PeakRNN and StatsRNN: Dynamic Pruning in Recurrent Neural Networks

Zuzana Jelčicová[1,2], Rasmus Jones[1], David Thorn Blix[1], Marian Verhelst[3], and Jens Sparsø[2]
[1] Demant A/S, Kongebakken 9, 2765 Smørum, DK
zuje@demant.com, rajn@demant.com, dbli@demant.com
[2] Technical University of Denmark, Richard Petersens Plads, Building 322, 2800 Kgs. Lyngby, DK
zuje@dtu.dk, jspa@dtu.dk
[3] Katholieke Universiteit Leuven, Oude Markt 13, 3000 Leuven, BE
marian.verhelst@kuleuven.be

***Abstract*—This paper introduces two dynamic real-time pruning techniques *PeakRNN* and *StatsRNN* for reducing costly multiplications and memory accesses in recurrent neural networks. The methods are demonstrated on a gated recurrent unit in a multi-layer network, solving a single-channel speech enhancement task with a wide variety of real-world acoustic environments and speakers. The performance is compared against the baseline gated recurrent unit and the DeltaRNN method. Compared to the unprocessed speech, the SNR and Perceptual Evaluation of Speech Quality were on average improved by 8.11 dB and 0.43 MOS-LQO, respectively. Additionally, the two proposed methods outperformed DeltaRNN by 0.7 dB and 0.11 MOS-LQO in the two objective measures, while using the same computational budget per timestep and reducing the original operations by 88%. Furthermore, *PeakRNN* is fully deterministic, i.e. it is always known in advance how many computations will be executed. Such worst-case guarantees are crucial for real-time acoustics applications.**

***Index Terms*— RNN, determinism, statistics, peaks, threshold, single-channel speech enhancement, hearing instruments**

I. Introduction

Speech enhancement (SE) is a classical problem in signal processing that focuses on attenuating background noise from a speech signal. Traditionally, statistical methods such as the Wiener filter [1], non-Negative Matrix Factorization [2], and Short-Time Spectral Amplitude [3] have been used to solve the single-channel SE task. Applications such as hearing instruments (HIs), (wireless) headsets, and mobile communications require algorithms that are able to handle a wide range of noise environments and speakers to be useful in real-life situations. However, such signals do not follow normal distributions and are often non-stationary [4], i.e. the statistical structure of the signal changes over time, which imposes a challenge for the traditional methods [5]. Deep neural networks (DNNs) are able to capture non-linear and complex relationships, and have proved successful in SE applications [6], [7], outperforming classical signal processing methods.

SE is a challenging problem that is usually solved by complex DNN models using several hidden layers consisting of hundreds to thousands of neurons, resulting in a model with millions of parameters. Often such models can, however, be pruned, leading to computational savings that are crucial for low-power edge devices such as HIs. Static pruning [8] results in a smaller dense model where, however, the capabilities of the pruned neurons and weights are irreversibly gone. Moreover, static pruning cannot capture the importance of neurons and weights that are highly input-dependent. Dynamic approaches [9], on the other hand, enable to use parts of the NN relevant for the current input. Yet these methods are often complex, and the deployment hence still remains challenging.

In this work, we propose two dynamic pruning techniques, called *PeakRNN* and *StatsRNN*, which during inference reduce the number of memory accesses (MAs) and multiply-accumulates (MACs) dynamically in a data-driven way. The reduction is demonstrated on a SE task using a gated recurrent unit (GRU) hidden layer in a three-layer network. The evaluations are based on the computational costs and objective measures such as SNR and Perceptual Evaluation of Speech Quality (PESQ). While *PeakRNN* offers determinism and robustness without any prior data analysis, *StatsRNN* approaches pruning by exploring the underlying statistical properties of the data. These two pruning techniques can be used to find an optimal model and they outperform the current state-of-the-art DeltaRNN [10] technique.

II. Related work

Recurrent neural networks (RNNs) and their variants, such as long short-term memory (LSTM) units [11] and GRU [12], are suited for time-series tasks since they are able to model complex temporal structures. GRUs are computationally more efficient and thus preferred over LSTMs in real-time SE tasks [13]. However, they still require many matrix-vector multiplications and memory accesses. Works targeting single-channel SE primarily focus on a high-level exploration, i.e. comparing different objective measures for speech quality and intelligibility without considering the complexity of the proposed algorithms.

