ON THE COMPRESSION OF RECURRENT NEURAL NETWORKS WITH AN APPLICATION TO LVCSR ACOUSTIC MODELING FOR EMBEDDED SPEECH RECOGNITION

Rohit Prabhavalkar[†] *Ouais Alsharif*[†] *Antoine Bruguier* *Ian McGraw*

Google Inc.
{prabhavalkar,oalsha,tonybruguier,imcgraw}@google.com

ABSTRACT

We study the problem of compressing recurrent neural networks (RNNs). In particular, we focus on the compression of RNN acoustic models, which are motivated by the goal of building compact and accurate speech recognition systems which can be run efficiently on mobile devices. In this work, we present a technique for general recurrent model compression that jointly compresses both recurrent and non-recurrent inter-layer weight matrices. We find that the proposed technique allows us to reduce the size of our Long Short-Term Memory (LSTM) acoustic model to a third of its original size with negligible loss in accuracy.

Index Terms— model compression, LSTM, RNN, SVD, embedded speech recognition

1. INTRODUCTION

Neural networks (NNs) with multiple feed-forward [1, 2] or recurrent hidden layers [3, 4] have emerged as state-of-the-art acoustic models (AMs) for automatic speech recognition (ASR) tasks. Advances in computational capabilities coupled with the availability of large annotated speech corpora have made it possible to train NN-based AMs with a large number of parameters [5] with great success.

As speech recognition technologies continue to improve, they are becoming increasingly ubiquitous on mobile devices: voice assistants such as Apple's Siri, Microsoft's Cortana, Amazon's Alexa and Google Now [6] enable users to search for information using their voice. Although the traditional model for these applications has been to recognize speech remotely on large servers, there has been growing interest in developing ASR technologies that can recognize the input speech directly "on-device" [7]. This has the promise to reduce latency while enabling user interaction even in cases where a mobile data connection is either unavailable, slow or unreliable. Some of the main challenges in this regard are the disk, memory and computational constraints imposed by these devices. Since the number of operations in neural networks is proportional to the number of model parameters, compressing the model is desirable from the point of view of reducing memory usage and power consumption.

In this paper, we study techniques for compressing recurrent neural networks (RNNs), specifically RNN acoustic models. We demonstrate how a generalization of conventional inter-layer matrix factorization techniques (e.g., [8, 9]), where we jointly compress both recurrent and inter-layer weight matrices, allows us to compress acoustic models up to a third of their original size with negligible loss in accuracy. While we focus on acoustic modeling, the techniques presented can be applied to RNNs in other domains, e.g., handwriting recognition [10] and machine translation [11] inter alia. The technique presented in this paper encompasses both traditional recurrent neural networks (RNNs) *as well as* Long Short-Term Memory (LSTM) neural networks.

In Section 2, we review previous work that has focussed on techniques for compressing neural networks. Our proposed compression technique is presented in Section 3. We examine the effectiveness of proposed techniques in Sections 4 and 5. Finally, we conclude with a discussion of our findings in Section 6.

2. RELATED WORK

There have been a number of previous proposals to compress neural networks, both in the context of ASR as well as in the broader field of machine learning. We summarize a number of proposed approaches in this section.

It has been noted in previous work that there is a large amount of redundancy in the parameters of a neural network. For example, Denil et al. [12] show that the entire neural network can be reconstructed given the values of a small number of parameters. Caruana and colleagues show that the output distribution learned by a larger neural network can be approximated by a neural network with fewer parameters by training the smaller network to directly predict the outputs of the larger network [13, 14]. This approach, termed "model compression" [13] is closely related to the recent "distillation" approach proposed by Hinton et al. [15]. The redundancy in a neural network has also been exploited in the HashNet approach of Chen et al. [16], which imposes parameter tying in

[†]Equal contribution. The authors would like to thank Haşim Sak and Raziel Alvarez for helpful comments and suggestions on this work, and Chris Thornton and Yu-hsin Chen for comments on an earlier draft.

network based on a set of hash functions.

In the context of ASR, previous approaches to acoustic model compression have focused mainly on the case of feed-forward DNNs. One popular technique is based on sparsifying the weight matrices in the neural network, for example, by setting weights whose magnitude falls below a certain threshold to zero [1] or based on the second-derivative of the loss function in the "optimal brain damage" procedure [17]. In fact, Seide et al. [1] demonstrate that up to two-thirds of the weights of the feed-forward network can be set to zero without incurring any loss in performance. Although techniques based on sparsification do decrease the number of effective weights, encoding the subset of weights which can be 'zeroed out' requires additional memory. Further, if the weight matrices are represented as dense matrices for efficient computation, then the parameter savings on disk will not translate in to savings of runtime memory. Other techniques to reduce the number of model parameters is based on changing the neural network architecture, e.g., by introducing bottleneck layers [18] or through a low-rank matrix factorization layer [19]. We also note recent work by Wang et al. [20] which uses a combination of singular value decomposition (SVD) and vector quantization to compress acoustic models.

