Odyssey 2018 The Speaker and Language Recognition Workshop
26-29 June 2018, Les Sables d'Olonne, France

BUT/Phonexia Bottleneck Feature Extractor

Anna Silnova[1], *Pavel Matějka*[1,2], *Ondřej Glembek*[1], *Oldřich Plchot*[1],
Ondřej Novotný[1], *František Grézl*[1], *Petr Schwarz*[2], *Lukáš Burget*[1], *Jan "Honza" Černocký*[1]

[1]Brno University of Technology, Speech@FIT and IT4I Center of Excellence, Czechia
[2] Phonexia, Czechia

{isilnova|matelkap}@fit.vutbr.cz

Abstract

This paper complements the public release of the BUT/Phonexia bottleneck (BN) feature extractor. Starting with a brief history of Neural Network (NN)-based and BN-based approaches to speech feature extraction, it describes the structure of the released software. It follows by describing the three provided NNs: the first two trained on the US English Fisher corpus with monophone-state and tied-state targets, and the third network trained in a multi-lingual fashion on 17 Babel languages. The NNs were technically trained to classify acoustic units, however the networks were optimized with respect to the language recognition task, which is the main focus of this paper. Nevertheless, it is worth noting that apart from language recognition, the provided software can be used with any speech-related task. The paper concludes with a comprehensive summary of the results obtained on the NIST 2015 and 2017 Language Recognition Evaluations tasks.

1. Introduction

The BUT group has a long tradition of research of NN-derived features. Back in the years 2000-2002, most of its (now senior) members worked with Hynek Hermansky and his OGI group on temporal pattern (TRAP) features that followed the early work of Sharma et al. [1]. Already at that time, the processing worked in two stages: the temporal evolutions of energies in spectral bands were processed by the first-stage neural networks, whose outputs were then merged by a merging NN. In *tandem*-based ASR[1], the NN-based features did not outperform the classical MFCC and PLP coefficients. However, they have become the basis of the very successful BUT phone recognizer, which, in the early 2000's, produced the state-of-the-art results on TIMIT [2], and was released publicly[2]. It then served BUT and many other groups as an excellent tokenizer for phonotactic language identification [3, 4].

In the follow-up research, it became evident that phone posterior probabilities were not the best features for tandem ASR; better results could rather be achieved by taking features from a narrow hidden NN layer, commonly referred to as *bottleneck*. The newly introduced bottleneck (BN) features have, for the first time, outperformed the classical MFCC/PLP coefficients [5]. We have then settled on stacked bottleneck (SBN) feature architecture (described in detail in Sec. 3.3), that again comprises two NN stages: the first looking at a rather short temporal context, and the second processing several down-sampled outputs of the first, reaching a temporal context of 310 ms. These features have been for long time our production features for ASR, and have surrendered to simpler features (filter-bank outputs again) only recently, with the introduction of LSTMs that are more powerful in modeling the temporal evolution of features [6].

In addition to ASR, SBN features have a long and successful track in language and speaker recognition. The use of BN features for LID was investigated in [7, 8] with 45% relative improvement to acoustic features baseline on DARPA RATS database. A thorough summary of our recent work with BN features in LID has been presented in [9]. Finally, SBN features again contributed to a superior performance of BUT-UAM-Phonexia-PoliTo system in 2017 NIST Language recognition evaluation (LRE17) [10].

Although trained to model the phonetic content (and therefore discard any speaker information), BN features have been successfully used in speaker recognition, as well. Numerous sites reported good results either with BN features themselves, or in a conjunction with conventional acoustic features, or as DNN alignment for GMMs [11, 12, 13, 14, 15].

This paper and the release of the BN feature extractor continue in our tradition of releasing state-of-the-art techniques to the R&D community with the aim of faster innovation and reproducible science.

The rest of the paper covers i) the internals of the software release in Sec. 2, ii) details on the training of included NNs in Sec. 3, and iii) a summary of results achieved with BN features extracted with the released package on NIST LRE15 and LRE17 data [10, 16]. Sec. 5 provides the necessary download information.

