Interspeech 2022
18-22 September 2022, Incheon, Korea

Incremental learning for RNN-Transducer based speech recognition models

Deepak Baby, Pasquale D'Alterio, Valentin Mendelev

Amazon Alexa

Abstract

This paper investigates an incremental learning framework for a real-world voice assistant employing RNN-Transducer based automatic speech recognition (ASR) model. Such a model needs to be regularly updated to keep up with changing distribution of customer requests. We demonstrate that a simple fine-tuning approach with a combination of old and new training data can be used to incrementally update the model spending only several hours of training time and without any degradation on old data. This paper explores multiple rounds of incremental updates on the ASR model with monthly training data. Results show that the proposed approach achieves 5-6% relative WER improvement over the models trained from scratch on the monthly evaluation datasets. In addition, we explore if it is possible to improve recognition of specific new words. We simulate multiple rounds of incremental updates with handful of training utterances per word (both real and synthetic) and show that the recognition of the new words improves dramatically but with a minor degradation on general data. Finally, we demonstrate that the observed degradation on general data can be mitigated by interleaving monthly updates with updates targeting specific words.

Index Terms: speech recognition, end-to-end models, RNN-T, incremental learning, targeted updates

1. Introduction

Traditional hybrid automatic speech recognition (ASR) systems consist of an acoustic model (AM), a language model (LM) and a pronunciation model (lexicon), all of which are trained independently. In contrast, end-to-end automatic speech recognition systems use neural networks to directly transduce an input sequence of acoustic features to an output sequence of tokens [1–4], without requiring the independent training of different components. In particular, recurrent neural network transducer (RNN-T) [1] has shown superior performance in such applications as streaming [5–7] and on-device ASR [8, 9].

While end-to-end model training process is simpler than that of a hybrid model, a large amount of transcribed audio data is needed to achieve competitive performance. Hence, a lot of computational resources are consumed and training time is usually high. On the other hand, in order to maintain model performance in a real-word voice assistant application, it needs to be updated frequently to capture the various shifts occurring over time in the distribution of the incoming requests (seasonal changes, new trending words, acoustics from new device types, etc.).

For hybrid ASR models this can be effectively addressed by updating the LM and the lexicon. Such updates require only textual data and phonetic knowledge and are usually sufficient enough to enable the system to recognize new words. Even though in theory end-to-end models could construct any word using word-pieces, words that were not present in the training set are often incorrectly recognized in practice [10, 11]. Since end-to-end models are trained in a single step, to recognize new words and to capture distribution shifts, the model is typically trained from scratch with the addition of newly available training data. However, frequently training end-to-end models from scratch is costly in terms of both time and computational resources.

One of the techniques used for faster model updates and reducing the training costs is incremental learning (IL) [12, 13], where an existing ASR model is incrementally updated as new training data become available. There exist prior works exploring IL for ASR, conducted in a hypothetical setup where the data changes acoustically (reverberation, noise, accent) or semantically (new domain, read speech vs conversational) [14–16]. However, for applications such as voice assistants, such drastic changes in acoustics or semantics are not expected. This paper investigates a real-world scenario where an RNN-T-based streaming ASR model is incrementally updated to ingest newly available training data based on real production traffic. Specifically, we investigate two use-cases for incrementally updating a model:

1. **Periodic Update**: Incrementally update the RNN-T model to ingest all the newly available training data to keep the model up-to-date. This update could capture distribution shifts and also possibly new words.
2. **Targeted Update**: Update the RNN-T model to learn specifically new words with a few available training samples. The training samples could be transcribed audio data or generated using text-to-speech (TTS) if no training data is available and to reduce data scarcity.

Ideally, we want to apply multiple rounds of periodic and targeted updates without degrading on *average data*. The term average data refers to the test data associated with the training data used in the seed model. In addition, we assume that all the previously used training data are available (*data-replay*). With this setup, we investigate the following key research questions:

- Is fine-tuning sufficient to apply IL to ASR models in a production scenario or are more sophisticated techniques (e.g. different losses or architectural changes) actually required in order to avoid overfitting to the newly ingested data? Can the potential overfitting be mitigated by using data-replay?
- How many incremental updates can be performed before there is a performance degradation and a from-scratch training is required?
- Are a few training examples sufficient for targeted updates? Can they be replaced by TTS utterances?

