NETWORK ARCHITECTURES FOR MULTILINGUAL SPEECH REPRESENTATION LEARNING

Tom Sercu, George Saon, Jia Cui, Xiaodong Cui, Bhuvana Ramabhadran, Brian Kingsbury, Abhinav Sethy

IBM Watson, IBM T. J. Watson Research Center, Yorktown Heights, NY.

ABSTRACT

Multilingual (ML) representations play a key role in building speech recognition systems for low resource languages. The IARPA sponsored BABEL program focuses on building speech recognition (ASR) and keyword search (KWS) systems in over 24 languages with limited training data. The most common mechanism to derive ML representations in the BABEL program has been with the use of a two-stage network, the first stage being a convolutional network (CNN) from where multilingual features are extracted, expanded contextually and used as input to the second stage which can be a feed-forward DNN or a CNN. The final multilingual representations are derived from the second network. This paper presents two novel methods for deriving ML representations. The first is based on Long-Short Term Memory (LSTM) networks and the second is based on a very deep CNN (VGG-net). We demonstrate that ML features extracted from both models show significant improvement over the baseline CNN-DNN based ML representations, in terms of both speech recognition and keyword search performance and draw the comparison between the LSTM model itself and the ML representations derived from it on Georgian, the surprise language for the OpenKWS evaluation.

Index Terms— LSTM, VGG, multilingual, acoustic model, keyword search

1. INTRODUCTION

Multilingual ASR has been investigated over the last two decades. With the recent success of Deep Neural Networks (DNNs), and their ability to generalize and learn useful acoustic representations of languages, there has been increased interest in using neural network based multilingual representations. Multilingual (ML) models have been shown to outperform unilingual models for ASR in low resource languages [1, 2]. The IARPA sponsored-BABEL program has demonstrated their advantages in keyword search (KWS) tasks [3, 4, 5, 6, 7]. ML models are also used in the Spoken Web Search Task held as part of MediaEval Benchmark [8].

The most popular framework for deriving these ML representations remains the hierarchical, two-stage DNN-DNN or CNN-DNN architecture [5, 7]. The first stage is typically a convolutional network (CNN) from where multilingual features are extracted, expanded contextually and used as input to the second stage which can be a feed-forward DNN or a CNN. The final multilingual representations are derived from the second network. In this paper, we propose two novel ML representations. Inspired by the recent success of Long-Short Term Memory(LSTM) networks as acoustic and language models, and the use of very deep convolutional neural networks as acoustic models [9], we explore these architectures for deriving ML representations. We demonstrate that these architectures lend themselves to the extraction of better ML representations than the well-benchmarked, hierarchical CNN-DNN architecture on the Babel Optional Period 3 (OP3) surprise language (Georgian) evaluation. Given the large amount of data (approximately 1000 hours spanning $24+$ languages) used for ML training, the derivation of good ML representations from complex, state-of-the-art networks can be very time consuming. We demonstrate that these new ML representations can be produced in one-fourth the training time of the baseline CNN-DNN representation.

The rest of the paper is organized as follows. Section 2 introduces the proposed LSTM and VGG architectures to derive ML representations. Section 3 covers details on training these networks and experimental results obtained with two different implementations of the VGG-inspired ML feature extractors. Next, we explore LSTM-based ML representations (Section 4). These representations not only improved ASR and KWS performance, but also shortened the training time significantly. We present the impact of fine-tuning the ML network on the target languages with different strategies in Section 5 and compare the performance of a system trained with LSTM-based ML representations with the LSTM acoustic model directly.

This work is supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0012. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government. This effort uses the IARPA Babel Program language collection release IARPA-babel{202b-v1.0d, 101-v0.4c, 102b-v0.5a, 103b-v0.4b, 104b-v0.4aY, 105bv0.4, 106-v0.2f, 107b-v0.7, 201b-v0.2b, 203b-v3.1a, 206bv0.1d, 204b-v1.1b}.

