PERSONALIZED SPEECH RECOGNITION ON MOBILE DEVICES

Ian McGraw, Rohit Prabhavalkar, Raziel Alvarez, Montse Gonzalez Arenas, Kanishka Rao, David Rybach, Ouais Alsharif, Haşim Sak, Alexander Gruenstein, Françoise Beaufays, Carolina Parada

Google Inc.

{imcgraw,prabhavalkar,raziel,montse,kanishkarao,rybach,oalsha,hasim,alexgru,fsb,carolinap}@google.com

ABSTRACT

We describe a large vocabulary speech recognition system that is accurate, has low latency, and yet has a small enough memory and computational footprint to run faster than real-time on a Nexus 5 Android smartphone. We employ a quantized Long Short-Term Memory (LSTM) acoustic model trained with connectionist temporal classification (CTC) to directly predict phoneme targets, and further reduce its memory footprint using an SVD-based compression scheme. Additionally, we minimize our memory footprint by using a single language model for both dictation and voice command domains, constructed using Bayesian interpolation. Finally, in order to properly handle device-specific information, such as proper names and other context-dependent information, we inject vocabulary items into the decoder graph and bias the language model on-the-fly. Our system achieves 13.5% word error rate on an open-ended dictation task, running with a median speed that is seven times faster than real-time.

Index Terms— embedded speech recognition, CTC, LSTM, quantization, model compression.

1. INTRODUCTION

Speech recognition for dictation, search, and voice commands has become a standard feature on smartphones and wearable devices. The vast majority of the literature devoted to improving accuracy for these tasks assumes that speech recognition will be run in datacenters on powerful servers. However, despite increases in speed and the availability of mobile internet, speech recognition requests frequently have high latency, or even completely fail, due to unreliable or unavailable network connections. An embedded speech recognition system that runs locally on a mobile device is more reliable and can have lower latency; however, it must be accurate and must not consume significant memory or computational resources.

In this paper we extend previous work that used quantized deep neural networks (DNNs) and on-the-fly language model rescoring to achieve real-time performance on modern smartphones [1]. We demonstrate that given similar size and computation constraints, we achieve large improvements in word error rate (WER) performance and latency by employing Long Short-Term Memory (LSTM) recurrent neural networks (RNNs), trained with connectionist temporal classification (CTC) [2] and state-level minimum Bayes risk (sMBR) [3] techniques. LSTMs are made small and fast enough for embedded speech recognition by quantizing parameters to 8 bits, by using context independent (CI) phone outputs instead of more numerous context dependent (CD) phone outputs, and by using Singular Value Decomposition (SVD) compression [4, 5].

SVD has elsewhere been shown to be effective for speech processing tasks [4, 6, 7] as have structured transforms [8] and low-rank matrix factorizations [9]. Vector quantization has also been shown to significantly reduce model size with only small accuracy losses [10], however it is unclear whether this algorithm can be implemented in a computationally efficient manner while minimizing runtime memory footprint. Such parameter reduction techniques have generally been applied to DNNs and not RNNs. For embedded speech recognition, some authors have avoided RNNs citing increased computational costs and instead evaluated methods for transferring knowledge from RNNs to DNNs [11].

We present results in two very different domains: dictation and voice commands. To keep the disk space requirements of the system small, we experiment with language model interpolation techniques that enable us to effectively share a single model across both domains. In particular, we demonstrate how the application of bayesian interpolation out-performs simple linear interpolation for these tasks.

Finally, we explore using language model personalization techniques to improve voice command and dictation accuracy. Many voice commands can be completed and executed on a device without a network connection, or can easily be queued up to be completed over an unreliable or slow network connection later in the background. For example, a command such as "Send an email message to Darnica Cumberland: can we reschedule?" can be transcribed by an embedded speech recognition system and executed later without a perceptual difference to the user. Accurate transcription, however, requires integrating personal information such as the contact name "Darnica Cumberland" into the language model. We demonstrate that the vocabulary injection and on-the-fly language model biasing techniques from [12, 13] can significantly improve accuracy without significant adverse computational overhead.

The remainder of this paper is organized as follows. We summarize the baseline sytem in Section 2. Section 3 describes our techniques to build a small but accurate acoustic model, Section 4 describes our LM training procedure and the interpolation techniques used in our system, Section 5 describes the decoder. Section 6 describes how we handle context or device-specific information, and finally Section 7 summarizes the footprint of our system. Conclusions are presented in Section 8.

