EfficientTTS: An Efficient and High-Quality Text-to-Speech Architecture

Chenfeng Miao [1] **Shuang Liang** [1] **Zhengchen Liu** [1] **Minchuan Chen** [1] **Jun Ma** [1] **Shaojun Wang** [1] **Jing Xiao** [1]

Abstract

In this work, we address the Text-to-Speech (TTS) task by proposing a non-autoregressive architecture called EfficientTTS. Unlike the dominant non-autoregressive TTS models, which are trained with the need of external aligners, EfficientTTS optimizes all its parameters with a stable, end-to-end training procedure, allowing for synthesizing high quality speech in a fast and efficient manner. EfficientTTS is motivated by a new monotonic alignment modeling approach, which specifies monotonic constraints to the sequence alignment with almost no increase of computation. By combining EfficientTTS with different feed-forward network structures, we develop a family of TTS models, including both text-to-melspectrogram and text-to-waveform networks. We experimentally show that the proposed models significantly outperform counterpart models such as Tacotron 2 (Shen et al.) and Glow-TTS (Kim et al., 2020) in terms of speech quality, training efficiency and synthesis speed, while still producing the speeches of strong robustness and great diversity. In addition, we demonstrate that proposed approach can be easily extended to autoregressive models such as Tacotron 2.

1. Introduction

Text-to-Speech (TTS) is an important task in speech processing. With rapid progress in deep learning, TTS technology has received widespread attention in recent years. The most popular neural TTS models are autoregressive models based on an encoder-decoder framework (Wang et al., 2017; Shen et al.; Ping et al., 2018; 2019; Li et al., 2019; Valle et al., 2021). In this framework, the encoder takes the text sequence as input and learns its hidden representation, while the decoder generates the outputs frame by frame, i.e., in an autoregressive manner. As the performance of autoregressive models has been substantially promoted, the synthesis efficiency is becoming a new research hotspot.

Recently, significant efforts have been dedicated to the development of non-autoregressive TTS models (Ren et al., 2019; 2021; Miao et al.; Peng et al., 2020). However, most non-autoregressive TTS models suffer from complex training procedures, high computational cost or training time cost, making them not suited for real-world applications. In this work, we propose EfficientTTS, an efficient and high-quality text-to-speech architecture. Our contributions are summarized as follows,

- We propose a novel approach to produce soft or hard monotonic alignments for sequence-to-sequence models. By constraining the sequence alignment to be monotonic, the proposed approach extends the vanilla attention mechanism with almost no increase in computation. Most importantly, the proposed approach can be incorporated into any attention mechanisms without constraints on network structures.
- We propose EfficientTTS, a non-autoregressive architecture to perform high-quality speech generation from text sequence without additional aligners. EfficientTTS is fully parallel, trained end-to-end, thus being quite efficient for both training and inference.
- We develop a family of TTS models based on EfficientTTS, including: (1) EFTS-CNN, a convolutional model learns melspectrogram generation with high training efficiency; (2) EFTS-Flow, a flow-based model enables parallel melspectrogram generation with controllable speech variation; (3) EFTS-Wav, a fully end-to-end model directly learns waveform generation from text sequence. We experimentally show that the proposed models achieve significant improvements in speech quality, synthesis speed and training efficiency, in comparison with counterpart models Tacotron 2 and Glow-TTS.
- We show that the proposed approach can be easily extended to autoregressive models such as Tacotron 2.

The rest of the paper is structured as follows. Section 2 discusses related works in non-autoregressive TTS models and

[1]Ping An Technology. Correspondence to: Chenfeng Miao <miao_chenfeng@126.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

monotonic alignment modeling respectively. We introduce monotonic alignment modeling using *index mapping vector* in Section 3. The EfficientTTS architecture is introduced in Section 4. Section 5 demonstrates experimental results. Finally, Section 6 concludes the paper.

2. Related Work

2.1. Non-Autoregressive TTS models

In TTS tasks, an input text sequence $\boldsymbol{x} = [x_0, x_1, ..., x_{T_1-1}]$ is transduced to an output sequence $\boldsymbol{y} = [y_0, y_1, ..., y_{T_2-1}]$ through an encoder-decoder framework (Bahdanau et al., 2015). Typically, $\boldsymbol{x}$ is first converted to a sequence of hidden states $\boldsymbol{h} = [h_0, h_1, ..., h_{T_1-1}]$ through an encoder f: $\boldsymbol{h} = f(\boldsymbol{x})$, and then passed through a decoder to produce the output $\boldsymbol{y}$. For each output timestep, an attention mechanism allows for searching the whole elements of $\boldsymbol{h}$ to generate a context vector $\boldsymbol{c}$:

