ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10096135

STRUCTURED STATE SPACE DECODER FOR SPEECH RECOGNITION AND SYNTHESIS

Koichi Miyazaki, Masato Murata, Tomoki Koriyama

CyberAgent, Inc.
{miyazaki_koichi_xa, murata_masato, koriyama_tomoki}@cyberagent.co.jp

ABSTRACT

Automatic speech recognition (ASR) systems developed in recent years have shown promising results with self-attention models (e.g., Transformer and Conformer), which are replacing conventional recurrent neural networks. Meanwhile, a structured state space model (S4) has been recently proposed, producing promising results for various long-sequence modeling tasks, including raw speech classification. The S4 model can be trained in parallel, similar to the Transformer model. In this study, we applied S4 as a decoder for ASR and text-to-speech (TTS) tasks, respectively, by comparing it with the Transformer decoder. For the ASR task, our experimental results demonstrate that the proposed model achieves a competitive word error rate (WER) of 1.88%/4.25% on the LibriSpeech test-clean/test-other set and a character error rate (CER) of 3.80%/2.63%/2.98% on the CSJ eval1/eval2/eval3 set. Furthermore, the proposed model is more robust than the standard Transformer model, particularly for long-form speech on both the datasets. In the TTS task, the proposed method outperforms the Transformer baseline.

Index Terms— Automatic speech recognition, text-to-speech, state space model, S4

1. INTRODUCTION

End-to-end automatic speech recognition (E2E-ASR) has become popular because of its simple training process and high recognition accuracy [1]–[3]. Generally, E2E-ASR is based on a sequence-to-sequence framework consisting of an encoder that processes acoustic features and a decoder that outputs linguistic information such as phonemes and characters. Many recent E2E-ASR systems are developed based on a Transformer [4] that uses self-attention layers in the encoder and decoder [5]. As a noticeable example, Gulati *et al.* proposed Conformer [6] that incorporates a convolutional neural network (CNN) into the encoder to explicitly capture local features and achieved state-of-the-art performance.

Transformer-based models have also achieved promising results in the speech field for various tasks such as text-to-speech (TTS), speech translations, and speech separation [7]. The advantage of self-attention is that it has a flexible function based on the similarity matrix that can capture global characteristics. However, self-attention has a computational complexity problem for long sequences because both the computation time and memory usage are quadratic in the sequence length. To address these issues, customized attention layers have been proposed to reduce attention computational complexity [8], [9].

Furthermore, as self-attention itself has no positional information for handling token order, such information must be explicitly provided as additional information. The position information in the vanilla Transformer is *positional encoding* in which the absolute position information is represented by a set of sinusoidal curves [4]. This absolute position information causes overfitting problems to the sequence lengths in training data, which results in the performance degradation for long-form sequences. Although one solution involves the use of relative position information [10]–[12], it tends to increase the computational complexity and cause incompatibility with the customized attention layers.

An alternative approach is to use other flexible function layers that can explicitly capture the position information. In this study, we focus on a structured state space model (S4) [13], [14] in which the relationship among latent state spaces are represented by linear transformation. S4 has the characteristics of recurrent neural network (RNN) that can be applied to autoregressive generation without masking and save memory usage during inference unlike Transformer. In addition, S4 has the property of CNN that enables parallel computing. S4 solves the computational complexity and position information issues in the Transformer and has outperformed it on several tasks [13]. It has also been reported that S4 is effective in autoregressive inference such as waveform generation [15], language modeling [13], and time-series forecasting [13].

In this paper, we propose an S4-based decoder for the sequence-to-sequence speech model. Specifically, we replace the self-attention-based decoder of Conformer ASR with the S4-based one. We expect that the performance of S4 on autoregressive inference can also be seen in the E2E-ASR framework. Furthermore, we evaluate the performance of the S4 decoder in autoregressive TTS. Our experimental evaluation results on ASR tasks demonstrate that our proposed model achieves a competitive recognition accuracy on the datasets of LibriSpeech and Corpus of Spontaneous Japanese (CSJ) compared with the Transformer and Conformer models. Furthermore, we show that our proposed model is more robust than the standard Transformer ASR, particularly for long-form speech. We also show that TTS with the S4 decoder enhances the naturalness of synthetic speech compared with Transformer-TTS.