2. Overview of BN extractor

BUT/Phonexia bottleneck feature extractor is a Python toolkit allowing to extract bottleneck features or phoneme classes posterior probabilities from a given audio signal. The package includes the following components.

audio2bottleneck.py - a Python script to extract SBN features for a given audio recording in `.wav` format. The script could be modified to save BN features in addition to SBNs. Also, by default, the full feature matrix is saved. One can change the script to save features only

The work was supported by Czech Ministry of Interior project No. VI20152020025 "DRAPAK", Technology Agency of the Czech Republic project No. TJ01000208 "NOSICI", and by Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602". The project was also supported by the Czech Science Foundation under project No. GJ17-23870Y.

[1]"Tandem" denoted NN feature extractor feeding a classical HMM/GMM system.

[2]http://speech.fit.vutbr.cz/software/phoneme-recognizer-based-long-temporal-context

10.21437/Odyssey.2018-40

for speech frames according to provided VAD labels or to internally computed energy-based VAD.

bottleneck2posterior.py - a Python script to calculate matrix of phoneme class posterior probabilities from SBN features extracted with audio2bottleneck.py. The choice of the network to extract posteriors has to be consistent with the network used for SBN feature extraction.

utils.py, nn_def.py, gmm.py - supplementary scripts used by two main ones (audio2bottleneck.py and bottleneck2posterior.py)

README - text file with detailed description of software structure and usage

nn_weights - directory containing six `.npz` files with neural network weights. There are three files with network weights to extract SBN features from audio and three files with the weights transforming SBNs into posteriors. It is worth noticing that by merging corresponding weight files, one can restore the originally trained neural networks. In total, three networks were trained for BUT BN feature extractor:

- FisherEnglish_FBANK_HL500_SBN80_PhnStates120.npz + FisherEnglish_SBN80_PhnStates120.npz
- FisherEnglish_FBANK_HL500_SBN80_triphones2423.npz + FisherEnglish_SBN80_triphones2423.npz
- Babel-ML17_FBANK_HL1500_SBN80_PhnStates3096.npz + Babel-ML17_SBN80_PhnStates3096.npz

In the rest of the paper we will refer to them as to *FisherMono*, *FisherTri* and *BabelMulti*, respectively.

Also, the directory contains three files with lists of class labels the networks were trained to predict.

nn_weights/FisherEnglish_SBN80_PhnStates120.dir - directory containing set of scripts and models for generating phoneme lattices using HTK tool HVite. It is possible to create phoneme posterior files, phoneme strings and phoneme lattices - which can replace our previously released phoneme recognizer. The main script for this is phnrec.sh. The phoneme lattices or phoneme soft counts might be used for phonotactic language recognition, topic recognition or other tasks, usually together with some classifier, such as SVM/LM/Decision Trees/NN or other.

example - directory with a simple example of how to extract and save SBN features and how to extract phoneme posterior probabilities.

3. Details of released networks

3.1. Feature extraction

For all of the networks distributed with BUT/Phonexia BN feature extractor, we used the same input features and feature preprocessing.

Acoustic features presented to the network are 24 log Mel-scale filter bank outputs extracted from 25 ms-long frames every 10 ms. For every utterance, global mean normalization on speech frames is applied. We add a context of 10 frames (± 5) to each log filter bank feature vector. Then, along the time trajectory of each feature coefficient, we apply a Hamming window followed by projection into the first 6 DCT bases (0th to 5th). This results in $6 \times 24 = 144$-dimensional input vector to the NN.

In our previous works [9, 17], we used to augment the outputs of the log filter bank with fundamental frequency features. In this work, we decided not to include these in order to keep the BN extractor package self-contained and independent of other third-party toolkits, which the users of BN extractor might not have available. Experiments showed that there was no significant degradation in language recognition performance when fundamental frequency features were not used.

3.2. Training data and targets

The BN extractor package provides three neural networks. Two of them, *FisherMono* and *FisherTri* were trained on Fisher English, Parts 1 and 2 dataset. This dataset contains approximately 2000 hours of clean telephone speech in English. The two networks differ in the targets they were trained to predict. *FisherMono* is trained to assign one of 120 phoneme state class labels (3 states per phoneme) to each frame. *FisherTri* assigns triphone state labels to the frames, 2423 targets in total. These states correspond to the original triphone state tying obtained during GMM-HMM training.

