INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Ensembles of Multi-scale VGG Acoustic Models

Michael Heck[1,2], *Masayuki Suzuki*[1], *Takashi Fukuda*[1], *Gakuto Kurata*[1], *Satoshi Nakamura*[2]

[1]Watson Multimodal, IBM Research - Tokyo, Japan
[2]Graduate School of Information Science, Nara Institute of Science and Technology, Japan

michael-h@is.naist.jp, szuk@jp.ibm.com, fukuda1@jp.ibm.com, gakuto@jp.ibm.com, s-nakamura@is.naist.jp

Abstract

We present our work on constructing multi-scale deep convolutional neural networks for automatic speech recognition. Several VGG nets have been trained that differ solely in the kernel size of the convolutional layers. The general idea is that receptive fields of varying sizes match structures of different scales, thus supporting more robust recognition when combined appropriately. We construct a large multi-scale system by means of system combination. We use ROVER and the fusion of posterior predictions as examples of late combination, and knowledge distillation using soft labels from a model ensemble as a way of early combination. In this work, distillation is approached from the perspective of knowledge transfer pre-training, which is followed by a fine-tuning on the original hard labels. Our results show that it is possible to bundle the individual recognition strengths of the VGGs in a much simpler CNN architecture that yields equal performance with the best late combination.

Index Terms: acoustic modeling, knowledge distillation, ensembles, multi-scale, speech recognition, VGG

1. Introduction

The construction of large-scale systems for automatic speech recognition is a multi-faceted challenge that involves a substantial amount of work. Along the path to a final system, one typically has to solve complex tasks such as model selection, efficient combination of sub-systems, and system integration for the application to target data. Very large high performance systems such as [1] are usually built by combining sufficiently diverse models, so that individual model hypotheses complement each other and result in an overall better system output. Systems that differ in the acoustic front-end, training objective, model type or the used training data are frequent candidates for combination. Often, more than one distinction is exploited to maximize positive effects. However, finding good pairs for such a combination is not a trivial task. Moreover, the development of various distinctive systems in parallel can bind many resources.

One of the most popular methods for system combination is ROVER [2], where the combination is done on hypothesis level after decoding the same data with multiple systems. The more complex confusion network combination method [3] generates a lattice given 1-best hypotheses from different systems to find a path with higher scores. Lastly, the combination of decoder output lattices [4] prior to a refined decoding tries to make use of all the information each individual system can provide. All these methods have in common that they are expensive in terms of development costs and especially in accumulated decoding time: Each system has to transcribe the target data entirely before any combination can be performed to produce the final output. This circumstance might be negligible for small target data such as the test sets of popular annual speech recognition evaluations [5, 6], but it is a knockout argument for handling large data sets. Semi-supervised training methods - with more and more data being freely available in the world wide web - make use of massive amounts of automatically transcribed audio data. To produce such transcriptions in reasonable time, fast yet powerful speech recognizers are in dire need.

Combination of neural network based systems can be done by fusing posterior predictions after acoustic model (AM) scoring, for instance by stacking the posteriors of individual models [7] or joint training of different network topologies [8]. The advantage over the methods mentioned above is that the language model (LM) scoring has to be performed only once, which saves a good portion of decoding complexity. The disadvantage is the need of a more complex training due to the combined architecture. A method called *knowledge distillation* can be used to follow a fundamentally different approach to model combination. Knowledge distillation [9] is derived from the principle of model compression [10] and drew attention anew in the wake of regained popularity of fundamental deep learning approaches to speech processing [11]. The general idea is to train a simple model with the help of information provided by a much more complex model (as done in [12], for instance). Thus, the knowledge of a "teacher" is compressed – or distilled – into a "student". Knowledge distillation from ensembles [13] extends this idea by using a whole ensemble of models as teacher, which effectively combines multiple models into a single model.

