INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Robust Continuous On-device Personalization for Automatic Speech Recognition

Khe Chai Sim, Angad Chandorkar, Fan Gao*, Mason Chua*, Tsendsuren Munkhdalai*, Françoise Beaufays*

Google, USA

{khechai, angadc, fgao, aftran, tsendsuren, fsb}@google.com

Abstract

On-device personalization of an all-neural automatic speech recognition (ASR) model can be achieved efficiently by fine-tuning the last few layers of the model. This approach has been shown to be effective for adapting the model to recognize rare named entities using only a small amount of data. To reliably perform continuous on-device learning, it is important for the training process to be completely autonomous without manual intervention. Our simulation studies show that training over many rounds may eventually lead to a significant model drift if the personalized model is indiscriminately accepted at the end of each training round. It is important to have appropriate acceptance criteria in place to guard the model against drifting. Moreover, for storage efficiency, it is desirable to persist the model weights in quantized form. We found that quantizing and dequantizing the model weights in between training rounds can prevent the model from learning effectively. This issue can be circumvented by adding noise to the quantized weights at the start of each training round.

Index Terms: speech recognition, on-device learning, continual learning

1. Introduction

Voice is an important input modality for mobile devices, and it is now common to use speech-to-text (STT) or automatic speech recognition (ASR) in mobile applications, including voice search [1] and dictation [2]. Many speech-to-text services are deployed on a server and therefore require a network connection to send the audio data to the server for processing. The latency and reliability of a server-based speech-to-text service is greatly affected by the network bandwidth and quality. On the other hand, it has been shown that large vocabulary continuous speech recognition (LVCSR) [3] can be performed entirely on mobile devices to avoid the cost of sending audio data to a server [4]. This is made possible by using an all-neural model whose weights are quantized for computational efficiency, compactness and smallness of memory footprint.

Personalization of ASR models is an important way to improve performance for specific speakers. The ASR personalization approaches that have been investigated in the past can be divided into two broad categories. The first category of approaches attempts to personalize the language model (LM) component. For end-to-end models, such as listen-attend-spell [5] and recurrent neural network transducer (RNN-T) [6, 7], biasing [8, 9], shallow/deep fusion [10] and cold fusion [11] techniques have been proposed. However, these techniques do not take into consideration any acoustic mismatch between the base model and the target speakers. On the other hand, many speaker adaptation techniques have been explored in the past to handle acoustic mismatch for neural network acoustic models [12]. For example, structured parameterization methods are used to partition the model weights such that a portion of the weights can be reliably fine-tuned using a small amount of adaptation data [13, 14, 15, 16, 17]. Feature augmentation techniques, such as appending speaker i-vectors to the input features [18, 19, 20], have also been shown to be effective.

When the recognition is performed on mobile devices, personalization needs to happen on device as well to avoid sending user data and models to a server. This poses some challenges due to the limited resources available on mobile devices. Previously, we studied how the training memory footprint can be reduced by computing the gradients for a subset of layers at a time [21]. We also showed that ASR personalization can be achieved by fine-tuning the decoder of the RNN-T model to better recognize named entities [22]. To avoid overfitting, it is important to tune the learning rate and the number of training epochs. For server-side training, such as domain adaptation [23, 24] with only a few target domains, it is relatively straightforward to set up a suitable validation data set for hyper-parameter tuning and manually decide which models to deploy. However, for on-device learning, in order to prevent the personalized model from drifting into a bad state, it is important to have a mechanism that automatically determines whether a personalized model should be accepted without manual intervention.

The remainder of the paper is organized as follows. Section 2 outlines the major considerations for continuous on-device personalization and our proposed solutions. Section 3 describe the RNN-T speech recognition model and the data sets used for the simulation experiments. Section 4 presents the experimental results and section 5 presents some analyses.

