Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Efficient decoding self-attention for end-to-end speech synthesis*

Wei ZHAO[1,2], Li XU‡[1,2]

[1]*College of Electrical Engineering, Zhejiang University, Hangzhou 310027, China*

[2]*Institute of Robotics, Zhejiang University, Yuyao 315400, China*

E-mail: weizhao_ee@zju.edu.cn; xupower@zju.edu.cn

Received Oct. 21, 2021; Revision accepted Jan. 9, 2022; Crosschecked Apr. 12, 2022; Published online May 31, 2022

Abstract: Self-attention has been innovatively applied to text-to-speech (TTS) because of its parallel structure and superior strength in modeling sequential data. However, when used in end-to-end speech synthesis with an autoregressive decoding scheme, its inference speed becomes relatively low due to the quadratic complexity in sequence length. This problem becomes particularly severe on devices without graphics processing units (GPUs). To alleviate the dilemma, we propose an efficient decoding self-attention (EDSA) module as an alternative. Combined with a dynamic programming decoding procedure, TTS model inference can be effectively accelerated to have a linear computation complexity. We conduct studies on Mandarin and English datasets and find that our proposed model with EDSA can achieve 720% and 50% higher inference speed on the central processing unit (CPU) and GPU respectively, with almost the same performance. Thus, this method may make the deployment of such models easier when there are limited GPU resources. In addition, our model may perform better than the baseline Transformer TTS on out-of-domain utterances.

Key words: Efficient decoding; End-to-end; Self-attention; Speech synthesis
https://doi.org/10.1631/FITEE.2100501 **CLC number:** TN912.3

1 Introduction

Recent advances in deep learning have substantially promoted text-to-speech (TTS) systems to generate high-quality speech that is close to human speech. Specifically, there are two types of methods for building a modern TTS system: multi-stage approaches (Ren et al., 2019, 2021; Lim et al., 2020; Zeng et al., 2020) and end-to-end approaches (Wang YX et al., 2017; Ping et al., 2018; Shen et al., 2018; Tachibana et al., 2018). Each type has its own advantages and disadvantages. With a non-autoregressive architecture, the multi-stage models can synthesize speech much faster than the end-to-end methods. However, they are often more complicated and face the challenge of the one-to-many mapping problem (Ren et al., 2021). Despite the lower inference speed, the end-to-end models greatly reduce the workload by employing a single attention-based sequence-to-sequence network which is widely used in neural machine translation tasks (Bahdanau et al., 2015). Therefore, these end-to-end models are deserving more discussion and development.

The sequence-to-sequence network mentioned above usually consists of an encoder, a decoder, and the interactive attention mechanism between the encoder and the decoder. During the inferring procedure, the decoder generates spectra autoregressively to maximize the joint probability distribution. Researchers have introduced various neural architectures into end-to-end speech synthesis. The

‡ Corresponding author

* Project supported by the National Key Research and Development Program of China (No. 2019YFB1312603) and the Robotics Institute of Zhejiang University, China (No. K11801)

ORCID: Wei ZHAO, https://orcid.org/0000-0001-8452-9339; Li XU, https://orcid.org/0000-0001-7874-629X

recurrent neural network (RNN) based models, such as Tacotron (Wang YX et al., 2017) and Tacotron 2 (Shen et al., 2018), are prevalent owing to their inherent autoregressive attributes and outstanding performances. However, their recurrence would seriously reduce the training efficiency. Because all target spectra are known in the training phase, parallelizable networks can be used to improve training efficiency. Consequently, end-to-end TTS models with convolutional neural networks (CNNs) have been proposed (Ping et al., 2018; Tachibana et al., 2018). Nevertheless, it is difficult for CNNs to learn dependencies between distant positions (Vaswani et al., 2017), which might limit the model's performance. Currently, the Transformer network, which relies solely on self-attention, outperforms both the RNN and CNN across many natural language processing tasks with a parallel structure (Vaswani et al., 2017; Dai et al., 2019; Yang et al., 2019). The performance of the Transformer network in speech synthesis is comparable to those of recurrent TTS models, and the Transformer network is much faster in training (Li et al., 2019).

Even though the Transformer network can be fully parallelized in the training phase, it inevitably follows a sequential pattern in the inference phase. When inferring, self-attention in the decoder must assign attention weights for all previous positions to calculate the hidden state at the current time step, which makes the number of operations scale quadratically in sequence length. As a result, the Transformer's inference speed becomes rather low, especially under conditions without graphics processing units (GPUs). Preserving the Transformer's training efficiency on one hand and accelerating its inference on the other hand has become an urgent problem to be solved. To this end, many efficient Transformer models have been investigated (Tay et al., 2020). However, though the Transformer has been popularized by the sequence-to-sequence machine translation task, not many of these new efficient Transformer models are applicable to sequence-to-sequence tasks.

For the neural machine translation task, researchers have conducted some studies to improve the efficiency of Transformer networks. Zhang et al. (2018) proposed replacement of the vanilla self-attention in the decoder with an average attention network paired with an accelerated decoding algorithm. Wu et al. (2019) introduced lightweight convolutions as alternatives for the vanilla self-attention. The Reformer (Kitaev et al., 2020) and Synthesizer (Tay et al., 2021) were then proposed and evaluated for machine translation. In contrast, few attempts have been made to fill the gap in speech synthesis. Therefore, we present efficient decoding self-attention (EDSA), inspired by the studies of Zhang et al. (2018) and Wu et al. (2019) for Transformer-based TTS. Unlike their works, which use only the global average operation or local convolutional operation to replace self-attention, our proposed EDSA takes a step further and can use both long- and short-range information.

The major contributions of our study are as follows: (1) We propose EDSA as an alternative to the original self-attention in the Transformer's decoder. Through a dynamic programming decoding algorithm, the inference of the model can be accelerated effectively with linear computational complexity. (2) Experiments on both Mandarin and English are conducted to verify the proposed method. Our EDSA model can synthesize speech faster than the baseline model on either CPU or GPU devices, while almost keeping the same audio quality. (3) Because of dynamic programming decoding, it is easy to forcibly impose incremental constraints on the attention alignments between text and spectra at inference, which may lead to better performance on out-of-domain utterances.