Experimental results show that the proposed fine-tuning techniques work well for multiple rounds of periodic or targeted incremental updates with and without TTS data. We show that data-replay can be used to effectively address overfitting

10.21437/Interspeech.2022-10795

and demonstrate that no additional regularization terms are required. We believe this is important in production scenarios, since it allows to streamline the IL setup without making training procedure or loss function more complicated and without increasing the amount of hyper-parameters to tune. Furthermore, the incrementally updated models outperform the models trained from scratch, while simple fine-tuning could better capture the distribution shifts. The proposed approach achieves $>$50x faster model updates, as it requires only $\sim$ 3 hours to update the model, whereas training a model from scratch takes $\sim$ 7 days.

2. Related Work

Since there exists no standard benchmark for IL in the area of ASR, most prior works investigate IL on a hypothetical setup where for each IL iteration, the data distribution shift is very strong. Thus, incrementally updating the model with the new data leads to overfitting and performance degradation on previously learned data. Such setup requires additional efforts to improve regularization, for example by using elastic weight consolidation [16] and distillation loss [15]. This implies additional hyper-parameters to be tuned such as related to computing the Fisher information matrix and the associated loss weight [16, 17], making the training process less scalable, harder to design and maintain for production use.

This paper simulates a realistic scenario for periodic and targeted updates, and investigates whether a simple fine-tuning technique would reliably work for both scenarios. Moreover, we explore multiple rounds of both periodic and targeted updates and use data-replay to address the overfitting.

3. Methodology

3.1. Problem description

This paper simulates an incremental learning setup on a real-world voice assistant application employing an RNN-T based ASR system. Given such an RNN-T model trained from scratch with all the data up to a certain month, say M_0, this paper investigates incrementally training the model for periodic and targeted updates as formulated below. We denote the RNN-T model from M_0 as M_0-RnnT.

Periodic Updates: Perform incremental updates on the M_0-RnnT model using the newly available training data from upcoming time periods: $\{M_1, M_2, \ldots, M_n\}$. Typically, such training data help capturing distribution shifts, i.e. it could be dominated by popular words which are already present in the M_0 training data.

Targeted Updates: This setup simulates the *hot-fixing* scenario where there are new words which were not present in the training data. In this case, the RNN-T model is updated using the data containing these new words so that it learns to emit them. Further, we explore multiple rounds of targeted updates at regular intervals as new words appear over time, which are denoted as $\{T_1, T_2, \ldots, T_n\}$. In a practical scenario, each targeted update could capture multiple new words with limited transcribed training data per word available.

Ideally, the ASR model needs to recognize new words before they are actually seen in production traffic and hence before any transcribed real speech is available. This may be crucial, e.g. with the release of new apps that respond to a new keyword or for future trending words before they actually become popular (e.g. being able to recognize names of Olympic athletes before the Olympics start). In all these scenarios, TTS can be a powerful tool, since text-only data is enough to enable the model to recognize such words. Therefore, this paper also includes experiments which use synthetic audio data for targeted updates.

The incrementally updated RNN-T models after each IL iteration should satisfy the following conditions:

- Perform well on the data associated with each incremental update: for periodic updates the model should perform better or on-par on the test set from M_i $\forall i \in \{1, 2, \ldots n\}$ compared to the RNN-T model trained from scratch with all the data up to M_i. For targeted updates, the RNN-T model should learn to emit the new words;
- Should not overfit to the datasets used for incremental updates, i.e., no degradation or degradation within an acceptable threshold on a *control* dev set from M_0 and also on the dev sets associated with the previous IL updates.

For each IL iteration the incrementally updated model from the previous stage is used as the seed model. Unlike the prior works on IL, we assume that training data from all the previous IL iterations are available (data-replay).

3.2. IL with simple fine-tuning

For both periodic and targeted updates, it is not expected that the training data contain any drastic acoustic or semantic distribution shifts. Therefore, we propose using a simple IL recipe where the seed RNN-T model is updated with the new training data and a small constant learning rate.

The main advantage of such a setup is that it streamlines the IL process without requiring any changes to the model architecture or to the training loss whilst limiting the introduction of new hyper-parameters. As a consequence, the implementation of IL in this setup can be executed with minimal human intervention.