2. Continuous On-device Personalization

Fig. 1 depicts the continuous on-device ASR personalization workflow, which is also described in [22]. According to this workflow, users will use voice input on their mobile device and optionally make corrections to the ASR transcripts. The audio data, along with the user-corrected transcripts, are stored in a training cache on the user's device. The cached data are then split into train and validation sets. When the device is idle and charging, the training data set is used to fine-tune the model for personalization while the validation data set is used to determine whether the personalized model should be accepted or rejected. If the personalized model is accepted, it will be used for subsequent voice input. This forms a closed loop for *continuous* learning. To achieve reliable continuous on-device learning, there are two important aspects to be considered, which

* Equal contribution.

http://dx.doi.org/10.21437/Interspeech.2021-318

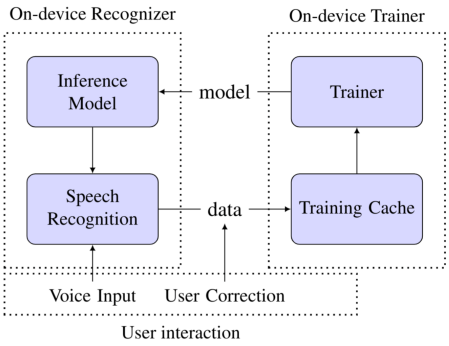


Figure 1: *Continuous on-device ASR personalization workflow.*

will described in the following two sub-sections.

2.1. Model Persistence

On-device ASR models are usually first trained with high precision (e.g. 32-bit floats) and then stored with lower precision (e.g. 8-bit integers) to achieve a compact model for inference [4]. In order to do so in a continuous learning setting, it is necessary to quantize and dequantize the model weights in between training rounds. We apply a symmetric quantization method [25] to the weight matrices. Specifically, a 32-bit float value for the ith weight, w_i, is quantized into a signed 8-bit integer, $\hat{w}_i$ as follows:

$$\hat{w}_i = \left\lfloor \frac{w_i \times 127}{\alpha} \right\rceil \tag{1}$$

where the scaling factor, α, is computed as the maximum absolute weight value for each weight matrix to yield $-127 \leq \hat{w}_i \leq 127$ and $\lfloor \cdot \rceil$ denotes the integer rounding operator.

We restore the weight, $\tilde{w}_i$, from the quantized integer value, $\hat{w}_i$ with

$$\tilde{w}_i = \frac{(\hat{w}_i + \sigma_i) \times \alpha}{127} \tag{2}$$

where σ_i is a uniform random noise in $[-0.5, 0.5]$. For training to work properly, we found it crucial to add a small amount of noise to the quantized integer value before scaling. Without adding noise, all the weights will be at the quantization point at the start of each training round. Weights whose updates are not big enough to cross the quantization boundaries will be quantized to their original values, which effectively unlearns those weights. We chose to add the noise prior to scaling so that the choice of the noise range is agnostic to the dynamic range of the weight values. Furthermore, we limit the noise range to be within $[-0.5, 0.5]$ so that a round trip of dequantization/quantization will not change the weights. We will compare the effect of using different noise scale in Section 4.2.

2.2. Acceptance Criteria

It is important for the on-device learning process to be completely autonomous. Therefore, a reliable mechanism is needed to automatically determine whether a personalized model should be accepted. This is crucial, especially for continuous learning, to ensure that the personalized model does not end up in a bad state due to bad training data or overfitting.

In this paper, the acceptance decision is determined by the *loss* and *word error rate* metrics on the validation set, with the reference labels provided by the user-corrected transcripts. A personalized model will be rejected if either metric increases due to personalization. However, given that the validation set is acquired from the users on their devices, its quality and quantity cannot be guaranteed. As a precaution, we also consider an additional acceptance requirement that the word error rate performance on a custom-written *regression* evaluation data set does not exceed a hand-optimized threshold. This data set comprises 10 hand-picked stock phrases that happen to be correctly recognized by the baseline model, like "the quick brown fox jumps over the lazy dogs", spoken by one of Google's user-facing text-to-speech voices.

3. Data Sets

3.1. Wiki-Names

The *Wiki-Names* corpus [22] was designed for named entity personalization. This data set contains read speech collected from 100 speakers. The text prompts used for the data collection were extracted from US English Wikipedia pages that contain named entities that are misrecognized by the baseline ASR model. Each speaker has 50 train, 10 dev and 10 test utterances that cover 5 named entities. There are on average 4.6 minutes of training data per speaker.

