VERY DEEP MULTILINGUAL CONVOLUTIONAL NEURAL NETWORKS FOR LVCSR

Tom Sercu[1,2] *Christian Puhrsch*[1] *Brian Kingsbury*[2] *Yann LeCun*[1]

[1] Center for Data Science, Courant Institute of Mathematical Sciences, New York University
[2] IBM T. J. Watson Research Center, Yorktown Heights, NY, 10598, U.S.A.
[2]{tsercu,bedk}@us.ibm.com, [1]cpuhrsch@nyu.edu,yann@cs.nyu.edu

ABSTRACT

Convolutional neural networks (CNNs) are a standard component of many current state-of-the-art Large Vocabulary Continuous Speech Recognition (LVCSR) systems. However, CNNs in LVCSR have not kept pace with recent advances in other domains where deeper neural networks provide superior performance. In this paper we propose a number of architectural advances in CNNs for LVCSR. First, we introduce a very deep convolutional network architecture with up to 14 weight layers. There are multiple convolutional layers before each pooling layer, with small 3×3 kernels, inspired by the VGG Imagenet 2014 architecture. Then, we introduce multilingual CNNs with multiple untied layers. Finally, we introduce multi-scale input features aimed at exploiting more context at negligible computational cost. We evaluate the improvements first on a Babel task for low resource speech recognition, obtaining an absolute 5.77% WER improvement over the baseline PLP DNN by training our CNN on the combined data of six different languages. We then evaluate the very deep CNNs on the Hub5'00 benchmark (using the 262 hours of SWB-1 training data) achieving a word error rate of 11.8% after cross-entropy training, a 1.4% WER improvement (10.6% relative) over the best published CNN result so far.

Index Terms— Convolutional Networks, Multilingual, Acoustic Modeling, Speech Recognition, Neural Networks

1. INTRODUCTION

Convolutional Neural Networks (CNNs) [1] have recently pushed the state of the art on large-scale tasks in many domains dealing with natural data, most notably in computer vision tasks like image classification [2, 3], object detection [4, 5], object localization [6] and segmentation [7].

Early applications of neural nets to speech recognition used Time-Delay Neural Nets [8] which can be seen as simple forms of CNNs without pooling or subsampling. Full-fledged CNNs with pooling and subsampling were soon applied to speech recognition and combined with dynamic time warping [9, 10]. While the globally-trained combination of neural nets and HMMs for speech and handwriting goes back to the 1990s [11, 1], only due to recent developments [12, 13, 14] HMM/DNN hybrid modeling became dominant in ASR. In the context of these hybrid models, the use of CNNs is relatively recent [15]. CNNs were shown to achieve state of the art performance on the benchmark datasets Broadcast News and Switchboard 300 [16]. However, in contrast to the trend in other domains where deeper architectures are often shown to gain performance, the classical CNN architecture in LVCSR [16, 17, 18] has only two convolutional layers.

Our network architecture (Section 2.1) is strongly inspired by the work of Simonyan et al. [3] (subsequently referred to as "VGG Net") which obtained second place in the classification section of the Imagenet 2014 competition. The central idea of VGG Net is to replace large convolutional kernels by a stack of 3×3 kernels with ReLU nonlinearities without pooling between these layers; The authors argue the advantage of this is twofold: (1) additional nonlinearity hence more expressive power, and (2) a reduced number of parameters. Using these principles, very deep networks are trained with up to 19 weight layers (of which 16 are convolutional and 3 fully connected). By contrast, the classical CNNs deployed in LVCSR have typically only two convolutional layers, use large (9×9) kernels in the first layer, and use sigmoid activation functions. The first goal of this work is to adapt the VGG Net architecture to LVCSR. Most closely related to this is [19], which also uses VGG Net-inspired CNNs for LVCSR [1]. In contrast to our work, the architectures investigated in [19] are quite different and the paper only provides results from training on a non-standard Switchboard-51h dataset, with WER not close to state of the art performance on Hub5'00.