2. BASELINE SYSTEM

We model our baseline system after the embedded speech recognition system presented in [1]. Instead of using a standard feed-forward DNN, however, we use deep LSTM models which have been shown to achieve state-of-the-art results on large-scale speech recognition tasks [14, 15, 16]. The LSTM architecture of our baseline consists of three hidden layers with 850 LSTM cells in each. We

make use of a recurrent projection layer as described in [14] of size 450 for each of hidden layers. This LSTM is trained to predict 2,000 CD states, analogous to the system described in [1]. This system is also trained to optimize the standard (CE) criterion on the training set, with the output labels delayed by 5 frames [14].

The input features are 40-dimensional log mel-filterbank energies calculated on a 25ms window every 10ms. Unlike in [1], where frames are stacked to provide right and left context to the net, we rely on the LSTM's memory capabilities and supply only one frame every 10ms as input. This model was trained to optimize the standard cross-entropy (CE) criterion on the training set described in Section 3.1, with frame-level labels derived from a larger system.

The language model presented in this work also follows along the lines of [1]. The vocabulary size is restricted to 64K so that an index into the lexicon only requires 16-bits of storage. The small decoder graph is constructed from a tiny LM containing 70K n-grams (almost entirely of unigrams). During decoding the partial paths are rescored on-the-fly with a large LM containing roughly 1.5M n-grams. This rescoring LM is made extremely compact using the LOUDS [17] compression mechanism. More details of the LM can be found in Section 4.

3. ON-DEVICE ACOUSTIC MODELING

In this section we describe an LSTM configuration that can successfully be deployed to a mobile device and contrast this with the baseline system described in Section 2.

In particular, the LSTM architecture that we investigate is a *CTC model* [15, 16]: the system consists of five hidden layers with 500 LSTM cells in each, that predict 41 context independent (CI) phoneme targets plus an additional "blank" target that can be hypothesized if the system is unsure of the identity of the phoneme at the current frame. The system is trained to optimize the connectionist temporal classification (CTC) criterion [2] as described in [15, 16].

Similar to the baseline, we use standard 40-dimensional log mel-filterbank energies over the 8Khz range, computed every 10ms on 25ms of input speech. In order to stabilize CTC training, our CTC models use the strategy proposed in [16]: we stack together 8 consecutive frames (7 frames of right context) and only present every third stacked frame as input to the network. In addition to stabilizing CTC training, this has the additional benefit of speeding up computation since the network is only evaluated every 30ms.

3.1. AM Experiments

Our AMs are trained on 3M hand-transcribed anonymized utterances extracted from Google voice search traffic (approximately 2,000 hours). All models in our work are trained using distributed asynchronous stochastic gradient descent (ASGD) [18]. In order to improve robustness to noise and reverberation, we generate "multi-style" training data by synthetically distorting each training utterance using a room simulator with a virtual noise source, to generate 20 distorted versions of each utterance. Noise samples are extracted from YouTube videos and environmental recordings of daily events.

Results in this section are reported on a set of 13.3K anonymized utterances in the domain of open-ended dictation extracted from Google traffic. The LM used in these experiments was described in Section 2 and detailed further in Section 4. We benchmark our systems to determine runtime speed by decoding a subset of 100 utterances on a Nexus 5 smartphone which contains a 2.26 GHz quad-core CPU and 2 GB of RAM. We report median real-time factors (RT50) on our test set. Our results are presented in Table 1.

AM Setup	WER	Params	Size	RT50
LSTM 2,000 CD States	23.4	9.9M	39.4 MB	2.94
LSTM CTC CI Phones	19.4	9.7M	38.8 MB	0.64
+ sMBR	15.1	9.7M	38.8 MB	0.65
+ SVD Compression	14.8	3M	11.9 MB	0.22
+ adaptation	12.9	3M	11.9 MB	0.22
+ quantization	13.5	3M	3 MB	0.14
LSTM CTC (Server-size)	11.3	20.1M	80.4 MB	-

Table 1. Word Error Rates (%) on an open-ended dictation task, evaluating various acoustic models, using the same language model described in Section 4, along with median RT factor.