The third network, *BabelMulti*, was trained on 17 languages from the IARPA Babel program[3]. These are also mainly telephone conversational speech data, although small amount of far field and scripted recordings is present. The target output of this network is a vector of stacked phoneme states for all 17 languages. In total, the output consists of 3096 units. The output layer utilizes block softmax activation function [18], meaning that it is divided into 17 subsets corresponding to the phoneme sets of the individual languages. During training, only the part of the output layer corresponding to the language of the given training example is updated.

3.3. SBN architecture

All released networks extract stacked bottleneck (SBN) features. This term refers to a topology where each of the provided networks is essentially a cascade of two bottleneck networks (see Fig. 1). The first of them is a standard bottleneck network taking as an input features described in Sec. 3.1. This network has four hidden layers, third of which is 80-dimensional bottleneck layer. In other words, the configuration of this network is $144 \times D_{HL} \times D_{HL} \times 80 \times D_{HL} \times D_{out}$, where D_{HL} is the size of hidden layer, and D_{out} is the number of target classes. All layers utilize logistic sigmoid activation function. The exceptions are linear bottleneck layer and softmax (block-softmax) output. The second network has exactly the same architecture as the first one except that it has different input dimensionality. It takes as an input for the frame t BN features extracted from the first network sampled at times $t-10$, $t-5$, t, $t+5$, $t+10$, which results in 400-dimensional vector. By stacking bottlenecks from the first network, the second effectively sees 31 frames in the original feature space at a time. The activations of the bottleneck layer of this network are used as final features. The architecture of all three released networks is identical except for the size of hidden layers. In both networks trained on Fisher English, hidden layers have 500 neurons, while each hidden layer of the multilingual network has 1500 neurons.

Although SBNs were originally introduced for the ASR task [19], they were successfully adapted for language recognition. Previously, we have shown that the advantage of stacked architecture can be compensated by increasing input context for the simpler traditional bottleneck network [9]. However, we de-

[3] Collected by Appen, `http://www.appenbutlerhill.com`

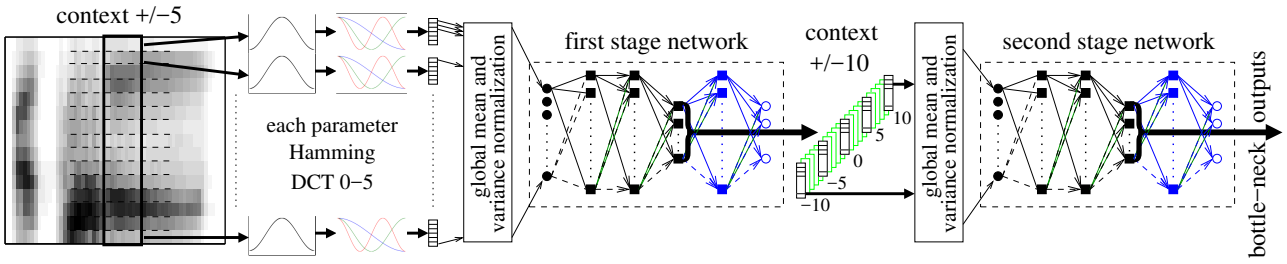

Figure 1: Block diagram of stacked bottleneck (SBN) feature extraction. The blue parts of neural networks are used only during training. The green frames in context gathering between the two stages are skipped. Only frames with shift -10, -5, 0, 5, 10 form the input to the second stage NN.

cided to provide the full stacked BN network as it can be used also for extraction of standard BN features (by propagating input data only to the bottleneck layer of the first network). That allows to compare which architecture (traditional or stacked) suits better for a desired application. In Sec. 4.2.3, we provide results of language recognition systems based on BN features extracted from the networks distributed with the software.

To summarize, training schemes and architectures of all three networks are almost identical, except for a few differences highlighted in Tab. 1.

4. Experiments

4.1. Experimental setup

We report results of the released features on two most recent NIST Language Recognition Evaluation tasks, LRE15 and LRE17 [20, 21]. We trained similar systems for both conditions, the main difference between the two being the data the systems were trained on.