In the context of low-resource language tasks, it can be crucial to leverage training data in languages other than the target language. Therefore we trained multilingual deep CNNs, which we describe in Section 2.2. This is related to multilingual neural networks in hybrid NN-HMM systems [20] which have been extended to multilingual bottleneck architectures for tandem models [21, 22] and have proven valuable for spoken term detection [23]. To our knowledge, no work has been published that extends the multilingual setup to CNNs.

The multi-scale features described in Section 2.1 aim at exploiting more context at very low computational cost. They are inspired by the recent success of combining information at multiple scales in tasks like traffic sign recognition [24], semantic segmentation [7, 25] and depth map prediction [26]. In LVCSR the multi-scale idea has been explored in tandem systems [27] and the CLDNN architecture [28].

As training becomes more challenging with increasing depth, we used two recently proposed optimization algorithms, Adadelta [29] and Adam [30] (Section 2.4). Both algorithms are first order gradient-based optimization methods, which keep track of an estimate of the first and second order moment of the gradient to tune the step size of each weight separately.

The rest of the paper is organized as follows. In Section 2 we introduce the novel aspects of our work: very deep CNN architectures in 2.1, multilingual CNN training in 2.2, multi-scale features in 2.1, and training details in 2.4. We then show experimental results on Babel in 3.1 and on Switchboard in 3.2.

[1] This work was pursued independently of ours, and was published about two weeks before submission of this paper.

# Fmaps	Classic [16, 17, 18]	VB(X)	VC(X)	VD(X)	WD(X)
64		conv(3,64) conv(64,64) pool 1x3	conv(3,64) conv(64,64) pool 1x2	conv(3,64) conv(64,64) pool 1x2	conv(3,64) conv(64,64) pool 1x2
128		conv(64, 128) conv(128, 128) pool 2x2	conv(64, 128) conv(128, 128) pool 2x2	conv(64, 128) conv(128, 128) pool 1x2	conv(64, 128) conv(128, 128) pool 1x2
256			conv(128, 256) conv(256, 256) pool 1x2	conv(128, 256) conv(256, 256) pool 2x2	conv(128, 256) conv(256, 256) conv(256, 256) pool 2x2
512	conv9x9(3,512) pool 1x3 conv3x4(512,512)			conv(256, 512) conv(512, 512) pool 2x2	conv(256, 512) conv(512, 512) conv(512, 512) pool 2x2
	FC 2048 FC 2048 (FC 2048) FC output size				
	Softmax				

Table 1. The configurations of our very deep CNNs for LVCSR. In all but the classic convnet, convolutional layers have 3×3 kernels, thus kernel size is omitted. The depth of the networks increases from left to right. The deepest configuration, WDX, has 10 convolutional and 4 fully connected layers. The leftmost column indicates the number of output feature maps in each layer. The optional X means there are four fully connected layers instead of three (output layer included).

2. ARCHITECTURAL AND TRAINING NOVELTIES

2.1. Very Deep Convolutional Networks

The very deep convolutional networks we describe here are adaptations of the VGG Net architecture [3] to the LVCSR domain, where until now networks with two convolutional layers dominated [16, 17, 18]. Table 1 shows the configurations of the deep CNNs. The deepest configuration, WDX, has 14 weight layers: 10 convolutional and 4 fully connected. As in [3], we omit the Rectified Linear Unit (ReLU) layers following every convolutional and fully connected layer. The convolutional layers are written as conv({input feature maps}–{output feature maps}) where each kernel is understood to be size 3×3. The pooling layers are written as (time x frequency) with stride equal to the pool size.

For architectures VDX and WDX, we apply zero padding of size 1 at every side before every convolution, while for architecture VC(X) and VB(X) we use the convolutions to reduce the size of the feature maps, hence only in the higher layers of VC(X) padding is applied.

In contrast to [3], we do not reinitialize the deeper models with the shallower models. Each model is trained from scratch with random initialization from a uniform distribution in the range $[-a, a]$ where $a = (\text{kW} \times \text{kH} \times \text{numInputFeatureMaps})^{-\frac{1}{2}}$. This follows the argument of [31] to initialize the weights such that the variance of the activations on each layer does not explode or vanish during the forward pass.

