Interspeech 2018
2-6 September 2018, Hyderabad

Compression of End-to-End Models

Ruoming Pang, Tara N. Sainath*, Rohit Prabhavalkar, Suyog Gupta, Yonghui Wu, Shuyuan Zhang, Chung-cheng Chiu*

Google Inc., U.S.A

{rpang,tsainath,prabhavalkar,suyoggupta,yonghui,syzhang,chungchengc}@google.com

Abstract

End-to-end models, which directly output text given speech using a single neural network, have been shown to be competitive with conventional speech recognition models containing separate acoustic, pronunciation, and language model components. Such models do not require additional resources for decoding and are typically much smaller than conventional models. This makes them particularly attractive in the context of on-device speech recognition where both small memory footprint and low power consumption are critical. This work explores the problem of compressing end-to-end models with the goal of satisfying device constraints without sacrificing model accuracy. We evaluate matrix factorization, knowledge distillation, and parameter sparsity to determine the most effective methods given constraints such as a fixed parameter budget.

1. Introduction

There has been growing interest in developing end-to-end models for the task of automatic speech recognition (ASR). Such models fold the different modules of a conventional speech recognition system – the acoustic model (AM), the pronunciation model (PM), and the language model (LM) – into a single neural network. Examples of such approaches include connectionist temporal classification (CTC) [1] with word targets [2], the recurrent neural network transducer (RNN-T) [3, 4], the recurrent neural aligner (RNA) [5], and attention-based models such as listen attend and spell (LAS) [6, 7, 8, 9].

End-to-end approaches to ASR have a number of advantages over conventional ASR systems. One of the main advantages lies in the simplicity of the overall system. Since these models do not require external hand-designed resources, such as pronunciation lexica, models can be built and deployed with relative ease. Moreover, there is an important advantage of these techniques which makes them particularly attractive for deployment on embedded devices [10] which are limited in terms of the available memory: these models can perform comparably with conventional ASR even with only a small fraction of the total number of parameters required in a conventional system [4, 9]. While end-to-end models certainly offer size advantages over conventional models, the best models are still too large and computationally expensive to run on embedded devices. Further reduction of the number of effective parameters (and thus computation) can serve to reduce both processing time and power consumption. With this goal in mind, in the present work, we explore a variety of approaches to compress end-to-end models.

Neural network compression has been well explored in the literature. There are numerous proposals including distillation [11, 12], low-rank matrix factorizaton [13], and model sparsity [14]. To the best of our knowledge, however, these techniques have not been explored in the context of end-to-end ASR models. We therefore examine the impact of these techniques on compressing a specific end-to-end model: the attention-based encoder-decoder architecture (LAS) [9].

We begin by exploring distillation as a compression method. Most previous works in ASR (e.g., [15, 16] inter alia) have explored the use of distillation at the 'token level': the frame-level posterior distribution is 'transferred' from the teacher to the student. However, since end-to-end models are optimized with a sequence-level criterion, it makes sense to apply a sequence-level distillation objective. Motivated by recent end-to-end distillation work in machine translation [17], and ASR [18, 19], in this work we explore applying a sequence-level distillation objective to train end-to-end speech models. Next, we leverage low-rank matrix factorization to compress models. While these techniques have been explored previously, our goal in this work is to analyze the importance of this factorization when it is applied to the encoder network, versus the decoder network, or both. Finally, we explore applying sparsity [14] to the parameters in the model, which we find to be effective to reduce the effective number of parameters in the model.

Our goal is to answer the following research question: given a specific parameter budget, what is the best approach to compress the attention-based encoder-decoder model? We attempt to answer this question by comparing the above-mentioned techniques individually, as well as when applied jointly. We consider three different model sizes: 16M, 32M, and 57M parameters, which are chosen to study the effectiveness of the techniques over a range of model sizes. Our experiments are conducted on a $\sim$15,000 hour Voice Search task. We find that for small models, distillation gives a 10% improvement in WER. For larger models, using a combination of distillation and factorization gives $\sim$8% relative improvement. Random sparsity is the best for all models, giving more than 10% relative WER reduction. Finally, when we consider the problem of having a fixed number of parameters per layer, we find that factorization of both the encoder and the decoder, results in a 15% improvement in WER at a comparable model size.