2 End-to-end speech synthesis

2.1 Sequence-to-sequence network

End-to-end speech synthesis refers to TTS models with a single sequence-to-sequence network. This network follows the encoder–decoder framework with the interactive attention between the encoder and the decoder. The input sequence $\boldsymbol{x} = (x_1, x_2, ..., x_m)$ is first processed by an encoder to obtain more compact representations $\boldsymbol{h} = (h_1, h_2, ..., h_m)$. Then the decoder predicts the output $\boldsymbol{y} = (y_1, y_2, ..., y_n)$ sequentially with the interactive attention mechanism conditioned on states $\boldsymbol{h}$. As an illustration, the detailed RNN-based sequence-to-sequence network process is denoted as follows:

$$\boldsymbol{h} = \text{Encoder}(\boldsymbol{x}), \tag{1}$$

$$s_j = \text{RNN}_{\text{attn}}(s_{j-1}, y_{j-1}, c_{j-1}), \tag{2}$$

$$\boldsymbol{\alpha}_j = \text{Attention}(\boldsymbol{h}, s_j, ...),\ c_j = \sum_i \alpha_{j,i} h_i, \quad (3)$$

$$d_j = \text{Decoder}(d_{j-1}, c_j, s_j),\ y_j = O(d_j). \quad (4)$$

To predict the output element y_j, the RNN unit, RNN_{attn}, computes a new hidden state s_j, based on the previous state s_{j-1}, a context state c_{j-1} derived from the encoder output $\boldsymbol{h}$, and the previous predicted spectrum frame y_{j-1}. The attention mechanism, denoted in Eq. (3), consumes the encoder output $\boldsymbol{h}$ and the hidden state s_j to determine attention weights $\boldsymbol{\alpha}_j$ for each element of the encoder output $\boldsymbol{h}$. Then the current context state c_j is obtained via the weighted sum. Finally, the decoder RNN is fed both the context state c_j and the hidden state s_j to compute the hidden state d_{j-1}, and a linear layer produces the output with the expected shape as denoted in Eq. (4).

In this way, the sequence-to-sequence neural network factorizes the probability distribution over possible outputs into a chain of conditional probabilities with a left-to-right causal structure:

$$p(\boldsymbol{y}|\boldsymbol{x};\theta) = \prod_{j=1}^{n} p(y_j|y_{1:j-1}, x_{1:m};\theta), \quad (5)$$

where θ represents the parameter of the network and can be acquired through maximum likelihood training.

2.2 TTS with Transformer network

Although the sequence-to-sequence network can be conveniently built with RNNs, training such models is very time-consuming for the speech synthesis task where the output spectrum may include more than 1000 frames. To boost the training efficiency, the Transformer network, which has a parallelizable structure, is applied to TTS. Using the powerful self-attention module and proper masking techniques, the Transformer network can perform the same function as the RNN-based sequence-to-sequence network. We recommend the original work (Vaswani et al., 2017) for details about the Transformer network.

Fig. 1 shows the overall architecture of the TTS system with the Transformer network, which converts the source text sequence into the target spectrum sequence. Both the encoder and the decoder comprise N identical blocks. The encoder block consists of a self-attention module and a position-wise, fully connected, feed-forward module. Around each module, residual connection (He et al., 2016) and layer normalization (Ba et al., 2016) are applied. The decoder block is similar to the encoder block in the structure, except that an inter-self-attention module is added between the intra-self-attention and feed-forward modules to align the output spectrum with the input text, as explained in Eq. (3).

In addition to the Transformer backbone, a three-layer CNN and a two-layer fully connected network are employed as the pre-net for the encoder and decoder, respectively. A Mel linear layer is responsible for converting output hidden states from Transformer's decoder into spectra with the expected shape, and a stop linear layer is used to decide whether to stop the generation. In addition, a five-layer CNN is employed as the post-net to refine the output spectrum. Finally, the audio waveform is reconstructed from the spectrum using a neural vocoder. More detailed hyper-parameters of the Transformer TTS model can be found in Li et al. (2019).

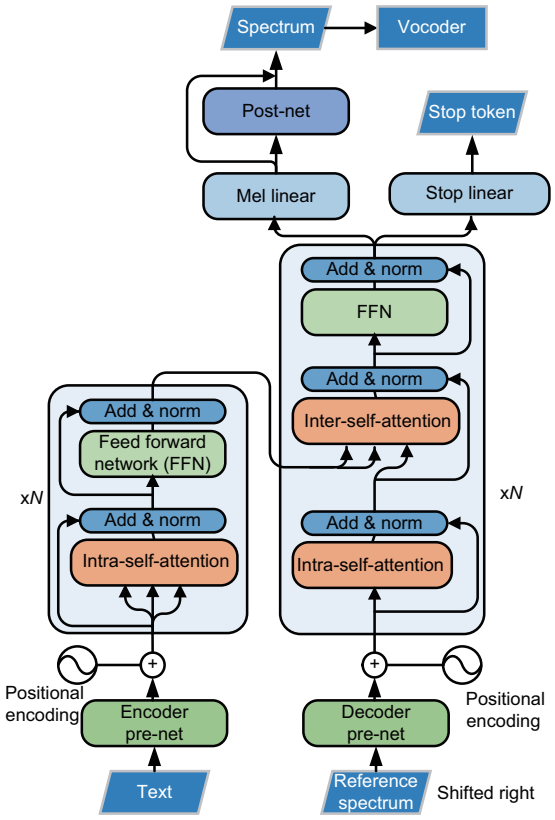

Fig. 1 Detailed architecture of the text-to-speech (TTS) system with the Transformer network

3 Efficient decoding self-attention

3.1 Analysis of vanilla self-attention

The vanilla self-attention mechanism processes a state sequence $\boldsymbol{v} = (v_1, v_2, ..., v_n)$ and returns the new hidden state sequence $\boldsymbol{v}' = (v'_1, v'_2, ..., v'_n)$. Considering that v_t represents the current token to be processed, the vanilla self-attention mechanism is shown in Fig. 2a: it assigns weights $\omega_{t,j}$ to each token v_j from $\boldsymbol{v}$, according to the degree of similarity between v_j and v_t. To quantify the similarity, the dot product is used, followed by softmax normalization, which limits weight values in the range $[0, 1.0]$. Finally, the weighted sum $\sum_j \omega_{t,j} v_j$ is computed as the new hidden state v'_t corresponding to v_t. A multi-head operation has also been proposed which groups input states along channels and applies self-attention to each group to make the module more powerful.