This work describes our approach to building a large but lightweight and robust automatic speech recognizer that can be utilized for offline transcription of extensive amounts of data. We resort to conventional system combination methods to make use of multiple models for decoding. Our strategy to construct a set of models for combination is inspired by a multi-scale convolutional neural network (MS-CNN) architecture that has been successfully applied to object detection [14]. The MS-CNN consists of sub-networks with receptive fields that vary in size. The idea is that these receptive fields match objects of different scales, which is hoped to support robust recognition. We adapt the concept of multi-scaling by training several deep convolutional neural network models with VGG net architecture [15] that solely differ in their kernel size for the convolutional layers. We compare ROVER to fusing posterior predictions by averaging, both being examples of late combination, i.e., a combination that happens late in the decoding pipeline. We show that the difference in kernel size is producing sufficiently diverse model sets for successful combination. We further demonstrate the effectiveness of knowledge distillation from ensembles to train a classical CNN model [16] that

This work was done during the first author's internship period in IBM

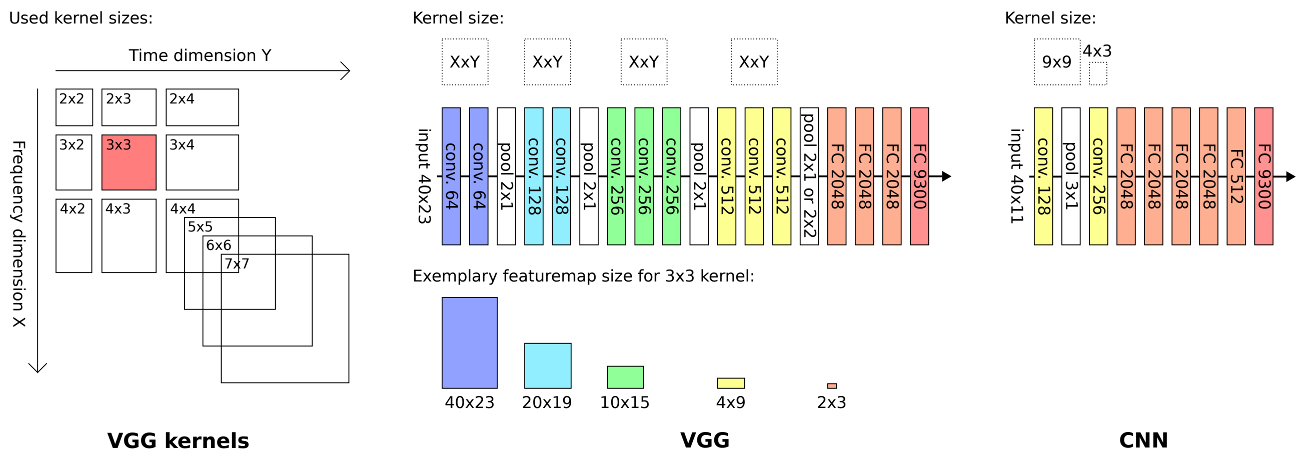

Figure 1: Left: *Set of permitted kernel sizes for the VGG nets.* Center: *Layout of the VGG nets. All convolutional layers share the same kernel size. Padding, pooling and the featuremap sizes depend on the selected kernel size. See also Table 1.* Right: *Classical CNN that is used as student in the knowledge distillation.*

is much simpler in structure but yields equal performance when compared to the best conventional system combination. We differ from [13] in that we successfully distill knowledge from a fairly homogeneous ensemble. [17] utilizes knowledge distillation as a pre-training step and shows that it can lead to faster initialization of entire networks and help train complex models with better performance even with a weak teacher model. Similarly, we utilize the distillation to pre-train an experienced, i.e., already initialized model to support a subsequent fine-tuning in finding a better local optimum.

2. Multi-scale ensemble

What we call *multi-scale ensemble* is a combination of several very deep convolutional networks that share the same base architecture and are trained on the same data in the same manner, but differ in the size of their receptive fields. Our base architecture is derived from the *VGG net* that is well known from the Imagenet 2014 challenge [15] and adjusted for the purpose of LVCSR. In the VGG net architecture, the large convolutional kernels of standard CNNs are replaced with small kernels that are arranged in stacks of layers, which leads to a much deeper net, and thus more non-linearity. This change is desired as it produces the same receptive field with less parameters.

We use the “WDX” network layout of [18] that features 10 convolutional and 4 fully connected weight layers. A graphical representation of this layout is given in Figure 1. The default network uses a kernel size of 3x3 (frequency domain x time domain). Zero padding with size 1 is applied in the frequency dimension before each convolution. The max pooling layers have size 2x1, with a stride that is equal to the pooling size, i.e., 1 in the time domain. This configuration has the effect, that the frequency dimension is kept fixed during the convolutions, and is reduced only through max pooling. Figure 1 depicts the details of the base model. The network input is a stack of frames with stacking context $1 + 2c$, where $c = 11$ leading and succeeding frames are stacked to the center frame. Each frame is a 40 dimensional logMel feature vector. The last fully connected layer of the network represents 9300 context dependent HMM states, thus the output is a vector of posterior probabilities over states for each stacked input frame.