In both cases, we trained a diagonal-covariance UBM with 2048 components. The total i-vector extractor was trained in 10 iterations and the dimensionality of i-vectors was set to 600. A simple Gaussian Linear Classifier (GLC) [22, 23] was used on top of the i-vectors in order to obtain the vector of class-conditional log-likelihoods for each segment. Model of each language is represented by a Gaussian distribution with mean estimated over i-vectors of given language and covariance matrix that is estimated over all training data and shared across all models. We trained a multi-class logistic regression to obtain a scaling factor and an offset vector to calibrate the scores.

In case of LRE15, we used dataset provided by NIST for the LRE15 fixed condition. The dataset consists of recordings from 20 different languages, clustered into six groups according to language similarities [20]. We split training data into two disjoint parts: training and development (dev) datasets [16]. These datasets were created by randomly selecting 60% for the training part and 40% for the dev set. The segments belonging to the development set were further split into short cuts of different durations that contain from 3 to 30 seconds of speech. Moreover, a balanced training subset (up to 15 h per per language) was randomly selected and used for UBM training in order to partially compensate for big differences in the amounts of available training data per language. However, no data augmentation was performed for the classes with less than 15 h of speech. Finally, we used the full training set to train i-vector extractor and the GLC. The dev set was used to obtain calibration parameters. We evaluated this system on the full LRE 2015 evaluation dataset.

For LRE17 experiments, we have utilized the data supplied by NIST (training and dev) for fixed condition. This dataset contains recordings from 14 languages and, as in the case of LRE 2015, they are split into five clusters of similar languages. We downsampled all data to 8 kHz. Similarly to LRE15, we created a balanced subset of training data having up to 15 hours of speech per language. This set was used to train UBM; i-vector extractor and GLC were trained using all training data. In LRE17 dev, we split segments that contain more than 40 seconds of speech into multiple short cuts ranging from 2.5 to 40 seconds. We also kept the original long segments that we cut. The resulting dev set was used for calibration.

4.2. Results

4.2.1. Performance of SBNs on language recognition task

We evaluated performance of three types of SBN features on LRE15 and LRE17 tasks. We report the results in terms of official metrics of evaluations in 2015 and 2017, average C_{avg} and $C_{\text{avg}}^{\text{act}}$, as defined in [20, 21]. Tab. 2 presents the results achieved. The first part of Tab. 2 describes results on 2015. They are as expected: the worst results for *FisherMono*, then *FisherTri* and the best ones for Multilingual bottleneck features *BabelMulti*. The second part of Tab. 2 shows results on 2017 data. They are similar except for Multilingual BN features *BabelMulti* being a bit worse than *FisherTri*.

4.2.2. Effect of VAD

Our bottleneck feature extractor allows the users to utilize their own VAD labels. In case the user supplies the BN extraction script with a label file, it will be used to compute feature mean normalization, also with slight modification of the script (uncommenting a single line of code) the same VAD will be applied to resulting bottleneck features. When the user does not provide any label file, energy-based VAD will be computed: it is used for feature normalization and also could be used to eliminate non-speech frames from BN feature matrix.

We performed a set of experiments on NIST LRE15 to see the effect of VAD. In all experiments, SBN features were extracted using exactly the same neural network and processed with the same LID system, only VAD was varying. Here, the network with 30-dimensional bottleneck layer was trained on 250 hours of Fisher English with 2423 triphone targets. These bottlenecks were used as an input to LID system as described in Sec. 4.1. Here, we only look at the performance on LRE15 task. Tab. 3 summarizes the results. The first line of the table

Table 1: Summary of differences in architecture and training of three SBN networks released with BN extractor.

Network title	Training data	D_{HL}	D_{out}	Output activation
FisherMono	Fisher English 1+2	500	120 phone states	Softmax
FisherTri	Fisher English 1+2	500	2423 triphone states	Softmax
BabelMulti	17 languages from IARPA Babel	1500	3096 phone states	Block softmax

Table 2: Results of provided SBN features on NIST LRE 2015 and 2017 tasks. The performance metrics are the official LRE metrics C_{avg} and $C_{\text{avg}}^{\text{act}}$, reported in [%].