2. LAS Model

The LAS model used for all experiments in this paper uses the architecture from [9] and is described as follows. The *listener*, also known as the encoder network, is akin to an acoustic model in a conventional ASR system. This network, takes the input features, $\mathbf{x}$, and maps them to a higher-order feature representation $\mathbf{h}^{\text{enc}}$ through a multi-layer LSTM. The output of the encoder is passsed to an *attender*, which acts as an alignment mechanism, determining which encoder features in $\mathbf{h}^{\text{enc}}$ should be attended to in order to predict the next output symbol, y_i. The output of the attention module is passed to the *speller*,

* Equal contribution.

10.21437/Interspeech.2018-1025

i.e., the decoder network, which is akin to the pronunciation and language models. The decoder, also a multi-layer LSTM, takes the attention context and the embedding of the previous prediction, y_{i-1}, in order to produce a probability distribution, $P(y_i|y_{i-1}, \ldots, y_0, \mathbf{x})$, over the current sub-word unit, y_i, given the previous units, $y_{i-1}, \ldots, y_0$, and input, $\mathbf{x}$.

3. Compression Techniques

3.1. Knoweldge Distillation

Knowledge distillation encompasses a class of methods in which a *student* network is trained to mimic the behavior of the *teacher* network [11, 12]. These two networks differ in structure: e.g., the student network might contain fewer parameters, or might belong to a different model class than the teacher. When the student network contains fewer parameters than the teacher, it is empirically found to be more effective to train the student network using labels derived from the teacher, rather than training the network from scratch using ground-truth labels. This process can be seen as compressing the larger teacher network into a more compact student network.

Many previous ASR works [15, 16] implement distillation by training the frame-level posterior distribution of the student to be close to that of the teacher by minimizing the Kullback-Leibler (KL) divergence between the two distributions. This is equivalent to standard cross-entropy training, where the target distribution corresponds to the teachers output:

$$\mathcal{L}_{KD}(\theta, \theta_T) = -\sum_{c=1}^{|C|} Q(y = c|x; \theta_T) \log P(y = c|x; \theta) \quad (1)$$

where, θ_T corresponds to the parameters in the teacher model (held fixed) and $Q(y = c|x; \theta_T)$ represents the teacher distribution; θ represents the parameters to be learned in the student model, and $P(y = c|x; \theta)$ represents the student distritbution. The criterion in Equation 1 transfers the frame-level distribution at each frame from the teacher to the student. However an end-to-end model defines a probability distribution over a *sequence of tokens* $P(\mathbf{y}|\mathbf{x}) = P(y_N, \cdots, y_1|\mathbf{x})$, which motivates a modification to the basic approach that we describe in subsequent sections.

3.1.1. Sequence-level Distillation

The simplest way to apply Equation 1 in the context of end-to-end models is to compute teacher logits for each step in the target sequence by feeding back previous ground-truth labels, and to use these as the target distribution for the student model. Specifically, for an utterance with N labels in the ground-truth target sequence $y_1^*, \cdots, y_N^*$, we minimize the following:

$$\mathcal{L}_{\text{TRUTH-KD}}(\theta, \theta_T) = -\sum_{n=1}^{N} \sum_{c=1}^{|C|} Q(y_n = c|y_{n-1}^*, \cdots, y_0^*, x; \theta_T) \times \log P(y_n = c|y_{n-1}^*, \cdots, y_0^*, x; \theta) \quad (2)$$

where, $y_0^* = \langle \text{sos} \rangle$ is a special label which indicates the start of the sentence and is fed in to the model at the first step.