2.1. Models

In order to produce candidates for a model ensemble, we copy the base model architecture and modify the kernel size, where we distinct between the sizes X and Y in the frequency domain and time domain, respectively, meaning that the kernels are allowed to be non-quadratic. We tried to keep the output dimensionality of the feature extraction sub-network before the fully connected layers the same for each individual model, if possible. Thus, we adjust the zero padding for the convolutional layers and the max pooling size to compensate for changes in the kernel size: For a larger dimensionality, more excessive zero padding is required, and a larger max pooling window and stride is neccessary for smaller kernel sizes. Table 1 lists the specifics of all possible candidates that we generated for a model ensemble. Having the default kernel size 3x3 in mind, we allow variants with any size in {2,3,4}x{2,3,4}. In addition to that, we also use 5x5, 6x6 and 7x7 kernels.

2.2. Combination

We build model ensembles by means of late model combination in different stages of the decoding process. We use the AMs from Subsection 2.1 to run multiple decodings on the target data for ROVER combination. Our setup performs a majority vote on token level, without considering confidence measures.

Alternatively, we combine models on posterior probability level after scoring the audio data with multiple AMs. For each feature vector $\boldsymbol{x}_i$ of frame i, we compute the non-weighted average of posterior probabilities for all context-dependent (CD) states $\boldsymbol{s}_i$:

$$P_{\text{ensemble}}(\boldsymbol{s}_i|\boldsymbol{x}_i) = \frac{\sum_{m=1}^{M} P_m(\boldsymbol{s}_i|\boldsymbol{x}_i)}{M}, \forall i \in \{1,\ldots,N\} \quad (1)$$

This is straightforward as we use the same CD state target layout for the output layer of all M AMs in the multi-scale ensemble. After posterior probability combination, we complete decoding with the same pipeline as used for the individual ROVER systems. The latter approach has the advantage that full decoding has do be performed only once, which naturally reduces overall run-time by a considerable amount. Our third way of model combination is based on knowledge distillation and is explained in the following section.

Table 1: *Frequency padding, time padding and max pooling parameters for each kernel size. The values are set so that the final feature map before the fully connected layers has size 2x3. For a kernel size of 2 in the time dimension, the feature map size is 2x5.*

Layer	2x2	2x3	2x4	3x2	3x3	3x4	4x2	4x3	4x4	5x5	6x6	7x7
conv	fPad=1	fPad=1	fPad=1,tPad=1	fPad=1	fPad=1	fPad=1,tPad=1	fPad=3	fPad=3	fPad=3,tPad=1	fPad=3,tPad=1	fPad=4,tPad=2	fPad=4,tPad=2
conv,pool			fPad=1	fPad=1	fPad=1	fPad=1,tPad=1	tPad=2	fPad=2	fPad=2,tPad=1	fPad=3,tPad=1	fPad=3,tPad=2	fPad=4,tPad=2
conv	fPad=1	fPad=1	fPad=1,tPad=1	fPad=1	fPad=1	fPad=1,tPad=1	fPad=3	fPad=3	fPad=3,tPad=1	fPad=3,tPad=1	fPad=3,tPad=2	fPad=3,tPad=2
conv,pool	fPad=1	fPad=1	fPad=1,tPad=1	fPad=1	fPad=1	fPad=1,tPad=1	fPad=2	fPad=2	fPad=2,tPad=1	fPad=2,tPad=1	fPad=2,tPad=2	fPad=2,tPad=2
conv	fPad=1	fPad=1	fPad=1,tPad=1	fPad=1	fPad=1	fPad=1,tPad=1	fPad=3	fPad=3	fPad=3,tPad=1	fPad=3,tPad=1	fPad=3,tPad=2	fPad=3,tPad=2
conv				fPad=1	fPad=1	fPad=1		fPad=2	fPad=2	fPad=1,tPad=1	fPad=2,tPad=1	fPad=3,tPad=2
conv,pool	pool=2x2			pool=2x2			pool=2x2			tPad=1	fPad=2,tPad=1	fPad=2,tPad=2
conv	fPad=1	fPad=1	fPad=1	fPad=1	fPad=1	fPad=1	fPad=3	fPad=3	fPad=3	fPad=3,tPad=1	fPad=3,tPad=1	fPad=3,tPad=2
conv				fPad=1	fPad=1	fPad=1				fPad=3,tPad=1	fPad=3,tPad=1	fPad=3,tPad=2
conv,pool				fPad=1	fPad=1	fPad=1				tPad=1	fPad=1,tPad=1	fPad=3,tPad=2

