INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Phonetic Context Embeddings for DNN-HMM Phone Recognition

Leonardo Badino

CTNSC, Istituto Italiano di Tecnologia
leonardo.badino@iit.it

Abstract

This paper proposes an approach, named phonetic context embedding, to model phonetic context effects for deep neural network - hidden Markov model (DNN-HMM) phone recognition. Phonetic context embeddings can be regarded as continuous and distributed vector representations of context-dependent phonetic units (e.g., triphones). In this work they are computed using neural networks. First, all phone labels are mapped into vectors of binary distinctive features (DFs, e.g., nasal/not-nasal). Then for each speech frame the corresponding DF vector is concatenated with DF vectors of previous and next frames and fed into a neural network that is trained to estimate the acoustic coefficients (e.g., MFCCs) of that frame. The values of the first hidden layer represent the embedding of the input DF vectors. Finally, the resulting embeddings are used as secondary task targets in a multi-task learning (MTL) setting when training the DNN that computes phone state posteriors. The approach allows to easily encode a much larger context than alternative MTL-based approaches. Results on TIMIT with a fully connected DNN shows phone error rate (PER) reductions from 22.4% to 21.0% and from 21.3% to 19.8% on the test core and the validation set respectively and lower PER than an alternative strong MTL approach.

Index Terms: embeddings, multi-task learning, speech recognition, acoustic modeling

1. Introduction

This paper addresses the problem of phonetic context modeling for hybrid DNN-HMM acoustic modeling [1, 2, 3, 4] where the use of senones, i.e., clustered context-dependent (CD) phone state targets, is the standard approach to model the effects of coarticulation [2].

While senones have been shown to work well in practice they may have some potential disadvantages. These include the senone's intrinsic clustering problem [5], i.e., same-senone CD-phone states (e.g., triphone states) are assigned the same probability even when their frequency can be very different, and the "equality problem" where same-monophone senones and different-mononophone senones are equally discriminated by the DNN [6, 7].

Recently, multi-task learning (MTL) approaches to context modeling have been proposed as alternative or complementary approaches to senones [8, 6, 7]. MTL [9] is a training strategy that aims at improving the generalization performance of a classifier (or regressor) on a task by forcing it to simultaneously learn related secondary tasks. The rationale is that domain specific information of related tasks acts as an inductive bias for primary task learning.

In MTL-based approaches to phonetic context modeling MTL can be either used as an alternative to senones or combined with them. In [7] the primary senone classification task is jointly learned with secondary tasks that can be either prediction of (i) (p_l, q_n), (p_r, q_n) tuples where p_l and p_r are the left and right context phones respectively and q_n is the current phone state; or (ii) of (a_l, q_n), (a_r, q_n) tuples where p_l and p_r are left and right context articulatory/distinctive feature values (e.g., a_r = open vowel).

In [8] a primary monophone state classification task is paired with two secondary tasks: classification of p_l and of p_r. Apart from the general motivation of MTL, learning to classify context phones while learning to classify target mononophone states, may force the DNN to learn context-dependent features (in its hidden layers) given a monophone class. Such context-dependent feature space partition can be useful during recognition even if context information is not provided at test time.

In this paper I propose phonetic context embedding vectors as secondary task target of a DNN that classifies monophone states. Here phonetic context embeddings are real-valued distributed vector representations of a target monophone and its neighboring phones. To some extent they could be regarded as continuous vector representations of context-dependent phonetic units (e.g., triphones). The concept of embedding, defined as a continuous vector representation of discrete entities, has been successfully applied to words, in natural language processing (e.g., [10]), and in language modeling [11, 12, 13] and acoustic modeling/lattice re-scoring [14] for ASR.

Phonetic-context embeddings are extracted at each speech frame from the first hidden layer of a neural network (NN) that takes as input the vector of distinctive features (DFs, also referred to as articulatory features) of the target monophone and the DF vectors associated to the left and right context frames to estimate the acoustic coefficients of the frame of the target monophone, i.e., the central frame (see Figure 1 left). The expectation is that embeddings group together patterns of DFs and of DF vectors that sound alike.