2. S4 DECODER FOR SEQUENCE-TO-SEQUENCE SPEECH MODELS

2.1. Structured state space model (S4)

S4 is based on a linear state space layer (LSSL) [14]. Let $\mathbf{u}(t) \in \mathbb{R}^{D_{\text{in}}}$ and $\mathbf{y}(t) \in \mathbb{R}^{D_{\text{out}}}$ be the input and output continuous-time sequences and $\mathbf{x}(t) \in \mathbb{R}^{N}$ be a latent space sequence; then, the output sequence can be obtained with the following equation:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \tag{1}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t). \tag{2}$$

Using bilinear discretization, the LSSL for the discrete-time sequence sampled with a trainable step size Δ can be represented

as

$$\mathbf{x}_k = \overline{\mathbf{A}}\mathbf{x}_{k-1} + \overline{\mathbf{B}}\mathbf{u}_k, \tag{3}$$

$$\mathbf{y}_k = \overline{\mathbf{C}}\mathbf{x}_k + \overline{\mathbf{D}}\mathbf{u}_k, \tag{4}$$

$$\overline{\mathbf{A}} = (\mathbf{I} - \Delta/2 \cdot \mathbf{A})^{-1}(\mathbf{I} + \Delta/2 \cdot \mathbf{A}), \tag{5}$$

$$\overline{\mathbf{B}} = (\mathbf{I} - \Delta/2 \cdot \mathbf{A})^{-1}\Delta\mathbf{B}, \tag{6}$$

$$\overline{\mathbf{C}} = \mathbf{C}, \quad \overline{\mathbf{D}} = \mathbf{D}. \tag{7}$$

As indicated by the equations, the LSSL has the characteristics of an RNN.

The architecture of LSSL can be considered as a CNN. By setting $\mathbf{x}_{-1} = \mathbf{0}$ and unrolling Eq. (3), we obtain

$$\mathbf{y}_k = \overline{\mathbf{C}}\overline{\mathbf{A}}^k\overline{\mathbf{B}}\mathbf{u}_0 + \cdots + \overline{\mathbf{C}}\overline{\mathbf{A}}\overline{\mathbf{B}}\mathbf{u}_{k-1} + \overline{\mathbf{C}}\overline{\mathbf{B}}\mathbf{u}_k + \overline{\mathbf{D}}\mathbf{u}_k. \tag{8}$$

Hence, the output $\mathbf{y}_k$ was calculated by a convolution kernel $\overline{\mathbf{K}}$ as follows:

$$\mathbf{y}_k = (\overline{\mathbf{K}} * (\mathbf{u}_{k-L}, \ldots, \mathbf{u}_k)) + \overline{\mathbf{D}}\mathbf{u}_k, \tag{9}$$

$$\overline{\mathbf{K}} = (\overline{\mathbf{C}}\overline{\mathbf{B}}, \overline{\mathbf{C}}\overline{\mathbf{A}}\overline{\mathbf{B}}, \ldots, \overline{\mathbf{C}}\overline{\mathbf{A}}^{L-1}\overline{\mathbf{B}}), \tag{10}$$

where L is the kernel size. Thus, we can utilize the fast parallel computation with GPU by calculating the convolution kernel in advance.

The LSSL has problems with regard to the instability of the state space sequence and the computational complexity in the kernel calculation. Thus, a structured state space model called *S4* is proposed to overcome the problems by restricting the state matrix $\mathbf{A}$ as a normal plus low-rank (NPLR) matrix [13][1], which is parameterized by $\mathbf{A} = \mathrm{diag}[\boldsymbol{\lambda}] - \mathbf{p}\mathbf{p}^*$ and $\boldsymbol{\lambda} \in \mathbb{C}^N$, $\mathbf{p} \in \mathbb{C}^N$. We refer to this LSSL layer as the *S4 layer*. This parameterization is a structured representation that allows faster computation based on efficient computation algorithms.

