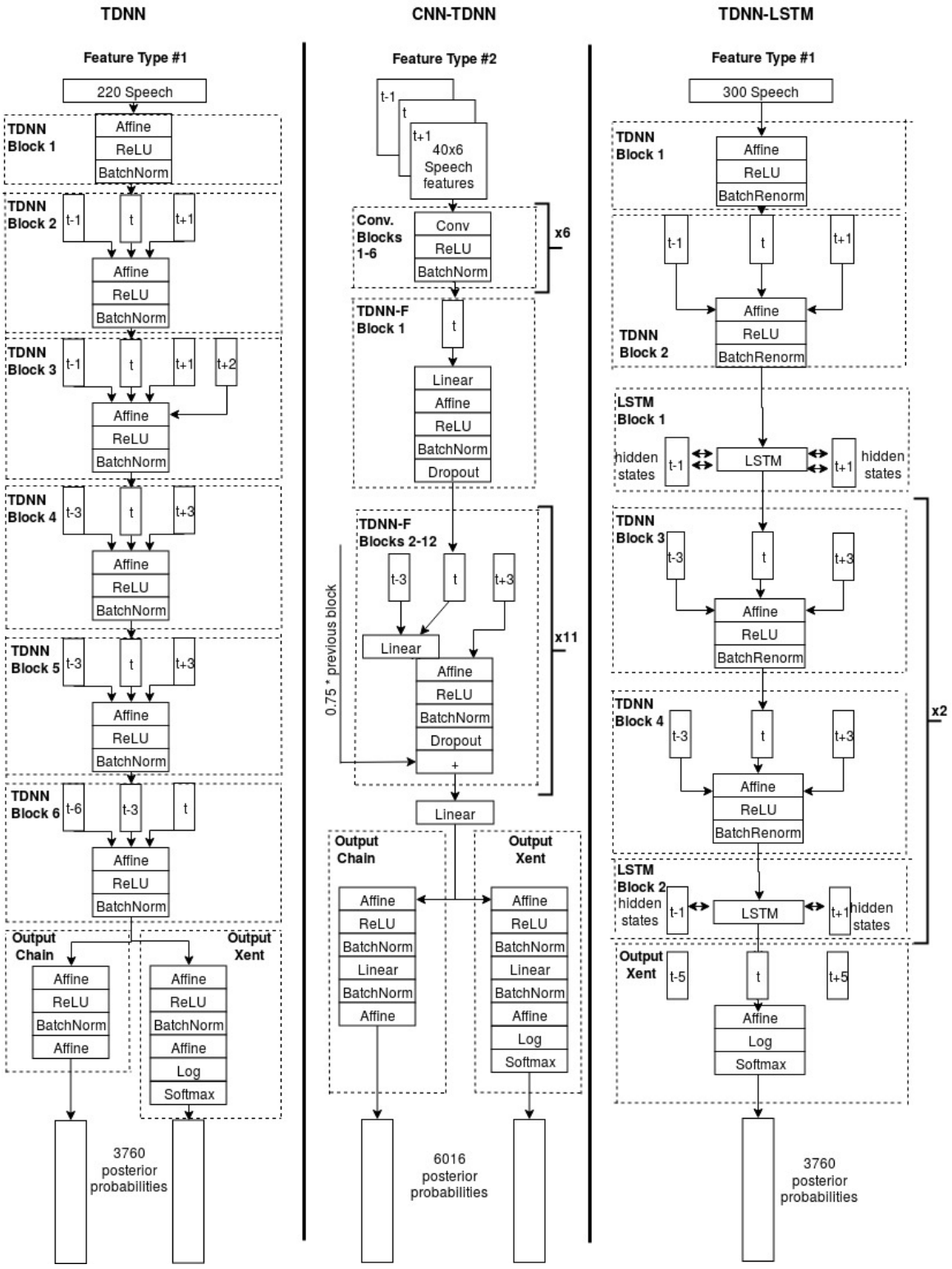

Figure 2. Neural network architectures for acoustic modeling

V. EXPERIMENTAL RESULTS

Table 2 comprises the results obtained using the Kaldi-based DNN networks for acoustic modeling. The general impression over the results confirm that the pure TDNN is the best algorithm, followed by the CNN-TDNN, which outperforms the TDNN-LSTM-Attention and TDNN-LSTM. This conclusion is pretty clear regarding the RSC-eval corpus, while for the SSC-eval, the CNN-TDNN exchanges places in this hierarchy with pure TDNN. Relative improvements up to 33% on RSC-eval and up to 17% on SSC-eval are obtained when the best TDNN, respectively CNN-TDNN are chosen in detriment of the other models.

Almost all the time, the RNN-LM rescoring outperforms the 4-gram rescoring, which is better than just 2-gram decoding. This trend is kept for the RSC-eval, while in the case of the SSC-eval, sometimes the RNN-LM is worse than n-gram rescoring.

Reported to our previously published results [7], the overall relative improvement of the Romanian ASR system is around 38% on RSC-eval and more than 17% on SSC-eval.