3. Distillation from ensembles

Knowledge distillation is a method for model compression and knowledge transfer that requires a teacher model to train or guide a student model. The guidance is given through soft outputs, typically in the form of posterior probabilities, produced by the teacher model, given the shared data intended for training. Knowledge is *distilled* by increasing the temperature T in the tempered softmax function that is used in the last layer of a neural network to convert logits $\boldsymbol{z}_i$ into posterior probabilities:

$$p(c_k|\boldsymbol{x}) = \frac{\exp \frac{\boldsymbol{z}_k}{T}}{\sum_{l=1}^{K} \exp \frac{\boldsymbol{z}_l}{T}} \qquad (2)$$

where $P(\boldsymbol{s}_i|\boldsymbol{x}_i) = (p(c_1|\boldsymbol{x}_i), \ldots, p(c_K|\boldsymbol{x}_i))$.

A temperature higher than the default value of $T = 1$ produces a softer probability distribution over the CD states of the network. Typically, the student is showing competitive performance to the teacher model after distillation, commonly with the advantage of being less complex and therefore faster than its advisor model. The teacher is not restricted to being a single model. Ensembles of models can likewise serve as teacher. We use an ensemble of multi-scale VGG net models as teacher-ensemble to train a classical CNN student with much simpler architecture than each of the VGGs. Figure 1 shows the details of the CNN architecture. The ensemble is constructed by averaging the CD state posterior probabilities of the single VGG nets, that all share the same layout of the output layer. This is identical to the model combination on posterior level mentioned in Section 2.2, except that now the target data is intended for training. We use the distillation in the sense of knowledge transfer pre-training, i.e., a student model is first trained with the soft labels of the ensemble as targets to push the model parameters into a good direction in the parameter space. The pre-training is followed by a fine-tuning with the original hard labels as targets, using a lower learning rate, to reach a good local optimum.

4. Experiments

The training data for our models is comprised of 50 hours of conversational interview-style English. The evaluation data is of the same nature, with the speaker sets being disjoint. The number of speakers for the entire data set is 971, the average utterance length is 3.15 seconds. The evaluation set is split into three subsets A (3.5h, 24108 words), B (2.9h, 32539 words) and C (2.2h, 18713 words), sorted by increasing difficulty for a speech recognizer, caused by higher amounts of speakers with various English accents.

All models are trained using the cross-entropy criterion. The VGG nets are implemented in torch [19]. We use balanced sampling as described in [20] and optimize the training using stochastic gradient descent (SGD) with a batch size of 128 and an L2 weight penalty of $1\mathrm{e}{-6}$. The inital learning rate is set to 0.03 and is divided by 3 after 25M, 30M and 35M frames. Training is stopped after 40M frames. The classical CNN is built with an in-house toolkit. The initial learning rate is $5\mathrm{e}{-3}$ for the knowledge distillation pre-training and $5\mathrm{e}{-4}$ for the fine-tuning and is reduced using the Newbob strategy. We use the average word error rate (WER) over utterances for evaluation. For decoding we utilize a vocabulary that covers 250K words and a 4-gram LM with 200M n-grams. The latter was built by estimating a model for the training data transcripts and interpolating it with a general purpose LM.

4.1. Multi-scale VGGs

The nine different models with their kernel sizes circling around the default all performed quite similarly, with the difference between best (25.0% WER) and worst system being 0.6% absolute in WER. The model with default kernel produced 25.2% WER. With the more extreme squared kernel sizes, the accuracy of the models clearly suffer. The poorest model is using the 6x6 kernel and yields 27.0% WER. Moreover, the training duration is growing considerably larger as the more aggressive padding and pooling in all dimensions take their toll.