3.3. Sampled data combination with data-replay

While the finetuning approach offers a simpler solution, as IL involves multiple rounds of such updates, there is a risk of overfitting, since there are no additional regularization techniques being applied. This could be particularly detrimental for targeted updates with limited amount of training data. Because it is assumed that data-replay is possible, a subset of the data used in the previous training stages could be leveraged to balance overfitting [5, 10]. In this setup every batch of training data for IL is sampled, so that it contains $x\%$ of the new data and $(100 - x)\%$ of the training data used to train the seed RNN-T model.

4. Experimental setup

4.1. RNN-T Model

The model consists of a 8 layers deep encoder, a 2 layers deep prediction network, a joint network as in [18] and an output layer with a softmax nonlinearity. Each layer of the encoder and the prediction network comprises 1024 Long Short-Term Memory [19] units. The size of the joint network layer is 512 and the output layer size is 4001 corresponding to 4000 word-pieces and a *blank* symbol. The word-piece model was trained on a large set of voice assistant requests using a unigram language model [20].

The model accepts 192 dimensional input feature vectors each comprising three 64 dimensional Log-Mel-Filterbanks extracted every 10 milliseconds and stacked together. Training objective is minimization of RNN-T loss function [1, 18] with Adam optimizer [21], with total batch size of 1536 utterances and warmup-hold-decay learning rate schedule. We also use SpecAugment [22].

4.2. Data

The experiments were performed on a German RNN-T ASR system trained using in-house de-identified far-field data. The seed RNN-T model (M_0-RnnT) is trained using a combination of human transcribed and machine-transcribed data up to M_0. This constitutes around 300k hours audio in total. The model was trained for 100 epochs with 5000 steps per epoch on 48 GPUs with a total batch size of 1536 samples/step. Training such a model from scratch takes several days.

In this work, we explore only using human transcribed training data for both periodic and targeted updates. For periodic updates, we consider three rounds of incremental updates with monthly training data from M_1, M_2 and M_3 which on average contains 250 hours of training data per month. For targeted updates, we identified 30 new words which were not present in the training data up to M_0. The training data for targeted updates is created so that the number of training samples per word varies from 10 to 15 samples.

For the experiments on targeted updates using TTS data, the synthetic utterances were produced as following: each transcription in the targeted training sets $\{T_1, T_2, \ldots, T_n\}$ is passed to an all-neural TTS model [23] to generate a synthetic utterance with a speaker profile randomly sampled from a pool of German voice profiles (containing both male and female voices). This produces n TTS targeted training datasets $\{T'_1, T'_2, \ldots, T'_n\}$, where each $T'_i, 0 < i \leq n$ contains synthetic utterances generated from the set of transcriptions from T_i.

Since fine-tuning using such small datasets could lead to overfitting, to study its impact and to find a reasonably good setup, two different scenarios are considered: 1) three rounds of targeted IL with 10 new words per iteration and 2) six rounds of targeted IL with 5 new words per iteration.

4.3. IL with data-replay

For incrementally updating the ASR model for both periodic and targeted updates, we explore fine-tuning the model at a small learning rate for a small number of steps. We investigated different learning rates and $1e-5$ was found to perform well. In addition, the model is updated only for up to 3000 steps for periodic updates and 600 steps for targeted updates. To reduce the overfitting, data-replay at varying sampling weights are investigated.

4.4. Evaluation

For evaluating IL periodic updates, at i-th iteration of incremental update the test data from M_i and M_0 were used to evaluate the performance on the current month and the old data to keep track of overfitting. In addition, the model was also evaluated on test data from M_{i+1} to assess hypothetical performance after deployment. The relative reduction in word error rate (rWERR) with respect to the M_0-RnnT model is used as the evaluation metric. For baseline models the monthly RnnT models trained from scratch were used (dubbed M_1-RnnT, M_2-RnnT and M_3-RnnT).

	Weights %		WERR (%)		
ID	(old, new)	m	Dev$_{\text{m}}$	Dev$_{\text{m+1}}$	Dev$_{\text{old}}$
1	(0, 100)	$+M_1$	5.4	0.9	1.7
2		$+M_2$	1.8	8.0	2.2
3		$+M_3$	8.9	4.6	1.7
4	(50, 50)	$+M_1$	6.0	0.4	2.0
5		$+M_2$	4.8	8.3	2.2
6		$+M_3$	8.9	4.6	1.7
7	(90, 10)	$+M_1$	2.7	-0.7	0.2
8		$+M_2$	1.3	3.0	0.7
9		$+M_3$	3.4	2.8	1.0
10		M_1-RnnT	1.7	0.4	1.0
11		M_2-RnnT	1.5	2.3	2.4
12		M_3-RnnT	2.3	-1.6	2.2

Table 1: *Incremental periodic updates starting with the RNN-T model from $t = M_0$ as seed using M_1, M_2 and M_3 monthly training data. For each IL update, monthly data is incrementally added with the respective weight % given in the table. WERR w.r.t. the M_0 RNN-T model are shown.*

For targeted updates, the recall metric is used to evaluate how well the RNN-T model performs on recognizing the target word. Additionally, the rWERR on M_0 test data was evaluated to track overfitting as multiple rounds of targeted updates are performed.