2. PROPOSED ARCHITECTURES FOR ML REPRESENTATIONS

Convolutional neural networks (CNNs) have become an important class of machine learning models, achieving state-of-the-art results on tasks in computer vision [10] and speech recognition [11, 12]. Very deep "VGG" style convolutional networks were introduced in [13] for the computer vision domain as a submission in the ImageNet classification challenge. The central idea is to replace convolutional layes with large kernels with a stack of small 3×3 layers, increasing the depth and nonlinearity. VGG style CNNs were introduced as acoustic models in the speech domain [9], and demonstrated strong performance on the English conversational telephony task. Multilingual deep CNNs were introduced in [9, 17], though only in a small scale (18h of data) and for ASR evaluation only. These multilingual CNNs were analogous to multilingual DNNs, but with the additional variation of untying multiple fully connected layers per language. Long Short Term Memory (LSTM) networks were first introduced in [14] to overcome vanishing and exploding gradient issues in training recurrent networks for modeling sequences. LSTMs incorporate temporal memory using memory gates, control error flow through input gates and use forget gates to adaptively reset the cell's memory. Deep LSTM networks have been successfully used in ASR [15, 16]. We propose the use of these deep CNNs and LSTMs as feature extractors.

3. DERIVATION OF VGG-BASED ML REPRESENTATIONS

3.1. VGG-1

The first VGG-based ML feature stream, VGG-1, is trained on data from 24 Babel languages, and uses a two-stage, stacked bottleneck architecture. The first-stage CNN is configured in a VGG style architecture consisting of five convolutional/pooling layers followed by five fully connected layers, where the fourth is a bottleneck layer. The convolutional layers employ 3×3 windows and the corresponding pooling layers employ 3×1 windows (that is, they perform max-pooling in frequency, but not in time) . Both the convolutional and pooling layers use a stride of 1×1. The input to the first stage bottleneck CNN is 40-dimensional log-Mel features with temporal deltas and double deltas, which amounts to three input feature maps. The numbers of feature maps used in the five convolutional layers are 128, 128, 128, 128, and 256. These were determined empirically. In the fully connected layers, the bottleneck layer has 80 hidden units while the rest have 1024 hidden units. The last hidden layer is connected to the output layer with 3000 output units from 22 languages and 2500 output units from 2 languages. Sigmoid activation functions are used in the convolutional and fully connected layers while softmax functions are used in the language-specific output layers.

In the second stage of the VGG-1 pipeline, a bottleneck DNN with only fully connected layers is used. The input to the second stage DNN is the output of the bottleneck layer of the first stage CNN augmented with its four preceding and following frames. There are five hidden layers in total and the fourth one is a bottleneck layer. The bottleneck layer has 80 hidden units while the rest have 1024 hidden units. Like in the first stage CNN, the last hidden layer is connected to the output layer with 3000 output units from 22 languages and 2500 output units from 2 languages. Sigmoid activation functions are used in all hidden layers and softmax functions are used in the language-specific output layers. The ML representations are derived from the linear output of the bottleneck layer.

The final VGG-1 features are speaker-adapted. To compute the speaker-adapted feature stream, 9 successive frames of raw features are spliced into a supervector, projected to 80 dimensions using an LDA transformation, and subsequently diagonalized using a global semi-tied covariance (STC) transform. Finally, a single CMLLR transform is learned per conversation side to further normalize the features. These are the speaker-adapted, ML VGG features.

