INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

GhostRNN: Reducing State Redundancy in RNN with Cheap Operations

Hang Zhou[1,2], *Xiaoxu Zheng*[3], *Yunhe Wang*[2*] *Michael Bi Mi*[3], *Deyi Xiong*[1,4*], *Kai Han*[2]

[1] College of Intelligence and Computing, Tianjin University, Tianjin, China
[2] Huawei Noah's Ark Lab, China
[3] Huawei International Pte Ltd, Singapore
[4] School of Computer Science and Technology, Kashi University, Kashi, China

{zhouhang25, zhengxiaoxu1, yunhe.wang, Michael.Bi.Mi}@huawei.com, dyxiong@tju.edu.cn, kai.han@huawei.com

Abstract

Recurrent neural network (RNNs) that are capable of modeling long-distance dependencies are widely used in various speech tasks, eg., keyword spotting (KWS) and speech enhancement (SE). Due to the limitation of power and memory in low-resource devices, efficient RNN models are urgently required for real-world applications. In this paper, we propose an efficient RNN architecture, GhostRNN, which reduces hidden state redundancy with cheap operations. In particular, we observe that partial dimensions of hidden states are similar to the others in trained RNN models, suggesting that redundancy exists in specific RNNs. To reduce the redundancy and hence computational cost, we propose to first generate a few *intrinsic* states, and then apply cheap operations to produce *ghost* states based on the *intrinsic* states. Experiments on KWS and SE tasks demonstrate that the proposed GhostRNN significantly reduces the memory usage (∼40%) and computation cost while keeping performance similar. Codes will be available at https://gitee.com/mindspore/models/tree/master/research/audio/ghostrnn

Index Terms: RNN, keyword spotting, speech enhancement

1. Introduction

Recent years have witnessed that substantial improvements have been made in a wide range of speech tasks with the rapid development of neural networks. Among these neural networks, RNNs, e.g., LSTMs [1] or GRUs [2], are widely employed in various speech-related tasks in low-resource devices (e.g., mobile phones), such as KWS [3, 4], SE [5, 6], automatic speech recognition [7], acoustic echo cancellation [8, 9], etc., although they are less parallelizable than Transformer [10].

Due to the high demand of AI model deployment on edge devices with limited power and memory, designing efficient models with low computation cost while maintaining high performance is desirable. A variety of efforts have been made in this direction. Dey and Salem [11] propose efficient GRU variants, which reduce the size of the gate matrix by adjusting the calculation method of the gate, e.g, calculating the gate vector with only hidden states as input. The Light-Gated-GRU (Li-GRU) is proposed by [12], which removes the reset gate and developes a single-gate RNN. Batch normalization is also used to optimize the model performance. Amoh and Odame [13] propose the Embedded Gated Recurrent Unit, which has only one gate with the Single Gate Mechanism. Simliar to Li-GRU, Fanta et al. [14] discard the reset gate of GRU and replace the activation function Tanh with Sigmoid in their proposed SITGRU. Zhang et al. [15] compress RNN models with a twin-gated mechanism. Although these methods can effectively reduce the number of parameters of the model, reducing the number of gate matrices may undermine the exploration of contextual information.

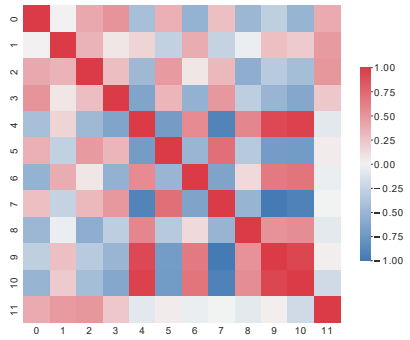


Figure 1: *The cosine similarity matrix of RNN hidden states (first 12 dimensions) in a well-trained DCCRN_GRU with 128 hidden units.*