4.2. ROVER combination

The ROVER combination proved very effective, even though most models achieve very similar WERs. We combined between 3 and 9 models in intuitive constellations such as models with squared kernels of increasing size, or with kernels that differ in size along one or both axes. Generally, the tendency was that the more models in the combination, the better the outcome in terms of WER. If we constrained our models to only use squared kernels, combining all systems from kernel size 2x2 to 7x7 produced the best hypotheses, despite the poorer performance of the large-sized kernel models. The best ROVER combination was achieved by considering 9 systems whose kernel sizes circle around the 3x3 default, improving the accuracy by 4.7% relative compared to the baseline architecture, and 4% relative compared to the single best architecture. The details are listed in Table 2.

4.3. Posterior combination

The best ensembles for ROVER were our blueprint for the posterior combination after AM scoring and prior to LM scoring. The posterior averaging is unweighted, as is the ROVER combination. For computing the combined posteriors, we first extract the logit values from the layer right before the final softmax layer. We do this for each frame, given all networks of the en-

Table 2: *Performance of the ROVER combination and posterior combination, compared to decoding with our default VGG net.*

Ensemble	**Kernels**												**ROVER**				**Posterior**			
	2x2	2x3	2x4	3x2	3x3	3x4	4x2	4x3	4x4	5x5	6x6	7x7	A	B	C	all	A	B	C	all
VGG x1					✓								17.6	24.5	33.4	25.2	17.6	24.5	33.4	25.2
VGG x2					✓				✓				-	-	-	-	17.5	23.6	32.4	24.5
VGG x3	✓				✓				✓				17.2	24.2	32.9	24.8	17.2	23.4	32.2	24.3
VGG x6	✓				✓				✓	✓	✓	✓	17.0	23.7	32.1	24.3	17.0	23.3	32.0	24.1
VGG x9	✓	✓	✓	✓	✓	✓	✓	✓	✓				16.8	23.4	31.9	24.0	17.1	23.0	31.8	24.0

Table 3: *Performance of the distilled CNN model, after pre-training (pt) and fine-tuning (ft) the experienced student.*

System	**A**	**B**	**C**	**all**
CNN (exp. student)	46.2	34.5	44.5	41.7
CNN (adapt)	18.3	22.9	33.3	24.8
VGG x1 → CNN (pt)	19.2	23.7	34.5	25.8
VGG x6 → CNN (pt)	18.7	23.3	34.0	25.3
VGG x9 → CNN (pt)	18.7	23.2	33.9	25.3
VGG x1 → CNN (ft)	17.7	22.6	32.6	24.3
VGG x6 → CNN (ft)	17.6	22.6	32.5	24.2
VGG x9 → CNN (ft)	17.4	22.4	32.4	24.1

semble. The values are averaged frame-wise and dumped, ready to be loaded for the rest of the decoding. Besides the smaller overhead for decoding, one advantage of the posterior combination is that 2-system ensembles are feasible, as opposed to the majority vote based ROVER, which has to rely on at least 3 systems for it to be reasonable. Our results in Table 2 show that the posterior combination can be superior when only a few models are involved in the combinations: A 3-system posterior combined ensemble can compete even with a 6-system ROVER. With increasing size of the ensemble, both methods converge to the same overall performance, with the posterior averaging still holding the advantage of a simpler decoding. Interestingly, the posteriorgram combination leads to the largest improvements on the more difficult portions of the evaluation data (subsets B and C). The richer informations captured by the soft labels seem more valuable for handling tougher data.

4.4. Knowledge distillation

We use knowledge distillation to pre-train a classical CNN, where the teacher is an ensemble of VGGs. Our student is already *experienced* in that it has been initialized by a training on about 2000h of out-of-domain data from the Switchboard corpus. The teacher ensemble is represented by averaged soft labels that are computed by the ensemble members using the tempered softmax function in Equation 2 with the temperature parameter set to $T = 2$. Initial experiments with untempered softmax did not lead to satisfactory results, thus we resorted to a parameter value that has already been confirmed to be a very reasonable choice [9, 17]. We display the performance of this system before and after knowledge distillation in Table 3.

We also compare to the standard approach of domain adaptation via fine-tuning on the 50h of in-domain data using the original hard labels, which leads to a strong baseline that can beat the single best VGG, but does not triumph over our best multi-scale VGG ensemble. Our numbers reveal that the knowledge distillation alone, i.e., the pre-training on soft labels, is not enough to challenge the conventionally adapted model. However, we see that after fine-tuning on the original hard labels the performance can be greatly increased. This phenomenon is in conformity with the observations made in [17]. The fine-tuned CNN model outperforms the conventionally adapted model and achieves equal performance with the best VGG ensemble, albeit its much simpler structure. This is remarkable insofar that even a model initialized on a fairly large amount of data can still benefit from the guidance given by the knowledge distillation to reach a better optimum in the parameter space.