There are three stages at which the self-attention mechanism is employed in the Transformer network: an inter-self-attention stage (at which the spectrum is aligned with text) and two intra-self-attention stages (at which the information is aggregated for the encoder and decoder separately). For the inter-self-attention stage, it is indispensable to use the pair-wise dot product to obtain successful alignment. For the intra-self-attention stage in the encoder, the non-autoregressive work pattern is used, which can enjoy full parallelization during both training and inferring. As a result, the inference efficiency is not adversely affected by these two modules.

However, the intra-self-attention in the decoder follows an autoregressive decoding scheme as depicted in Eq. (5), which may slow down the generation of the spectrum significantly. As shown in Fig. 2b, all the previous states before v_t are required to compute the new hidden state v'_t, and this operation is sequentially conducted from $t = 1$ to $t = n$. Hence, the intra-self-attention mechanism in the decoder has quadratic complexity in sequence length and cannot be parallelized during inference. Because the speech spectrum is frequently very long, using such self-attention in the decoder results in an undesirable inference speed. To overcome this problem, we suggest the EDSA module as an alternative to the original intra-self-attention mechanism in the decoder, as demonstrated in Fig. 3.

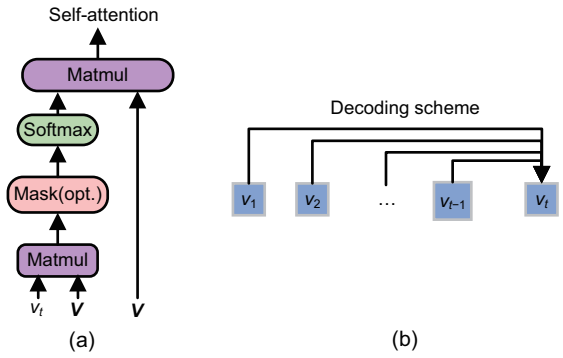

Fig. 2 Vanilla self-attention mechanism in the Transformer network (a) and its decoding scheme for a specific token v_t (b)

3.2 Structure of EDSA

As presented in Fig. 4, the EDSA module factorizes the feature extraction process of a sequence into two parts: a global average attention (GAA) network producing long-range dependencies and a local predictive attention (LPA) network capturing short-range dependencies.

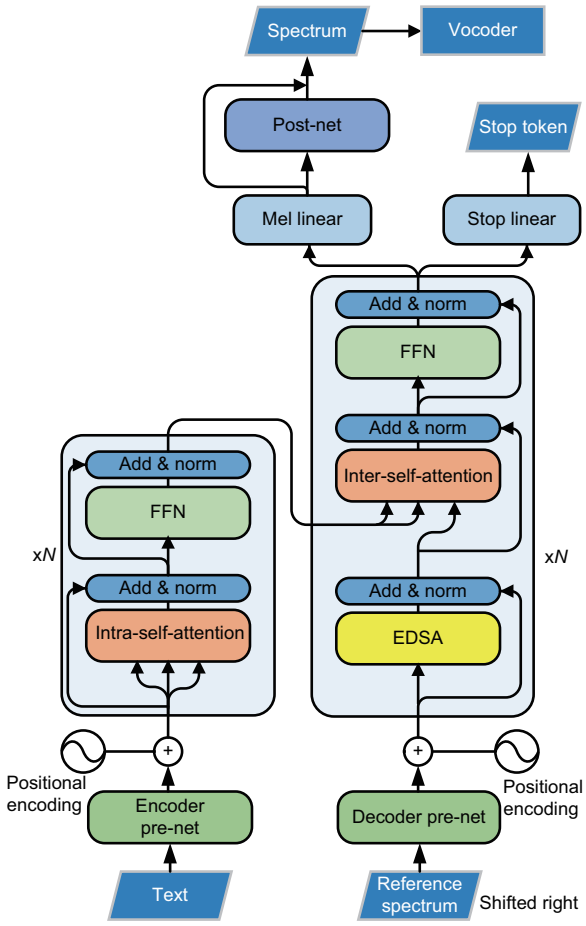

Fig. 3 Transformer-based TTS system with the efficient decoding self-attention (EDSA) module

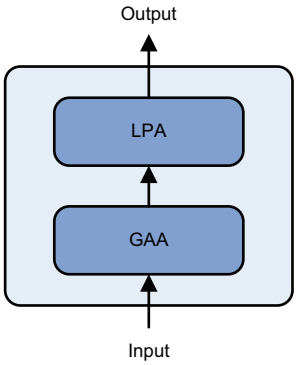

Fig. 4 Structure of the EDSA module
GAA denotes the global average attention network and LPA denotes the local predictive attention network

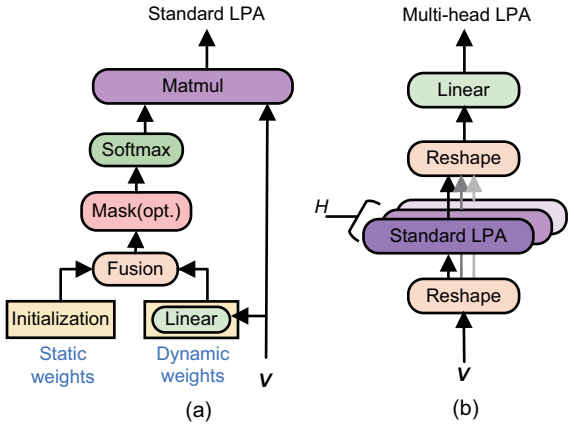

Fig. 5 Standard (a) and multi-head (b) local predictive attention networks

3.2.1 Global average attention

The original self-attention can form relationships between any two states in a sequence via pairwise dot products. To ensure the capacity for building global dependencies, we employ the GAA network as a submodule that directly assigns averaged weights for each token in the sequence. This average operation can be realized conveniently with the mask matrix indicated in Eq. (6):

$$\begin{pmatrix} v'_1 \\ v'_2 \\ v'_3 \\ \vdots \\ v'_n \end{pmatrix} = \begin{pmatrix} v_1 \\ (v_1+v_2)/2 \\ (v_1+v_2+v_3)/3 \\ \vdots \\ (v_1+v_2+...+v_n)/n \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & ... & 0 \\ 1/2 & 1/2 & 0 & ... & 0 \\ 1/3 & 1/3 & 1/3 & ... & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 1/n & 1/n & 1/n & ... & 1/n \end{pmatrix} \times \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{pmatrix}, \tag{6}$$

where v_t represents the input state at time step t and n denotes the number of elements in the sequence. By multiplying all the input states by the above mask matrix, we can obtain the final output of the GAA network and assure the training efficiency.