3.2. Librispeaker

The Wiki-Names data set has a relatively small amount of training data per speaker. In order to better study the model drift effect in a continuous learning setting, we also created a *Librispeaker* data set, which is a subset of the LibriSpeech data set [26] with 110 speakers selected from the `train-other-500` subset. We randomly select 160 utterances from each speaker for our simulation experiments, which are split into 140 train, 10 dev and 10 test utterances. Each speaker has about 25.6 minutes of training data.

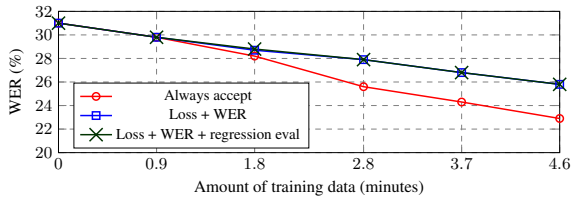
3.3. Librispeaker TTS

Finally, to simulate an even longer training horizon, we also created a *Librispeaker TTS* data set where the speech data are synthesized with text-to-speech using all the texts from the Librispeaker data set described above. We use 15 voices to simulate different speakers, each one speaking the same 1020 transcripts from the beginning of the corpus, with 10 dev and 10 test utterances held out from the beginning. The average amount of training data per voice is 138 minutes.

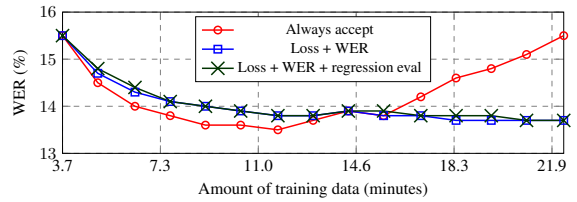
4. Experimental Results

We use an RNN-T [6, 7] model with long short-term memory (LSTM) [27] layers, similar to those described in [22]. There are 8 LSTM layers for the encoder and 2 for the decoder. 128-dimensional log Mel features are computed every 10 milliseconds. 4 consecutive feature vectors are stacked with a stride of 3 frames to yield a 512-dimensional input features every 30 milliseconds. There are 4096 output units that correspond to the word piece tokens.

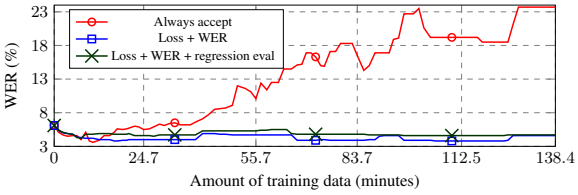
For the simulation experiments, we fine-tune only the joint layers, which amount to about 3.4 million parameters. We use Adafactor optimization [28] with a learning rate of 0.005 and updating clipping of 0.1 for stability. First-order momentum is turned off ($\beta_1 = 0$) to reduce training memory. All the models are trained and evaluated using TensorFlow [29]. The RNN-T



(a) *Wiki-Names*



(b) *Librispeaker*



(c) *Librispeaker TTS*

Figure 2: *Comparison of WER performance with increasing training rounds using different acceptance criteria.*

loss and gradients are computed using the algorithm described in [30]. On-device training is performed with 2 epochs per training round and a mini-batch size of 5.

4.1. Acceptance Criteria

We study the effectiveness of using the criteria described in Section 2.2 to accept or reject the personalized models. Figures 2a, 2b and 2c show how WER varies with increasing training rounds on the Wiki-Names, Librispeaker and Librispeaker TTS test sets, respectively. We compare three types of acceptance criteria: (1) always accept the personalized model; (2) accept conditioned on loss and word error rate metrics; and (3) accept based on the above and passing the regression evaluation. On Wiki-Names, the best performance improvements were achieved by always accepting the model. The acceptance criteria hurt performance because the personalized models are prematurely rejected before seeing enough training data to learn new named entities. However, on the Librispeaker and Librispeaker TTS data sets, after about 11 minutes of training data (6 or 10 training rounds, respectively), we observe a substantial WER increase with subsequent training rounds, which can be prevented by using the appropriate acceptance criteria. In our use case, the benefits of using the acceptance criteria on Librispeaker-like data outweigh its disadvantage on Wiki-Names-like data. In practice, unlike Wiki-Names data, the training examples on device will often not contain any useful information to learn from. This result also shows that the regression eval criterion does not add much additional benefit if the loss and WER criteria are available.