Table 4: *Initial results on the Switchboard 300h data set for combining three multi-scale VGG models.*

Kernels			**ROVER**	**Posterior**
2x2	3x3	4x4		
12.4	12.3	12.0	11.8	11.8

4.5. Scalability

Table 4 lists results of scalability experiments on the Switchboard 300h data set. As can be seen, both ROVER and posteriorgram combination clearly improve decoding performance, even with just three multi-scale VGGs in the model ensemble. We also analyzed the impact of the kernel size on the real-time factor (RTF) for the model training with a single GPU (NVIDIA Tesla K80). Using the default 3x3 kernel yields an RTF of 0.12. The lowest (0.08) and highest (0.5) RTFs are achieved with the smallest (1x2) and largest (7x7) kernels, respectively.

5. Conclusions

We have shown that deep convolutional neural network acoustic models that solely differ in their kernel size hold sufficient diversity and complementarity for successful model combination, which greatly simplifies building ensembles of models. Both ROVER and posterior probability based combination of such multi-scale models are efficient for improving recognition accuracy, the latter being superior in terms of reduced decoding complexity, higher robustness on difficult data and greater stability even on small ensembles. We successfully distilled the knowledge of these ensembles into a classical CNN with much simpler architecture and achieved equal performance compared to the best performing conventional combination. The aggregate of our results shows that multi-scale model ensembles can be used successfully in a variety of ways and that the method scales with the training data.

6. Acknowledgements

We express our gratitude to Tom Sercu for providing the Torch code that we used to build the VGG nets. Part of this work was supported by JSPS KAKENHI Grant Number JP17H06101.

7. References

[1] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dimitriadis, X. Cui, B. Ramabhadran, M. Picheny, L.-L. Lim *et al.*, "English conversational telephone speech recognition by humans and machines," *arXiv preprint arXiv:1703.02136*, 2017.

[2] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)," in *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*. IEEE, 1997, pp. 347–354.

[3] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus among words: lattice-based word error minimization." in *Eurospeech*, 1999.

[4] H. Xu, D. Povey, L. Mangu, and J. Zhu, "An improved consensus-like method for minimum Bayes risk decoding and lattice combination," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4938–4941.

[5] M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, R. Cattoni, and M. Federico, "The IWSLT 2016 evaluation campaign," in *International Workshop on Spoken Language Translation (IWSLT)*, 2016.

[6] A. Ali, P. Bell, J. Glass, Y. Messaoui, H. Mubarak, S. Renals, and Y. Zhang, "The MGB-2 challenge: Arabic multi-dialect broadcast media recognition," *arXiv preprint arXiv:1609.05625*, 2016.

[7] L. Deng and J. Platt, "Ensemble deep learning for speech recognition," 2014.

[8] H. Soltau, G. Saon, and T. N. Sainath, "Joint training of convolutional and non-convolutional neural networks." in *ICASSP*, 2014, pp. 5572–5576.

[9] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[10] C. Buciluă, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 535–541.

[11] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[12] W. Chan, N. R. Ke, and I. Lane, "Transferring knowledge from a RNN to a DNN," *arXiv preprint arXiv:1504.01483*, 2015.

[13] Y. Chebotar and A. Waters, "Distilling knowledge from ensembles of neural networks for speech recognition," *Interspeech 2016*, pp. 3439–3443, 2016.

[14] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *European Conference on Computer Vision*. Springer, 2016, pp. 354–370.

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

[16] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on*. IEEE, 2013, pp. 8614–8618.

[17] Z. Tang, D. Wang, Y. Pan, and Z. Zhang, "Knowledge transfer pre-training," *arXiv preprint arXiv:1506.02256*, 2015.

[18] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. LeCun, "Very deep multilingual convolutional neural networks for LVCSR," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4955–4959.

[19] R. Collobert, S. Bengio, and J. Mariéthoz, "Torch: a modular machine learning software library," Idiap, Tech. Rep., 2002.

[20] T. Sercu and V. Goel, "Advances in very deep convolutional neural networks for LVCSR," *arXiv preprint arXiv:1604.01792*, 2016.