2.2. Multilingual Convolutional Networks

Figure 1 shows a multilingual VBX network, which we used for most of our Babel experiments. It is similar to previous multilingual deep neural networks [20], with the main difference that the shared lower layers of the network are convolutional.

A second difference is that we untie more than only the last layer,

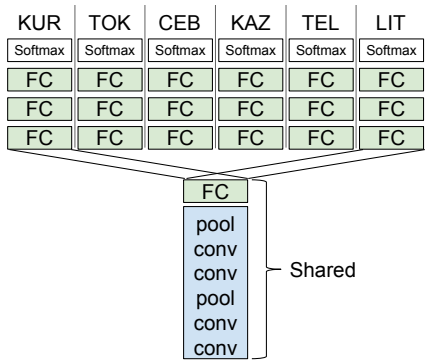

Fig. 1. Multilingual VBX network with the last three layers untied. FC stands for Fully Connected layers.

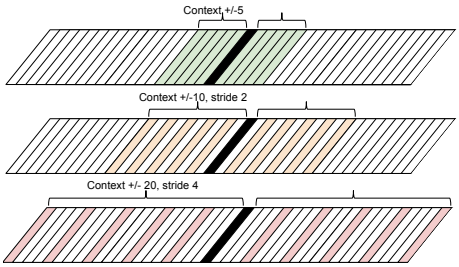

Fig. 2. Multi-scale feature maps with context ±5 and strides {1,2,4} (3S/5). The final size of each feature map along the time dimension is 11. The three 11×40 input feature maps are stacked as input to the CNN, similar to how RGB channels form 3 input feature maps in an image.

meaning that the weights and biases of multiple fully connected layers are different for each language. Since the output dimension of the convolutional stages is typically large when using large context windows, most of the weights are in the first fully connected layer, which acts on the flattened output of the convolutional stages. This is an argument to share this large, first fully connected layer across languages. We experimentally confirmed that for all architectures, untying all fully connected layers except the lowest one gives optimal performance, with strong degradation if the first fully connected layer is also untied. This untying corresponds to a view of the shared layers and the first fully connected layer as a shared multilingual feature extractor, while the fully connected layers higher up form the classifier.

The multilingual CNN is trained in a round-robin fashion: we process a mini-batch for each language before making an update to the weights. In the shared part of the network the gradients of all mini-batches are accumulated between weight updates.

2.3. Multi-scale feature maps

The main goal of constructing multi-scale feature maps is to add more context without increasing the computational cost. Figure 2 illustrates the concept of multi-scale feature maps, where additional input feature maps contain a larger view of the context of the frame by downsampling larger context windows with different strides. Kernels on the first convolutional layer are able to combine information from multiple scales, i.e. different distances from the central frame. Because the only difference for the convnet configuration is the first convolutional layer having more feature maps, the additional computational cost and number of parameters is small.

We found this style of multi-scale training to give small gains. Increasing the context size had a stronger positive impact, though at the expense of increased computational cost.

2.4. Training

We use Adadelta [29] and Adam [30] to do initial training of the deep CNNs. Using Adadelta has two main advantages. Firstly, in our experience the optimization problem converges much faster than with SGD; for the Babel experiments we typically see convergence after about 40 million frames using the 18 hours of Babel training data (after silence removal about 5.8 million frames). Secondly, the optimal working point of Adadelta's hyperparameters ϵ and ρ was stable across architectures, always giving optimal performance. This was crucial in order to explore architectural variations. After initial training with Adadelta, we fine tune using SGD with a small learning rate.

Another aspect of training that improved our results is data balancing (something similar was done in [6]). We construct batches on the fly by sampling target $y = CD_i$ with probability p_i, where p_i is related to the frequency f_i of context dependent state CD_i as $p_i = \frac{f_i^\gamma}{\sum_j f_j^\gamma}$. After sampling y, we sample uniformly across all frames with that target. The exponent γ takes values between balanced training ($\gamma = 0$) and unbalanced training using the natural frequencies ($\gamma = 1$).