TABLE 1. SPEECH CORPUS

Purpose	Set	Duration	
Training	RSC-train	94 h 46 m	225h 30 m
	SSC-train	130 h 44 m	
Evaluation	RSC-eval	5 h 29 m	8 h 58 m
	SSC-eval	3 h 29m	

TABLE 2. WORD ERROR RATES USING VARIOUS AMS AND LMS

Acoustic model	Language model	Task	RSC-eval	SSC-eval
TDNN	2-gram	decoding	4.27	19.70
	4-gram	rescoring	3.32	16.85
	RNN	rescoring	**2.79**	16.63
CNN-TDNN	2-gram	decoding	4.32	18.38
	4-gram	rescoring	3.44	16.00
	RNN	rescoring	3.33	16.18
TDNN-LSTM-Attention	2-gram	decoding	4.93	22.05
	4-gram	rescoring	4.02	19.01
	RNN	rescoring	3.68	19.43
TDNN-LSTM	2-gram	decoding	5.49	21.96
	4-gram	rescoring	4.55	19.25
	RNN	rescoring	4.20	19.21

VI. CONCLUSION

In this paper, we provided an overview about how a pipeline ASR system based on Kaldi works. The main components, like the feature extractor, acoustic model and language model were described. Several neural network architectures from the NNET3 Kaldi setup were in-depth analyzed. We provided details about each network, regarding the component blocks and their specific layers and dimensionality.

Moreover, ASR experiments on Romanian speech corpora were performed. For the acoustic modeling, we used the analyzed architectures: TDNN, CNN-TDNN, TDNN-LSTM and TDNN-LSTM-Attention. For the language modeling, n-gram decoding, as well as n-gram and recurrent based rescoring were applied.

It has been shown that the pure TDNN generally obtains better results than the other networks. The n-gram rescoring provides more accurate transcripts over the decoding and that rescoring using an RNN-LM usually performs the best.

The overall relative WER improvements, compared to the 2017 Romanian ASR system [7], are 38% on RSC-eval and 17% on SSC-eval.

ACKNOWLEDGMENT

This work was supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI – UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0818, within PNCDI III.

REFERENCES

[1] Kaldi ASR Toolkit. https://kaldi-asr.org/

[2] A. Ali et al., "A complete KALDI recipe for building Arabic speech recognition systems," IEEE Spoken Language Technology Workshop (SLT), South Lake Tahoe, NV, 2014, pp. 525-529.

[3] N. Bertin et al., "A French corpus for distant-microphone speech processing in real homes", Interspeech 2016, San Francisco, United States.

[4] I. Kipyatkova, A. Karpov, "DNN-based acoustic modeling for Russian speech recognition using Kaldi", International Conference on Speech and Computer. Springer, Cham, 2016. p. 246-253.

[5] H. Bu et al., "Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline", Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA). IEEE, 2017.

[6] H.T. Luong, H.Q. Vu, "A non-expert Kaldi recipe for Vietnamese speech recognition system", In Proceedings of the Third International Workshop on Worldwide Language Service Infrastructure and Second Workshop on Open Infrastructures and Analysis Frameworks for Human Language Technologies, 2016, pp. 51-55.

[7] H. Cucu, A. Buzo, L. Petrică, D. Burileanu and C. Burileanu, "Recent improvements of the SpeeD Romanian LVCSR system," 10th International Conference on Communications (COMM), Bucharest, 2014, pp. 1-4.

[8] A.L. Georgescu, H. Cucu, C. Burileanu, "SpeeD's DNN approach to Romanian speech recognition", In International Conference on Speech Technology and Human-Computer Dialogue (SpeD), 2017, pp. 1-8).

[9] P. Somervuo., "Experiments with linear and nonlinear feature transformations in HMM based phone recognition", IEEE Int. Conf. on Acoustics, Speech and Signal processing,, vol. 1, pages 52–55, 2003.

[10] M. J. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition", Computer speech & language, 1998, pp. 75-98.

[11] D .Povey et. al., "Boosted MMI for model and feature-space discriminative training", In ICASSP 2008, pp. 4057-4060.

[12] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," INTERSPEECH, 2015.

[13] D. Povey et. al., "Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks", In Interspeech 2018, pp. 3743-3747.

[14] V. Peddinti et. al., "Low latency acoustic modeling using temporal convolution and LSTMs", IEEE Signal Processing Letters, 25(3), 2017, 373-377.

[15] D V. Peddinti et. al., "Purely sequence-trained neural networks for ASR based on lattice-free MMI", In Interspeech 2016, pp. 2751-2755.

[16] A. Graves et al., "Connectionist temporal classification: labelling unsegmented se-quence data with recurrent neural networks," In Proceedings of the 23rd international conference on Machine learning", 2006, pp. 369–376

[17] T. Mikolov et al., "Recurrent neural network based language model." Eleventh annual conference of the international speech communication association. 2010.

[18] H. Xu et al., "A pruned rnnlm lattice-rescoring algorithm for automatic speech recognition. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 5929-5933). IEEE.