In [14] the authors introduce FastGRNNs that use a scalar weighted residual connection for each coordinate of the hidden state h. They have lower training times and prediction costs, as well as 2-4x fewer parameters than LSTMs and GRUs, while matching the state-of-the-art prediction accuracies. However, this static method does not consider temporal dependencies in data. DeltaRNN [10] addresses the computational issues by exploiting the temporal stability of inputs and activations, i.e. by caching neuron activations, operations can be skipped where no significant changes occur from the previous update. This data-driven approach saves fetches of entire columns of weight matrices, leading to substantial speedups of 5.7-100x for a RNN on classification tasks with negligible accuracy loss.

Our work builds on and outperforms the idea of DeltaRNNs with novel PeakRNN and StatsRNN techniques. The algorithms are demonstrated on the single-channel SE regression task.

III. Peak RNN algorithm

Similar to DeltaRNN, the objective of PeakRNN is to transform a dense matrix-vector multiplication into a highly-sparse matrix-vector multiplication to save both MAC operations and, above all, MAs. These two methods therefore share the underlying computations targeting GRU, equations (1)-(12), that are detailed in [10]. The actual difference arises in selecting the elements for computations (1)-(4). DeltaRNN applies a single threshold θ on both the input vector x and the activation vector h, resulting in a variable number of required computations. In contrast, PeakRNN selects a desired number of *peak* elements N_p from both vectors individually in every timestep. Hence, unlike DeltaRNN where the number of computations is non-deterministic, PeakRNN saves computations in a *deterministic manner* as it is always known in advance how many operations will be executed. Therefore, it provides *worst-case execution guarantees*, which is a crucial aspect in real-time low-power devices, and not ensured by DeltaRNN.

$$\hat{x}(t) = \begin{cases} x(t) & \text{if } |x(t) - \hat{x}(t-1)| \text{ among } N_p \\ \hat{x}(t-1) & \text{otherwise} \end{cases} \tag{1}$$

$$\hat{h}(t-1) = \begin{cases} h(t-1) & \text{if } |h(t-1) - \hat{h}(t-2)| \text{ among } N_p \\ \hat{h}(t-2) & \text{otherwise} \end{cases} \tag{2}$$

$$\Delta x(t) = \begin{cases} x(t) - \hat{x}(t-1) & \text{if } |x(t) - \hat{x}(t-1)| \text{ among } N_p \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\Delta h(t-1) = \begin{cases} h(t-1) - \hat{h}(t-2) & \text{if } |h(t-1) - \hat{h}(t-2)| \text{ among } N_p \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$M_r(t) = W_{xr}\Delta x(t) + W_{hr}\Delta h(t-1) + M_r(t-1) \tag{5}$$

$$M_u(t) = W_{xu}\Delta x(t) + W_{hu}\Delta h(t-1) + M_u(t-1) \tag{6}$$

$$M_{xc}(t) = W_{xc}\Delta x(t) + M_{xc}(t-1) \tag{7}$$

$$M_{hc}(t) = W_{hc}\Delta h(t-1) + M_{hc}(t-1) \tag{8}$$

$$r(t) = \sigma[M_r(t)] \tag{9}$$

$$u(t) = \sigma[M_u(t)] \tag{10}$$

$$c(t) = tanh[M_{xc}(t) + r(t) \odot M_{hc}(t)] \tag{11}$$

$$h(t) = u(t) \odot h(t-1) + (1 - u(t)) \odot c(t) \tag{12}$$

Moreover, PeakRNN is robust to the variations of the input data as the algorithm selects the top elements regardless of the threshold. Additionally, the number of peaks for the x and h vectors can either be equal or different. In our experiments, we used the same number of peaks for both vectors, but this combination can be optimized depending on a given task. The top N_p elements might be selected using sorting, but an alternative solution is to approximate N_p elements by exploring the underlying statistical properties of the data as introduced in Section IV.