The working hypothesis is that phonetic context embeddings have at least two potential advantages w.r.t. discrete context representations such as, e.g., context phones [8] or [7]'s tuples. First, they can easily encode very long contexts into fixed length real-valued vectors. Second, they are secondary task targets that by construction implicitly take into account whether context phones affect the acoustic realization of the target phone, while, e.g., in [8], all context phones are equally important independently of how they actually influence the target phone through coarticulation.

On the other hand if the embeddings are poor and encode only little information (e.g., different target monophones are collapsed into almost identical representations because the embedding dimensionality is too low) they might act as a too strong regularization term for primary task learning.

Note that the task performed by the neural network from which the embeddings are extracted is very similar to that of the decision tree that cluster context-dependent classes into senones

http://dx.doi.org/10.21437/Interspeech.2016-1036

[15] or to that of DNNs in statistical parametric synthesis that maps linguistic features into first order statistics of speech parameters [16].

Finally the present work is inspired by our work on "articulatory" DNN-HMM phone recognition [17, 18], which uses measured articulatory information and is based on the assumption that compact models of coarticulation (as opposed to thousands of senones) can be achieved through a more explicit use of speech production information. However, recorded articulatory datasets are very small and the articulatory data provide largely incomplete information about speech production. The phonetic context embeddings proposed here were born as an attempt to substitute measured real-valued articulatory information with real-valued speach production information extracted from linguistic binary articulatory features.

2. Phonetic context embeddings

In the first definition of phonetic context embedding proposed here the embedding is a continuous and distributed vector representation of a sequence of phone labels associated to a sequence of consecutive speech frames. I refer to this embedding as pc-embedding.

In this work the embeddings are learned by neural networks (NNs), not necessarily deep. The figure within the dashed line rectangle of Figure 1 shows how a pc-embedding is computed. Given a window of speech frames centered at frame n, each frame phone label is converted by a look-up table into a vector of binary distinctive features. The DFs are listed in Table 1. The binary vectors are then concatenated and used as input to a neural network that is trained to predict the acoustic coefficients of the central frame.The embedding is always represented by the first hidden layer, independently of the number of (optional) hidden layers used.

Note that one-hot representations of phones could be directly used as input to the NN without mapping phones into DF binary vectors.

In my experiments it turned out that using NNs with rectified linear units (ReLUs, [19, 20, 21]) is a key factor to extract "good" embeddings. Alternative non-linearities, e.g., sigmoids, did not produce any useful embedding. One reason seems to be the sparse representations naturally produced by ReLUs. Sparsity is also the reason why embeddings are always represented by the first hidden layer, which encodes a much sparser representation than the next hidden layers. Even when a penalty was applied on the activations of the topmost layers the first layer always turned out to generate much better embeddings for phone recognition.

2.1. Context-dependent phonetic unit embeddings

Formally, the pc-embedding proposed above is carried out by a function $f(\mathbf{x}, \mathbf{w})$ that takes an input vector $\mathbf{x}$ of dimensionality $d = l \times k$ where l is the number of frames in the context window and k is the overall number of phones (and we can assume a phone to be represented by a one-hot vector with dimensionality k). Its learning parameters are learned by learning the parameters of the composite function $(g \circ f) = g(f(\mathbf{x}, \mathbf{w}), \mathbf{w}')$, where $\mathbf{w}'$ are learning parameters of g, that maps the sequences of phones within the context window onto the acoustic coefficients of the window central frame.

The resulting embeddings can be regarded as embeddings of context-dependent phonetic units (e.g., triphones), although it must be pointed out that they are not exactly embeddings of context-dependent phonetic units. To generate, e.g., triphone embeddings, we only need to consider a different input $\mathbf{x}'$ represented by a one-hot vector of dimensionality $d =$ no. of all logical triphones.

This is a variant to the original phonetic context embedding (i.e., pc-embedding), which I named n-phone embedding. Note that in this case, to compute, e.g., triphone embeddings, $\mathbf{x}'$ will be first mapped on a triple of DF vectors.