However, since we are optimizing for a sequence-level end-to-end task, it might be better to mimic the teacher distribution at the *sequence level*. Motivated by recent work on sequence-level distillation in the context of end-to-end machine translation [17], in this work, as an alternative to frame-level distillation, we compute a set of hypotheses from the teacher model using beam-search [20] and distill towards these hypotheses. Specifically, we represent by $\mathbf{y}^i \in \mathcal{H}$, for $1 \leq i \leq M$, the set of hypotheses decoded from the teacher model using beam search, ordered by decreasing probability, so that: $Q(\mathbf{y^i}|\mathbf{x}) > Q(\mathbf{y^j}|\mathbf{x})$ for $i < j$. We assume that the hypothesis, $\mathbf{y}^i$ consists of N_i targets: $\mathbf{y}^i = [\mathbf{y}_1^i, \cdots, \mathbf{y}_{N_i}^i]$. For each hypothesis $\mathbf{y}^i$, we can compute distillation loss by teacher forcing on $\mathbf{y}^i$:

$$\mathcal{L}_{\text{HYP-KD}}(\theta, \theta_T|\mathbf{y}^i) = -\sum_{n=1}^{N_n} \sum_{c=1}^{|C|} Q(y_n = c|\mathbf{y}_{n-1}^i, \cdots, \mathbf{y}_0^i, \mathbf{x}; \theta_T) \times \log P(y_n = c|\mathbf{y}_{n-1}^i, \cdots, \mathbf{y}_0^i, \mathbf{x}; \theta) \quad (3)$$

Our first approach is simply distillation with the teacher's top hypothesis (i.e., $\mathbf{y}^1$):

$$\mathcal{L}_{\text{TOP-KD}}(\theta, \theta_T) = \mathcal{L}_{\text{HYP-KD}}(\theta, \theta_T|\mathbf{y}^1) \quad (4)$$

More generally, we can distill towards all hypotheses in the beam, rather than the top one. If we denote by α, the total probability of all hypotheses in the beam search decoding: $\alpha = \sum_{\mathbf{y}^i \in \mathcal{H}} Q(\mathbf{y}^i|\mathbf{x}; \theta_T)$, and $Q'(\mathbf{y}^i|\mathbf{x}; \theta_T) = \frac{Q(\mathbf{y}^i|\mathbf{x}; \theta_T)}{\alpha}$, then we can optimize the sequence-level KL divergence between the teacher and the student as follows:

$$\mathcal{L}_{\text{BEAM-KD}}(\theta, \theta_T) = -\sum_{\mathbf{y}^i \in \mathcal{H}} Q'(\mathbf{y}^i|\mathbf{x}; \theta_T) \times \mathcal{L}_{\text{HYP-KD}}(\theta, \theta_T|\mathbf{y}^i) \quad (5)$$

3.1.2. Pre-training the Student Network

In the distillation formulations in Equation 4 and 5, the student is never trained directly with the ground-truth targets. Thus, if the teacher makes a mistake this can be carried over to the student. Previous work has shown that distillation performance can be improved by including a cross-entropy loss between the student and ground-truth targets in the objective function [17]. As an alternative to employing a dual objective function, which would require tuning the mixing weight between the two objectives, in this work we consider initializing the student from a CE-trained model trained partially to convergence. The choice of only using a partially trained model is based on the intuition that the pre-training is only required to get a reasonable initialization which can then be fine-tuned during the distillation process, but is not completely influenced by the initial CE-training process, similar in spirt to discriminative pre-training [21].