As can be seen in Table 1, and consistent with previous work [15], the CTC-trained LSTM model that predicts CI phones outperforms the CE-trained LSTM that predicts 2,000 CD states. Furthermore, although both systems are comparable in terms of the number of parameters, the CTC-trained model is about $4\times$ faster than the CE-trained baseline. Sequence discriminative training with the sMBR criterion [3, 19] further improves system performance by 20% relative to the CTC-trained sytem.

In order to reduce memory consumption further, we compress our acoustic models using projection layers that sit between the outputs of an LSTM layer and both the recurrent and non-recurrent inputs to same and subsequent layers [14]. Of crucial importance, however, is that when a significant rank reduction is applied, it is not sufficient to simply initialize the projection layer's weight matrix randomly for training with the CTC criterion. Instead we use the larger 'uncompressed' model without the projection layer and jointly factorize its recurrent and (non-recurrent) inter-layer weight matrices at each hidden layer using a form of singular value decomposition to determine a shared projection layer. This process yields an initialization that results in stable convergence as described in detail in [5]. In our system, we introduce projection matrices of rank 100 for the first four layers, and a projection matrix of rank 200 for the fifth hidden layer. Following SVD compression, we once again train the system to optimize the CTC criterion, followed by discriminative sequence training with the sMBR criterion. As can be seen in Table 1, the proposed compression technique allows us to compress the AM by about $3\times$.

Finally, we note that adapting the AM using a set of 1M anonymized hand-transcribed utterances from the domain of open-ended dictation (processed to generate multi-style training as described in Section 3.1) results in a further 12.8% relative improvement over the SVD compressed models. The combination of all of these techniques allows us to significantly improve performance over the baseline system. For completeness, we also trained a DNN system with topology described in [1]. As expected, this 2,000 CD state DNN performed significantly worse than all of the LSTMs in Table 1.

For reference, we also present results obtained using a much larger 'server-sized' CTC model, which predicts 9287 CD phones (plus "blank"), but is evaluated with the same LM and decoder graph as our other systems, which serves as a sort of upperbound performance on this task[1].

[1]This model uses 80-dimensional filterbank features in the frontend, since this resulted in slightly improved performance. Frame stacking and frame skipping are as in the CI LSTM CTC model.

3.2. Efficient Representation and Fast Execution

Since the 11.9 MB floating point neural network acoustic model described above consumes a significant chunk of the memory and processing-time, we quantize the model parameters (i.e. weights, biases) into a more compact 8-bit integer-based representation. This quantization has an immediate impact on the memory usage, reducing the acoustic model's footprint to a fourth of the original size. The final footprint of our AM is 3 MB as shown in Table 1. Using 8-bit integers also has the advantage that we can also achieve 8-way parallelism in many matrix operations on most mobile platforms.

Although we could have applied a number of compression schemes [20, 21], with simplicity and performance in mind, and validated by previous work [22], we adopt a uniform linear quantizer that assumes a uniform distribution of the values within a given range. First, we find the minimum and maximum values of the original parameters. We then use a simple mapping formula which determines a scaling factor that when multiplied by the parameters spreads the values evenly in the smaller precision scale, thus obtaining a quantized version of the original parameters. The inverse operation is used when converting a quantized value back to its 32-bit floating point equivalent.

During neural network inference, we operate in 8-bit integers everywhere except in the activation functions and the final output of the network, which remain in floating point precision (by converting between quantized 8-bit values and their 32-bit equivalents as needed). Our quantization scheme and the inference computation approach provides a $2\times$ speed-up in evaluating our acoustic models as compared to the unquantized model, with only a small performance degredation (compare 'adaptation' vs. 'quantization' in Table 1).

4. ON-DEVICE LANGUAGE MODELLING

In this work, we focus on building a compact language model for the domains of dictation and voice commands. To maintain a small system footprint, we train a single model for both domains. As described in Section 2, we limit the vocabulary size to 64K. Our language models are trained using unsupervised speech logs from the dictation domain ($\sim$100M utterances) and voice commands domain ($\sim$2M utterances). The voice command utterances were extracted by filtering general voice search queries through the grammars usually used to parse voice commands at runtime. Those queries that parsed were added to the training set. A Katz-smoothed 5-gram LM is then trained and entropy-based pruning is employed to shrink the LM to the sizes described in Section 2.