One big difference between pc-embeddings and n-phone embeddings is that in the latter case the embedding represents a fixed number of phones, i.e. the current phone and its neighboring phones (e.g., $1 + 2$ for triphone embeddings), while in the first strategy the length of the context window is fixed but the number of phones it includes can vary.

3. Multi-task learning with embeddings

Once embeddings are computed they are used as secondary task targets in a MTL setting when training the DNN that has to classify phone states as primary task (Figure 1).

The overall target cost that the DNN is trained to minimize is

$$E_{tot}(\theta) = E_A(\theta) + \lambda E_B(\theta)$$

where θ is the vector of all learning parameters, λ is a weighting term.

$$E_A(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbf{t}_{nk}^A \ln \mathbf{y}_{nk}^A(\mathbf{x}, \theta)$$

is the usual cross-entropy target cost function used for classification (primary task). N is the overall number of training frames, K is the number of monophone states, $\mathbf{y}_n(\mathbf{x}, \theta)^A$ is the DNN K-dimensional softmax output for the primary task and $\mathbf{t}_n^A$ is the one-hot target vector of the primary task at frame n.

$$E_B(\theta) = \sum_{n=1}^{N} (\mathbf{y}_n^B(\mathbf{x}, \theta) - \mathbf{t}_n^B)^2$$

is the sum-of-squares error cost for the secondary task. $\mathbf{y}_n(\mathbf{x}, \theta)^B$ is the DNN (ReLU) output for the secondary tasks and $\mathbf{t}_n^B$ is the embedding vector at frame n. The additional ReLU output layer for the secondary task is only used during training and removed during evaluation.

4. Experimental setup

Experiments with the DNN-HMM phone recognition systems were conducted on the TIMIT dataset [22] following the standard procedure where all "sa" utterances are removed and keeping the usual splitting for training, validation and testing.

4.1. DNN-HMM phone recognition systems

Recognition was performed on the original set of 61 phones and 3-state HMMs were used to model monophones. The HMM emission probabilities were computed through the DNN phone state posteriors. HMMs were combined with a standard bigram phone language model. After decoding the 61 phones were collapsed in the usual set of 49 phones [23].

As in previous work (e.g., [8, 24]) the acoustic coefficient vectors used as input to the DNN consisted of 40 log mel-filtered spectral coefficients (MFSCs) plus deltas and delta-deltas, and log energy plus its derivatives, all computed every 10ms from 25ms long frames . The DNN had 4 hidden layers

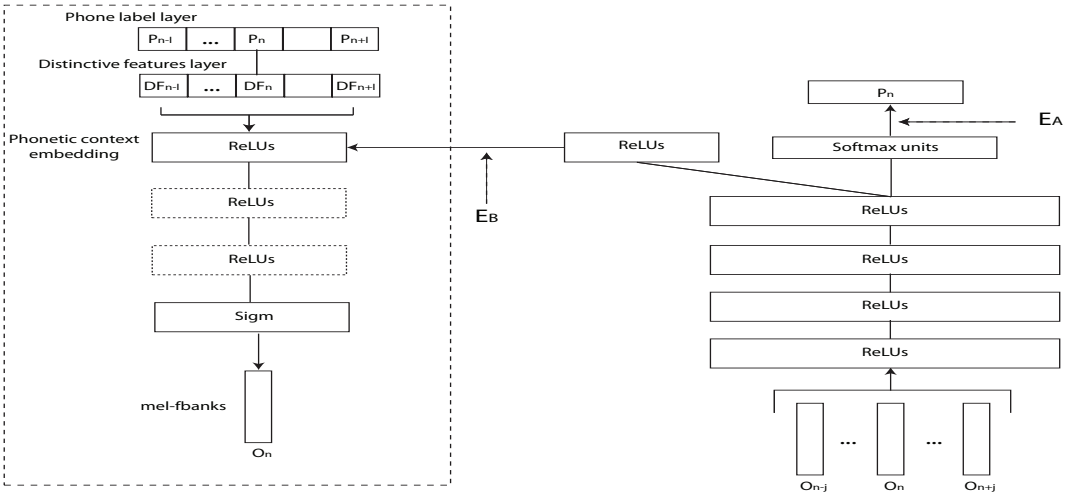

Figure 1: *Phonetic context embedding extraction and multi-task learning.*

with 2000 ReLUs each. The single-task learning (STL) baseline DNN had a single output layer with 61 x 3 = 183 softmax units while the MTL DNNs had an additional output layer with a number of ReLUs that depended on the dimensionality of the embedding vector.