5. Results and Discussion

5.1. IL periodic updates

The evaluation results for periodic updates with different data-replay configurations are summarised in Table 1. Incremental updates with monthly data leads to improvements on recent data with minimal degradation on the average dataset Dev$_{\text{old}}$. Data-replay with (50, 50) sampling ratio is found to be a good operating point as it yields the best performance on recent datasets coupled with no degradation on Dev$_{\text{old}}$. In addition, this setup yields significant performance improvements on the future datasets compared to the train from scratch baseline systems. These results demonstrate the effectiveness of the proposed IL approach as it yields 5-6% WERR improvements over the train from scratch models (cfr. on Dev$_{\text{m+1}}$ lines 11 vs. 5 and 12 vs. 6).

Moreover, periodic updates take only 3000 steps of RNN-T training that is $\sim$ 3 hours compared to $\sim$ 7 days required to train a model from scratch. Thus, the proposed approach enables 50× speedup in addition to improved accuracy compared to RNN-T models trained from scratch.

5.2. IL targeted updates

In this setup the seed RNN-T model is incrementally updated to improve recognition of a small set of words. Table 2 summarizes the results obtained for three and six rounds of targeted IL updates. The seed model M_0-RnnT model struggled to generate words that were not present in the training data, with a recall rate of below 15% on average.

Table 2(a) shows the results obtained for three rounds of targeted updates with real and TTS data while varying data-replay weights. Even though fine-tuning without data-replay (cfr. (0, 100) setup, lines 1-3) leads to significant improvements in recall for multiple rounds, WER degradation of 33% on the

	Weights % (old, target)	t	Real Data Recall$_t$	Real Data Test$_{old}$	TTS Data Recall$_t$	TTS Data Test$_{old}$
1	(0, 100)	+ T_1	85.4	-8.6	68.3	-7.5
2		+ T_2	95.2	-13.9	64.3	-11.9
3		+ T_3	90.2	-33.0	85.4	-17.5
4	(90, 10)	+ T_1	82.9	0.2	63.4	+0.5
5		+ T_2	78.6	-1.2	61.9	0.2
6		+ T_3	90.2	-1.7	85.4	0.0
7	(95, 5)	+ T_1	80.5	0.2	63.4	0.2
8		+ T_2	78.6	-0.2	57.1	-0.2
9		+ T_3	90.2	-1.5	85.4	0.0
10	(99, 1)	+ T_1	65.9	-1.0	41.5	1.0
11		+ T_2	57.1	-0.7	47.6	-1.2
12		+ T_3	82.9	-1.0	73.2	-0.7

(a) *Three rounds of IL targeted updates (10 new words per round)*

	Weights % (old, target)	t	Real Data Recall$_t$	Real Data Test$_{old}$	TTS Data Recall$_t$	TTS Data Test$_{old}$
1	(95, 5)	+ T_1	94.7	-0.5	78.9	-0.5
2		+ T_2	81.8	0.0	54.5	-0.5
3		+ T_3	100.0	0.2	90.9	-0.2
4		+ T_4	70.0	-1.2	40.0	0.5
5		+ T_5	100.0	-0.7	100.0	1.0
6		+ T_6	77.8	-2.0	72.2	-0.7
7	(99, 1)	+ T_1	89.5	-1.0	68.4	-0.7
8		+ T_2	72.7	-0.2	31.8	-0.7
9		+ T_3	90.9	-0.7	86.4	-0.5
10		+ T_4	45.0	-1.2	25.0	-0.2
11		+ T_5	100.0	-1.2	100.0	0.0
12		+ T_6	77.8	-2.4	66.7	-0.5