In this work, we have empirically observed redundancy in hidden states of RNN models in addition to that in gate matrices studied in aforementioned previous works. We hence propose to fully explore the redundancy of hidden states to compress RNNs, which has not been investigated for speech tasks. Particularly, we consider SE task with RNN models and analyze the hidden states of RNN layers. The feature map of hidden states with m hidden units after n time steps can be represented as $states \in \mathbb{R}^{m \times n}$, and the feature map for the i_{th} hidden unit can be represented as $state_i \in \mathbb{R}^{1 \times n}$. Firstly, we perform a singular value decomposition on the feature map of hidden states from a well trained DCCRN_GRU model. We find that only half of the singular values can reach 99% contribution rate of principal component analysis (PCA), which indicates some $state_i$ are relatively redundant. In addition, as shown in Figure 1 where the cosine similarities of $state_i$ at different indexes are calculated , some similarity values are relatively large, indicating a high similarity [16].

Inspired by the above observation, we propose GhostRNN to reduce the redundancy in hidden states and thus construct efficient RNN models. In particular, a small part of RNN hidden states are generated by the vanilla RNN model to serve as intrinsic states. Next, cheap operations including simple linear transformation and activation function, are implemented to generate ghost states based on the intrinsic states. The ghost states are then concatenated with the intrinsic states to serve as the complete feature representation of previous time step and be passed to the next time step for further calculation. Compared to the

*Corresponding author.

10.21437/Interspeech.2023-2417

vanilla RNN matrix multiplication, our cheap operations have fewer FLOPs and parameters, so the computation and memory costs of the GhostRNN model are significantly reduced.

Experiments on the KWS and SE tasks are conducted to examine the effectiveness of our proposed method. Experimental results demonstrate that our method achieves a 0.1% accuracy improvement on the Google Speech Commands dataset while compressing the parameters of baseline model by 40%. In the SE task, our method improves SDR and Si-SDR by approximately 0.1dB with around 40% compression rate.

2. Proposed Method

In this section, we elaborate the proposed GhostRNN with details in model compression. Without loss of generality, we use GRU to illustrate the definition of GhostRNN. Our method can be applicable to other RNNs, e.g., LSTM.

2.1. RNN

RNNs are a class of model structures that utilize hidden states to store and leverage contextual information [17], including popular variants GRUs and LSTMs. As one of the most commonly used RNN models, GRU is a simplified version of the LSTM, which is defined as follows:

$$\boldsymbol{r}_t = \sigma\left(\boldsymbol{W}_{ir}\boldsymbol{x}_t + \boldsymbol{b}_{ir} + \boldsymbol{W}_{hr}\boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{hr}\right) \quad (1)$$

$$\boldsymbol{z}_t = \sigma\left(\boldsymbol{W}_{iz}\boldsymbol{x}_t + \boldsymbol{b}_{iz} + \boldsymbol{W}_{hz}\boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{hz}\right) \quad (2)$$

$$\boldsymbol{c}_t = \tanh\left(\boldsymbol{W}_{ic}\boldsymbol{x}t + \boldsymbol{b}_{ic} + \boldsymbol{r}_t * \left(\boldsymbol{W}_{hc}\boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{hc}\right)\right) \quad (3)$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) * \boldsymbol{c}_t + \boldsymbol{z}_t * \boldsymbol{h}_{(t-1)} \quad (4)$$

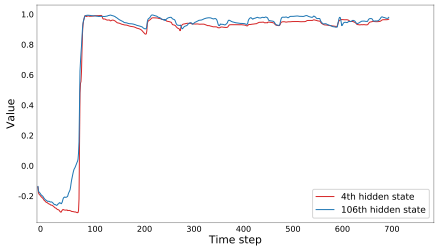
where $\boldsymbol{r}_t$, $\boldsymbol{z}_t$ and $\boldsymbol{c}_t$ are the reset gate, update gate and candidate vector respectively. $\boldsymbol{h}_t$ and $\boldsymbol{h}_{t-1}$ are hidden states at time step t and time step $t-1$, and $\boldsymbol{x}_t$ is the input feature at time step t. As shown in Eq. (1) to (4), six matrices are involved in computation, among which $\boldsymbol{W}_{ir}$, $\boldsymbol{W}_{iz}$ and $\boldsymbol{W}_{ic}$ have the same size, $\boldsymbol{W}_{ir}$, $\boldsymbol{W}_{iz}$ and $\boldsymbol{W}_{ic}$ are also with the same size. In addition, it's worth noting that the size of all the matrices of a GRU model is closely related to both the dimension of its hidden states and input features. Consequently, to compress the number of parameters of GRU, reducing the dimensionality of hidden states is an effective way. Also as mentioned in Section 1, it is feasible to decrease the number of gating matrices by reducing the number of required gating vectors, which ultimately leads to the reduction in the number of parameters of the model.