For on-device personalization, users might not correct all the ASR errors. To study the effect of noisy labels, we conducted experiments for the extreme scenario where the ASR

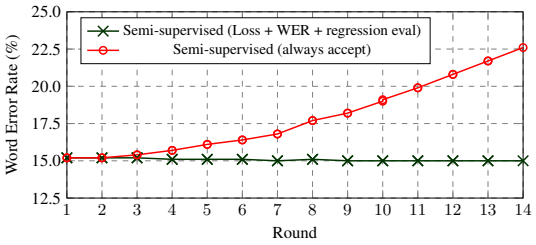


Figure 3: *Comparison of word error rate on the Librispeaker test set for semi-supervised training with and without acceptance criteria.*

Table 1: *LibriSpeech WER with different round sizes.*

Examples per round	10	20	30	40
Average duration (minutes)	1.8	3.7	5.5	7.3
WER (%)	12.4	12.5	12.8	12.7

transcripts are used as reference without user corrections. This is effectively performing semi-supervised learning. Due to the lack of user correction signal, and the fact that only the joint layer of the model is being updated, we expect model performance to stay the same at best, with plenty of opportunity for worsening. Fig. 3 shows that this worsening is significant but successfully prevented by our acceptance criteria.

We also studied how the WER performance varies with the frequency of making acceptance decisions by changing the number of examples per round. Table 1 shows that smaller number of examples per round (more frequent acceptance decision making) achieved better WER, but the difference is small.

4.2. Effect of Quantization

As described in Section 2.1, model weights are stored as quantized 8-bit integers and converted to 32-bit floats for training. In a continuous learning setting, the model weights are quantized and dequantized between training rounds. Table 2 shows the effect of (de)quantization on word error rate performance on the Wiki-Names test set. The WER of the baseline model is 30.9%. If the model weights are not quantized between training rounds, the WER performance improves to 25.2% after 5 training rounds. However, with quantization, the personalized model achieves no improvement over the baseline. The average relative weight change between the initial and personalized models was only 0.2% (compared to 5.6% for the case without quantization). This confirms our hypothesis that most of the adjustments made to the weights are effectively unlearned after quantization. This effect remains if we increase the learning rate or add uniform noise in the range of [-0.25, 0.25]. However, with a wider uniform noise range of [-0.5, 0.5], the resulting personalized model achieves 25.3% WER, which is very close to not using quantization.

5. Analysis of Results

5.1. Catastrophic Forgetting

Catastrophic forgetting [31] is a phenomenon by which a neural network forgets about the previous tasks when trained on multiple tasks sequentially. Its effect for ASR personalization was

Table 2: *Comparison of WER on Wiki-Names test set after 5 training rounds.*

Quantization	Uniform Noise Range	Learning Rate	WER (%)
Baseline	—	—	30.9
No	—	0.005	25.2
Yes	0.00	0.005	30.9
	0.00	0.010	29.6
	0.00	0.050	30.9
	[-0.25, 0.25]	0.005	30.9
	[-0.50, 0.50]	**0.005**	**25.3**

Table 3: *Comparing the catastrophic forgetting effect on WER with and without acceptance criteria.*

Training Set	Acceptance Criteria	WER (%) on Voice Search
Baseline	—	6.7
Wiki-Names	No	7.5
	Yes	**7.2**
Librispeaker	No	10.6
	Yes	**7.3**

studied in [22] and mitigated by elastic weight consolidation (EWC) in [32]. However, EWC requires additional statistics for the weights during training. This significantly increases memory usage, which is not good for on-device training. Acceptance criteria are an effective alternative, as shown in Table 3. The baseline model, which was trained with data from multiple domains including voice search [24], achieved 6.7% WER on the voice search test sets. Without any acceptance criterion ("always accept"), the forgetting effect is worse with a longer training horizon. With acceptance criteria ("loss + WER + regression eval"), the effect is suppressed substantially.