SBN	LRE15		LRE17	
	C_{avg}	$C_{\text{avg}}^{\text{act}}$	C_{avg}	$C_{\text{avg}}^{\text{act}}$
FisherMono	16.20	17.95	23.08	23.33
FisherTri	15.00	17.00	21.10	21.19
BabelMulti	13.82	15.98	21.87	21.99

Table 3: Performance of 30-dimensional SBN features w.r.t. VAD method. C_{avg}, reported in [%], on NIST LRE 2015.

VAD	C_{avg}
No VAD	32.27
Energy-based VAD	18.52
NN-based VAD	18.22

shows the performance of LID system, when no VAD was applied at all. The second line shows the performance of the same system, when we used energy-based VAD supplied with BN feature extractor. The last line corresponds to the experiment where we used our own NN-based VAD. As the results indicate, not using any VAD could significantly degrade the results, while basic energy-based one can already provide reasonably good performance. Using good VAD labels can help to improve performance even further, but the difference is less significant.

4.2.3. Performance of BN features

As mentioned before, provided networks could be used not only for stacked bottleneck feature extraction, but to extract standard BN features as well. One can extract BN features alone, it would allow to slightly speed up the generation of features. Our tests show that extraction of SBNs takes about 10-15% longer than standard BNs. Also, just because bottlenecks are anyway computed in order to get SBNs, one can save them and use as an alternative or supplementary set of features. In this case, one gets them practically for free, the only additional time cost is due to saving them.

Here, we show the results achieved with standard BN features on LRE17 task: Tab. 4 shows the performance of BN features extracted from three networks distributed with BN feature extractor. The results indicate that BNs perform significantly worse than SBNs extracted from the same network (see Tab. 2). As we have shown before [9], BNs could achieve comparable results to SBNs when the input context is large enough. In our case however, the first stage network "sees" rather short temporal context, only 10 frames around the current one, that explains poor performance of these features on language ID task. However, one can still find these features useful for some other application.

Table 4: Performance of BN features on NIST LRE 2017 task. The performance metrics are the official LRE metrics C_{avg} and $C_{\text{avg}}^{\text{act}}$, reported in [%].

BN	C_{avg}	$C_{\text{avg}}^{\text{act}}$
FisherMono	27.65	27.77
FisherTri	27.03	27.17
BabelMulti	28.63	28.77

5. Download

The package homepage is available at
`http://speech.fit.vutbr.cz/software/but-phonexia-bottleneck-feature-extractor`
It contains a download link and detailed installation instructions, as well as a list of system and software requirements. In short, the extractor requires Python 2 with a working numpy package.

6. Conclusion

In the last 10 years, bottleneck features have brought significant advantage to BUT and Phonexia, both in the R&D work and in production systems. This paper complements the public release of our BN feature extractor, which was conducted with the aim to level the field for newcomers in language identification, and to generate new interesting research results. Note that while the intended use of the BN extractor is primarily in language ID, it can be equally well used in speaker recognition, ASR, and maybe other applications such as unsupervised approaches, querying-by-example, and others. We will be happy to learn about your results!

7. References

[1] Hynek Hermansky, Daniel PW Ellis, and Sangita Sharma, "Tandem connectionist feature extraction for conventional hmm systems.," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2000)*, 2000.

[2] Petr Schwarz, Pavel Matějka, and Jan Černocký, "Hierarchical structures of neural networks for phoneme recognition," in *Proceedings of ICASSP 2006*, 2006, pp. 325–328.

[3] Pavel Matějka, Lukáš Burget, Petr Schwarz, and Jan Černocký, "Brno University of Technology system for NIST 2005 language recognition evaluation," in *Proceed-*

ings of Odyssey 2006: The Speaker and Language Recognition Workshop, 2006.

[4] Pavel Matějka, Lukáš Burget, Ondřej Glembek, Petr Schwarz, Valiantsina Hubeika, Michal Fapšo, Tomáš Mikolov, Oldřich Plchot, and Jan Černocký, "BUT language recognition system for NIST 2007 evaluations," in *Proc. Interspeech 2008*. 2008, International Speech Communication Association.

[5] František Grézl, Martin Karafiát, Stanislav Kontár, and Jan Černocky, "Probabilistic and bottle-neck features for LVCSR of meetings," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, 2007, pp. 757–760.