In addition to the dictation test set described in Section 3, in this section we present results on a voice commands test set. This set includes utterances from 3 types of commands: Device ($\sim$2K utterances) - which includes commands for device control (e.g., "Turn up volume"), Planning ($\sim$9K utterances) - consisting of utterances relevant to planning calendar events (e.g., "Set an alarm at 6 p.m."), and Communication ($\sim$8K utterances) with utterances relevant to chat messaging, emails, or making phone calls. The Communication set, also includes some open-ended dictation corresponding to the message (e.g. "Text Jacob, I'm running 10 minutes late, can we reschedule?").

All results in this section are evaluated using the quantized LSTM CI CTC acoustic model described in Section 3, thus allowing us to focus on the impact of the LM.

In order to build a single LM to use across both dictation and command domains, we explore different interpolation techniques.

LM Setup	Dictation WER	Commands WER
Linear Interpolation	12.9	10.0
Bayesian Interpolation	12.3	8.9
Bayesian + Rescoring	13.5	9.0

Table 2. Word Error Rates (%) on an open-ended dictation domain and the commands domain.

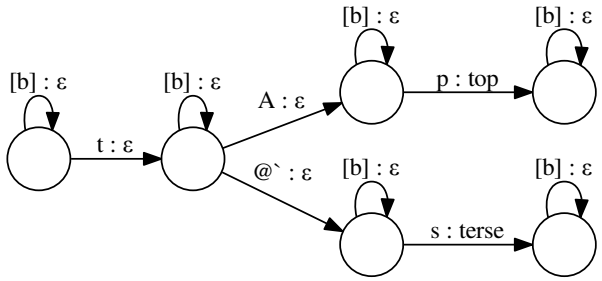

Fig. 1. Example of a part of a decoder graph with *blank* labels $[b]$.

As our baseline, we consider a linearly interpolated LM with interpolation weights estimated over a separate held-out development set sampled from speech logs. We compare performance obtained from the baseline system to a Bayesian interpolated LM [23], where voice commands and dictation are each represented as a unique task and the corresponding task priors are determined by sweeping parameters on a held-out development set to minimize word error rates rather than setting these based on the log counts.

Our results are presented in Table 2. The first two rows of the table highlight the utility of Bayesian interpolation over linear interpolation for both domains. The decoder graph used to produce these results was constructed with a single large language model, and therefore rescoring on-the-fly was not used. The third row of Table 2 shows the effects of on-the-fly rescoring on WER. Whereas the fully composed decoder graph is an unacceptable 29 MB, breaking them down into first-pass and on-the-fly rescoring models yields a 8.3 MB decoder graph and a 6.8 MB rescoring LM (with LOUDS compression [17]).

5. DECODER

In this section, we describe our decoder setup and a modification thereof that takes advantage of CTC's simple topology. In contrast to a conventional 3-state HMM structure, each phoneme is represented by a single AM output state in combination with a generic *blank* (or "non-perceiving") state. An FST-based decoder graph for the CTC model is created by the usual construction and composition of lexicon and LM transducers [24]. We do not require a context-dependency transducer, since we use context-independent phone models. Self-loop transitions are added to each state for the *blank* label. An example is shown in Figure 1.

We use an FST-based decoder with optimizations for CTC models in terms of both computation time and memory usage. By applying the *blank* self-loop transitions in the decoder, we can avoid adding them explicitly as arcs in the decoder graph. Furthermore, the dynamic expansion of HMM state sequences used in our generic FST-based decoder can be removed, which allows for a more compact search space in memory and a simpler search hypothesis expansion procedure.

	Communication WER	Names WER	RT50
No Contacts	13.7	70.3	0.14
2 Contacts	9.0	30.0	-
+ biasing	-	12.8	-
50 Contacts	9.2	38.2	-
+ biasing	-	17.7	0.17

Table 3. Impact of contact injection and biasing on WER and latency.

Component	Size
Acoustic Model	3.0 MB
Decoder Graph	8.3 MB
Rescoring LM	6.8 MB
G2P Model	497 KB
Text Normalizers	1.1 MB
Endpointer	22 KB
Personalization Components	2.5 KB
Total	20.3 MB

Table 4. Size of various components in the overall system architecture.