5.2. Acceptance Rate

Fig. 4 shows the rate of acceptance with increasing training rounds for three cases. For personalization on Wiki-Names, the acceptance rate remains relatively high with a slight downward trend over 5 training rounds. This could be explained by the fact that all the train, dev and test utterances contain overlapping difficult names and the likelihood of improving the dev set is high. For the Librispeaker data set, which lacks this characteristic, the decline in the acceptance rate over time is more pronounced. In the semi-supervised case, there is not much information to learn from (the reference labels were generated by the model itself), so the acceptance rate is consistently lower across all the training rounds. In all cases, the acceptance criteria are useful to prevent the model from drifting into a bad state due to overfitting and/or noisy training labels.

5.3. Per-speaker WER Improvements

Finally, we analyze the breakdown of the WER improvements per speaker. Fig. 5a and 5b show the absolute WER improvements for the 110 speakers on the Librispeaker test set using supervised and semi-supervised learning, respectively. Always accepting the personalized model caused 16.3% of the 110 speaker models to get worse after supervised learning and all of them to get worse after semi-supervised learning. Having the appropriate acceptance criteria in place successfully prevented all the speaker models from worsening for both supervised and semi-supervised training.

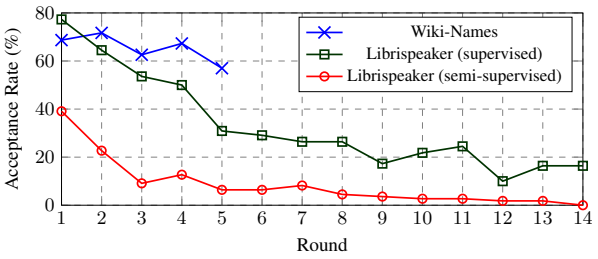
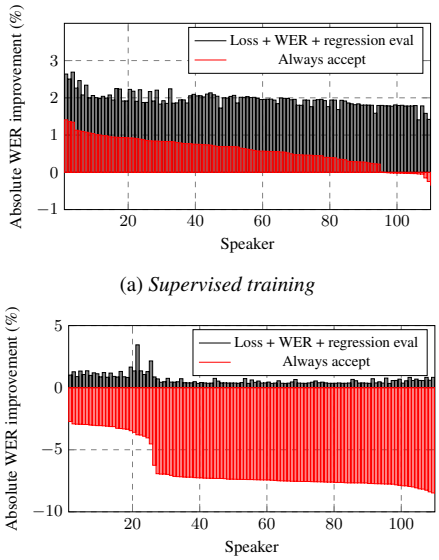


Figure 4: *Acceptance rates for different training data sets.*



(a) *Supervised training*

(b) *Semi-supervised training*

Figure 5: *Comparison of per-speaker absolute WER improvements on Librispeaker after 14 training rounds.*

6. Conclusions

We studied two important aspects of continuous on-device personalization for speech recognition. Firstly, we found that the necessary quantization of the model weights between training rounds causes unlearning of the finer-grained weight adjustments done by personalization. This issue can be mitigated by adding a small amount of random noise at the start of each training round to perturb the weights from the quantization point and encourage them to move across the quantization boundaries. Secondly, continuous learning over a long period of time could result in significant model drift if the personalized models are indiscriminately accepted without checks. Defining an appropriate set of acceptance criteria based on the loss and word error rate metrics on a validation set offers an effective safety net to stop the model from drifting into a bad state, suppresses the catastrophic forgetting effect, and improves the chances of achieving a better personalized model for users.

7. References

[1] 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*. Springer, 2010, pp. 61–90.

[2] J. Schalkwyk. (2019) An all-neural on-device speech recognizer. [Online]. Available: https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html

[3] S. Young, "A review of large-vocabulary continuous-speech," *IEEE signal processing magazine*, vol. 13, no. 5, p. 45, 1996.