(b) *Six rounds of IL targeted updates (5 new words per round)*

Table 2: *Incremental targeted updates on the M_0-RnnT model using real and synthetic training data to learn new words. Three rounds of training were performed with training data containing utterances with* 10 *new words per round. Recall in % is shown to evaluate the recognition of the new words. The recall levels of the seed M_0-RnnT model were below* 15% *for every target word. In addition, rWERR on Dev$_{old}$ w.r.t. the seed model is included to track the overfitting.*

average dataset is unacceptable. Adding data-replay effectively addresses the overfitting but with a slight degradation in recall compared to the no data-replay setting. Replacing real data with TTS data also leads to similar conclusions except that TTS data provides a smaller, but still significant, improvement in the recall. This is expected since the acoustics of the TTS data does not match the real production environment. Overall, data replay (95, 5) is found to be a good operating point, as it provides significant improvements in terms of recall with minimal degradation on the average dataset.

Next, we investigated six rounds of targeted incremental updates. The training data available for each round is $<$ 75 words. Since (0,100) and (90,10) data-replay configurations showed overfitting in the previous experiments, only the (95, 5) and (99, 1) setups were investigated. The evaluation results in Table 2(b) show that the same recipe can be used for more number of incremental updates.

Targeted IL updates require up to 600 steps of RNN-T training, which takes only 30-40 minutes. Compared to training the model from scratch, the proposed targeted IL setup works $250\times$ faster and yields significant improvements in recall on the target words with minimal overfitting ($<$ 2% WER degradation on average dataset for most cases). Interestingly, with synthetic data the degradation on the average dataset is lower than with real data and is not degrading further when more incremental training iterations are performed.

5.3. Alternating periodic and targeted updates

Finally, we simulate a realistic scenario where periodic and targeted updates are performed in an alternating fashion. For this experiment, we selected the best configurations obtained for periodic and targeted updates from the above experiments, i.e., data-replay of (50, 50) for periodic and (95, 5) for targeted IL updates. Here we used only the real data for IL updates and the results are provided in Table 3. Notice that there could be overlap between targeted and periodic training datasets.

To summarize, the proposed simple fine-tuning setup together with data-replay achieves significant performance improvements over the models trained from scratch. In addition, we can perform targeted updates which are required for RNN-T models to be able to correctly recognize new words. Furthermore, this setup has a reduced risk of overfitting and even achieves improvement on average data.

	Dev sets rWERR (%)					Recall (%)		
m/t	M_1	M_2	M_3	M_4	old	T1	T2	T3
M_0-RnnT	0.0	0.0	0.0	0.0	0.0	7.3	11.9	7.3
1 + M_1	5.8	0.0	4.0	4.6	2.4	12.2	16.7	9.8
2 + T_1	5.6	1.1	3.4	3.8	2.2	82.9	19.0	22.0
3 + M_2	4.3	4.8	6.7	4.8	2.2	85.4	21.4	22.0
4 + T_2	5.4	4.6	7.1	3.2	2.7	85.4	81.0	22.0
5 + M_3	4.7	3.1	7.8	3.4	2.4	85.4	88.1	22.0
6 + T_3	4.3	4.9	6.7	1.4	2.4	82.9	83.3	90.2
M_1-RnnT	1.7	0.7	3.1	2.0	1.5	12.2	26.2	12.2
M_2-RnnT	3.1	1.5	2.1	2.2	2.9	7.3	28.6	9.8
M_3-RnnT	4.3	3.9	2.3	-1.6	2.4	7.3	26.2	17.1

Table 3: *Incremental training with alternating periodic and targeted updates. For periodic updates we used a data sampling ratio of* $(50, 50)$ *and for targeted updates* $(95, 5)$.

6. Conclusions

This paper investigated incrementally updating an RNN-T ASR model used a real-world production scenario using fine-tuning with data-replay. The proposed approach is shown to yield significant performance improvements over the model trained from scratch with reduced risk of overfitting. We also demonstrated that the same recipe can be used successfully for updates targeting a small set of new words with very limited amount of real or synthetic training data. Finally, we explored a setup with alternating periodic and targeted updates which could help to mitigate the overfitting problems observed during targeted updates. The proposed approach allows to capture the distribution shift in training data with time while spending only hours of computations instead of days.

7. References

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

[2] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition." in *IEEE-ICASSP*, 2016, pp. 4960–4964.

[3] G. Zweig, C. Yu, J. Droppo, and A. Stolcke, "Advances in all-neural speech recognition." in *IEEE-ICASSP*, 2017, pp. 4805–4809.