6. PERSONALIZATION

Our final set of experiments highlight the advantages of integrating personal information into the language model. These experiments are aimed at determining the impact of incorporating device-specific information (e.g., the user's list of contact names) on the word error rate for individual users. We experiment with two test sets related to contact name recognition. The first is the 8K utterance Communication test set described in Section 4, containing contact names in the context of messages, e.g., "Text Jacob, . . .". The second set consists of 1.2K utterances containing *only* contact names. This second set is representative of the utterances that might follow a text-to-speech (TTS) prompt such as: "Which Jacob?" or perhaps a more general prompt such as "Who would you like to email?". The number of candidate contacts injected will depend on whether the TTS prompt is requesting disambiguation or just any name from the contact list. In either context, we can perform the additional step of using on-the-fly rescoring as in [13] to bias the language model towards recognizing only these contact names.

Given the lexical limits of the language model described above, it is unlikely that the recognizer will be able to handle the long tail of contact names as is. This motivates the incorporation of dynamic classes into our language model. In the general spirit of class-based LMs, and following the work of Aleksic et. al. [12] we annotate our training data with a special *$CONTACTS* symbol in place of contact names and train a language model that includes this placeholder token. At run-time we inject a small FST representing the user's personal contacts into the decoder graph at these locations. It should be noted that this is particularly simple in our system as our AM uses context-independent phonemes.

In order to generate pronunciations for contacts we train a LSTM-based grapheme-to-phoneme (G2P) model on human transcribed word-pronunciation pairs. The G2P problem is treated as a sequence transcription task as described in [25]. The LSTM-G2P system consists of four LSTM layers with 64 cells in each layer, and is trained to optimize the CTC objective function. The LSTM-G2P performs better in terms of word accuracy compared to traditional joint-sequence models represented as finite state transducers (FSTs) (a detailed comparison can be found in [25]). More importantly, the LSTM-G2P is considerably smaller in size compared to the FST implementation, 500 KB vs. 70 MB, making it an ideal solution for on-device pronunciation generation.

Table 3 summarizes our results on the two contact test sets. For each utterance recognized, N contacts are injected into the decoder graph. If the transcript does indeed contain a contact name, one of these N is the correct contact. For the set containing only contact names, we additionaly evaluate performance obtained using on-the-fly biasing [13] towards contact names.

Unsurprisingly, adding in personal contact names has a significant impact on WER, since many of the terms in these test sets are out-of-vocabulary items. In contexts when a single contact name is the expected user-response, these results indicate that biasing recognition towards the unigram *$CONTACTS* can yield dramatic improvements, especially if the set of candidate names can be whittled down to just two, as is often the case when disambiguating between contacts ("Do you mean John Smith or John Snow?"). While in practice one can often precompute these graphs, we also show here that median RT factors are not significantly affected even when 50 pronunciations are compiled and injected on-the-fly in the system.

7. SYSTEM FOOTPRINT

We present the sizes of the various components in our overall system architecture in Table 4. Using a combination of SVD-based compression and quantization, along with a compact first-pass decoding strategy and on-the-fly rescoring with a larger LM, we can build a system that is about 20.3 MB in size, without compromising accuracy or latency.

8. CONCLUSION

We presented our design of a compact large vocabulary speech recognition system that can run efficiently on mobile devices, accurately and with low latency. This is achieved by using a CTC-based LSTM acoustic model which predicts context-independent phones and is compressed to *a tenth of its original size* using a combination of SVD-based compression [4, 5] and quantization.

In order to support the domains of both open-ended dictation and voice commands in a single language model we use a form of Bayesian interpolation. Language model personalization is achieved through a combination of vocabulary injection and on-the-fly language model biasing [12, 13].

For efficient decoding, we use a on-the-fly rescoring strategy following [1] with additional optimizations for CTC models which reduce computation and memory usage. The combination of these techniques allows us to build a system which runs $7\times$ faster than real-time on a Nexus 5, with a total system footprint of 20.3 MB.

9. ACKNOWLEDGEMENTS

The authors would like to thank our colleagues: Johan Schalkwyk, Chris Thornton, Petar Aleksic, and Peter Brinkmann, for helpful research discussions and support for the implementation.

10. REFERENCES

[1] Xin Lei, Andrew Senior, Alexander Gruenstein, and Jeffrey Sorensen, "Accurate and compact large vocabulary speech recognition on mobile devices.," in *INTERSPEECH*. 2013, pp. 662–665, ISCA.

[2] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006, pp. 369–376.