3.2. VGG-2

The second VGG-based ML feature stream, called VGG-2, is trained on the 24 Babel languages and the data (Full Language Pack of 40hours) provided for Georgian, which was the surprise language in the evaluation. Similar to VGG-1, VGG-2 takes 40-dimensional log-Mel features with temporal deltas and double deltas as input. The VGG-2 network comprises 12 convolutional layers, with a max-pooling layer inserted after every 3 convolutional layers, followed by 5 fully connected layers. The network uses language-dependent output layers, similar to the VGG-1 network and the baseline hierarchical, CNN-DNN, multilingual network. All hidden layers use ReLU nonlinearities, while the outputs use softmax nonlinearities. The first convolutional layer uses 7×7 kernels, while all remaining layers use 3×3 kernels. The convolutional layers output 64, 64, 64, 128, 128, 128, 256, 256, 256, 512, 512, and 512 feature maps, proceeding from the first layer to the twelfth. The first two pooling layers downsample in frequency by a factor of two, but do not downsample in time, while latter two pooling layers downsample in time and frequency by a factor of two. The five fully connected layers contain 2048, 2048, 2048, 256, and then 3000 or 2500 units (depending on the number of HMM states used for a given language). Training uses the variant of batch normalization [18] described in [17] and Nesterov's accelerated optimization with momentum. In contrast to [9], in order to accommodate the bottleneck layer before the output layer shared across all languages, we do not untie more than one layer. The final ML representation is derived from the fourth, fully-connected, 256-unit bottleneck layer.

4. DERIVATION OF LSTM-BASED ML REPRESENTATIONS

We trained two multilayer bidirectional LSTMs on $logmel + \Delta + \Delta\Delta$ features. The LSTMs share a common architecture consisting of 4 layers with 1024 cells per layer (512 per direction), a bottleneck layer of 256 units, and an output layer of size 3000, but differ in the type of training data. The output layer uses a softmax non-linearity. The first LSTM (LSTM-1) was trained on 24 languages with the cross-entropy objective. The second LSTM (LSTM-2) was initialized with the fully CE-trained multilingual LSTM and fine-tuned on the evaluation language (Georgian) training data. The sequence training criterion used was sMBR with cross-entropy smoothing [19]. For the second LSTM, we also experimented with varying the number of outputs by growing phonetic decision trees to 3000, 6000 and 10000 leaves. The final ML representation is derived from the 256-unit bottleneck layer.

5. EXPERIMENTAL RESULTS

5.1. Acoustic and Language Models

Context-dependent realizations of graphemic targets serve as the output units for the neural networks used throughout this paper. The acoustic models presented in this paper are DNNs built on ML features, unless explicitly stated. They only differ in the input feature representations and are all trained with: (1) Layer-wise discriminative pretraining using stochastic gradient and cross-entropy loss, (2) Cross-entropy training with Stochastic gradient, and (3) Distributed Hessian-free optimization and the state-level minimum Bayes risk loss.

The input features used to train these acoustic models are either the ML representations derived from the baseline CNN-DNN, VGG or LSTM architectures. This final DNN has 5 hidden layers with 1024 hidden units in each layer and one output layer with 3000 output units. ReLU activation functions are used for the three bottom-most hidden layers and sigmoid activation functions are used for the top-most hidden layers. Softmax functions are used in the output layer.

Two baseline ML models were trained. The ML representations for both were derived from the hierarchical CNN-DNN architecture. One ML representation was derived from a network trained with 24 languages, referred to as ML-24. The other, referred to as ML-28, was trained with 24 languages and in addition, English, Arabic, Mandarin and Spanish. Each of these four new languages contain about 200 hours of training data. The language model is a bigram model built on 95M words (Full Language Pack plus web text), with a lexicon of 313K words. The decoder and on-the-fly lattice generation used for key word search are described in [7].

5.2. Baseline ML Features

Table 1 illustrates the speech recognition performance obtained with the baseline, VGG-2 and LSTM-based ML representations. None of these models have seen the target language, Georgian, during training.

ML feature	Cross-entropy (WER %)	sMBR loss objective (WER %)
ML-24	46.0	41.6
ML-28	45.1	41.6
VGG-2	44.9	41.4
LSTM-1	44.5	41.2
ML-24+ VGG-2 +LSTM-1	43.4	40.4

Table 1. *Word Error Rates (WERs) of DNNs built on various speaker-independent multilingual representations*

It can be seen that the additional of the four new languages did not make a difference in the performance of the ASR system after sequence training, although a difference of 1% absolute can be seen at the cross-entropy loss based training. The speaker independent ML representations derived from VGG and LSTM networks are slightly better than the baseline ML representations. The last row in Table 1 is a DNN built by fusing all three ML representations. The ML representations seem complimentary enough to provide a reduction in WER of 1% absolute. The VGG-1 speaker adapted ML representations yielded a WER: of 41.1%. Interestingly ,VGG-2 speaker-independent ML representations performed similar to VGG-1, even though the underlying network architectures are also very different.