2.2. S4 decoder and its application to sequence-to-sequence speech modeling

We propose an S4 decoder that inherits the Transformer decoder architecture [4], which consists of stacks of a feed-forward block, a multi-head attention block, and a masked multi-head self-attention block. Specifically, we replaced the masked multi-head self-attention block with an S4 block and removed positional encoding to compose our proposed model (see Fig. 1). We utilized source-target attention to connect the encoder output and the S4 decoder in the same way as the Transformer decoder. Each block contains a residual connection, dropout, and layer normalization [16]. We employed a linear layer and gated linear unit (GLU) [17] activation for ensuring non-linearity after the S4 layer. Unlike self-attention, S4 does not require positional encoding to handle positional information. Therefore, we feed input vectors to S4 blocks without adding positional information. While training the S4 decoder, we performed parallel computing using the convolution kernel of S4. The prediction was executed in an autoregressive manner based on the RNN nature of S4.

In this paper, we applied the proposed S4 decoder to sequence-to-sequence ASR and TTS models. In ASR, a text sequence was predicted in an autoregressive manner using the S4 decoder. In TTS, the S4 decoder was used to generate acoustic features such as mel-spectrograms.

[1]We referred to the updated version available at `https://arxiv.org/abs/2111.00396v3`

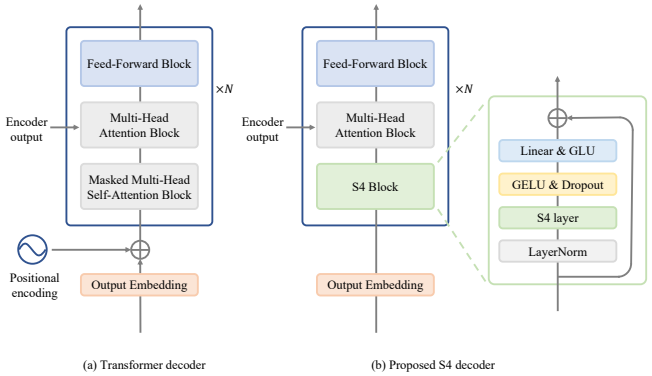


Fig. 1. *Overview of our proposed S4 decoder architecture. (a) illustration of the Transformer decoder, and (b) our proposed S4 decoder. Compared to the Transformer decoder, the masked multi-head self-attention block is replaced with the S4 block and positional encoding is removed.*

3. EXPERIMENTS

To verify the effectiveness of the proposed method, we evaluated its performance on ASR and TTS tasks. The proposed model was compared by simply replacing the self-attention of the baseline transformer decoder with the S4 layer.

3.1. Automatic speech recognition

3.1.1. Experimental conditions

We evaluated our proposed model on two corpora: CSJ [18] and LibriSpeech [19]. The CSJ corpus contains 581 h of Japanese speech sampled at 16 kHz and its transcription. The LibriSpeech corpus contains 960 h of English speech sampled at 16 kHz for training an acoustic model and an additional 800M word token text-only corpus for building the language model. For the evaluation on CSJ, we used the same evaluation sets as in ESPnet. For the evaluation on LibriSpeech, we used the sets of dev-clean, dev-other, test-clean, and test-other. Each evaluation set contained 5 h data, and the "other" sets were more challenging to recognize than the "clean" ones.

For all the experiments, we used the ESPnet [20] toolkit for training and evaluation. The basic configuration and preprocessing followed the LibriSpeech Conformer recipe[2], which consists of 12 encoder layers and 6 decoder layers. We trained 50 epochs for the CSJ corpus and 60 epochs for the LibriSpeech corpus using the AdamW [21] optimizer with an exponential learning rate decay scheduler (40,000 steps for warmup and a peak learning rate of 0.025). Moreover, we excluded the weight decay from the embedding layer, normalization layer, bias parameters, and S4 parameters. The numbers of dimensions in the hidden layer and state space were 512 and 64, respectively. We employed SpecAugment [22] and speed perturbation[23] as data augmentation. After finishing all the training epochs, we applied model averaging among the weights of the 10-best validation accuracy models during training. For the decoding process, we used beam search decoding. The beam size was 25 for CSJ and 60 for LibriSpeech. For each residual connection, we applied stochastic depth ($p = 0.1$) regularization [24].