2.2. GhostRNN

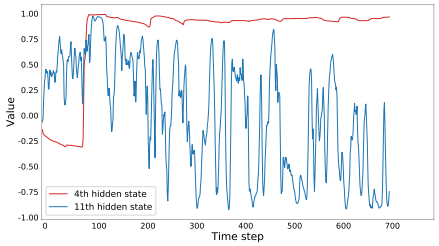
Aiming at reducing hidden state redundancy, our proposed GhostRNN employs extremely low-cost transformations to generate ghost states from intrinsic states.

Observation on the redundancy of hidden states

As discussed in Section 1, previous studies usually focus on decreasing the number of gate matrices for model compression while the redundancy of hidden states is seldom investigated, which is also crucial to effectively reduce the number of model parameters. Therefore, the redundancy of hidden states is thoroughly under investigation in this section. Initially, the PCA contribution rate is adopted as the evaluation metric. The accumulation of hidden state vectors over time is considered as a feature map and the singular value decomposition is performed on it. Based on the result, only approximately half of the singular values are necessary and the feature map can reach a 99%



(a) *Value of the 4th hidden state and 106th hidden state*



(b) *Value of the 4th hidden state and 11th hidden state*

Figure 2: *The value of hidden states at different indexes.*

PCA contribution rate, which indicates that with only about half of hidden states, the complete feature information is possible to be constructed. Furthermore, as shown in Figure 1, we take the accumulation of hidden states of different dimensions over time steps as the state components $state_i$ and calculate the cosine similarity between different components. It shows that the cosine similarity of certain components nearly approaches 1.0. A group of components with highly correlated trends is shown in Figure 2a, which indicates that hidden states of GRU contain redundancy. On the other hand, as shown in Figure 2b, the cosine similarity between specific components is close to 0, indicating that these state components are almost orthogonal and hence necessary and irreplaceable. Therefore keeping the necessary state components and eliminating redundant components is a straightforward and practical way to compress the RNN models.

GhostRNN module

Based on the above results and analysis, we propose the GhostRNN module to construct the ghost states based on the intrinsic states as shown in Figure 3, which can be defined as follows:

$$\boldsymbol{r}_t = \sigma\left(\boldsymbol{W}_{ir}\boldsymbol{x}_t + \boldsymbol{b}_{ir} + \boldsymbol{W}_{hr}[\boldsymbol{h}_{(t-1)}\ \boldsymbol{g}_{(t-1)}] + \boldsymbol{b}_{hr}\right) \quad (5)$$

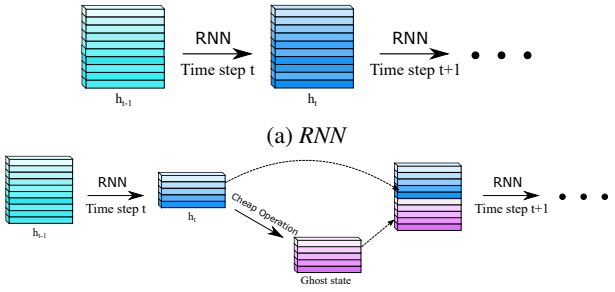
$$\boldsymbol{z}_t = \sigma\left(\boldsymbol{W}_{iz}\boldsymbol{x}_t + \boldsymbol{b}_{iz} + \boldsymbol{W}_{hz}[\boldsymbol{h}_{(t-1)}\ \boldsymbol{g}_{(t-1)}] + \boldsymbol{b}_{hz}\right) \quad (6)$$