Table I provides an overview of the theoretical estimations of MACs and MAs required for a GRU every timestep as these, particularly MAs, are among the most costly operations [15]. These estimations enable us to compare DeltaRNN and PeakRNN on equal terms. The calculations are derived from equations (1) - (12). N_x and N_h refer to the dimensionality of an input vector x and activation vector h, respectively. o_x and o_h, called *occupancy* [10], correspond to the fraction of non-zero values of an input vector x and activation vector h, respectively. These fractions define how many operations will be executed. As it can be seen in Table I, there is MA overhead compared to GRU due to keeping track of additional states. However, these excess operations are negligible for huge RNNs as the MAs for weights scale approximately quadratically. Therefore, significant savings and increased speedup can be achieved with a sparse occupancy. This is pronounced even more for MACs, and the results are presented in Section VI.

TABLE I: Theoretical cost calculations of MACs and MAs.

	GRU	**DeltaRNN/PeakRNN**	**Equations**
MACs (x + h)	$3(N_xN_h)$ + $3(N_hN_h)$	$o_x[3(N_xN_h)]$ + $o_h[3(N_hN_h)]$	(5) - (8)
MACs (pointwise)	$3N_h$	$3N_h$	(11) - (12)
MAs (x+h weights)	$3(N_xN_h)$ + $3(N_hN_h)$	$o_x[3(N_xN_h)]$ + $o_h[3(N_hN_h)]$	(5) - (8)
MAs (x+h read)	N_x + N_h	$2N_x$ + $2N_h$	(1) - (4)
MAs (h write)	N_h	N_h	(12)
MAs (M states read)	-	$4N_h$	(5) - (8)
MAs (M states write)	-	$4N_h$	(5) - (8)
MAs ($\hat{x}$ + $\hat{h}$ write)	-	o_xN_x +o_hN_h	(1) - (2)

IV. Statistical RNN algorithm

StatsRNN finds the top N_p elements by exploiting the statistical properties of the $|x(t)-\hat{x}(t-1)|$ and $|h(t-1)-\hat{h}(t-2)|$ computations that determine whether the elements should be zeroed out. We hypothesized that our training dataset is representative of our SE application. Consequently, we can assume that the underlying statistical distributions within the network are a good approximation of distributions in the application. We created an average histogram with 256 logarithmic bins for the x and h computations separately, based on the entire training dataset. Figure 1 shows a histogram example for $|x(t)-\hat{x}(t-1)|$ ($|h(t-1)-\hat{h}(t-2)|$ has a similar distribution) where it can be observed that the data is correlated. Also, due to ReLU in the first fully connected (FC) layer, a lot of values are zero, ∼31% and 21% for x and h, respectively, which enables to exploit sparsity to a high degree. After training, the threshold can be statistically determined for x and h separately using the bin boundaries, i.e. selecting a boundary where all the bins to its right (greater values) represent the percentage of elements (o_x and o_h) equivalent to the number of peak elements (PeakRNN) that should be processed. This approach preserves the idea of PeakRNN and eliminates sorting. In the presented experiments, the same percentage of top elements to extract was set for x and h.

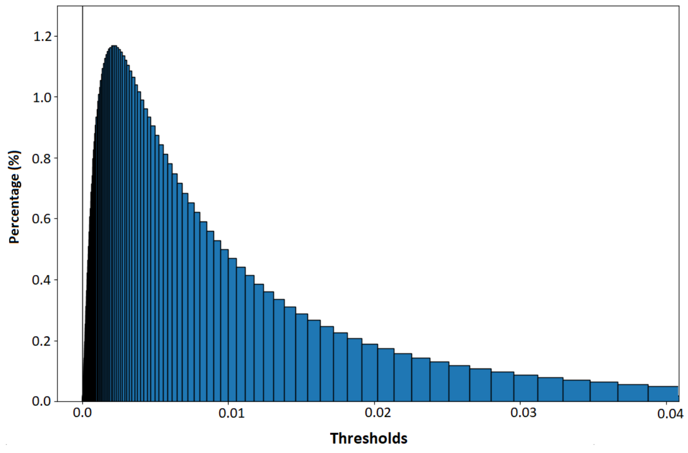


Fig. 1: A zoomed view on a part of the histogram with logarithmic bins for the delta calculation $|x(t)-\hat{x}(t-1)|$ from the training dataset. The x-axis represents bin boundaries that are used to determine x and h thresholds. The thin black vertical line to the very left corresponds to the first bin that contains ~31% of zeros.