[2]`https://github.com/espnet/espnet/blob/master/egs2/librispeech/asr1/conf/tuning/train_asr_conformer10_hop_length160.yaml`

Table 1. *CER[%] results on CSJ. The values except "This work" are those reported in other reference papers. #Params(M) refers to the number of parameters in millions, and † refers to the result obtained using LM rescoring.*

Method	#Params(M)	CER[%](↓)		
		eval1	eval2	eval3
AED				
Transformer [25]†	-	4.7	3.7	3.9
Conformer [7]†	91	4.5	3.3	3.6
Transducer				
Conformer [26]	120	4.1	3.2	3.5
This work (AED)				
Transformer dec.	113.5	3.81	2.82	3.12
S4 dec.	110.5	**3.80**	**2.63**	**2.98**

3.1.2. ASR results

Table 1 shows the results of the CER on CSJ. We confirmed that S4 decoder (S4 dec.) yielded lower CER values compared with the Transformer decoder (Transformer dec.) on all eval1/eval2/eval3 sets. Table 2 shows the results of our model on the LibriSpeech dataset compared with the result of recently published models. In our experiment comparing S4 decoder with Transformer decoder, the WERs without the language model (LM) of S4 decoder were lower than those of Transformer decoder, and the WERs with the LM were comparable to those of Transformer decoder

3.1.3. Robustness on long-form input

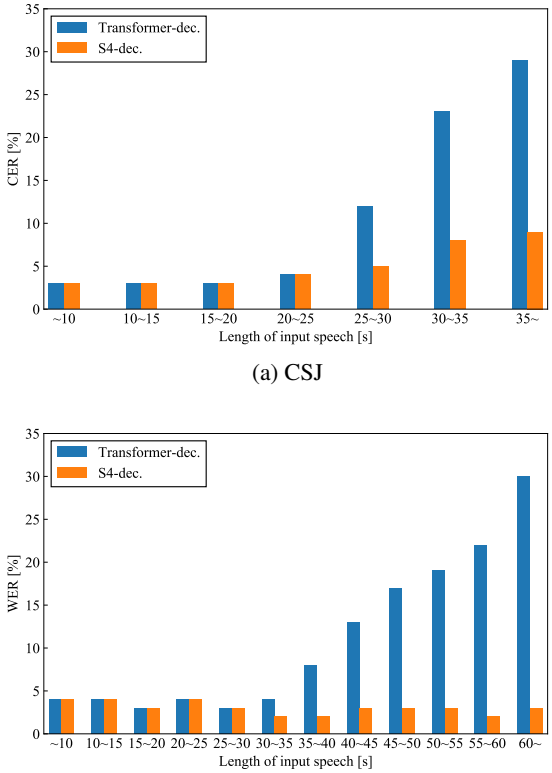
We evaluated our model on the long-form utterance input situation because this also occurs in a real environment (e.g., difficult situations for voice activity detection, such as multi-speaker environment and noisy situations). Pan *et al.* [27] showed that the positional encoding layer in the Transformer-based model caused a lack of robustness in long-form input scenarios. Further, we verified the robustness of S4 in long-form input scenarios by comparing it with the Transformer model. We followed the experiment settings in [27]. First, we prepared new evaluation datasets containing long-form audio. We concatenated these consecutive three utterances in the CSJ/LibriSpeech corpora into one long-form speech utterance. Second, we compared each result of the CER/WER for the Transformer decoder and the S4 decoder on both the new evaluation datasets.

Fig. 2 shows the CER/WER distribution in terms of the speech length. We found that the S4 decoder was more robust than the Transformer decoder, particularly for audio longer than 30 s on both the datasets, consistent with the result reported in [27]. The results suggest that S4 better dealt with long-form sequences whose lengths were unseen during training than the Transformer model. We speculated that the cause for the difference of error rates could be that the Transformer model had a non-trivial positional encoding layer; therefore, it could not deal with such a long-form audio. However, the S4 model contained positional information implicitly in their model architecture, instead of positional encoding.