$$\begin{aligned}\boldsymbol{c}_t = &\tanh\left(\boldsymbol{W}_{ic}\boldsymbol{x}_t + \boldsymbol{b}_{ic} + \boldsymbol{r}_t * \left(\boldsymbol{W}_{hc}\boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{hc}\right)\right.\\ &\left.+\boldsymbol{W}_{gc}\boldsymbol{g}_{t-1} + \boldsymbol{b}_{gc}\right)\end{aligned} \quad (7)$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) * \boldsymbol{c}_t + \boldsymbol{z}_t * \boldsymbol{h}_{(t-1)} \quad (8)$$

$$\boldsymbol{g}_t = \phi\left(\boldsymbol{h}_t\right) \quad (9)$$

where $\boldsymbol{g}_t$ and $\boldsymbol{g}_{t-1}$ are the ghost states of the GRU model at time step t and time step $t-1$ generated by the original intrinsic states $\boldsymbol{h}_t$ and $\boldsymbol{h}_{t-1}$ through simple linear transformation operations and activations denoted by ϕ. The intrinsic states at



(a) *RNN*

(b) *GhostRNN*

Figure 3: *Schematic diagram of the RNN layer and proposed Ghost RNN.*

time step t are obtained by following Eqs (5) to (8). [ ] represents the concatenation operation to concatenate intrinsic and ghost states. The concatenated states are used as the input for the next time step. Specifically, at time step t the GhostRNN receives both the previous hidden state $\boldsymbol{h}_{t-1}$ and corresponding ghost state $\boldsymbol{g}_{t-1}$ synchronously with the current input feature $\boldsymbol{x}_t$. This process is repeated to complete the calculation of the GhostRNN. As mentioned above, the GhostRNN method, which leverages the redundancy of hidden states, can be applied to compress all RNN models.

2.3. Analysis on the Number of Parameters and Computational Complexity

The number of parameters of vanila GRU and our proposed GhostRNN can be calculated as follows:

$$\mathrm{Param}_{\mathrm{GRU}} = 3\left(\mathrm{Dim}_{\mathrm{feature}} + \mathrm{Dim}_{\mathrm{state}}\right)\mathrm{Dim}_{\mathrm{state}} \tag{10}$$

$$\begin{aligned}\mathrm{Param}_{\mathrm{GhostRNN}} =& 3\left(\mathrm{Dim}_{\mathrm{feature}} + \mathrm{Dim}_{\mathrm{state}}\right)\left(\mathrm{Dim}_{\mathrm{state}}/\mathrm{r}\right) \\ &+ \mathrm{Param}_{\phi}\end{aligned} \tag{11}$$

$$\begin{aligned}\mathrm{Param}_{\phi} &= \left(\mathrm{Dim}_{\mathrm{state}}/\mathrm{r}\right)\left(\mathrm{Dim}_{\mathrm{state}} - \left(\mathrm{Dim}_{\mathrm{state}}/\mathrm{r}\right)\right) \\ &= {\mathrm{Dim}_{\mathrm{state}}}^{2}\left((\mathrm{r}-1)/\mathrm{r}^{2}\right)\end{aligned} \tag{12}$$

where r represents the ratio of the complete state to the intrinsic state. As shown in Eqs (10) to (11), the dimension of output hidden states in GhostRNN is divided by the ratio r compared with those in the vanila GRU. Although an additional cheap operation module consisting of linear layers is applied, according to Eq. (12) the number of parameters of the cheap operation module is far smaller than that of the GRU model. As a result, the total number of parameters of the GhostRNN will be compressed by the factor r. If the cheap operation module is constructed by other calculation methods such as vanilla linear transformation without parameters, the compression ratio can reach up to r. As for the computational complexity, since all the matrices in GRU are linear layers, the computational complexity of GRU is almost proportional to the number of parameters. Thus, the computational complexity of GhostRNN will also be reduced by the same factor r.