5.3. Fine tuning Strategies

In this section, we study the impact of fine-tuning the multilingual networks on the target language, Georgian. The fine-tuning strategy used for the baseline model was a few training iterations of the second DNN in the hierarchical architecture on the target language. With the baseline ML-24 features, this did not yield any further reduction in WER over 41.6% reported in Table 1. Fine-tuning of both CNN and DNN stages, yielded a slight reduction in WER to 41.3%. Exploring this strategy on the VGG net did not provide any appreciable decrease in WER (44.7%). Next, we explored a fine-tuning strategy of sampling equal parts of Georgian and the 24 languages (fVGG-2). As can be seen from Table 2, this was successful in adapting the features to Georgian while roughly keeping the same performance on the other 24 Babel languages, yielding the best performing system at 39.7% WER. However, with the LSTM-based ML representation, we found that initializing with the cross-entropy trained multilingual LSTM (LSTM-1) followed by a few training passes on the Georgian data, was sufficient to yield an absolute 0.5% improvement in WER (LSTM-2).

ML feature	Cross-entropy (WER %)	sMBR (WER %)
fVGG-2	42.7	39.7
LSTM-2	43.2	40.7

Table 2. *Impact of fine-tuning the multilingual networks on the target language*

In Table 3, we present the ASR and key word search performance of the two novel ML representations proposed in this paper. The first two rows in the table correspond to the two baseline ML features. It can be seen that the KWS performance of the two systems is quite different even though the ASR performance is identical. The fine tuned VGG-2 features yield the best performance for ASR and keyword search.

Model	WER (%)	MTWV
ML-24	41.6	0.681
ML-28	41.6	0.6947
VGG-2	41.4	0.6928
VGG-2 (fine tuned)	39.7	0.705
VGG-1 (speaker adapted ML features)	41.1	0.6942
LSTM-2	40.7	0.6993

Table 3. *ASR (WER) and KWS performance (MTWV) of VGG and LSTM based representations*

5.4. Training Speed ups

The number of parameters in VGG-style CNNs is similar to the classical 2-layer CNNs. However, the increased context and number of convolutional layers makes for a significantly larger amount of computation per frame (or per epoch). We compensate for this increased computation per frame (w.r.t. classical CNNs) in two ways. First, the number of frames before convergence is strongly reduced from 15 to 2 epochs with no loss in performance. Second, our torch-based implementation lends itself to easy parallelization across multiple GPUs. We use data parallelization, where a batch of 512 frames is split into 4 batches of 128 frames. This provided an overall speedup of a factor of 3.5 per batch. Training took 10.25 days followed by equal parts finetuning (fVGG-2) for 1.25 days.

The LSTM implementation was done in Torch and is based on NVIDIA's highly optimized cuDNN 5.0 library. Training of LSTM-1 took 7.7 days on a single K-80 GPU. Training for LSTM-2 took 5 hours for CE and 20 hours for SGD-based sequence discriminative training. This model was one of the fastest models to train compared to the VGG or baseline (approximately a month) models.

5.5. Decision Tree Sizes

In order to get the best possible performance, we experimented with Increasing the number of leaves in decision trees (on the target language) for the baseline, VGG and LSTM networks. The word error rates and MTWVs for VGG and LSTM based ML DNNs are presented in Table 4 and Table 5. As can be seen, increasing the number of context-dependent HMM states for the final DNN built off any ML representations is beneficial for both WER and KWS performance.