[4] E. Battenberg, J. Chen, R. Child, A. Coates, Y. Gaur, Y. Li, H. Liu, S. Satheesh, A. Sriram, and Z. Zhu, "Exploring neural transducers for end-to-end speech recognition." in *IEEE-ASRU*, 2017, pp. 206–213.

[5] J. Li, Y. Wu, Y. Gaur, C. Wang, R. Zhao, and S. Liu, "On the comparison of popular end-to-end models for large scale speech recognition." in *INTERSPEECH*. ISCA, 2020, pp. 1–5.

[6] X. Zhang, F. Zhang, C. Liu, K. Schubert, J. Chan, P. Prakash, J. Liu, C.-F. Yeh, F. Peng, Y. Saraf, and G. Zweig, "Benchmarking lf-mmi, ctc and rnn-t criteria for streaming asr." in *IEEE-SLT*, 2020.

[7] C.-C. Chiu, A. Kannan, R. Prabhavalkar, Z. Chen, T. N. Sainath, Y. Wu, W. Han, Y. Zhang, R. Pang, S. Kishchenko, P. Nguyen, A. Narayanan, H. Liao, and S. Zhang, "A comparison of end-to-end models for long-form speech recognition." in *IEEE-ASRU*, 2019, pp. 889–896.

[8] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S.-Y. Chang, K. Rao, and A. Gruenstein, "Streaming end-to-end speech recognition for mobile devices." in *IEEE-ICASSP*, 2019, pp. 6381–6385.

[9] T. N. Sainath, Y. He, B. Li, A. Narayanan, R. Pang, A. Bruguier, S.-Y. Chang, W. Li, R. Alvarez, Z. Chen, C.-C. Chiu, D. Garcia, A. Gruenstein, K. Hu, A. Kannan, Q. Liang, I. McGraw, C. Peyser, R. Prabhavalkar, G. Pundak, D. Rybach, Y. Shangguan, Y. Sheth, T. Strohman, M. Visontai, Y. Wu, Y. Zhang, and D. Zhao, "A streaming on-device end-to-end model surpassing server-side conventional model quality and latency." in *IEEE-ICASSP*, 2020, pp. 6059–6063.

[10] X. Zheng, Y. Liu, D. Gunceler, and D. Willett, "Using synthetic audio to improve the recognition of out-of-vocabulary words in end-to-end asr systems," in *IEEE-ICASSP*, 2021, pp. 5674–5678.

[11] C. Peyser, S. Mavandadi, T. N. Sainath, J. Apfel, R. Pang, and S. Kumar, "Improving tail performance of a deliberation E2E ASR model using a large text corpus," in *INTERSPEECH*. ISCA, 2020, pp. 4921–4925.

[12] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning." in *ECCV*, vol. 11216, no. 12. Springer, 2018, pp. 241–257.

[13] Z. Li and D. Hoiem, "Learning without forgetting." *IEEE-TPAMI*, vol. 40, no. 12, pp. 2935–2947, 2018.

[14] S. Sadhu and H. Hermansky, "Continual learning in automatic speech recognition," in *INTERSPEECH*. ISCA, 2020, pp. 1246–1250.

[15] L. Fu, X. Li, L. Zi, Z. Zhang, Y. Wu, X. He, and B. Zhou, "Incremental learning for end-to-end automatic speech recognition," in *IEEE-ASRU*, 2021, pp. 320–327.

[16] S. Ghorbani, S. Khorram, and J. H. L. Hansen, "Domain expansion in dnn-based acoustic models for robust speech recognition," in *IEEE-ASRU*, 2019, pp. 107–113.

[17] K. C. Sim, L. Johnson, G. Motta, L. Zhou, F. Beaufays, A. Benard, D. Guliani, A. Kabel, N. Khare, T. Lucassen, P. Zadrazil, and H. Zhang, "Personalization of end-to-end speech recognition on mobile devices for named entities," in *IEEE-ASRU*, 2019, pp. 23–30.

[18] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee, 2013, pp. 6645–6649.

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

[20] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[23] I. Vallés-Pérez, J. Roth, G. Beringer, R. Barra-Chicote, and J. Droppo, "Improving multi-speaker TTS prosody variance with a residual encoder and normalizing flows," *arXiv preprint arXiv:2106.05762*, 2021.