In our experiments on Babel it proved optimal to start with $\gamma = 0$ and raise it during training to its final value of $\gamma = 1$. In our experiments on switchboard we varied γ typically from 0.4 to 0.8 and decoded with HMM priors adjusted to match the final p_i distribution.

3. EXPERIMENTAL RESULTS

3.1. Babel

	DNN	Classic	VB	VBX	VC	VCX
KUR	82.7	80.6	79.3	78.3	78	77.7
TOK	62.6	59.4	57.1	56.1	54.3	54.5
CEB	76.3	74.2	72.6	71.6	70.6	70.6
KAZ	77.3	75.2	73.5	72.7	71	71.4
TEL	87.0	85.4	83.7	83.7	82.4	82.7
LIT	71.0	69.5	67.8	67.7	66	66.4
IMPR	0.00	2.10	3.82	4.47	**5.77**	5.60

Table 2. WER on Babel for different model architectures. Left to right is increasing depth. The bottom row shows the absolute WER improvement over the CE PLP DNN baseline. Note that adding a fully connected layer for the 6-layer convolutional VC model (i.e. VCX) degrades performance.

	DNN	1L Clas	6L Clas	1L VC	6L VC
KUR	82.7	82.8	80.6	81.3	78
TOK	62.6	63.3	59.4	59.5	54.3
CEB	76.3	76.7	74.2	73.2	70.6
KAZ	77.3	77.7	75.2	74.4	71
TEL	87.0	86.8	85.4	84.8	82.4
LIT	71.0	72.7	69.5	69.8	66
IMPR	0.00	-0.52	2.10	2.32	**5.77**

Table 3. WER on Babel for monolingual (1L, 3 hours of training data) versus multilingual (6L, 18 hours of training data). When trained on a single language, the classical CNN architecture does slightly worse than the baseline DNN. However, the VC architecture gives an average 2.5 WER improvement even when trained on one language. As expected, for both models training multilingual gives a strong performance boost.

Our first set of experiments on Babel focuses on the multilingual and multi-scale aspects of this work. The IARPA Babel program is aimed at developing robust keyword search technology for low resource languages. Though the word error rates reported here are too high to be useful for simple speech to text applications, useful keyword search (KWS) systems can still be built based on these ASR models.

As training data we use a combination of 6 languages, with 3 hours of training data per language. The languages used for training are languages from the second Option Period of the Babel program, i.e. Kurmanji (KUR), Tok Pisin (TOK), Cebuano (CEB), Kazakh (KAZ), Telugu (TEL), and Lithuanian (LIT). The features used in these experiments are standard log-Mel features, standardized with a global mean and variance shared across the speakers and langauges. Unless explicitly mentioned, we use multi-scale features with context ± 20 in the Babel experiments. We report results after cross-entropy training with adadelta ($\rho = 0.985$, $\epsilon = 1e{-}10$), and γ varying from 0 to 1.

We trained the multilingual deep CNN architecture on 6 Babel languages using alignments from 6 baseline speaker independent HMM/DNN systems using PLP features, with 1000 context dependent states. The context dependent states are specific to each language. Each baseline system is cross-entropy trained on a single language with 3 hours of data. We will report the WER of the CNNs

	DNN	3S/20	1S/20	3S/8	1S/8
KUR	82.7	78.1	78.4	78.4	79.2
TOK	62.6	54.2	54.7	55.8	56.7
CEB	76.3	70.3	70.4	71.6	71.8
KAZ	77.3	71.1	71.8	72.5	72.8
TEL	87.0	82.5	83.1	83.5	83.6
LIT	71.0	66.2	67.3	66.9	67.5
IMPR	0.00	**5.75**	5.20	4.70	4.22

Table 4. WER for VC multi-scale training with different context windows. 3S/20 stands for three scales with a context of ± 20. For 3S we use strides of 1, 2, and 4, while 1S just has stride 1, i.e. regular input features. Multi-scale features provide a modest gain. Using larger context size gives a better improvement, however this comes at the cost of extra computation proportional to the context size in the convolutional layers.

compared to the baseline DNN, and summarize this in the average absolute WER improvement over the baseline DNN, which gives one number to compare different models. The WER improvements over the baseline DNN are fairly consistent across languages.