The SLT DNN parameters were randomly initialized and then updated using backpropagation and stocastic gradient descent with initial learning rate = 0.075, momentum = 0.9 and mini-batches of 1000 training examples. At each epoch the learning rate was decreased by multiplying it to a learning rate decay factor = 0.75. Training was when the classification error on the development set did not decrease for 2 consecutive epochs. The same hyper-parameters were used for the MTL DNNs. The hyper-parameters were chosen with the goal of a trade-off between a strong baseline DNN and fast training of MTL DNNs.

The secondary task weight λ ranged from 0 to 2 with 0.2 increments and its optimal value was the one producing the best recognition performance on the validation set. In all training examples where the central frame was silence λ was set to 0 (i.e., the secondary task target was ignored).

It is worth to point out that, contrary to most MTL settings where target costs and output activations for primary and secondary tasks are of the same type, here both target costs (cross-entropy vs. square error) and output activations (softmax vs. ReLU) are different. That makes difficult to find a principled way to scale λ (which, in our setting correspond to scale the learning rate for the secondary task) depending on the secondary task target dimensionality.

4.2. Context embeddings

This sections describes the neural networks that extract phonetic pc-embeddings and n-phone embeddings.

The input to the NNs from which pc-embeddings were extracted (pceNN) consisted of vectors of 37 DFs (shown in Table 1). The number of DF vectors ranged from 11 to 31 (with same-length left and right context). The net target was a vector of 20 MFSCs plus deltas and delta-deltas (computed every 10ms from 25ms long frames) , then normalized to lie in the [0 1] range.

I experimented with net with either 1 or 3 hidden layers, with a number of ReLU nodes of either 300 or 600, and a sigmoid output layer. All examples where the target frame was silence were removed from the training set (as all DFs, excluding the silence DF, do not affect non-speech sounds) while the silence DF was taken into account to build phonetic embeddings. The net was trained as the phone state classifier DNN but with larger learning rate (0.1) and learning decay factor (0.99). The target cost was the sum of squares error.

consonant, voiced, unvoiced, fricative, nasal, stop
approximant, affricate, labial, dental, alveolar, lateral
post-alveolar, palatal, velar, glottal, syllabic, flapping
vowel, diphtong, nasalized, r-merged, close, close-mid
mid, open-mid, open, front, central, back, long, short
close2, close-mid2, mid2, open-mid2, open2
front2, central2, back2, long2, short2, silence

Table 1: *List of all 37 binary distinctive features used in the experiments. Number 2 refers to the 2nd vowel of a diphtong.*

An alternative net (qeNN) was trained to compute n-phone embeddings described in section 2.1. Specifially the net was trained to extract embeddings of quin-phones, so it received as input 5 vectors of distinctive features (one per each phone of the quin-phone). qeNN had the same architecture and learning schedule of a 3-hidden layer pceNN.

5. Results

Figure 2 plots the validation and test core PERs over the λ weight term of 2 different phone recognition systems (continuous lines) where the MTL DNNs were trained using pc-embeddings as secondary task targets. $\lambda = 0$ corresponds to the baseline system (i.e., system with STL DNN). It can be seen that for both systems the optimal λ values, computed on the validation set, are values that produce PER values in the core test very close to the lowest PER in the core test. All of them are significantly lower than the baseline.