The StatsRNN approach was applied to the entire test dataset, trying to obtain various percentages of x and h elements individually, varying from 50% down to 1%. Due to a big variety of scenes, some environments naturally deviate from the required percentage more while others less. The smallest difference between the expected and the actual obtained percentage was only 0.05% (Quiet Street), where, on average, 40.05% of the h elements were processed instead of 40%. The biggest outliers were Pink and, in particular, White noise, where ~31.65% of the h vector elements was processed instead of 50%. All the details about the datasets are described in Section V-C. StatsRNN analytically defines a threshold for x and h individually. This is a very important property since the vectors have different sparsity as also shown in [10], and their individual handling might contribute to additional improvements. Therefore, exploiting a priori statistical knowledge about the data will lead to a better and more deterministic algorithm with consistent performance compared to DeltaRNN. Furthermore, the estimations provided in Table I can be directly used for StatsRNN as well. Instead of N_p in equations (1)-(4), two statistically derived thresholds Θ_x and Θ_h will be applied for StatsRNN (a single threshold Θ in original DeltaRNN [10]).

V. Experimental setup

This section describes the entire system setup used for performing the experiments.

A. Hearing-instrument application

Figure 2 illustrates a simplified system extracted from a real HI setup where the DNN replaces a typical noise reduction module for obtaining postfilter gain. Firstly, the *Analysis Filter Bank* applies a 1024-point FFT and a square-root Hanning window on the 20 kHz microphone signal (*mic*), resulting in 512 frequency sub-bands and downsampling to 40 Hz (a new frame every 25 ms, no overlapping). The output is then passed to the *DNN* that will learn to estimate a postfilter gain ($pfGain$) to be applied on the original signal. Finally, the *Synthesis Filter Bank* reconstructs a wideband signal and passes it to the speaker. The proposed techniques are demonstrated on a single microphone but they can be also applied in a setup with a microphone array (e.g. HIs with two microphones). In such case, the *Minimum Variance Distortionless Response (MVDR) Beamformer* could be used for multi-channel processing where interferences from undesired directions would be attenuated.

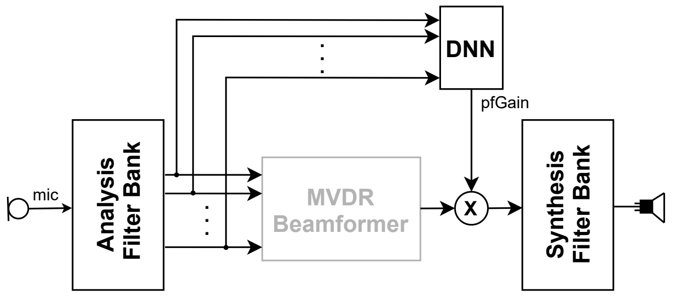


Fig. 2: Full system overview representing simplified internals of a HI. The highlighted parts (black) are used for the single-channel SE experiments.

B. DNN architecture

The DNN architecture used in the experiments consists of three layers: FC-GRU-FC, each having 512 output neurons, with 512 inputs to the first FC layer. The first FC layer is followed by ReLU activation function, while the GRU layer uses tanh and sigmoid activation functions. The final output of the network are 512 $pfGain$ values that are applied on the original signal. The GRU component is replaced with DeltaRNN, PeakRNN, and StatsRNN during the experiments.

C. Dataset

The DNN input is a mixed signal y, i.e. clean speech corrupted with noise, constructed by adding 30-second segments of noise n and clean speech s together ($y = s + n$). The 30-second segments contain one to three speakers with a maximum gap of 300 ms and up to 30% overlap, creating seemingly natural flow and tempo of a conversation. The speech was obtained from the *VCTK Corpus* [16] and *Akustiske Database for Dansk* [17]. Fifteen different types of audio environments (referred to as background noise) were used, reflecting the most relevant acoustic situations that people are exposed to in the real world in order to obtain the required variations in SNR estimates. Eight scenes (Beach, Busy Street, Park, Pedestrian Zone, Quiet Street, Shopping Centre, Train Station, Woodland) were obtained from [18], five scenes are a part of the internal database of the Demant company (Bar, Cafe, Canteen, Car, Office), and Pink and White stationary noises were simulated. The entire dataset contains ~25 hours (left and right channels together) of mixed signal, divided into training (~19.5 h), test, and validation (each ~2.7 h) subsets. Each of the three subsets is unique, i.e. the speakers and the background noise sections are not shared across the subsets.