[4] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6381–6385.

[5] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4960–4964.

[6] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[7] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 38, 03 2013.

[8] P. Aleksic, C. Allauzen, D. Elson, A. Kracun, D. M. Casado, and P. J. Moreno, "Improved recognition of contact names in voice commands," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5172–5175.

[9] K. Hall, E. Cho, C. Allauzen, F. Beaufays, N. Coccaro, K. Nakajima, M. Riley, B. Roark, D. Rybach, and L. Zhang, "Composition-based on-the-fly rescoring for salient n-gram biasing," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015, pp. 1418–1422.

[10] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," *arXiv preprint arXiv:1503.03535*, 2015.

[11] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold fusion: Training seq2seq models together with language models," *arXiv preprint arXiv:1708.06426*, 2017.

[12] K. C. Sim, Y. Qian, G. Mantena, L. Samarakoon, S. Kundu, and T. Tan, "Adaptation of deep neural network acoustic models for robust automatic speech recognition," in *New Era for Robust Speech Recognition*, S. Watanabe, M. Delcroix, F. Metze, and J. Hershey, Eds. Springer, 2017, ch. 9, pp. 219–243.

[13] B. Li and K. C. Sim, "Comparison of discriminative input and output transformations for speaker adaptation in the hybrid nn/hmm systems," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010, pp. 526–529.

[14] L. Samarakoon and K. C. Sim, "Factorized hidden layer adaptation for deep neural network based acoustic modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 12, pp. 2241–2250, 2016.

[15] T. Tan, Y. Qian, M. Yin, Y. Zhuang, and K. Yu, "Cluster adaptive training for deep neural network," in *Proc. ICASSP*. IEEE, 2015, pp. 4325–4329.

[16] 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. ICASSP*. IEEE, 2014, pp. 6359–6363.

[17] Y. Zhao, J. Li, and Y. Gong, "Low-rank plus diagonal adaptation for deep neural networks," in *Proc. ICASSP*. IEEE, 2016, pp. 5005–5009.

[18] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors." in *ASRU*, 2013, pp. 55–59.

[19] A. Senior and I. Lopez-Moreno, "Improving DNN speaker independence with i-vector inputs," in *Proc. ICASSP*. IEEE, 2014, pp. 225–229.

[20] L. Sarı, N. Moritz, T. Hori, and J. Le Roux, "Unsupervised speaker adaptation using attention-based speaker memory for end-to-end asr," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7384–7388.

[21] K. C. Sim, P. Zadrazil, and F. Beaufays, "An investigation into on-device personalization of end-to-end automatic speech recognition models," in *Interspeech*, 2019.

[22] K. C. Sim, F. Beaufays, A. Benard, D. Guliani, A. Kabel, N. Khare, T. Lucassen, P. Zadrazil, H. Zhang, L. Johnson, G. Motta, and L. Zhou, "Personalization of end-to-end speech recognition on mobile devices for named entities," in *ASRU*, 2019.

[23] K. C. Sim, A. Narayanan, A. Misra, A. Tripathi, G. Pundak, T. N. Sainath, P. Haghani, B. Li, and M. Bacchiani, "Domain adaptation using factorized hidden layer for robust automatic speech recognition," *Proc. Interspeech 2018*, pp. 892–896, 2018.

[24] A. Narayanan, A. Misra, K. C. Sim, G. Pundak, A. Tripathi, M. Elfeky, P. Haghani, T. Strohman, and M. Bacchiani, "Toward domain-invariant speech recognition via large scale training," *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 441–447, 2018.

[25] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2017.

[26] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5206–5210.

[27] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014, pp. 338–342.

[28] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," 2018.

[29] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016. [Online]. Available: http://arxiv.org/abs/1603.04467

[30] T. Bagby, K. Rao, and K. C. Sim, "Efficient implementation of recurrent neural network transducer in TensorFlow," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 506–512.

[31] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[32] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *CoRR*, vol. abs/1612.00796, 2016. [Online]. Available: http://arxiv.org/abs/1612.00796