3. Experiments

Experiments on two tasks were conducted to evaluate the effectiveness of our method: KWS and SE.

Table 1: *Overall performance of KWS*

System	# Params	# MACs	Accuracy (%)
GRU	498K	24.1M	94.68
GRU	295K	14.2M	94.49
Li-GRU	334K	16.1M	93.49
SITGRU	334K	16.1M	94.26
GhostRNN	292K	14.0M	**94.79**

3.1. Datasets

The Google Speech Commands dataset v0.02 [18] which contains thousands of one-second audio samples divided into 30 categories was used in our experiments for KWS. Following the previous work [3, 19], 12 categories were selected: "yes," "no," "up," "down," "left," "right," "on," "off," "stop," "go," silence, and unknown. We used 36,923, 4,445, and 4,890 of these samples for training, validation, and testing, respectively. The 10-dimensional Mel-frequency cepstral coefficients (MFCC) were used as the speech feature with a window length of 40 ms and a window shift of 20 ms which results in a feature map of size 49x10 for each speech sample and the data augmentation techniques such as adding background noise and random shift as suggested in [3] were employed to enhance the roubustness of the models.

To evaluate the performance of our method on the SE task, we used the LibriMix dataset [20] which generates noisy speech clips by combining clean speech from LibriSpeech [21] and noise from the WHAM! dataset [22]. We used the 16 kHz version of the train-360 data with a total of 50,800 training samples, 3000 validation samples and 3,000 testing samples, resulting in a total of 234 hours of data. We implemented the same data preprocessing as that in Asteroid [23].

3.2. Settings and Evaluation Metrics

During training of KWS, the standard cross-entropy loss and the Adam optimizer with a batch size of 100 were employed. We used a step-down learning rate strategy, where the initial learning rate was 5e-4 with the step setting [10,000,20,000]. All the models were trained from scratch for a total of 30,000 iterations and evaluated by the accuracy metric [3]. To ensure the reliability of our results, each model was trained with the same configuration for three times and the average experiment results are reported here.

During the training process of SE, the permutation invariant loss and the ADAM optimizer were used and the batch size was set to 12 for DCRNN [5] and 32 for GRU-TasNet [24]. The learning rate decay strategy and early stopping strategy were both applied in all experiments. The initial learning rate was set to 0.001 and a weight decay of 1e-5 was applied. In terms of the filterbank, our settings were consistent with those described in [23]. For evaluation, 5 metrics, namely Signal-to-Distortion Ratio (SDR), SDR improvements (SDRi), Scale-Invariant Signal-to-Distortion Ratio (Si-SDR) [25], Si-SDR improvements (Si-SDRi), and Short-Time Objective Intelligibility (STOI) [26] were used.

3.3. Baselines

KWS model

As demonstrated in Table 1, a GRU model of approximately 500k in size and another GRU model of 295k in size were employed as baseline models for KWS [3], while Li-GRU [12] and SITGRU [14] were implemented for comparison.

Table 2: *Performance of DCCRN, DCCRN_Ghost, GRU-TasNet and GhostRNN-TasNet*

System	# Parameters	# RNN MACs[1]	SDR (dB)	SDRi (dB)	Si-SDR (dB)	Si-SDRi (dB)	STOI (%)
DCCRN_GRU128	3.4M	0.7M	13.99	10.49	13.47	10.02	91.6
DCCRN_GRU80	3.1M	0.4M	13.89	10.39	13.36	9.91	91.4
DCCRN_Ghost128	3.1M	0.4M	**13.93**	**10.43**	**13.40**	**9.95**	**91.5**
GRU512-TasNet	5.6M	4.7M	13.14	9.64	12.63	9.18	89.5
GRU384-TasNet	3.4M	2.8M	13.09	9.59	12.56	9.11	89.4
GhostRNN512-TasNet	3.4M	2.6M	**13.26**	**9.76**	**12.73**	**9.28**	**89.7**
GRU192-TasNet	1.6M	0.8M	12.82	9.31	12.28	8.83	88.9
GRU136-TasNet	1.2M	0.5M	12.48	8.98	11.92	8.47	88.1
GhostRNN192-TasNet	1.2M	0.5M	**12.61**	**9.10**	**12.05**	**8.60**	**88.4**

[1] The MACs of RNN with single time step is presented to show the difference in computation cost clearly.