The unprocessed speech has SNR and PESQ of 4.39 dB and 1.85 MOS-LQO (Mean Opinion Score - Listening Quality

Objective) [19], respectively. The starting SNRs for the scenes in the test dataset vary between -12.7 dB (e.g. White noise) to 14.4 dB (e.g. Park), with most acoustic scenes having SNR up to 8 dB (80%). The high SNRs are present in only a few cases to cover the necessary variations as mentioned before.

D. Training

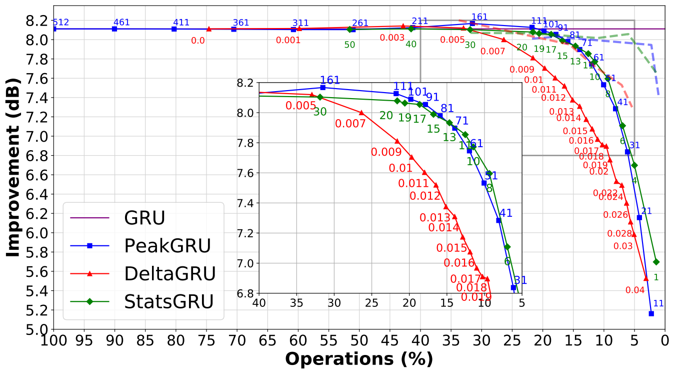
The DNN is trained on a linear ideal ratio mask (IRM) where the mask value is a continuous gain between zero and one. The IRM is defined as follows:

$$IRM = \left(\frac{|s(t,f)|}{|s(t,f)| + |n(t,f)|} \right) \tag{13}$$

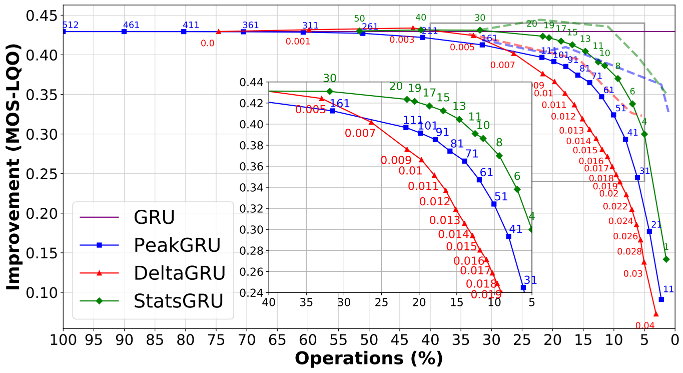
The $s(t,f)$ and $n(t,f)$ represent the clean speech and noise, respectively, with time frame t and frequency channel f. The mean squared error between the target IRM and the denoised speech is used as the loss function. The final DNN was trained using a batch size of 128 and a sequence length of 100 samples, which corresponds to 2.5 seconds. The DNN with a GRU layer was trained in Tensorflow using 32-bit floating-point, and the weights and biases were transferred to DeltaRNN, PeakRNN, and StatsRNN (further referred to as DeltaGRU, PeakGRU, and StatsGRU) for inference. The results are presented in Section VI. Transferring the learned parameters replaces computationally expensive and time demanding training from scratch for each configuration, and enables to compare all the methods in a fair manner. The DeltaGRU, PeakGRU, and StatsGRU networks were finally also retrained using transfer learning, where the transferred GRU model served as a starting point. This approach is discussed at the end of Section VI.

VI. Results and Discussion

Figure 3 shows the improvement of the different GRU implementations to unprocessed speech in SNR and PESQ for decreasing percentage of MAC operations per timestep. It also provides a zoomed view on a part of the plot discussed in this section. The annotations on the PeakGRU, DeltaGRU, and StatsGRU curves describe the tested number of processed peaks (512-11), thresholds (0.0-0.04) and the desired percentages of elements to extract (50-1), respectively. The GRU performance of 8.11 dB and 0.43 MOS-LQO in (a) and (b), respectively, is shown as baseline with 100% MAC operations, corresponding to 1.5744 MOps. The same number of operations is also required for MAs. Only a plot for MAC results is presented since MA reduction is done in the same linear manner and its diagram would, therefore, look almost the same (max ∼0.4% more MAs from the original number of operations). Due to the MA overhead estimated in Table I, all the three modified GRU algorithms have a total of ∼1.581 MOps when processing all features. The computations are based on an *overall* scene, i.e. an average across all the acoustic environments and x and h subsets. However, a similar trend can be also observed for the individual scenes, i.e. PeakGRU and StatsGRU outperforming DeltaGRU. The only scene where DeltaGRU has subtly better performance is White noise, and it is on par with PeakGRU for Pink noise and Busy Street scenes. The performance of StatsGRU is very similar to PeakGRU as it approximates the number of top elements to be processed. The largest SNR improvement on average was obtained for the White noise environment (∼15.72 dB), while the smallest one for the Office scene (∼4.26 dB).