3.2. Factorization

In this section, we describe our approach for applying a low-rank matrix factorization to LSTM layers in the encoder, which follows the work in [22] for conventional LSTM CE-trained models. A standard LSTM [23] layer in the network, can be described by the following equations, for all time-steps $t = 1, \cdots, T$:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + b_i) \quad (6)$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1} + b_f) \quad (7)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \quad (8)$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + b_o) \quad (9)$$

$$m_t = o_t \odot h(c_t) \quad (10)$$

where i_t, f_t, c_t and o_t denote the input, forget, memory cell and output gate activations at time t; x_t is the input at step t; c_t is the cell state; and m_t is the output of the LSTM layer. The various

matrices W are the parameters of the model, e.g., W_{ix} is the weight matrix from the input gate to the input. Finally, $\odot$ represents an element-wise dot product, $\sigma(\cdot)$ is the sigmoid function, and g and h are suitable activation functions such as $\tanh$. We replace the standard output computation in Equation 10 with a low-rank factorized representation:

$$m_t = W_{mc}(o_t \odot h(c_t)) \tag{11}$$

where, W_{mc}, of shape $|m| \times |c|$, projects the cell output to a lower dimension than the cell state: $|m| << |c|$. Typically, $|m|$ is chosen experimentally [24].

In this work, we utilize multiple stacked LSTM layers in both the encoder network, as well as in the decoder network. Thus, we investigate both an appropriate choice of $|m|$ and $|c|$ as well as examining the effectiveness of such a factorization in the encoder network and the decoder network to examine how it impacts performance at a given parameter budget.

3.3. Sparsity

Pruning neural network models to remove the less salient connections in the network has been demonstrated as an effective method for model compression [14, 25, 26]. Model pruning seeks to introduce a sparse structure into the various weight matrices in the network, thus reducing the model size by virtue of sparse matrix storage. We follow the gradual pruning approach [14] and prune the LSTM layers of the encoder and the decoder during training using the following sparsity function:

$$s_t = s_f \left(1 - \left(1 - \frac{t - t_0}{t_f - t_0}\right)^3\right) \tag{12}$$

Pruning starts at training step t_0, terminating at step t_f when the target sparsity s_f is achieved. At training step $t \in [t_0, t_f]$, the smallest magnitude weights are masked to zero to achieve the sparsity level s_t. Note that this method of model pruning induces unstructured sparsity in the weight matrices, i.e., the distribution of the zero-valued weights is unconstrained.

4. Experimental Details

Our experiments are conducted on a $\sim$15,000 hour training set consisting of 22 million English utterances. The training utterances are anonymized and hand-transcribed, and are representative of Google's voice search traffic. Multi-style training (MTR) data are created by artificially corrupting the clean utterances using a room simulator, adding varying degrees of noise and reverberation with an average SNR of 12dB [27]. The noise sources are drawn from YouTube and daily life noisy environmental recordings. We report results on a set of $\sim$14,800 anonymized, hand-transcribed Voice Search utterances extracted from Google traffic.

All experiments use 80-dimensional log-mel features, computed with a 25-ms window and shifted every 10ms. Similar to [28, 29], at the current frame, t, these features are stacked with 2 frames to the left and downsampled to a 30ms frame rate. The encoder network architecture consists of 5 unidirectional LSTM layers [23], with the size as specified in the Section 5. Following our previous work [9], all models in our experiments employ multi-head attention [30] with four attention heads. The decoder network is a 2 layer LSTM with the size as specified in Section 5. No external LM is used in this work. All networks are trained to predict 4,096 word pieces [31] which are derived using a large corpora of text transcripts.

Unless specified otherwise, all neural networks are trained to optimize the standard cross-entropy criterion, using synchronous stochastic gradient descent (SGD) optimization [32] with Adam [33] and are trained using TensorFlow [34]. We use synchronous replicated training with 16 P100 GPUs and per-GPU batch size of 128. Models are trained to >200K steps, at a learning rate of 8e-4 with linear warm-up and exponential decay starting at 50K steps. Following [9], we apply label smoothing [35] to all factorization and sparsity models, as well as while training the teacher models, but we do not apply minimum word error rate (MWER) training [36] or scheduled sampling [37], which are left as future work.