3.2.2 Local predictive attention

The LPA network works as a special self-attention mechanism by which only attention weights are assigned for the tokens inside the restricted context. As shown in Fig. 5a, the LPA layer determines attention weights according to the input state instead of the pair-wise dot-product operation between tokens. It has been proved in our previous work that the LPA-like layer is good at capturing local dependencies for speech synthesis (Zhao et al., 2021). However, the above work still suffers from slow inference with quadratic complexity because it focuses on improving the model's overall performance but ignores efficiency.

Specifically, the procedure of the LPA network is described as follows:

$$\tilde{\boldsymbol{w}}_t,\ \tilde{\boldsymbol{g}}_t = \mathrm{Linear}(\boldsymbol{v}_t), \tag{7}$$

$$\boldsymbol{w}_t = \mathrm{Sigmoid}(\tilde{\boldsymbol{g}}_t) \cdot \tilde{\boldsymbol{w}}_t + \overline{\boldsymbol{w}}, \tag{8}$$

$$\boldsymbol{v}'_t = \sum_j \mathrm{Dropout}(\mathrm{Softmax}(\boldsymbol{w}_t))_j \cdot \boldsymbol{v}_j. \tag{9}$$

Supposing that d is the input state's dimension and that k is the interval size of the restricted context, the LPA network first predicts a set of dynamic weights $\tilde{\boldsymbol{w}}_t \in \mathbb{R}^k$ and corresponding gate signals $\tilde{\boldsymbol{g}}_t \in \mathbb{R}^k$ through a fully connected layer according to the input $\boldsymbol{v}_t \in \mathbb{R}^d$ as presented in Eq. (7). Simultaneously, another set of static weights $\overline{\boldsymbol{w}}_t$, irrelevant to the input, is randomly initialized and updated with the training of the network. We fuse the dynamic weights and static weights via the nonlinear interpolation as in Eq. (8). It is expected that the static weights could capture the time-invariant dependencies, and the dynamic weights could build the time-variant dependencies. A mask technique, similar to self-attention, can be applied to ensure that the decoder always generates causal outputs. These attention weights $\boldsymbol{w}_t$ are also softmax normalized to limit them in the range $[0, 1.0]$, and dropout is applied for regularization. Finally, the output $\boldsymbol{v}'_t \in \mathbb{R}^d$

of the LPA network is computed by weighted summation as shown in Eq. (9).

We introduce the multi-head operation to endow the LPA network with more strength, as depicted in Fig. 5b. Considering an input sequence $\boldsymbol{V}$ of size $B \times n \times d$, where B represents the batch size and n represents the sequence length, it can first be reshaped and transposed to size $BH \times n \times \frac{d}{H}$, where H denotes the number of heads. Then, the standard LPA is employed to predict the attention weight for each head. Finally, the output is recovered to the original shape and sent to a linear layer for information exchanges across channels.

3.3 Dynamic programming decoding for EDSA

Paired with the EDSA module, a dynamic programming decoding algorithm is proposed to accelerate the inference, as demonstrated in Eqs. (10)–(13):

$$\boldsymbol{a}_t = \boldsymbol{a}_{t-1} + \boldsymbol{v}_t, \tag{10}$$

$$\tilde{\boldsymbol{w}}_t,\ \tilde{\boldsymbol{g}}_t = \mathrm{Linear}(\boldsymbol{a}_t/t), \tag{11}$$

$$\boldsymbol{w}_t = \mathrm{Sigmoid}(\tilde{\boldsymbol{g}}_t) \cdot \tilde{\boldsymbol{w}}_t + \overline{\boldsymbol{w}}, \tag{12}$$

$$\boldsymbol{o}_t = \sum_{\boldsymbol{v}_i \in B_t} \mathrm{Dropout}(\mathrm{Softmax}(\boldsymbol{w}_t))_i \cdot \boldsymbol{v}_i, \tag{13}$$

where $\boldsymbol{a}_t$ is the accumulation of input states for EDSA over the past t time steps, and $\boldsymbol{a}_0 = \boldsymbol{0}$. By maintaining the accumulation $\boldsymbol{a}_t$ as the hidden state for each time step, the averaged output can be calculated immediately just based on the previous accumulation $\boldsymbol{a}_{t-1}$ and the current input $\boldsymbol{v}_t$, as denoted in Eq. (11). We keep the last k input states within the local context window as a hidden buffer B_t for each time step, and thus the final output can be obtained by reading the buffer and performing the weighted sum.

In this manner, the inference of EDSA has only linear complexity and is therefore much more efficient than the original self-attention. At the same time, the TTS model with EDSA can be completely parallelized during training, which ensures the training efficiency. Another advantage of employing the dynamic programming decoding algorithm is that alignments between text and spectra are produced sequentially. Therefore, it is capable of applying a forcibly incremental attention strategy to the decoding process (Ping et al., 2018; Tachibana et al., 2018).

During inference, the attention weights at the current time step are forcibly compared with the weights at previous time steps to ensure that the attention alignment goes forward in a monotonic way. For our proposed model, we apply a strategy similar to that introduced by Tachibana et al. (2018) to constrain the attention weights of several dominant heads of the inter-self-attention module.