(a) SNR improvement vs MACs (unprocessed speech = 4.39 dB).



(b) PESQ improvement vs MACs (unprocessed speech = 1.85 MOS-LQO).

Fig. 3: Plots (a) and (b) show the percentage of executed MACs with respect to improvement in SNR/PESQ (including a zoomed part of the plot). 100% of MACs corresponds to the baseline GRU illustrated as a constant horizontal line, i.e. the x-axis does not apply to it. The data labels annotated on the lines represent the number of processed peaks, thresholds, and the desired % of elements to extract for PeakGRU, DeltaGRU, and StatsGRU, respectively. The dashed lines show the benefit of retraining that further optimizes the performance.

As it can be observed in Figure 3(a), all the methods have equal and no loss in performance down to ∼30%, corresponding to ∼0.4723 MOps and thus reducing the computations by 70%. DeltaGRU, and likewise StatsGRU, already saves 25% of computations with Θ=0.0 without any SNR/PESQ degradation due to the ReLU in the first FC layer that produces a sparse input to the GRU. However, with decreasing % of computations, the objective metrics start to noticeably degrade especially for DeltaGRU, while PeakGRU and StatsGRU have a less steep decrease. StatsGRU and PeakGRU are aligned regarding SNR, and StatsGRU has slightly better PESQ. This behavior can be explained by the fact that StatsGRU selects a specific percentage of x and h separately across the *entire dataset*, while PeakGRU does so *every timestep*, and thus sometimes chooses less significant values, which is subtly reflected in speech quality. At ∼7.8 dB SNR improvement, the two new methods reduce the computations down to ∼12%, which is a reduction by almost a factor of 2 compared to ∼21% for DeltaGRU. Even at 7 dB, which represents an acceptable loss of 1 dB, PeakGRU and StatsGRU execute on average 7% of operations while DeltaGRU 11%. Similarly, a better PESQ performance of the two proposed techniques can be seen in Figure 3(b). On the other hand, if the number of operations is fixed at 12%, PeakGRU and StatsGRU outperform DeltaGRU by 0.7 dB on average. Figure 4 further presents the results per scene, where the SNR improvements for each of the methods are plotted as overlapping bars.

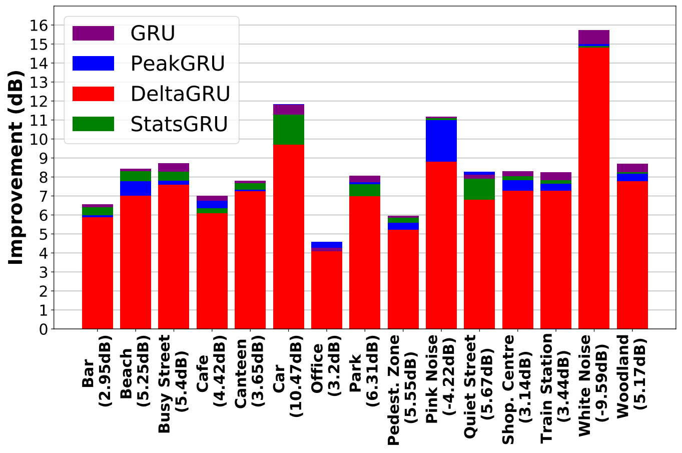


Fig. 4: SNR improvement per scene under the fixed number of operations per timestep (12%) plotted as overlapping bars, with PeakGRU and StatsGRU outperforming DeltaGRU by 0.7 dB on average. The configurations use 61 peaks, Θ=0.016, and top 10% of elements for PeakGRU, DeltaGRU, and StatsGRU, respectively. The x-axis denotes each tested scene along with its initial average SNR. PeakGRU is on par with GRU, and StatsGRU with DeltaGRU for the Car and Office scene, respectively.