For comparison with alternative context modeling MTL-based approaches, Figure 2 shows (in dashed line) the best performing MTL-based system proposed in [8] and described in the introduction. That system uses two secondary tasks, classification of the left and right context phone labels. Although the

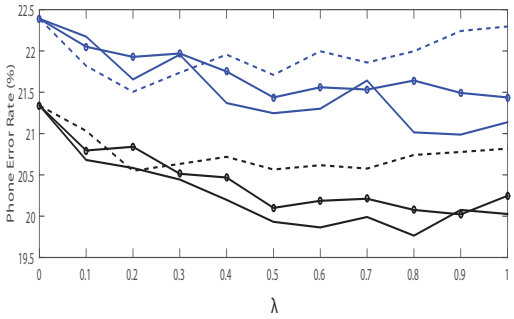

Figure 2: *Core test (blue lines) and development (black lines) PERs vs. λ of 2 pc-embedding based systems (continuous lines) and [8]'s best performing system (dashed line) . For comparison with the dash line system, the λ of the 2 pc-embedding systems was divided by 2. pc-embeddings were extracted from 3-hidden-layer nets with 300 ReLUs per layer. The 2 pc-embedding systems only differ in the number of input DF vectors, 25 vs 13 (circle marker).*

System	Secondary task target	λ	Dev PER(%)	Core Test PER(%)
Baseline SLT	-	-	21.34	22.39
S&D[8]	l-p r-p	0.2	20.55	21.51
n-phone E 3H	5-phone E	1.2	20.59	21.93
pc-E 1H 500u	11w E	1.4	20.52	21.81
pc-E 1H 500u	27w E	1	20.27	21.76
pc-E 3H 300u	13w E	1.8	20.02	21.49
pc-E 3H 300u	25w E	1.6	**19.76**	**21.01**

Table 2: *Core test and development PERs for baseline and MTL systems. E stands for embedding. 1H and 3H indicate the number of hidden layers in the embedding neural net. l-p and r-p stand for left and rigth phone respectively. 11w indicates the number of input distinctive feature vectors used in pc-embedding. u indicates the number of units per hidden layer.*

system significantly outperforms the baseline it does not perform as well as the systems based on pc-embedding.

The better performance of pc-embeddings can be mainly attributed to two factors. First, in these experiments pc-embeddings encode larger context. The approach by [8] could use a larger context but that would imply more secondary tasks and computation as opposed to the pc-embedding based approach. Second, pc-embeddings encode higher level information about the context, i.e., patterns of context elements (and current phone), while in [8] approach each context element, i.e., phone in the context, is considered independently of the other elements.

Note that in [8] the relative PER reduction produced by their approach (over a baseline stronger than the baseline used here) may be due to a learning schedule that is more appropriate for a MTL setting or might be due to some specific learning schedule for the secondary tasks. However, in both cases any MTL-based approach, included the pc-embedding approach proposed here, could show better performance.

Table 2 shows PERs on TIMIT development and test sets for the baseline and different embedding-based systems with optimal λ (computed on the development set). One clear result

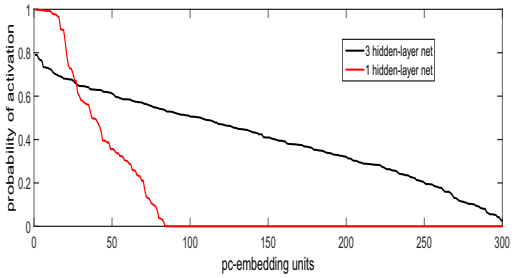

Figure 3: *Probability of activation of pc-embedding ReLUs extracted from a 1-hidden layer net and a 3-hidden layer net. Units are sorted according to their probability of activation. In this case both nets have 300 ReLUs per hidden layer.*

is that the pc-embeddings outperform n-phone embeddings.

Within pc-embedding based systems embeddings of larger phonetic contexts usually produce better results.

Another interesting result is that pc-embeddings extracted with 3-hidden layer NNs outperform those extracted from 1-hidden layer NNs (which is more evident when plotting PERs vs. λ, not shown). Despite in both cases the embedding is represented by the first hidden layer, adding more hidden layers produces better embeddings. One possible explanation is that the additional layers might act as a sort of regularization on the embedding since complex feature extraction is left to the deeper layers. Figure 3 may provide a better understanding of the difference between the two cases. It shows he probability of activation [20] of each pc-embedding ReLU unit (i.e., the average number of times a unit value is greater than 0) in the pc-embedding vector extracted from a 1-hidden layer net (red) and a 3-hidden layer net. The figure explains why in the phone recognition experiments the best 1-hidden layers "required" more hidden units than 3-hidden layer nets. Note that in the 1-hidden layer pc-embedding case all units with 0 probability of activations were removed from the final pc-embedding vector, i.e., the vector used in MTL DNN training. In both cases the average sparsity ratios of the embedding vector, i.e., the average number of 0-values, were both around 50%.