4 Experimental results

We conducted experiments on datasets of two different languages: the Mandarin dataset called CSMSC (Databaker, 2019) and the English dataset called LJSpeech (Ito and Johnson, 2017). For each dataset, 50 longest sentences were reserved for the out-of-domain evaluation. We then divided each dataset into three subsets with a proportion of 96% for training, 2% for validation, and 2% for testing. We used a 22 kHz sampling rate across all experiments. Experimental samples of the synthetic speech are available on GitHub (https://zju-ee-robotics.github.io/edsa-audios/).

The input text was converted into phonemes using the open-source front-end tools G2PC (for Mandarin) (Park, 2019) and G2P (for English) (Park and Kim, 2019). Each phoneme was then embedded into a 512-dimensional vector and fed to the TTS network. As the target to be predicted, the Mel spectrum was calculated via a short-time Fourier transform (STFT) using a 50 ms frame size, 12.5 ms frame hop, a Hann window function, and 80 Mel scale filterbanks followed by log compression. The detailed architecture of our model was the same as that of the baseline Transformer model, except for the newly introduced EDSA. For the EDSA module, we used head number $H = 16$ and context size $k = 31$. The Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$ was used during training, and the warm-up learning rate strategy (Vaswani et al., 2017) with warmup_steps = 4000 was applied. The source code of our model is available online (https://github.com/ZJU-EE-Robotics/transformer-edsa).

We used Tacotron 2 and Transformer TTS for comparison in our experiments because they are the most dominant end-to-end TTS models. Though some non-autoregressive studies can significantly speed up the inference of TTS, they usually employ

complicated pipelines. Because we focused on end-to-end speech synthesis, these non-autoregressive models were not included in this study. Replacing the Transformer's decoder with long-short term memory (LSTM) networks may also accelerate the inference. However, the whole model would become highly similar to Tacotron 2 and would not support parallelizable training. Thus, the Transformer TTS model with the LSTM-based decoder was not included to avoid redundancy. All experimental models were built referring to the speech synthesis toolkit named ESPnet-TTS (Hayashi et al., 2020), which provides unified and reproducible implementations of popular TTS networks. These models were trained to converge on two NVIDIA RTX 3090 GPUs with a batch size of 64 using Pytorch. Gradient accumulation was applied during training whenever needed.

We employed the WaveGlow (Prenger et al., 2019) model released by NVIDIA as our vocoder across the experiments. Considering that this model is trained only on LJSpeech, we fine-tuned its parameters on CSMSC to make it applicable to Mandarin.

4.1 Naturalness and similarity

It is challenging to evaluate TTS models because objective measurements do not always accurately reflect performance and may potentially lead to misunderstandings. For this reason, recent studies usually used subjective metrics to evaluate the quality of synthetic audios. We chose the widely used five-point mean opinion score (MOS), in which raters are asked to score the naturalness of each audio on a five-point scale (1: bad; 2: poor; 3: fair; 4: good; 5: excellent). In our experiments, 50 random samples from the test set were demonstrated to 15 listeners for the MOS test. In addition, the objective metric, Mel cepstrum distortion (MCD), was calculated on the entire test set as a supplement for measuring the distance from the synthetic spectrum to the real spectrum. The shorter the distance, the higher the similarity.

It can be observed from Table 1 that our model with EDSA may achieve comparable performance to the Transformer baseline, with only an MOS gap of 0.03 on the CSMSC dataset and 0.01 on the LJSpeech dataset. Our model may even obtain better MCD values on the Mandarin dataset. Our model and the Transformer both obtained slightly better performance than Tacotron 2 on MOS and MCD evaluations. As a result, both subjective and objective tests indicated that the suggested TTS model with EDSA can be employed as a replacement for the Transformer baseline.

A further ablation study was conducted with comparison mean opinion score (CMOS) tests to verify the effectiveness of the proposed GAA and LPA networks. In CMOS tests, the 15 raters listened to two audios (one generated by our model and the other generated by our model without GAA or LPA for the same input text) each time and evaluated how the latter performed compared to the former using a score in $[-3, 3]$. The order of the two audios was randomly changed for fairness. As shown in Table 2, removing either the GAA or LPA network led to negative CMOS values for both languages, suggesting that each proposed component contributes remarkably to the EDSA module.

Table 2 Comparison mean opinion score (CMOS) with 95% confidence intervals

Model	CMOS	
	CSMSC	LJSpeech
EDSA	0	0
Without GAA	−0.14±0.17	−0.11±0.16
Without LPA	−0.18±0.16	−0.23±0.15

GAA: global average attention; LPA: local predictive attention

Table 1 Mean opinion score (MOS) and Mel cepstrum distortion (MCD) with 95% confidence intervals

Model	MOS		MCD	
	CSMSC	LJSpeech	CSMSC	LJSpeech
Ground truth	4.48±0.06	4.51±0.05	N/A	N/A
Tacotron 2	4.26±0.07	4.15±0.07	6.78±0.09	7.62±0.11
Transformer	**4.31±0.07**	**4.20±0.08**	6.73±0.09	**7.40±0.10**
Ours	4.28±0.08	4.19±0.07	**6.67±0.08**	7.43±0.10

Bold values indicate the better results of the Transformer baseline and our model. N/A: not applicable

4.2 Inference speed

We compared the inference speed across different models using the real-time factor (RTF). The RTF value indicates the ratio of the speech generation time to the real speech duration. In the experiments, RTF values for the generation of spectra were measured on a personal workstation with Intel Core i7-8700K CPU, 1 NVIDIA RTX 2080Ti GPU, and a batch size of 1. We selected 20 utterances of about 400 frames from CSMSC and 20 much longer utterances of about 800 frames from LJSpeech for evaluation.

Table 3 summarizes the final results of the comparison of inference speed. Our model was distinctly faster than the Transformer model during inference. Specifically, our model can accelerate the inference by 4.5–7.2 times on a CPU and 0.2–0.5 times on a GPU. Note that the Transformer's inference speed deteriorated dramatically on the CPU, whereas our model with EDSA can mitigate this problem effectively. In particular, the Transformer baseline was unable to generate spectra in real time on a CPU, while our model could meet the requirements of real-time production across different devices. Moreover, with the increase of spectra length, it can be observed that the RTF value of our model was nearly unchanged. However, the RTF value of the Transformer baseline decreased significantly due to the quadratic complexity in the sequence length. Thus, our model may be more efficient for long input texts.