Model	WER	MTWV
ML28	41.1	0.6939
fVGG	39.4	0.7066

Table 4. *Word error rates and MTWVs for baseline and VGG ML features with 6000 output targets*

# outputs (leaves)	WER	MTWV
3000	40.7	0.6993
6000	40.3	0.7115
10000	40.0	0.7129

Table 5. *Word error rates and MTWVs for LSTM ML features with different number of output targets*

5.6. Comparison of LSTM and LSTM based ML features

Tabel 6 compares the performance of DNNs trained on the baseline and fine-tuned LSTM-2 ML representations with 6000 output targets. We also explored using LSTM-2 directly in the KWS pipeline,(LSTM-2 direct) without using it as a multilingual feature extractor. This gives a further improvement in KWS performance, perhaps, due to denser lattices while the WER stays the same. Although fine tuned VGG based ML features (Table 4 show the best MTWV performance among all the ML representations, the LSTM-2 direct model outperforms all ML representations in KWS performance.

Model	WER	MTWV
ML-28	41.1	0.6939
LSTM-2	40.3	0.0.6993
LSTM-2 direct	40.0	0.7149

Table 6. *LSTM performance with 6000 output targets*

6. SUMMARY

We have presented two new architectures, based on VGG and LSTM for deriving ML representations. The non-fine tuned versions of these features are only slightly better in WER than the CNN+DNN ML baseline. The fine-tuned VGG based ML representations provides a 1.7% absolute reduction in WER and a 0.01 absolute gain in MTWV for KWS. This is the best among all ML representations. The LSTM based ML representation are 1% absolute better than the baseline in WER while matching the KWS performance. However, the LSTM ML model when used directly for KWS gained 0.02 absolute[1] in MTWV.

[1] 3% relative which is very significant in this task

7. REFERENCES

[1] Stefano Scanzio, Pietro Laface, Luciano Fissore, Roberto Gemello, and Franco Mana, "On the use of a multilingual neural network front-end," in *Interspeech*, 2008.

[2] L. Burget, P. Schwarz, M. Agarwal, Pinar Akyazi, K. Feng, A. Ghoshal, O. Glembek, N. Goel, M. Karafiát, D. Povey, A. Rastrow, R. C. Rose, and S. Thomas, "Multilingual acoustic modeling for speech recognition based on subspace gaussian mixture models," in *Proc. ICASSP*. IEEE, 2010, pp. 4334–4337.

[3] Mary P. Harper, "http://www.iarpa.gov/index.php/research-programs/babel," .

[4] J. Fiscus, J. Ajot, and G. Doddington, "The spoken term detection (std) 2006 evaluation plan," *NIST USA Sep*, 2006.

[5] Z. Tuske, R. Schluter, and H. Ney, "Multilingual hierarchical MRASTA features for ASR," in *Interspeech*, 2013, pp. 2222–2226.

[6] K. Knill, M.J.F. Gales, S. Rath, P. Woodland, C. Zhang, and S.-X. Zhang, "Investigation of multilingual deep neural networks for spoken term detection," in *ASRU*, 2013.

[7] Jia Cui, Brian Kingsbury, Bhuvana Ramabhadran, Abhinav Sethy, Kartik Audhkhasi, Xiaodong Cui, Ellen Kislal, Lidia Mangu, Markus Nussbaum-Thom, Michael Picheny, et al., "Multilingual representations for low resource speech recognition and keyword search," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 259–266.

[8] Florian Metze, Xavier Anguera, Etienne Barnard, Marelie Davel, and Guillaume Gravier, "The spoken web search task at MediaEval 2012," in *Proc. ICASSP*. IEEE, 2013, pp. 8121–8125.

[9] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. LeCun, "Very deep multilingual convolutional neural networks for lvcsr," in *ICASSP*, 2016.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition," in *2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP)*. IEEE, 2012, pp. 4277–4280.

[12] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran, "Deep convolutional neural networks for lvcsr," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8614–8618.

[13] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[14] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] A. Graves, A. R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*, 2013.

[16] Haşim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *arXiv preprint arXiv:1402.1128*, 2014.

[17] T. Sercu and V. Goel, "Advances in very deep convolutional neural networks for lvcsr," in *Interspeech*, 2016.

[18] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[19] Hang Su, Gang Li, Dong Yu, and Frank Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 6664–6668.