Tables 2 through 4 show the results outlining the performance gains from the different architectural improvements discussed in Section 2.1, 2.2, and 2.1 respectively. From table 3 note that even in the monolingual case (3 hours of data) the VBX CNN architecture outperforms both the classical CNN and the baseline DNN.

3.2. Switchboard 300

	WER	# params (M)	#M frames
Classic 512 [17]	13.2	41.2	1200
Classic 256 ReLU (A+S)	13.8	58.7	290
VCX (6 conv) (A+S)	13.1	36.9	290
VDX (8 conv) (A+S)	12.3	38.4	170
WDX (10 conv) (A+S)	12.2	41.3	140
VDX (8 conv) (S)	11.9	38.4	340
WDX (10 conv) (S)	**11.8**	41.3	320

Table 5. Results on Hub5'00 SWB after training on the 262-hour SWB-1 dataset. We obtain 14.5% relative improvement over our baseline adaptation of the classical CNN and 10.6% relative improvement over [17]. (A+S) means Adadelta + SGD finetuning. (S) means the model was trained from random initialization using SGD. The last column gives the number of frames til convergence.

We evaluate our deep CNN architecture by training on the 262-hour SWB-1 training data, and report the Word Error Rates on Hub5'00 SWB (table 5). The Switchboard experiments focus on the very deep aspect of our work. Apart from not involving multilingual training, we did not use multi-scale features in the Switchboard experiments, but did use speaker-dependent VTLN and deltas and double deltas as this is shown to help performance for classical CNNs [16].

In the switchboard experiments, using a large context only gave marginal gains which were not worth the computational cost, so we worked with context windows of ± 8. We use a data balancing value of $\gamma = 0.8$, chosen from $[0.4, 0.6, 0.8, 1.0]$.

After training with multiple combinations of Adam, Adadelta and SGD, we settled on two possible strategies for optimization: the first strategy is to use Adadelta or Adam for initial training, followed by SGD finetuning. This way one can typically achieve good performance in minimal time. The second strategy, training from scratch using only SGD, requires more training, however the performance is slightly superior. Classical momentum yielded no gains and sometimes slight degradation over plain SGD. We provide the results and total number of frames until convergence. Note that with the first strategy, we achieve 12.2% WER after 140M frames, i.e. only 1.5 passes through the dataset (which has 92.1M frames). Using just SGD we achieve 11.8% WER in 3.5 passes through the data.

We only present results after cross-entropy training, so we compare against the best published cross-entropy trained CNNs. The baseline is the work of Soltau et al. [17] using classical CNNs with 512 feature maps on both convolutional layers. A second baseline is the work of Saon et al. [18] which introduces annealed dropout maxout CNN's with a large number of HMM states, achieving 12.6% WER after cross-entropy training (not in the paper, from personal communication). Note that these improvements could readily be integrated with our very deep CNN architectures.

4. DISCUSSION

In this paper we proposed a number of architectural advances in CNNs for LVCSR. We introduced a very deep convolutional network architecture with small 3×3 kernels and multiple convolutional layers before each pooling layer, inspired by the VGG Imagenet 2014 architecture. Our best performing model has 14 weight layers. We also introduced multilingual CNNs which proved valuable in the context of low resource speech recognition. We introduced multi-scale input features aimed at exploiting more acoustic context with minimal computational increase. We showed an improvement of 2.50% WER over a standard DNN PLP baseline using 3 hours of data, and an improvement of 5.77% WER by combining six languages to train on 18 hours of data. We then showed results on Hub5'00 after training on 262 hours of SWB-1 data where we get 11.8% WER, which is an improvement of 2.0% WER (14.5% relative) over our own baseline, and a 1.4% WER (10.6% relative) improvement over the best result published on classical CNNs after cross-entropy training [17].