To explicitly compare the model complexity in inference, we employed floating-point operations (FLOPs) to assess experimental models. For fair comparison, we conducted the FLOPs counted on the same two samples (about 400 and 800 frames) and the inference procedure was run with the "teacher forcing" strategy to guarantee the identical output spectrum length. As demonstrated in Table 4, our model had a substantially lower number of FLOPs, which was only 4%–7% that of the Transformer baseline, while having a similar number of parameters. The FLOP statistics was also listed for the separate intra-self-attention module in the decoder (denoted as SA). It can be observed that the number of FLOPs of the SA in our model, i.e., the proposed EDSA, varied only with the linear complexity in the spectrum length. As the spectrum went from about 400 frames to about 800 frames, the FLOP number of our EDSA module doubled; this behavior is similar to that of an RNN-based model. However, the number of FLOPs of the SA in Transformer, i.e., the original self-attention, increased dramatically with non-linear complexity.

Additionally, although Tacotron 2 is the most efficient in inference, the RNN-based models, as previously mentioned, take a long time to train. We benchmarked the training efficiency of different models on the same device. For each model, the batch size was set to 6, and the training procedure was run for one epoch. Then we calculated the time cost per batch/iteration to indicate the training efficiency. As shown in Table 5, the Transformer and our model could save roughly 91% of the training time per batch compared with Tacotron2, thanks to

Table 3 Real-time factors (RTFs) of different models with 95% confidence intervals

Model	RTF with about 400 frames		RTF with about 800 frames	
	CPU	GPU	CPU	GPU
Tacotron 2	4.044±0.015	15.702±0.079	3.926±0.023	16.051±0.217
Transformer	0.248±0.012	2.186±0.023	0.164±0.006	1.776±0.056
Ours	**1.370±0.010**	**2.643±0.026**	**1.350±0.005**	**2.656±0.009**

Bold values indicate the better results of the Transformer baseline and our model

Table 4 Number of employed floating-point operations (FLOPs) of different models in inference

Model	Number of parameters ($\times10^6$)	Number of FLOPs ($\times10^{12}$)			
		400	(400, SA)	800	(800, SA)
Tacotron 2	28.158	0.010	N/A	0.021	N/A
Transformer	52.949	2.183	0.654	10.032	3.534
Ours	48.245	**0.162**	**0.003**	**0.418**	**0.006**

Bold values indicate the better results of the Transformer baseline and our model. N/A: not applicable; SA: self-attention

the parallel network architecture. Even though the Transformer baseline and our model had more parameters and the gradient accumulation was applied when we trained them on the server, it took only about 2 d to converge well for the LJSpeech dataset, while Tacotron 2 required more than 5 d. It should be mentioned that the training efficiency of our model suffered slightly as current compute unified device architecture (CUDA) primitives do not support the local predictive attention very well. This problem was also discussed in Wu et al. (2019), and may be solved in the future via more delicate CUDA kernels.

Table 5 Training time per batch of different models with 95% confidence intervals

Model	Training time per batch (s)	
	CSMSC	LJSpeech
Tacotron 2	0.830±0.003	1.456±0.005
Transformer	0.069±0.001	0.128±0.001
Ours	0.072±0.001	0.134±0.001

4.3 Out-of-domain generalization

Researchers have found that attention-based end-to-end TTS models are prone to making errors for out-of-domain texts, such as repeating phonemes and missing phonemes, which severely degrades audio quality. Because the proposed dynamic programming decoding algorithm is combined with forcibly incremental attention, we supposed that our model would be more robust than the original Transformer. The AB preference tests were conducted to assess our model and Transformer on the reserved out-of-domain utterances. Given each pair of synthetic audios the same transcript, 15 raters were asked to give their preferences for the better ones.

Figs. 6a and 6b demonstrate that our model performed considerably better than the Transformer on the English dataset. However, this preference was inconspicuous on the Mandarin dataset. Unlike English dataset, where a word's pronunciation is determined by the alphabet and letter arrangements, Mandarin has pre-marked pronunciations for all words as Pinyin. As a result, the Mandarin Transformer model may not be as sensitive to out-of-domain utterances as the English Transformer model.

We conducted case studies on attention alignments between text and spectra to learn more about how our model is better for out-of-domain utterances. As seen in Fig. 7, the attention alignment of the Transformer model became discontinuous at the rear part, which results in mispronunciations. In contrast, our model can produce a robust and monotonic alignment and therefore was preferred, as we expected.

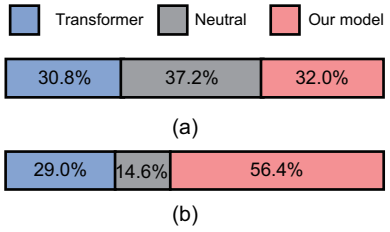

Fig. 6 AB preference results on out-of-domain utterances based on the CSMSC (a) and LJSpeech (b) datasets

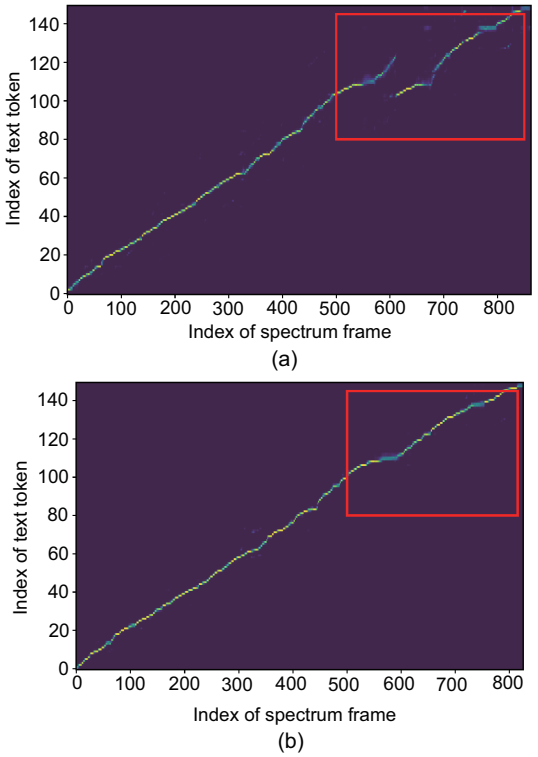

Fig. 7 Comparison of the alignment between text and spectra for the Transformer (a) and our model (b)