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

[4] Jian Xue, Jinyu Li, and Yifan Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *INTERSPEECH*, 2013, pp. 2365–2369.

[5] Rohit Prabhavalkar, Ouais Alsharif, Antoine Bruguier, and Ian McGraw, "On the compression of recurrent neural networks with an application to LVCSR acoustic modeling for embedded speech recognition," in *ICASSP*. 2016, IEEE.

[6] Yu-hsin Chen, Ignacio Lopez-Moreno, Tara N. Sainath, Mirkó Visontai, Raziel Alvarez, and Carolina Parada, "Locally-connected and convolutional neural networks for small footprint speaker recognition," in *INTERSPEECH*. 2015, pp. 1136–1140, ISCA.

[7] Preetum Nakkiran, Raziel Alvarez, Rohit Prabhavalkar, and Carolina Parada, "Compressing deep neural networks using a rank-constrained topology," in *INTERSPEECH*. 2015, pp. 1473–1477, ISCA.

[8] Vikas Sindhwani, Tara N. Sainath, and Sanjiv Kumar, "Structured transforms for small-footprint deep learning," in *NIPS (to appear)*, 2015.

[9] Tara N. Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *ICASSP*. 2013, pp. 6655–6659, IEEE.

[10] Yongqiang Wang, Jinyu Li, and Yifan Gong, "Small-footprint high-performance deep neural network-based speech recognition using split-VQ," in *ICASSP*. 2015, pp. 4984–4988, IEEE.

[11] William Chan, Nan Rosemary Ke, and Ian Lane, "Transferring knowledge from a RNN to a DNN," in *INTERSPEECH*. 2015, ISCA.

[12] Petar Aleksic, Cyril Allauzen, David Elson, Aleksandar Kracun, Diego Melendo Casado, and Pedro J. Moreno, "Improved recognition of contact names in voice commands," in *ICASSP*, 2015, pp. 5172–5175.

[13] Keith Hall, Eunjoon Cho, Cyril Allauzen, Françoise Beaufays, Noah Coccaro, Kaisuke Nakajima, Michael Riley, Brian Roark, David Rybach, and Linda Zhang, "Composition-based on-the-fly rescoring for salient n-gram biasing," in *INTERSPEECH*. 2015, ISCA.

[14] Haşim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *INTERSPEECH*. 2014, pp. 338–342, ISCA.

[15] Haşim Sak, Andrew Senior, Kanishka Rao, Ozan İrsoy, Alex Graves, Françoise Beaufays, and Johan Schalkwyk, "Learning acoustic frame labeling for speech recognition with recurrent neural networks," in *ICASSP*, 2015, pp. 4280–4284.

[16] Haşim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," in *INTERSPEECH*. 2015, pp. 1468–1472, ISCA.

[17] Jeffrey Sorensen and Cyril Allauzen, "Unary data structures for language models," in *INTERSPEECH*. 2011, ISCA.

[18] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng, "Large scale distributed deep networks," in *NIPS*, 2012, pp. 1223–1231.

[19] Haşim Sak, Oriol Vinyals, Georg Heigold, Andrew Senior, Erik McDermott, Rajat Monga, and Mark Mao, "Sequence discriminative distributed training of long short-term memory recurrent neural networks," in *INTERSPEECH*, 2014, pp. 1209–1213.

[20] Alan C. Bovik, *Handbook of Image and Video Processing (Communications, Networking and Multimedia)*, Academic Press, Inc., Orlando, FL, USA, 2005.

[21] Allen Gersho and Robert M. Gray, *Vector Quantization and Signal Compression*, Springer US, 1992.

[22] Vincent Vanhoucke, Andrew Senior, and Mark Mao, "Improving the speed of neural networks on cpus," in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[23] Cyril Allauzen and Michael Riley, "Bayesian language model interpolation for mobile speech input," in *INTERSPEECH*, 2011, pp. 1429–1432.

[24] Mehryar Mohri, Fernando Pereira, and Michael Riley, "Speech recognition with weighted finite-state transducers," in *Handbook of Speech Processing*, Jacob Benesty, M. Sondhi, and Yiteng Huang, Eds., chapter 28, pp. 559–582. Springer, 2008.

[25] Kanishka Rao, Fuchun Peng, Haşim Sak, and Françoise Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *ICASSP*, 2015.