We expect additional gains from sequence training, joint training with DNNs [17], and integrating improvements like annealed dropout and maxout nonlinearities [18].

5. ACKNOWLEDGEMENT

This effort uses the very limited language packs from IARPA Babel Program language collections IARPA-babel205b-v1.0a, IARPA-babel207b-v1.0e, IARPA-babel301b-v2.0b, IARPA-babel302b-v1.0a, IARPA-babel303b-v1.0a, and IARPA-babel304b-v1.0b. 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.

We gratefully acknowledge the support of NVIDIA Corporation.

The authors would like to thank Pierre Sermanet for the initial code base, George Saon, Vaibhava Goel, Etienne Marcheret and Xiaodong Cui for valuable discussions and comments.

6. REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proc. ICLR*, 2015.

[4] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proc. CVPR*, 2013, pp. 3626–3633.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, 2014, pp. 580–587.

[6] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *Proc. ICLR*, 2014.

[7] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2013.

[8] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 37, no. 3, 1989.

[9] L. Bottou, F. F. Soulié, P. Blanchet, and J. S. Liénard, "Experiments with time delay networks and dynamic time warping for speaker independent isolated digits recognition," in *Proc. Eurospeech*, 1989.

[10] L. Bottou, F. F. Soulié, P. Blanchet, and J. S. Liénard, "Speaker-independent isolated digit recognition: multilayer perceptrons vs. dynamic time warping," *Neural Networks*, vol. 3, no. 4, pp. 453–465, 1990.

[11] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe, "Global optimization of a neural network-hidden Markov model hybrid," *IEEE Trans. on Neural Networks*, vol. 3, no. 2, pp. 252–259, 1992.

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

[13] A.-r. Mohamed, T. N Sainath, G. Dahl, B. Ramabhadran, G. E Hinton, and Michael A P., "Deep belief networks using discriminative features for phone recognition," in *Proc. ICASSP*. IEEE, 2011, pp. 5060–5063.

[14] Brian Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *Proc. ICASSP*. IEEE, 2009, pp. 3761–3764.

[15] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *Proc. ICASSP*, 2012, pp. 4277–4280.

[16] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *Proc. ICASSP*, 2013.

[17] H. Soltau, G. Saon, and T. N. Sainath, "Joint training of convolutional and non-convolutional neural networks," *Proc. ICASSP*, 2014.

[18] G. Saon, H.-K. Kuo, S. Rennie, and M. Picheny, "The IBM 2015 english conversational telephone speech recognition system," *Proc. Interspeech*, 2015.

[19] M. Bi, Y. Qian, and K. Yu, "Very deep convolutional neural networks for LVCSR," in *Proc. Interspeech*, 2015.

[20] S. Scanzio, P. Laface, L. Fissore, R. Gemello, and F. Mana, "On the use of a multilingual neural network front-end," in *Proc. Interspeech*, 2008.

[21] S. Thomas, S. Ganapathy, and H. Hermansky, "Multilingual MLP features for low-resource LVCSR systems," in *Proc. ICASSP*, 2012.

[22] Z. Tüske, J. Pinto, D. Willett, and R. Schlüter, "Investigation on cross-and multilingual MLP features under matched and mismatched acoustical conditions," in *Proc. ICASSP*, 2013.

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

[24] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 2809–2813.

[25] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CVPR*, 2015.

[26] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proc. NIPS*, 2014, pp. 2366–2374.

[27] F. Grezl, M. Karafiát, and L. Burget, "Investigation into bottleneck features for meeting speech recognition.," in *Proc. Interspeech*, 2009, pp. 2947–2950.

[28] T. N Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," *Proc. ICASSP*, 2015.

[29] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," *Technical Report, arXiv:1212.5701*, 2012.

[30] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[31] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, 2010, pp. 249–256.