3.2. Text-to-speech

3.2.1. Experimental conditions

We evaluated the effectiveness of the S4 decoder using the TTS task. We used the ESPnet2-TTS [31] framework with the LJSpeech [32] dataset, containing 24 h of audiobook speech uttered by a single female speaker. The dataset had 13,100 utterances, and we used 250 utterances each for the development and evaluation set. For the proposed TTS task with the S4 decoder, we followed the Transformer recipe[3] by simply replacing the self-attention layer with a S4 layer in the Transformer decoder. For the baseline methods, we used two autoregressive models (Tacotron2 [33] and Transformer-TTS [34]) and one non-autoregressive model (Conformer-FastSpeech2 (CFS2) [7]). We used pre-trained models available on `espnet_model_zoo`[4]. Each model outputs mel-spectrograms as acoustic features. For the conversion from the generated mel-spectrogram to waveforms, we used the pre-trained HiFi-GAN vocoder [35][5] with the same split dataset.



(a) CSJ

(b) LibriSpeech

Fig. 2. *(a) CER on CSJ/(b) WER on LibriSpeech distribution by the speech length.*

3.2.2. TTS results

We observed the loss curves of the proposed model with the S4 decoder. Fig. 3 shows the L1 loss between the target and the generated acoustic features compared to the Transformer decoder. The S4 decoder was found to converge to a lower value than that of the Transformer decoder. This result suggests that the S4 decoder has the potential for better generalization.

[3] `https://github.com/espnet/espnet/blob/master/egs2/ljspeech/tts1/conf/tuning/train_transformer.yaml`

[4] `https://github.com/espnet/espnet_model_zoo`

[5] `https://github.com/kan-bayashi/ParallelWaveGAN`

Table 2. *WER[%] results on LibriSpeech. The values except "This work" are those reported in the reference papers. #Params(M) refers to the number of parameters in millions. AED refers to the attention-based encoder-decoder architecture. In our experiments, we trained the Transformer language model followed by the ESPnet recipe, and we obtained 30.88 of test perplexity.*

Method	#Params(M)	LM	WER[%](↓) w/ LM				WER[%](↓) w/o LM			
			dev-clean	dev-other	test-clean	test-other	dev-clean	dev-other	test-clean	test-other
AED										
Transformer [25]	-	Transformer	2.2	5.6	2.7	5.7	-	-	-	-
Conformer[2]	116.2	Transformer	1.8	4.1	1.9	4.3	2.1	5.4	2.3	5.4
SRU++ [27]	-	Transformer	1.9	4.8	2.0	4.7	-	-	-	-
Branchformer [28]	116.2	Transformer	1.9	4.2	2.1	4.5	2.2	5.5	2.4	5.5
Transducer										
Transformer[29]	139	Transformer	-	-	2.0	4.6	-	-	2.4	5.6
Conformer(L) [6]	118.8	LSTM	-	-	1.9	3.9	-	-	2.1	4.3
ContextNet(L) [30]	112.7	LSTM	-	-	1.9	4.1	-	-	2.1	4.6
This work (AED)										
Transformer dec.	116.2	Transformer	1.81	**3.98**	1.95	**4.21**	2.18	5.50	2.43	5.53
S4 dec.	113.2	Transformer	**1.72**	4.10	**1.88**	4.25	**2.07**	**5.31**	**2.29**	**5.13**

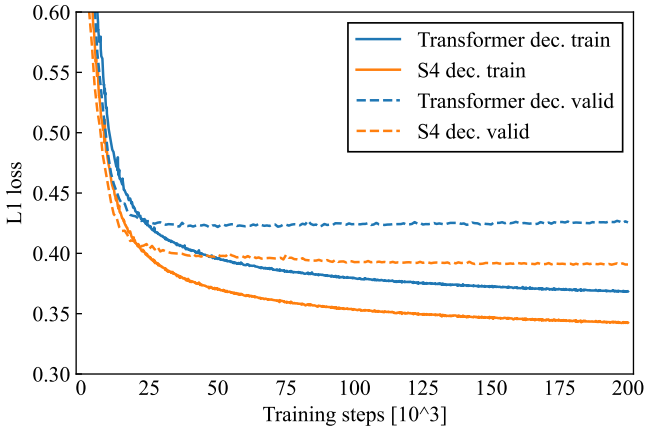


Fig. 3. *L1 loss curves between target and generated acoustic feature. Training loss curves are plotted using solid line, and validation loss curves are plotted using dashed line.*