Retraining the models further optimizes the performance, especially for the configurations with high thresholds/many peaks skipped. This is illustrated with dashed lines in Figure 3, where all the three methods benefit comparably from transfer learning. Therefore, the same relative gains for DeltaRNN, PeakRNN, and StatsRNN remain valid.

Future work further explores the most optimal ratio between x and h sparsities, since the best model might not necessarily use the same threshold/number of peaks for both. This could be done by making the sparsity ratio parameter differentiable, such that it can be trained together with the network itself.

VII. Conclusion

This paper introduced two new pruning techniques demonstrated on a single-channel SE task using complex environments and big variety of speakers. The results proved that computations in a RNN can be efficiently reduced nearly $2x$ compared to the state-of-the-art DeltaRNN while maintaining sufficient quality of objective measures. In addition, reducing the SNR quality by only ∼0.3 dB saves 88% of operations in PeakRNN and StatsRNN, while the same reduction in DeltaRNN is achieved by degrading SNR by 1 dB. Furthermore, PeakRNN is deterministic and thus provides worst-case execution guarantees required by real-time applications. If the deterministic aspect can be slightly relaxed, further savings can be achieved by StatsRNN using a statistical approximation. Overall, both algorithms are hence suitable for resource-constrained embedded devices such as HIs.

Acknowledgment

We would like to thank Asger Heidemann Andersen and Robert Rehr for all their advice, help, and knowledge they provided.

References

[1] P. C. Loizou, *Speech Enhancement: Theory and Practice*, 2nd ed. USA: CRC Press, Inc., 2013.
[2] Y. Wang, A. Narayanan, and D. Wang, "On Training Targets for Supervised Speech Separation," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 22, no. 12, p. 1849–1858, Dec. 2014.
[3] Y. Ephraim and D. Malah, "Speech Enhancement Using a Minimum Mean-Square Error Log-Spectral Amplitude Estimator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 2, pp. 443–445, 1985.
[4] A. Meynard and B. Torresani, "Spectral Analysis for Nonstationary Audio," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 26, no. 12, p. 2371–2380, Dec. 2018.
[5] Z. Duan, G. Mysore, and P. Smaragdis, "Speech Enhancement by Online Non-negative Spectrogram Decomposition in Non-stationary Noise Environments," in *Proc. Interspeech 2012*, Dec. 2012, pp. 594–597.
[6] Y. Wang, A. Narayanan, and D. Wang, "On Training Targets for Supervised Speech Separation," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 22, no. 12, p. 1849–1858, Dec. 2014.
[7] Y. Xu, J. Du, L. Dai, and C. Lee, "A Regression Approach to Speech Enhancement Based on Deep Neural Networks," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 23, no. 1, pp. 7–19, 2015.
[8] I. Fedorov, M. Stamenovic, C. Jensen, L.-C. Yang, A. Mandell, Y. Gan, M. Mattina, and P. N. Whatmough, "TinyLSTMs: Efficient Neural Speech Enhancement for Hearing Aids," 2020.
[9] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime Neural Pruning," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017, pp. 2181–2191.
[10] D. Neil, J. H. Lee, T. Delbruck, and S.-C. Liu, "Delta Networks for Optimized Recurrent Network Computation," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70, 2017, pp. 2584–2593.
[11] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997.
[12] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *CoRR*, vol. abs/1406.1078, 2014.
[13] J.-M. Valin, "A Hybrid DSP/Deep Learning Approach to Real-Time Full-Band Speech Enhancement," in *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, 2018, pp. 1–5.
[14] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma, "FastGRNN: A Fast, Accurate, Stable and Tiny Kilobyte Sized Gated Recurrent Neural Network," in *NeurIPS*, 2018.
[15] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, vol. 57, Feb 2014, pp. 10–14.
[16] C. Veaux, J. Yamagishi, and K. Macdonald, "CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit," 2017.
[17] G. Andersen, "Akustiske Database for Dansk," 2011. [Online]. Available: https://www.nb.no/sbfil/dok/nst_taledat_dk.pdf
[18] M. C. Green and D. Murphy, "EigenScape," Oct. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1012809
[19] ITU-T Recommendation P.862, "Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," 2001.