5. Results

5.1. Distillation Exploration

Our first set of experiments are aimed at understanding the behavior of the various proposed distillation objectives described in Section 3. For this experiment, our teacher model contains 110M parameters (5x1400 encoder, 2x1024 decoder) while the student is a 16M parameter model (5x400 encoder, 2x512 decoder). As can be seen in Table 1, distilling using the ground truth labels ($\mathcal{L}_{\text{TRUTH-KD}}(\theta, \theta_T)$) results in a WER of 10.6%, a small gain over using no distillation at all (11.0%). Distilling towards the top hypothesis obtained using beam search ($\mathcal{L}_{\text{TOP-KD}}(\theta, \theta_T)$) improves performance slightly to 10.5%, but the improvement is small; distilling towards all hypotheses in the beam ($\mathcal{L}_{BEAM-KD}(\theta, \theta_T)$) performs comparably to distilling towards the top hypothesis. We note that performance did not appear to be sensitive to the beam width for $M = 2, 4,$ or 8.

Next, we explore distillation towards the top hypothesis after beam search decoding, where we initialize the student model from an early checkpoint of the model trained with ground truth targets (i.e., the model which achieved 11.0% in Table 1), which improves the WER to 10.0%. Overall, with distillation we can achieve a WER of 10.0%, a 9% relative improvement compared to a model of the same size trained without distillation (11.0%). For the remainder of this paper, all distillation results are reported by first initializing the student network with a CE-trained model, and distilling towards ground-truth targets (which does not necessitate beam search decoding and is thus much faster).

Table 1: *WER Results With Distillation.*

Distillation From	WER
no distillation	11.0
ground truth	10.6
top beam	10.5
full beam	10.5
top beam + student init.	10.0

5.2. Matrix Factorization

Most of the parameters for the models considered in this work lie in the encoder network, rather than in the decoder network. Therefore, in these initial experiments, factorization is only applied to the encoder layers. In order to study the tradeoff between encoder versus the decoder size, we vary the size of the projection layer in the encoder, and the number of LSTM cells in the decoder, but keep the overall model size within a fixed budget. The results in Table 2 show that under the same parameter count constraint, applying factorization on encoder improves model quality significantly for all model sizes except for

the model with 16M parameters. Also, for all model configurations, different encoder/decoder sizes perform comparably so long as the total number of parameters in the model is similar.

Table 2: *WER Results With Factorization.*

Encoder Size	Proj Size	Decoder Size	Total Params	WER (%)
400	-	512	16M	11.0
512	128	600	16M	11.3
400	256	660	16M	11.3
660	-	660	32M	8.6
1024	256	660	32M	8.1
800	256	840	32M	8.2
700	-	1024	57M	7.9
1400	256	1024	57M	7.3
1024	256	1300	57M	7.4

5.3. Combining Compression Techniques

Now that we have explored how distillation and factorization behave individually, in this section, we combine these techniques and explore which combination of techniques works best for a given model size and layer size. We also compare these approaches to sparsity-based compression.

5.3.1. Comparison on Model Size Constraint

Table 3 compares WER for different compression techniques, across three different model sizes. In the sparsity experiments, we prune weights from the baseline 110M-parameter model and the model sizes are computed as the number of non-zero parameters. For a model trained to achieve s_t sparsity, the compression ratio is estimated using the following:

$$\eta = \frac{1}{1 - s_t + \frac{1}{32}} \tag{13}$$

In the equation above, the factor $\frac{1}{32}$ represents the overhead of sparse matrix storage. We assume a bit-mask sparse matrix representation which requires 1 bit per matrix element indicating whether the element is nonzero, and a vector containing all the nonzero matrix elements stored as a 32-bit float. Also note that we apply sparsity to only the LSTM layers of the encoder and the decoder, and do not prune the attention parameters.

Random sparsity gives the largest improvement individually, but of course requires having specialized hardware. In the absence of this, for small models, distillation appears to be the best choice, while factorization appears to be the best choice for larger models. Finally, for 32M and 57M parameter models, combining factorization with distillation improves performance over using either method individually. However, incorporating sparsity in addition to these does not seem to be better than applying sparsity alone.