We evaluated the performance of the proposed TTS with the S4 decoder via objective and subjective evaluations. For objective evaluation, we used the mel-cepstral distortion (MCD), log-F_0 root-mean-square error (logF_0RMSE) with the F_0 range restricted to [80, 400] (Hz), and CER. To calculate the CER, we used a pre-trained ASR model[6] and removed punctuation marks from the target text, and added 0.25 s of silence segments to the start and ending of the audio as a preprocessing. Table 3 shows the results of the objective evaluation. The experimental results show that the autoregressive model underperformed non-autoregressive models. This is because autoregressive TTS models suffered from attention errors, which caused misalignments between the input text and output acoustic feature. The distortions and errors of autoregressive models will be mitigated by applying the dedicated methods proposed [36] to overcome this misalignment problem. However, we did not apply them in this experiment and will address them in the future. Nevertheless, S4 decoder yielded lower MCD and logF_0RMSE than other autoregressive models.

Our subjective evaluation was conducted by a mean opinion score (MOS) test on Amazon Mechanical Turk. The number of participants was 50 and each participant listened to 30 speech samples composed of five randomly chosen sentences with six methods including recorded and vocoder reconstructed samples. Participants rated the naturalness of samples on a five-point scale (1 = bad, 2 = poor, 3 = fair, 4 = good, and 5 = excellent). Table 3 shows the MOS result. Our proposed S4 decoder yielded higher MOS than the Transformer, which indicated better acoustic feature generation than the self-attention. Furthermore, despite the existence of a misalignment, the MOS of the S4 decoder outperformed that of CFS2.

[6]https://zenodo.org/record/4037458

Table 3. *TTS results. GT refers to the recording sample, GT (mel) refers to a reconstructed sample with vocoder, and CI refers to 95 % confidence interval*

Method	MCD[dB](↓)	logF_0RMSE(↓)	CER[%](↓)	MOS(↑)±CI
GT	N/A	N/A	1.0	4.33 ± 0.10
GT(mel)	2.64	0.110	1.1	4.08 ± 0.11
Tacotron2	7.18	0.280	2.0	3.47 ± 0.13
Transformer	7.02	0.255	3.5	3.74 ± 0.12
CFS2	**6.46**	**0.227**	**1.2**	3.70 ± 0.13
S4 dec. (ours)	6.87	0.243	2.7	$\mathbf{3.92 \pm 0.12}$

4. CONCLUSION

In this study, we evaluated the effectiveness of the S4 decoder on ASR and TTS tasks. The S4 decoder produced a comparable performance with the Transformer decoder. We found that our S4 decoder could handle long-form sequence inputs without performance degradation. Moreover, our S4 decoder achieved better generalization performance than the Transformer decoder on a TTS task. Therefore, we believe that the S4 decoder has good application potential in various autoregressive models and tasks that require them. In the future, we plan to investigate the effects of using S4 as a transducer model and apply it to the encoder part. Furthermore, we aim to identify the relationship between lack of robustness and positional encoding layer in ASR tasks.

5. REFERENCES

[1] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. ICML*, 2014, pp. 1764–1772.

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

[3] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid CTC/attention architecture for end-to-end speech recognition," in *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, 2017, pp. 1240–1253.

[4] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Proc. NeurIPS*, vol. 30, 2017.

[5] L. Dong, S. Xu, and B. Xu, "Speech-Transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *Proc. ICASSP*, 2018, pp. 5884–5888.

[6] A. Gulati, J. Qin, C.-C. Chiu, *et al.*, "Conformer: Convolution-augmented Transformer for speech recognition," in *Proc. INTERSPEECH*, 2020, pp. 5036–5040.

[7] P. Guo, F. Boyer, X. Chang, *et al.*, "Recent developments on espnet toolkit boosted by conformer," in *Proc. ICASSP*, 2021, pp. 5874–5878.

[8] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are RNNs: Fast autoregressive Transformers with linear attention," in *Proc. ICML*, 2020, pp. 5156–5165.