SE models

Two types of models were selected as the baselines model for comparison for the SE task. The first one is the DCCRN [5], which is mainly based on convolution layers and supplemented with RNN layers. The second one is the GRU-TasNet [24], in which RNN is the primary module. A brief introduction of them is provided below:

- DCCRN. This model consists of three main components: a convolution encoder, a transpose convolution decoder and an RNN module. In our experiment, we chose the DCCRN-CL [5] model and replaced the LSTM with GRU, in which two sizes of the hidden unit 128 and 80 were chosen to construct the different baseline models with different size.
- GRU-TasNet. This model is optimized by the Time-domain Audio Separation Network, which consists of three parts: a 1-D convolutional encoder, a 1-D deconvolutional decoder, and a Deep LSTM separation module [24]. In our experiments, the LSTM was replaced with GRU. Four baseline models with different sizes were designed, differing in the hidden size of the GRU: 512, 384, 192, and 136.

3.4. Results on KWS

Table 1 presents the experiment results and model parameters. Our proposed GhostRNN model is compared with two baselines of 500k GRU and 300k GRU, as well as two other model compression methods: Li-GRU [12] and SITGRU [14]. The results show that our GhostRNN with about 40% fewer parameters achieves approximately a 0.1% improvement on the accuracy rate over the 500k GRU model and also outperforms the Li-GRU and SIT-GRU models with slightly more parameters, which indicates the effectiveness of our proposed GhostRNN.

3.5. Results on SE

Table 2 presents the results of DCCRN and DCCRN_Ghost128 on the librimix1 dataset. The results indicate that pruning the hidden size of the GRU layer in the DCCRN model by 10% for model compression leads to a decrease of approximately 0.1 dB in both SDR and Si-SDR metrics. In contrast, when applying our proposed GhostRNN compression method and compressing approximately 10% of the parameter, the SDR and Si-SDR metrics only decrease by approximately 0.05 dB. These findings clearly demonstrate the effectiveness of GhostRNN.

Table 2 presents the results of GRU-TasNet and GhostRNN-TasNet on the librimix1 dataset. The metrics show a slight decrease when the model is compressed from GRU512-TasNet to GRU384-TasNet, indicating that GRU512-TasNet has redundant parameters. In this case, the GhostRNN method yields a performance improvement of over 0.1 dB in SDR and Si-SDR, with 40% fewer parameters than GRU512-TasNet. However, when the model is further compressed from 1.6M (GRU192-TasNet) to 1.2M (GRU136-TasNet), the performance drops noticeably, demonstrating that the model has low redundancy. In this scenario, GhostRNN192 has an advantage of approximately 0.13 dB in SDR and Si-SDR compared to GRU136-TasNet. In summary, GhostRNN is an effective compression method for RNN models.

4. Conclusions

In this paper, we have presented GhostRNN for RNN model compression based on the observation of the redundancy in hidden states. In our GhostRNN, given the intrinsic hidden states, the extreme low-cost transformation layers are applied to generate the ghost states which significantly reduces the number of parameters and the computation cost of the vanllia GRU model but achieves competitive performance. Experimental results demonstrate that our method achieves a 0.1% accuracy improvement on the Google Speech Commands dataset while compressing the parameters of baseline model by 40%. In the SE task, our method improves SDR and Si-SDR by approximately 0.1 dB with around 40% compression rate. Additionally, our method outperforms the GRU based model with the same number of parameters by approximately 0.13 dB in terms of SDR and other evaluation metrics. Overall, the proposed GhostRNN is a simple yet effective method for RNN model compressing. In the future work, it is worth to investigate the extension of GhostRNN to other RNN structures, such as LSTM, and further explore novel ghost state generation methods to achieve better balance on the reduction of the model computational complexity and performance. Additionally, we plan to explore the potential benefits of combining GhostRNN with other existing RNN compression techniques.