The methods investigated in our work are most similar to previous work that has examined using SVD to reduce the number of parameters in the network in the context of feed-forward DNNs [8, 9, 21]. As we describe in Section 3, our methods can be thought of as an extension of the techniques proposed by Xue et al. [8], wherein we jointly factorize both recurrent and (non-recurrent) inter-layer weight matrices in the network.

3. MODEL COMPRESSION

In this section, we present a general technique for compressing individual recurrent layers in a recurrent neural network, thus generalizing the methods proposed by Xue et al. [8].

We describe our approach in the most general setting of a standard RNN. We denote the activations of the l-th hidden layer, consisting of N^l nodes, at time t by $\mathbf{h}_t^l \in \mathbb{R}^{N^l}$. The inputs to this layer at time t – which are in turn the activations from the previous layer or the input features – are denoted by $\mathbf{h}_t^{l-1} \in \mathbb{R}^{N^{l-1}}$. We can then write the following equations which define the output activations of the l-th and $(l+1)$-th layers in a standard RNN:

$$\mathbf{h}_t^l = \sigma(W_x^{l-1}\mathbf{h}_t^{l-1} + W_h^l\mathbf{h}_{t-1}^l + \mathbf{b}^l) \quad (1)$$

$$\mathbf{h}_t^{l+1} = \sigma(W_x^l\mathbf{h}_t^l + W_h^{l+1}\mathbf{h}_{t-1}^{l+1} + \mathbf{b}^{l+1}) \quad (2)$$

where, $b^l \in \mathbb{R}^{N^l}$ and $b^{l+1} \in \mathbb{R}^{N^{l+1}}$ represent bias vectors, $\sigma(\cdot)$ denotes a non-linear activation function, and $W_x^l \in \mathbb{R}^{N^{l+1} \times N^l}$ and $W_h^l \in \mathbb{R}^{N^l \times N^l}$ denote weight matrices that we refer to respectively as the *inter-layer* and the *recurrent* weight matrices, respectively[1]. Since our proposed approach can be applied independently for each recurrent hidden layer, we only describe the compression operations for a particular layer l. We jointly compress the recurrent and inter-layer matrices corresponding to a specific layer l by determining a suitable recurrent projection matrix [3], denoted by $P^l \in \mathbb{R}^{r^l \times N^l}$, of rank $r^l < N^l$ such that, $W_h^l = Z_h^l P^l$ and $W_x^l = Z_x^l P^l$, thus allowing us to re-write (1) and (2) as,

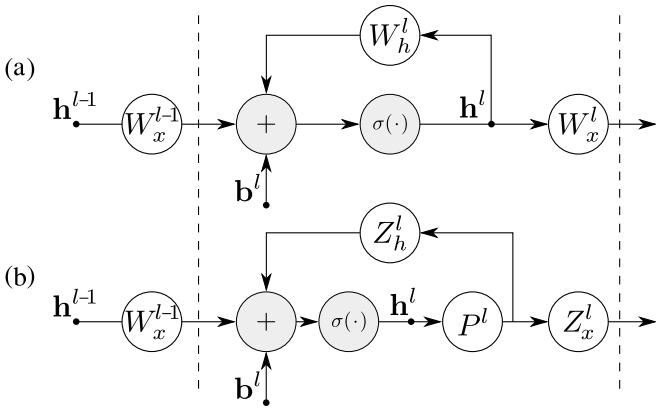

Fig. 1. The initial model (Figure (a)) is compressed by jointly factorizing recurrent (W_h^l) and inter-layer (W_x^l) matrices, using a shared recurrent projection matrix (P^l) [3] (Figure (b)).

$$\mathbf{h}_t^l = \sigma(W_x^{l-1}\mathbf{h}_t^{l-1} + Z_h^l P^l \mathbf{h}_{t-1}^l + \mathbf{b}^l) \quad (3)$$

$$\mathbf{h}_t^{l+1} = \sigma(Z_x^l P^l \mathbf{h}_t^l + W_h^{l+1}\mathbf{h}_{t-1}^{l+1} + \mathbf{b}^{l+1}) \quad (4)$$

where, $Z_h^l \in \mathbb{R}^{N^l \times r^l}$ and $Z_x^l \in \mathbb{R}^{N^{l+1} \times r^l}$. This compression process is depicted graphically in Figure 1.