[6] Martin Karafiát, Murali K. Baskar, Pavel Matějka, František Grézl, Lukáš Burget, and Jan Černocky, "BUT Babel system: Multilingual BLSTM acoustic model with i-vector based adaptation," in *Proceedings of Interspeech 2017*, 2017, pp. 719–723.

[7] Bing Jiang, Yan Song, Si Wei, Jun-Hua Liu, Ian Vince McLouglin, and Li-Rong Dai, "Deep bottleneck features for spoken language identification," *PLoS ONE*, vol. 9, 7 2014.

[8] Pavel Matějka et al., "Neural network bottleneck features for language identification," in *IEEE Odyssey: The Speaker and Language Recognition Workshop*, Joensu, Finland, 2014.

[9] Radek Fer, Pavel Matejka, Frantisek Grezl, Oldrich Plchot, Karel Vesely, and Jan Cernocky, "Multilingually trained bottleneck features in spoken language recognition," *Computer Speech and Language*, vol. 46, no. Supplement C, pp. 252 – 267, 2017.

[10] Plchot et al., "Analysis of BUT-PT submission for NIST LRE 2017," in *IEEE Odyssey: The Speaker and Language Recognition Workshop*, 2018.

[11] Pavel Matějka, Ondřej Glembek, Ondřej Novotný, Oldřich Plchot, František Grézl, Lukáš Burget, and Jan Černocký, "Analysis of DNN approaches to speaker identification," in *Proceedings of the 41th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2016), 2016*, 2016.

[12] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *ICASSP*, 2014.

[13] Y. Lei, L. Ferrer, M. McLaren, and N. Scheffer, "Comparative study on the use of senone-based deep neural networks for speaker recognition," *Submitted to IEEE Trans. ASLP*, 2014.

[14] Najim Dehak Fred Richardson, Douglas A. Reynolds, "A unified deep neural network for speaker and language recognition," in *Interspeech*, 2015.

[15] Mitchell McLaren, Martin Graciarena, and Yun Lei, "Advances in deep neural network approaches to speaker recognition," in *ICASSP*, 2015.

[16] Oldřich Plchot, Pavel Matějka, Radek Fér, Ondřej Glembek, Ondřej Novotný, Jan Pešán, Karel Veselý, Lucas Ondel, Martin Karafiát, František Grézl, Santosh Kesiraju, Lukáš Burget, Niko Brummer, Preez du Albert Swart, Sandro Cumani, Harish Sri Mallidi, and Ruizhi Li, "BAT System description for nist lre 2015," in *Proceedings of Odyssey 2016, The Speaker and Language Recognition Workshop*. 2016, vol. 2016, pp. 166–173, International Speech Communication Association.

[17] Martin Karafiát, František Grézl, Karel Veselý, Mirko Hannemann, Igor Szőke, and Jan Černocký, "BUT 2014 Babel system: Analysis of adaptation in NN based systems," in *Interspeech 2014*, 2014, pp. 3002–3006.

[18] Karel Veselý, Martin Karafiát, František Grézl, Miloš Janda, and Ekaterina Egorova, "The language-independent bottleneck features," in *Proceedings of IEEE 2012 Workshop on Spoken Language Technology*, 2012, pp. 336–341.

[19] František Grézl, Martin Karafiát, and Lukáš Burget, "Investigation into bottleneck features for meeting speech recognition," in *Interspeech 2009*, Brighton, England, 2009.

[20] "The 2015 NIST Language Recognition Evaluation Plan (LRE15)," http://www.nist.gov/itl/iad/mig/upload/LRE15_EvalPlan_v23.pdf.

[21] "The 2017 NIST Language Recognition Evaluation Plan (LRE17)," https://www.nist.gov/sites/default/files/documents/2017/09/29/lre17_eval_plan-2017-09-29_v1.pdf.

[22] David González Martínez, Oldřich Plchot, Lukáš Burget, Ondrej Glembek, and Pavel Matějka, "Language recognition in ivectors space," in *Proceedings of Interspeech 2011*, 2011, pp. 861–864.

[23] Sandro Cumani, Oldřich Plchot, and Radek Fér, "Exploiting i-vector posterior covariances for short-duration language recognition," in *Proceedings of Interspeech 2015*. 2015, International Speech Communication Association.