5 Discussion and conclusions

The self-attention mechanism has achieved enormous success across many deep learning tasks, such as natural language processing and computer

vision. In the past few years, researchers have applied self-attention to end-to-end speech synthesis, also known as the Transformer TTS. Most notably, the performance of the Transformer TTS can match those of the TTS models with RNNs, while being more efficient in training. Though the parallelization property of self-attention significantly speeds up the training process, the inference of the Transformer cannot be accelerated due to the sequential decoding schema. Because self-attention compares the element at the current time step to all the elements in the unrestricted context, the number of operations grows quadratically in spectrum length during inference. Hence, the Transformer TTS suffers from low inference speed, especially on devices with only CPUs.

Although many efficient Transformer models have been proposed, it is not straightforward to apply them to sequence-to-sequence tasks such as speech synthesis. In particular, the Transformer models can be divided into two categories (Tay et al., 2020): (1) encoder-only networks and (2) decoder-only or encoder-decoder networks. The main difference between the two categories is whether the model can support causal auto-regressive decoding. The self-attention mechanism in encoder-only models does not guarantee causality. Therefore, these models, such as Set Transformer (Lee et al., 2019), ETC (Ainslie et al., 2020), Linformer (Wang SN et al., 2020), and Big Bird (Zaheer et al., 2020), are not applicable to sequence-to-sequence problems where auto-regressive decoding is needed. Thus, the most potential self-attention mechanisms that may be used to improve decoding efficiency are Sparse Transformer (Child et al., 2019), Reformer (Kitaev et al., 2020), Longformer (Beltagy et al., 2020), Synthesizer (Tay et al., 2021), Performer (Choromanski et al., 2020), and Linear Transformer (Katharopoulos et al., 2020). However, among these works, Sparse Transformer, Performer, and Linear Transformer focus on evaluating the proposed methods with the decoder-only architecture. Although the Longformer work has been benchmarked on the summarization task with the Longformer-encoder-decoder (LED) network, this work employs an original Transformer decoder with the Longformer encoder. Therefore, the efficiency on the decoder side of the LED is not improved. Only Reformer (Kitaev et al., 2020) and Synthesizer (Tay et al., 2021) test their models on the sequence-to-sequence machine translation task. Further research is required to apply such efficient models with an auto-regressive decoder in speech synthesis.

Reformer-TTS (Ihm et al., 2020) with locality-sensitive hashing has recently been proposed to reduce GPU memory usage in training, but the inference efficiency has not been studied or reported. Our work is different from Reformer-TTS, because we focus on employing the EDSA module to speed up the inference process. Specifically, our EDSA module is somewhat like a variant of Synthesizer (Tay et al., 2021). As the dense synthesizer operation in Synthesizer predicts the attention weights globally, it still has quadratic complexity. Employing only the random synthesizer can further improve efficiency, but this strategy may negatively affect the performance on the Seq2Seq task (machine translation in their work). In contrast, the proposed EDSA can be considered a hybrid of a new dense synthesizer that predicts attention weights for a limited context and a non-random synthesizer that averages attention weights. Consequently, our model has a linear scale complexity during the inference and may perform comparably to popular end-to-end TTS models.

To investigate why the original self-attention in Transformer's decoder could be further optimized, we visualized the attention weights, as shown in Fig. 8. It was observed that the intra-self-attention module in Transformer's decoder learned a rather consistent pattern across multiple heads after training. Each element in the sequence most attended to its local context, thus allowing simpler approaches to replace the original self-attention.

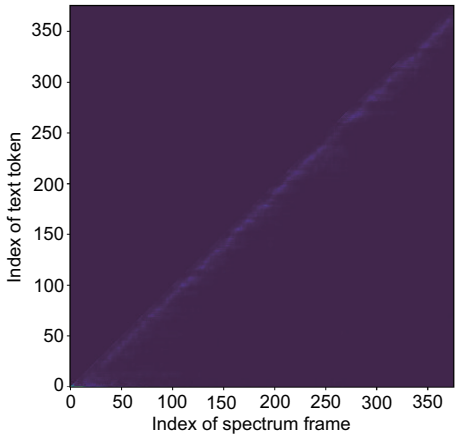

Fig. 8 Visualization of the intra-self-attention weights from a representative head in Transformer's decoder

In conclusion, we have presented EDSA for the popular end-to-end speech synthesis. EDSA can model both long- and short-range dependencies with the number of operations scaling linearly in sequence length during decoding. Experimental results demonstrated that the proposed EDSA can accelerate the inference effectively with almost no performance loss. Notably, our TTS model with EDSA can guarantee real-time inference even without GPUs. Our model might also perform better on out-of-domain utterances.

Contributors

Wei ZHAO and Li XU designed the research. Wei ZHAO performed the experiments and drafted the paper. Li XU revised and finalized the paper.

Compliance with ethics guidelines

Wei ZHAO and Li XU declare that they have no conflict of interest.

References

Ainslie J, Ontanon S, Alberti C, et al., 2020. ETC: encoding long and structured inputs in Transformers. Proc Conf on Empirical Methods in Natural Language Processing, p.268-284. https://doi.org/10.18653/v1/2020.emnlp-main.19

Ba JL, Kiros JR, Hinton GE, 2016. Layer normalization. https://arxiv.org/abs/1607.06450

Bahdanau D, Cho K, Bengio Y, 2015. Neural machine translation by jointly learning to align and translate. https://arxiv.org/abs/1409.0473v6

Beltagy I, Peters ME, Cohan A, 2020. Longformer: the long-document transformer. https://arxiv.org/abs/2004.05150

Child R, Gray S, Radford A, et al., 2019. Generating long sequences with Sparse Transformers. https://arxiv.org/abs/1904.10509

Choromanski KM, Likhosherstov V, Dohan D, et al., 2020. Rethinking attention with performers. https://arxiv.org/abs/2009.14794

Dai ZH, Yang ZL, Yang YM, et al., 2019. Transformer-XL: attentive language models beyond a fixed-length context. Proc 57th Annual Meeting of the Association for Computational Linguistics, p.2978-2988.