We note that sharing P^l across the recurrent and inter-layer matrices allows for more efficient parameterization of the weight matrices; as shown in Section 5, this does not result in a significant loss of performance. Thus, the degree of compression in the model can be controlled by setting the ranks r^l of the projection matrices in each of the layers of the network.

We determine the recurrent projection matrix P^l, by first computing an SVD of the recurrent weight matrix, which we then truncate, retaining only the top r^l singular values (denoted by $\widetilde{\Sigma_h^l}$) and the corresponding singular vectors from U_h^l and V_h^l (denoted by $\widetilde{U_h^l}$ and $\widetilde{V_h^l}$, respectively):

$$W_h^l = U_h^l \Sigma_h^l {V_h^l}^T \approx \left(\widetilde{U_h^l}\widetilde{\Sigma_h^l}\right)\widetilde{V_h^l}^T = Z_h^l P^l \quad (5)$$

where $Z_h^l = \widetilde{U_h^l}\widetilde{\Sigma_h^l}$ and $P^l = \widetilde{V_h^l}^T$. Finally, we determine Z_x^l, as the solution to the following least-squares problem:

$$Z_x^l = \arg\min_Y \|YP^l - W_x^l\|_{\mathcal{F}}^2 \quad (6)$$

[1]The equations are slightly more complicated when using LSTM cells in the recurrent layer, but the basic form remains the same. See Section 3.1.

where, $\|X\|_{\mathcal{F}}$ denotes the Frobenius norm of the matrix. In pilot experiments we found that the proposed SVD-based initialization performed better than training a model with recurrent projection matrices (i.e., same model architecture) but with random initialization of the network weights.

3.1. Applying our technique to LSTM RNNs

Generalizing the procedure described above in the context of *standard* RNNs to the case of LSTM RNNs [3, 22, 23] is straightforward. Using the notation in [3], note that the recurrent-weight matrix W_h^l in the case of the LSTM is the concatenation of the four *gate weight matrices*, obtained by stacking them vertically:

$$[W_{im}, W_{om}, W_{fm}, W_{cm}]^T$$

which represent respectively, recurrent connections to the input gate, the output gate, the forget gate and the cell state. Similarly, the inter-layer matrix W_x^l is the concatenation of the matrices:

$$[W_{ix}, W_{fx}, W_{ox}, W_{cx}]^T$$

which correspond to the input gate, the forget gate, the output gate and the cell state (of the next layer). With these definitions, compression can be applied as described in Section 3. Note that we do not compress the "peep-hole" weights, since they are already narrow, single column matrices and do not contribute significantly to the total number of parameters in the network.

4. EXPERIMENTAL SETUP

In order to determine the effectiveness of the proposed RNN compression technique, we conduct experiments on a open-ended large-vocabulary dictation task.

As we mentioned in Section 1, one of our primary motivations behind investigating acoustic model compression is to build compact acoustic models that can be deployed on mobile devices. In recent work, Sak et al. have demonstrated that deep LSTM-based AMs trained to predict either context-independent (CI) phoneme targets [22] or context-dependent (CD) phoneme targets [23] approach state-of-the-art performance on speech tasks. These systems have two important characteristics: in addition to the CI or CD phoneme labels, the system can also hypothesize a "blank" label if it is unsure of the identity of the current phoneme, and the systems are trained to optimize the connectionist temporal classification (CTC) criterion [24] which maximizes the total probability of correct label sequence conditioned on the input sequence. More details can be found in [22, 23].

Following [22], our baseline model is thus a *CTC model*: a five hidden layer RNN with 500 LSTM cells in each layer, which predicts 41 CI phonemes (plus "blank"). As a point of comparison, we also present results obtained using a much larger state-of-the-art 'server-sized' model which is too large to deploy on embedded devices but nonethless serves as an upper-bound performance for our models on this dataset. This model consists of five hidden layers with 600 LSTM cells per layer, and is trained to predict one of 9287 context-dependent phonemes (plus "blank").

Our systems are trained using distributed asynchronous stochastic gradient descent with a parameter server [25]. The systems are first trained to convergence to optimize the CTC criterion, following which these are discriminatively sequence trained to optimize the state-level minimum Bayes risk (sMBR) criterion [26, 27]. As discussed in Section 5, after applying the proposed compression scheme, we further fine-tune the network: first with the CTC criterion, followed by sequence discriminative training with the sMBR criterion. This additional fine-tuning step was found to be necessary to achieve good performance, particularly as the amount of compression was increased.