Table 3: *WER with Compression Techniques*

Method	16M	32M	57M	110M
Base	11.0	8.6	7.9	6.5
Factorization (F)	11.3	8.1	7.3	-
Distillation (D)	10.0	8.1	7.5	-
Sparsity (S)	**8.0**	**6.7**	**6.4**	-
F + D	-	7.9	7.2	-
F + D + S	-	**7.2**	**7.0**	-

5.3.2. Comparison on Layer Size Constraints

Hardware accelerators for embedded devices usually have a small amount of SRAM. Fitting all parameters of a layer into SRAM would make computation more efficient as the computation can be batched across timesteps. In this section we build on the conclusions drawn from our experiments, and repeat our evaluation with a constraint of $4M$ parameters per-layer, thus simulating performance given some SRAM. Model evaluated in this section are trained on 8×8 Tensor Processing Units (TPU) slices with global batch size 4096. Since the sparsity operation is not supported on the TPU, we emit those experiments here.

Our base model, `E1`, with 4M parameters per layer has 5 encoder layers and 2 decoder layers ('5+2'). Each encoder layer has 700 hidden units and each decoder layer has 512 units. It has a total of 29M parameters (including attention and softmax layers) and obtains a WER of 8.6%. In the factorization models we apply projection to both encoder and decoder layers so that every layer has no more 4M parameters. Each encoder layer has 1400 hidden units and projection size of 256 and each decoder layer has 1024 hidden units and the same projection size. As can be seen in Table 4, we find that both distillation (`E2`) and factorization (`E3`) improve WER. Applied together (`E4`), they improve WER to 7.2%, with most of the improvements in this case coming from factorization.

Next, we evaluated a model 2 times larger with deeper stacks: 10 encoder layers and 4 decoder layers, retaining the constraint of 4M parameters per layer (`E5`). It has similar number of parameters as a model with wider but shallower stacks (8M parameters per layer) (`E6`) and comparable WER at 7.1% to the model with factorization and distillation, which is 2 times smaller (`E4`).

In contrast, our 'large' model has 9-16M params per layer and WER of 6.4% (`E7`). Applying factorization on the deep model (`E8`), however, improves its WER to 6.1%, which is even better than our 'large' model `E7`. We have yet been able to find a large model with a better WER, and are thus unable to evaluate distillation on this deep factorized model.

Table 4: *WER, Limited by Parameters Per Layer. Unless otherwise stated, there are 5 encoder layers and 2 decoder layers and each layer has no more than 4M parameters. '10+4' means 10 encoder layers and 4 decoder layers.*

Exp-ID	Set-up	Total # Params	WER
`E1`	Base	29M	8.6
`E2`	Distillation	29M	8.3
`E3`	Factorization	28M	7.3
`E4`	F + D	28M	7.2
`E5`	Deep (10+4)	58M	7.1
`E6`	Wide (8M params/layer)	58M	7.1
`E7`	Large (<16M params/layer)	110M	6.4
`E8`	Deep + Factorization (10+4)	53M	**6.1**

6. Conclusions

In this work we empirically evaluate various techniques to compress end-to-end ASR models for deployment on embedded devices. Our experiments show that distillation, factorization, and sparsity are effective under different scenarios and that the combination of these techiniques results in larger improvements than those obtained using any single technique alone.

7. References

[1] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labeling Unsegmented Sequence Data with Recurrent Neural Networks," in *Proc. ICML*, 2006.

[2] H. Soltau, H. Liao, and H. Sak, "Neural Speech Recognizer: Acoustic-to-Word LSTM Model for Large Vocabulary Speech Recognition," in *Proc. Interspeech*, 2017.

[3] A. Graves, "Sequence Transduction with Recurrent Neural Networks," *CoRR*, vol. abs/1211.3711, 2012.

[4] K. Rao, H. Sak, and R. Prabhavalkar, "Exploring Architectures, Data and Units for Streaming End-to-End Speech Recognition with RNN-Transducer," in *Proc. ASRU*, 2017.