DataBaker, 2019. Chinese Standard Mandarin Speech Copus. https://www.data-baker.com [Accessed on June 1, 2020].

Hayashi T, Yamamoto R, Inoue K, et al., 2020. Espnet-TTS: unified, reproducible, and integratable open source end-to-end text-to-speech toolkit. Proc IEEE Int Conf on Acoustics, Speech and Signal Processing, p.7654-7658. https://doi.org/10.1109/ICASSP40776.2020.9053512

He KM, Zhang XY, Ren SQ, et al., 2016. Deep residual learning for image recognition. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.770-778. https://doi.org/10.1109/CVPR.2016.90

Ihm HR, Lee JY, Choi BJ, et al., 2020. Reformer-TTS: neural speech synthesis with reformer network. Proc Interspeech 21st Annual Conf of the Int Speech Communication Association, p.2012-2016. https://doi.org/10.21437/Interspeech.2020-2189

Ito K, Johnson L, 2017. The LJ Speech Dataset. https://keithito.com/LJ-Speech-Dataset/ [Accessed on June 1, 2020].

Katharopoulos A, Vyas A, Pappas N, et al., 2020. Transformers are RNNs: fast autoregressive transformers with linear attention. Proc 37th Int Conf on Machine Learning, p.5156-5165.

Kingma DP, Ba J, 2015. Adam: a method for stochastic optimization. https://arxiv.org/abs/1412.6980

Kitaev N, Kaiser L, Levskaya A, 2020. Reformer: the efficient Transformer. https://arxiv.org/abs/2001.04451v1

Lee J, Lee Y, Kim J, et al., 2019. Set Transformer: a framework for attention-based permutation-invariant neural networks. Proc 36th Int Conf on Machine Learning, p.3744-3753.

Li NH, Liu SJ, Liu YQ, et al., 2019. Neural speech synthesis with Transformer network. Proc AAAI Conf on Artificial Intelligence, p.6706-6713. https://doi.org/10.1609/aaai.v33i01.33016706

Lim D, Jang W, OG, et al., 2020. JDI-T: Jointly trained Duration Informed Transformer for text-to-speech without explicit alignment. Proc Conf of the Int Speech Communication Association, p.4004-4008.

Park K, 2019. g2pC. GitHub. https://github.com/Kyubyong/g2pC [Accessed on June 1, 2020].

Park K, Kim J, 2019. g2pE. GitHub. https://github.com/Kyubyong/g2p [Accessed on June 1, 2020].

Ping W, Peng KN, Gibiansky A, et al., 2018. Deep Voice 3: scaling text-to-speech with convolutional sequence learning. https://arxiv.org/abs/1710.07654

Prenger R, Valle R, Catanzaro B, 2019. WaveGlow: a flow-based generative network for speech synthesis. Proc IEEE Int Conf on Acoustics, Speech and Signal Processing, p.3617-3621.

Ren Y, Ruan YJ, Tan X, et al., 2019. FastSpeech: fast, robust and controllable text to speech. Proc Advances in Neural Information Processing Systems 32: Annual Conf on Neural Information Processing Systems, p.3165-3174.

Ren Y, Hu CX, Tan X, et al., 2021. FastSpeech 2: fast and high-quality end-to-end text to speech. https://arxiv.org/abs/2006.04558v3

Shen J, Pang RM, Weiss RJ, et al., 2018. Natural TTS synthesis by conditioning wavenet on Mel spectrogram predictions. Proc IEEE Int Conf on Acoustics, Speech and Signal Processing, p.4779-4783.

Tachibana H, Uenoyama K, Aihara S, 2018. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. Proc IEEE Int Conf on Acoustics, Speech and Signal Processing, p.4784-4788.

Tay Y, Dehghani M, Bahri D, et al., 2020. Efficient transformers: a survey. https://arxiv.org/abs/2009.06732

Tay Y, Bahri D, Metzler D, et al., 2021. Synthesizer: rethinking self-attention for Transformer models. Proc 38th Int Conf on Machine Learning, p.10183-10192.

Vaswani A, Shazeer N, Parmar N, et al., 2017. Attention is all you need. Proc Advances in Neural Information Processing Systems 30: Annual Conf on Neural Information Processing Systems, p.5998-6008.

Wang SN, Li BZ, Khabsa M, et al., 2020. Linformer: self-attention with linear complexity. https://arxiv.org/abs/2006.04768

Wang YX, Skerry-Ryan RJ, Stanton D, et al., 2017. Tacotron: towards end-to-end speech synthesis. Proc Interspeech 18th Annual Conf of the Int Speech Communication Association, p.4006-4010. https://doi.org/10.21437/Interspeech.2017-1452

Wu F, Fan A, Baevski A, et al., 2019. Pay less attention with lightweight and dynamic convolutions. https://arxiv.org/abs/1901.10430v2

Yang ZL, Dai ZH, Yang YM, et al., 2019. XLNet: generalized autoregressive pretraining for language understanding. Proc Advances in Neural Information Processing Systems 32: Annual Conf on Neural Information Processing Systems, p.5754-5764.

Zaheer M, Guruganesh G, Dubey KA, et al., 2020. Big Bird: Transformers for longer sequences. https://arxiv.org/abs/2007.14062

Zeng Z, Wang JZ, Cheng N, et al., 2020. AlignTTS: efficient feed-forward text-to-speech system without explicit alignment. Proc IEEE Int Conf on Acoustics, Speech and Signal Processing, p.6714-6718.

Zhang B, Xiong DY, Su JS, 2018. Accelerating neural transformer via an average attention network. Proc 56th Annual Meeting of the Association for Computational Linguistics, p.1789-1798.

Zhao W, He T, Xu L, 2021. Enhancing local dependencies for Transformer-based text-to-speech via hybrid lightweight convolution. *IEEE Access*, 9:42762-42770. https://doi.org/10.1109/ACCESS.2021.3065736