The language model used in this work is a 5-gram model trained on $\sim$100M sentences of in-domain data, with entropy-based pruning applied to reduce the size of the LM down to roughly 1.5M n-grams (mainly bigrams) with a 64K vocabulary. Since our goal is to build a recognizer to run efficiently on mobile devices, we minimize the size of the decoder graph used for recognition, following the approach outlined in [7]: we perform an additional pruning step to generate a much smaller *first-pass* language model (69.5K n-grams; mainly unigrams), which is composed with the lexicon transducer to construct the decoder graph. We then perform on-the-fly rescoring with the larger LM. The resulting models, when compressed for use on-device, total about 20.3 MB, thus enabling them to be run many times faster than real-time on recent mobile devices [28].

We parameterize the input acoustics by computing 40-dimensional log mel-filterbank energies over the 8Khz range, which are computed every 10ms over 25ms windowed speech segments. The server-sized system uses 80-dimensional features computed over the same range since this resulted in slightly improved performance. Following [23], we stabilize CTC training by stacking together 8 consecutive speech frames (7 right context frames); only every third stacked frame is presented as an input to the network.

4.1. Training and Evaluation Data

Our systems are trained on $\sim$3M hand-transcribed anonymized utterances extracted from Google voice search traffic ($\sim$2000 hours). We create "multi-style" training data by synthetically distorting utterances to simulate background noise and reverberation using a room simulator with noise samples extracted from YouTube videos and environmental recordings of everyday events; 20 distorted examples are created for each utterance in the training set. Systems are additionally adapted using the sMBR criterion [26, 27] on a set of

$\sim$1M anonymized hand-transcribed (in-domain) dictation utterances extracted from Google traffic, processed to generate "multi-style" training data as described above, which improves performance on our dictation task. All results are reported on a set of 13.3K hand-transcribed anonymized utterances extracted from Google traffic from an open-ended dictation domain.

5. RESULTS

In our experiments, we seek to determine the impact of the proposed joint SVD-based compression technique on system performance. In particular, we are interested in determining how system performance varies as a function of the degree of compression, which is controlled by setting the ranks of the recurrent projection matrices r^l as described in Section 3.

Notice that since the proposed compression scheme is applied to all hidden layers of the baseline system, there are numerous settings of the ranks r^l for the projection matrices in each layer which result in the same number of total parameters in the compressed network. In order to avoid this ambiguity, we set the various projection ranks using the following criterion: Given a threshold τ, for each layer l, we set the rank r^l of the corresponding projection matrix such that it corresponds to retaining a fraction of at most τ of the *explained variance* after the truncated SVD of W_h^l. More specifically, if the singular values in Σ_h^l in (5) are sorted in non-increasing order as $\sigma_1^l \geq \sigma_2^l \geq \cdots \geq \sigma_N^l$, we set each r^l as:

$$r^l = \underset{1 \leq k \leq N}{\arg\max} \left\{ \frac{\sum_{j=1}^{k} {\sigma_j^l}^2}{\sum_{j=1}^{N} {\sigma_j^l}^2} \leq \tau \right\} \quad (7)$$

Choosing the projection ranks using (7) allows us to control the degree of compression, and thus compressed model size by varying a single parameter, τ. In pilot experiments we found that this scheme performed better than setting ranks to be equal for all layers (given the same total parameter budget). Once the projection ranks r^l have been determined for the various projection matrices we fine-tune the compressed models by first optimizing the CTC criterion, followed by sequence training with the sMBR criterion and adaptation on in-domain data as described in Section 4.1. The results of our experiments are presented in Table 1.

As can be seen in Table 1, the baseline system which predicts CI phoneme targets is only $\sim$10% relative worse than the larger server-sized system, although it has half as many parameters. Since the ranks r^l are all chosen to retain a given fraction of the explained variance in the SVD operation, we also note that earlier hidden layers in the network appear to have lower ranks than later layers, since most of the variance is accounted for by a smaller number of singular values. It can be seen from Table 1 that word error rates increase as the amount of compression is increased, although performance of the compressed systems are close to the baseline for moderate compression ($\tau \geq 0.7$). Using a value of $\tau = 0.6$, enables the model to be compressed to a third of its original size, with only a small degradation in accuracy. However, performance begins to degrade significantly for $\tau \leq 0.5$. Future work will consider alternative techniques for setting the projection ranks r^l in order to examine their impact on system performance.