[5] H. Sak, M. Shannon, K. Rao, and F. Beaufays, "Recurrent Neural Aligner: An Encoder-Decoder Neural Network Model for Sequence to Sequence Mapping," in *Proc. Interspeech*, 2017.

[6] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, Attend and Spell," *CoRR*, vol. abs/1508.01211, 2015.

[7] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. NIPS*, 2015.

[8] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Proc. ICASSP*.

[9] C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani, "State-of-the-Art Speech Recognition With Sequence-to-Sequence Models," in *Proc. ICASSP*, 2018.

[10] I. McGraw, R. Prabhavalkar, R. Alvarez, M. G. Arenas, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Gruenstein, F. Beaufays, and C. Parada, "Personalized Speech Recognition on Mobile Devices," in *Proc. ICASSP*, 2016.

[11] C. Bucila, R. Caruana, and A. Niculescu-Mizi, "Model Compression," in *Knowledge Discovery and Data Mining (KDD)*, 2006.

[12] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *CoRR*, vol. abs/1503.02531, 2015.

[13] R. Prabhavalkar, O. Alsharif, A. Bruguier, and I. McGraw, "On the Compression of Recurrent Neural Networks with an Application to LVCSR acoustic modeling for Embedded Speech Recognition," in *Proc. ICASSP*, 2016.

[14] M. Zhu and S. Gupta, "To Prune, or not to Prune: Exploring the Efficacy of Pruning for Model Compression," in *Proc. NIPS Workshop on Machine Learning of Phones and other Consumer Devices*, 2017.

[15] A. Waters and Y. Chebotar, "Distilling knowledge from ensembles of neural networks for speech recognition," in *Proc. Interspeech*, 2016.

[16] L. Lu, M. Guo, and S. Renals, "Knowledge Distillation for Small-footprint Highway Networks," in *Proc. ICASSP*, 2017.

[17] Y. Kim and A. Rush, "Sequence-Level Knowledge Distillation," in *Proc. EMNLP*, 2016.

[18] J. H. M. Wong and M. J. F. Gales, "Sequence Student-Teacher Training of Deep Neural Networks," in *Proc. Interspeech*, 2016.

[19] R. Takashima, S. Li, and H. Kawai, "An investigatin of a knowledge distillation method for ctc acoustic models," in *in Proc. Interspeech*, 2018.

[20] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Of NIPS*, 2014.

[21] H. Soltau, H.-K. Kuo, L. Mangu, G. Saon, and T. Beran, "Neural Network Acoustic Models for the DARPA RATS Program," in *Proc. Interspeech*, 2013.

[22] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," in *Proc. Interspeech*, 2014.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov 1997.

[24] 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. ICASSP*, 2013.

[25] S. Narang, G. F. Diamos, S. Sengupta, and E. Elsen, "Exploring sparsity in recurrent neural networks," *CoRR*, vol. abs/1704.05119, 2017.

[26] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR*, vol. abs/1510.00149, 2015.

[27] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. N. Sainath, and M. Bacchiani, "Generated of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in google home," in *Proc. Interspeech*, 2017.

[28] G. Pundak and T. N. Sainath, "Lower Frame Rate Neural Network Acoustic Models," in *Proc. Interspeech*, 2016.

[29] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition," in *Proc. Interspeech*, 2015.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[31] M. Schuster and K. Nakajima, "Japanese and Korean voice search," *2012 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012.

[32] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large Scale Distributed Deep Networks," in *Proc. NIPS*, 2012.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of ICLR*, 2015.

[34] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," Available online: http://download.tensorflow.org/paper/whitepaper2015.pdf, 2015.

[35] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proc. CVPR*, 2016.

[36] R. Prabhavalkar, T. N. Sainath, Y. Wu, P. Nguyen, Z. Chen, C. C. Chiu, and A. Kannan, "Minimum Word Error Rate Training for Attention-based Sequence-to-sequence Models," in *Proc. ICASSP (accepted)*, 2018.

[37] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proc. NIPS*, 2015.