5. Acknowledgements

Deyi Xiong was partially supported by the Natural Science Foundation of Xinjiang Uygur Autonomous Region (No. 2022D01D43). We gratefully acknowledge the support of MindSpore [27], CANN(Compute Architecture for Neural Networks) and Ascend AI Processor used for this research. We would like to thank the anonymous reviewers for their insightful comments.

6. References

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

[2] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[3] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017.

[4] O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. Laurenzo, "Streaming keyword spotting on mobile devices," *arXiv preprint arXiv:2005.06720*, 2020.

[5] Y. Hu, Y. Liu, S. Lv, M. Xing, S. Zhang, Y. Fu, J. Wu, B. Zhang, and L. Xie, "Dccrn: Deep complex convolution recurrent network for phase-aware speech enhancement," *arXiv preprint arXiv:2008.00264*, 2020.

[6] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-path rnn: efficient long sequence modeling for time-domain single-channel speech separation," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 46–50.

[7] K. Rao, H. Sak, and R. Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 193–199.

[8] L. Pfeifenberger and F. Pernkopf, "Nonlinear residual echo suppression using a recurrent neural network." in *Interspeech*, 2020, pp. 3950–3954.

[9] L. Ma, H. Huang, P. Zhao, and T. Su, "Acoustic echo cancellation by combining adaptive digital filter and recurrent neural network," *arXiv preprint arXiv:2005.09237*, 2020.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[11] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (gru) neural networks," in *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE, 2017, pp. 1597–1600.

[12] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light gated recurrent units for speech recognition," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 92–102, 2018.

[13] J. Amoh and K. M. Odame, "An optimized recurrent unit for ultra-low-power keyword spotting," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 2, pp. 1–17, 2019.

[14] H. Fanta, Z. Shao, and L. Ma, "Sitgru: single-tunnelled gated recurrent unit for abnormality detection," *Information Sciences*, vol. 524, pp. 15–32, 2020.

[15] B. Zhang, D. Xiong, J. Su, Q. Lin, and H. Zhang, "Simplifying neural machine translation with addition-subtraction twin-gated recurrent networks," *arXiv preprint arXiv:1810.12546*, 2018.

[16] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "Ghostnet: More features from cheap operations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1580–1589.

[17] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[18] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[19] E. T. Mekonnen, A. Brutti, and D. Falavigna, "End-to-end low resource keyword spotting through character recognition and beam-search re-scoring," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8182–8186.

[20] J. Cosentino, M. Pariente, S. Cornell, A. Deleforge, and E. Vincent, "Librimix: An open-source dataset for generalizable speech separation," *arXiv preprint arXiv:2005.11262*, 2020.

[21] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[22] G. Wichern, J. Antognini, M. Flynn, L. R. Zhu, E. McQuinn, D. Crow, E. Manilow, and J. L. Roux, "Wham!: Extending speech separation to noisy environments," *arXiv preprint arXiv:1907.01160*, 2019.

[23] M. Pariente, S. Cornell, J. Cosentino, S. Sivasankaran, E. Tzinis, J. Heitkaemper, M. Olvera, F.-R. Stöter, M. Hu, J. M. Martín-Doñas, D. Ditter, A. Frank, A. Deleforge, and E. Vincent, "Asteroid: the PyTorch-based audio source separation toolkit for researchers," in *Proc. Interspeech*, 2020.

[24] Y. Luo and N. Mesgarani, "Real-time single-channel dereverberation and separation with time-domain audio separation network." in *Interspeech*, 2018, pp. 342–346.

[25] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "Sdr–half-baked or well done?" in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 626–630.

[26] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2125–2136, 2011.

[27] Huawei, "Mindspore," https://www.mindspore.cn/, 2020.