System	Projection ranks, r^l	Params	WER
server	-	20.1M	11.3
baseline	-	9.7M	12.4
$\tau = 0.95$	350, 375, 395, 405, 410	8.6M	12.3
$\tau = 0.90$	270, 305, 335, 345, 350	7.2M	12.5
$\tau = 0.80$	175, 215, 245, 260, 265	5.4M	12.5
$\tau = 0.70$	120, 150, 180, 195, 200	4.1M	12.6
$\tau = 0.60$	80, 105, 130, 145, 150	3.1M	12.9
$\tau = 0.50$	50, 70, 90, 100, 110	2.3M	13.2
$\tau = 0.40$	30, 45, 55, 65, 75	1.7M	14.4

Table 1. Word error rates (%) on the test set as a function of the percentage of explained variance retained (τ) after the SVDs of the recurrent weight matrices W_h^l in the hidden layers of the RNN.

6. CONCLUSIONS

We presented a technique to compress RNNs using a joint factorization of recurrent and inter-layer weight matrices, generalizing previous work [8]. The proposed technique was applied to the task of compressing LSTM RNN acoustic models for embedded speech recognition, where we found that we could compress our baseline acoustic model to a third of its original size with negligible loss in accuracy. The proposed techniques, in combination with weight quantization, allow us to build a small and efficient speech recognizer that run many times faster than real-time on recent mobile devices [28].

7. REFERENCES

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

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. 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, no. 6, pp. 82–97, 2012.

[3] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. of Interspeech*, 2014, pp. 338–342.

[4] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Proc. of ICASSP*, 2015, pp. 4580–4584.

[5] L. Deng and D. Yu, "Deep learning: methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[6] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope, ""Your Word is my Command": google search by voice: A case study," in *Advances in Speech Recognition*, pp. 61–90. Springer US, 2010.

[7] X. Lei, A. Senior, A. Gruenstein, and J. Sorensen, "Accurate and compact large vocabulary speech recognition on mobile devices," in *Proc. of Interspeech*, 2013, pp. 662–665.

[8] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Proc. of Interspeech*, 2013, pp. 2365–2369.

[9] J. Xue, J. Li, D. Yu, M. Seltzer, and Y. Gong, "Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network," in *Proc. of ICASSP*, 2014, pp. 6359–6363.

[10] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2009.

[11] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. of NIPS*, 2014, pp. 3104–3112.

[12] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas, "Predicting parameters in deep learning," in *Proc. of NIPS*, 2013, pp. 2148–2156.

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

[14] L. J. Ba and R. Caruana, "Do deep nets really need to be deep?," in *Proc. of NIPS*, 2014, pp. 2654–2662.

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

[16] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. of ICML*, 2015, pp. 2285–2294.

[17] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. of NIPS*, 1989, pp. 598–605.

[18] F. Grézl and P. Fousek, "Optimizing bottle-neck features for LVCSR," in *Proc. of ICASSP*, March 2008, pp. 4729–4732.

[19] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. of ICASSP*, 2013, pp. 6655–6659.

[20] Y. Wang, J. Li, and Y. Gong, "Small-footprint high-performance deep neural network-based speech recognition using split-VQ," in *Proc. of ICASSP*, 2015, pp. 4984–4988.

[21] P. Nakkiran, R. Alvarez, R. Prabhavalkar, and C. Parada, "Compressing deep neural networks using a rank-constrained topology," in *Proc. of Interspeech*, 2015, pp. 1473–1477.

[22] H. Sak, A. Senior, K. Rao, O. İrsoy, A. Graves, F. Beaufays, and J. Schalkwyk, "Learning acoustic frame labeling for speech recognition with recurrent neural networks," in *Proc. of ICASSP*, 2015, pp. 4280–4284.

[23] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," in *Proc. of Interspeech*, 2015, pp. 1468–1472.

[24] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. of ICML*, 2006, pp. 369–376.

[25] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *Proc. of NIPS*, 2012, pp. 1223–1231.

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

[27] H. Sak, O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, and M. Mao, "Sequence discriminative distributed training of long short-term memory recurrent neural networks," in *Proc. of Interspeech*, 2014, pp. 1209–1213.

[28] I. McGraw, R. Prabhavalkar, R. Alvarez, M. Gonzalez Arenas, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Gruenstein, F. Beaufays, and C. Parada, "Personalized speech recognition on mobile devices," in *Proc. of ICASSP*, 2016.