6. Conclusions

This paper addresses the problem of context modeling for DNN-based acoustic modeling. It proposes phonetic context embeddings, i.e., real-valued distributed vector representations of a phone and its interaction with the context. Phonetic embeddings are first extracted by neural networks that map phone labels into acoustic coefficients and are then used as secondary task targets when training the DNN of a DNN-HMM phone recognition system. Results on TIMIT show the utility of the context embeddings with PER reduction on the test core from 22.4% to 21.0% and lower PERs than an alternative approach. Although in this paper phonetic context embeddings were tested on fully connected DNNs the embedding-based approach can be applied to any DNN architecture, included stronger DNN architectures, e.g., max-out DNNs [24].

7. Acknowledgements

The author acknowledges the support of the European Union's Horizon2020 project ECOMODE (grant agreement No 644096).

8. References

[1] A. Mohamed, G. E. Dahl, and G. E. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 20, no. 1, pp. 14–22, 2012.

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

[3] D. Yu and L. Deng, *Automatic Speech Recognition - A Deep Learning Approach*. New York: Springer, 2015.

[4] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, 2012.

[5] G. Wang and K. C. Sim, "Regression-based context-dependent modeling of deep neural networks for speech recognition," *IEEE/ACM Transactions on Acoustics, Speech and Signal Processing*, vol. 22, no. 11, pp. 1660–1669, 2014.

[6] P. Bell and S. Renals, "Regularization of context-dependent deep neural networks with context-independent multi-task training," in *Proc. of ICASSP*, Brisbane, Australia, 2015.

[7] ——, "Complementary tasks for context-dependent deep neural network acoustic models," in *Proc. of Interspeech*, Dresden, Germany, 2015.

[8] M. Selzer and J. Droppo, "Multi-task learning in deep neural networks for improved phoneme recognition," in *Proc. of ICASSP*, Vancouver, Canada, 2013.

[9] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, pp. 41–75, 1997.

[10] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, Lake Tahoe, USA, 2013.

[11] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *The Journal of Machine Learning Research*, vol. 3, p. 11371155, 2003.

[12] H. Schwenk, "Continuous space language models," *Computer Speech and Language*, vol. 21, pp. 492–518, 2007.

[13] T. Mikolov, "Statistical language models based on neural networks," in *PhD Thesis*, Brno University of Technology, 2012.

[14] S. Bengio and G. Heigold, "Word embeddings for speech recognition," in *Proc. of Interspeech*, Singapore, 2014.

[15] S. J. Young and P. C. Woodland, "State clustering in hidden markov model-based continuous speech recognition," *Computer Speech Language*, vol. 8, no. 4, pp. 369–383, 1994.

[16] H. Heiga Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Proc. of ICASSP*, Vancouver, Canada, 2013.

[17] L. Badino, C. Canevari, L. Fadiga, and G. Metta, "Deep-level acoustic-to-articulatory mapping for dbn-hmm based phone recognition," in *Proc. of IEEE SLT*, Miami, Florida, 2012.

[18] ——, "Integrating articulatory data in deep neural network-based acoustic modeling," *Computer Speech and Language*, vol. 36, pp. 173–195, 2016.

[19] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, Haifa, Israel, 2010.

[20] A. Maas, A. Hannun, and A. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, Atlanta, USA, 2013.

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

[22] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren, "The darpa timit acoustic-phonetic continuous speech corpus cdrom," 1993.

[23] K. Lee and H. Hon, "Speaker-independent phone recognition using hidden markov models," *IEEE Transactions on Audio, Speech Language Processin*, vol. 37, no. 11, pp. 1641–1648, 1989.

[24] L. Tóth, "Convolutional deep maxout networks for phone recognition," in *Proc.of Interspeech*, Singapore, 2014.