[9] M. Zaheer, G. Guruganesh, K. A. Dubey, *et al.*, "Big Bird: Transformers for longer sequences," in *Proc. NeurIPS*, vol. 33, 2020, pp. 17 283–17 297.

[10] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proc. NAACL-HLT*, 2018, pp. 464–468.

[11] Z. Dai, Z. Yang, Y. Yang, *et al.*, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proc. ACL*, 2019, pp. 2978–2988.

[12] B. Wang, L. Shang, C. Lioma, *et al.*, "On position embeddings in BERT," in *Proc. ICLR*, 2020.

[13] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," in *Proc. ICLR*, 2022.

[14] A. Gu, I. Johnson, K. Goel, *et al.*, "Combining recurrent, convolutional, and continuous-time models with linear state-space layers," in *Proc. NeurIPS*, vol. 34, 2021.

[15] K. Goel, A. Gu, C. Donahue, and C. Ré, "It's raw! audio generation with state-space models," *arXiv preprint arXiv:2202.09729*, 2022.

[16] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[17] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. ICML*, 2017, pp. 933–941.

[18] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous speech corpus of Japanese," in *Proc. LREC*, 2000.

[19] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: An ASR corpus based on public domain audio books," in *Proc. ICASSP*, 2015, pp. 5206–5210.

[20] S. Watanabe, T. Hori, S. Karita, *et al.*, "ESPnet: End-to-end speech processing toolkit," in *Proc. INTERSPEECH*, 2018, pp. 2207–2211.

[21] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[22] D. S. Park, W. Chan, Y. Zhang, *et al.*, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. INTERSPEECH*, 2019, pp. 2613–2617.

[23] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proc. INTERSPEECH*, 2015, pp. 3586–3589.

[24] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. ECCV*, 2016, pp. 646–661.

[25] S. Karita, N. Chen, T. Hayashi, *et al.*, "A comparative study on Transformer vs RNN in speech applications," in *Proc. ASRU*, 2019, pp. 449–456.

[26] S. Karita, Y. Kubo, M. Bacchiani, and L. Jones, "A comparative study on neural architectures and training methods for Japanese speech recognition," in *Proc. INTERSPEECH*, 2021.

[27] J. Pan, T. Lei, K. Kim, K. J. Han, and S. Watanabe, "SRU++: Pioneering fast recurrence with attention for speech recognition," in *Proc. ICASSP*, 2022, pp. 7872–7876.

[28] Y. Peng, S. Dalmia, I. Lane, and S. Watanabe, "Branchformer: Parallel MLP-attention architectures to capture local and global context for speech recognition and understanding," in *Proc. ICML*, 2022, pp. 17 627–17 643.

[29] Q. Zhang, H. Lu, H. Sak, *et al.*, "Transformer transducer: A streamable speech recognition model with Transformer encoders and RNN-T loss," in *Proc. ICASSP*, 2020, pp. 7829–7833.

[30] W. Han, Z. Zhang, Y. Zhang, *et al.*, "ContextNet: Improving convolutional neural networks for automatic speech recognition with global context," in *Proc. INTERSPEECH*, 2020, pp. 3610–3614.

[31] T. Hayashi, R. Yamamoto, T. Yoshimura, *et al.*, "ESPnet2-TTS: Extending the edge of TTS research," *arXiv preprint arXiv:2110.07840*, 2021.

[32] K. Ito and L. Johnson, *The LJ speech dataset*, `https://keithito.com/LJ-Speech-Dataset/`, 2017.

[33] J. Shen, R. Pang, R. J. Weiss, *et al.*, "Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions," in *Proc. ICASSP*, 2018, pp. 4779–4783.

[34] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with Transformer network," in *Proc. AAAI*, vol. 33, 2019, pp. 6706–6713.

[35] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Proc. NeurIPS*, vol. 33, 2020, pp. 17 022–17 033.

[36] M. He, Y. Deng, and L. He, "Robust sequence-to-sequence acoustic modeling with stepwise monotonic attention for neural TTS," in *Proc. INTERSPEECH*, 2019, pp. 1293–1297.