$$c_j = \sum_{i=0}^{T_1-1} \alpha_{i,j} * h_i, \tag{1}$$

where $\boldsymbol{\alpha} \in \mathcal{R}^{(T_1,T_2)}$ is the alignment matrix. $\boldsymbol{c}$ is then fed to another network g to generate the output $\boldsymbol{y}$: $\boldsymbol{y} = g(\boldsymbol{c})$. Networks of f and g could be easily replaced with parallel structures because both of them obtain consistent lengths of input and output. Therefore, the key to build a non-autoregressive TTS model lies on parallel alignment prediction. In previous works, ParaNet (Peng et al., 2020) and FastSpeech (Ren et al., 2019) learn the sequence alignment through distillation from autoregressive models such as DeepVoice 3 (Ping et al., 2018) and TransformerTTS (Li et al., 2019). FastSpeech 2 (Ren et al., 2021) improves the two-staged training of FastSpeech by introducing an external aligner named *forced alignment* (McAuliffe et al., 2017), but *forced alignment* itself requires an unsupervised training. Flow-TTS (Miao et al.) is a flow-based non-autoregressive model, its alignment is predicted from text sequence only, which is not reliable, resulting in an unstable training. Similar limitation is encountered for EATS (Donahue et al., 2021). Although EATS alleviates the limitation by introducing *dynamic time warping* (DTW) (Sakoe, 1971; Sakoe & Chiba, 1978), the training of EATS is still expensive. Unlike most TTS models which require training a neural vocoder (van den Oord et al., 2016; Prenger et al., 2019; Valin & Skoglund, 2019) to produce the waveforms from the generated melspectrograms, EATS directly produces waveforms from text sequences without relying on intermediate representations such as melspectrograms or linguistic features in an end-to-end manner. Similar end-to-end models include FastSpeech 2s (Ren et al., 2021) and Wave-Tacotron (Weiss et al., 2021). FastSpeech 2s requires external aligners while Wave-Tacotron is autoregressive. Probably the most comparable model to EfficientTTS is Glow-TTS (Kim et al., 2020).

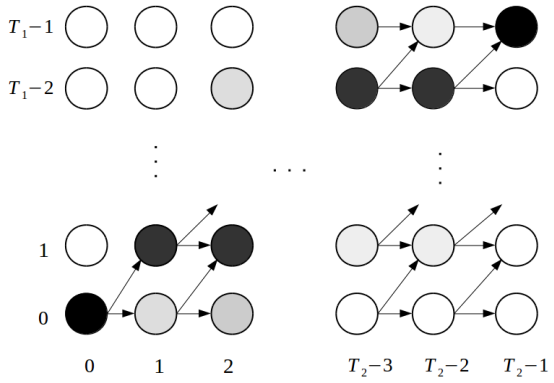

Figure 1. Schematics of the monotonic alignment. Each node $\alpha_{i,j}$ represents the possibility (shown as the shade of gray) that output timestep y_j(horizontal axis) attends on the input token x_i(vertical axis). At each output timestep, monotonic attention either move forward to next token or stay unmoved.

Glow-TTS is another flow-based non-autoregressive model without external aligner, it extracts the duration of each input token using an independent algorithm which precludes the use of standard back-propagation. Unlike the aforementioned models, EfficientTTS jointly learns sequence alignment and speech generation through a single network, while still maintaining a stable training.

2.2. Monotonic alignment modeling

As noted in section 2.1, a general attention mechanism inspects every input step at every output timestep. Such a mechanism often encounters misalignment and is quite costly to train, especially for long sequences. Therefore, it must be helpful if there is some prior knowledge incorporated. In general, as shown in Fig. 1, the alignment should follow strict criteria including: (1) **Monotonicity**, at each output timestep, the aligned position never rewinds; (2) **Continuity**, at each output timestep, the aligned position move forward at most one step; (3) **Completeness**, the aligned positions must cover all the positions of input tokens. Lots of prior studies have been proposed to ensure correct alignments (Li et al., 2020; Chiu & Raffel, 2018; Raffel et al., 2017), but most of them require sequential steps and often fail to meet all the criteria mentioned above. In this work, we propose a novel approach to produce monotonic alignment effectively and efficiently.

3. Monotonic Alignment Modeling Using IMV

We start this section by proposing the *index mapping vector* (IMV), and then we leverage IMV in monotonic alignment modeling. We further show how to incorporate IMV into a

general sequence-to-sequence model.

3.1. Definition of IMV

Let $\boldsymbol{\alpha} \in \mathcal{R}^{(T_1,T_2)}$ be the alignment matrix between input sequence $\boldsymbol{x} \in \mathcal{R}^{T_1}$ and output sequence $\boldsymbol{y} \in \mathcal{R}^{T_2}$. We define *index mapping vector* (IMV) $\boldsymbol{\pi}$ as sum of index vector $\boldsymbol{p} = [0, 1, \cdots, T_1 - 1]$, weighted by $\boldsymbol{\alpha}$:

$$\pi_j = \sum_{i=0}^{T_1-1} \alpha_{i,j} * p_i, \tag{2}$$

where, $0 \leq j \leq T_2 - 1$, $\boldsymbol{\pi} \in \mathcal{R}^{T_2}$, and $\sum_{i=0}^{T_1-1} \alpha_{i,j} = 1$. We can understand IMV as the *expected location* for each output timestep, where the expectation is over all possible input locations ranging from 0 to $T_1 - 1$.

3.2. Monotonic alignment modeling using IMV

Continuity and Monotonicity. We first show that given alignment matrix $\boldsymbol{\alpha}$ meets the continuity and monotonicity criteria, then we have:

$$0 \leq \Delta\pi_i \leq 1, \tag{3}$$

where, $\Delta\pi_i = \pi_i - \pi_{i-1}, 1 \leq i \leq T_2 - 1$. Detailed verification is shown in Appendix A.

Completeness. Given $\boldsymbol{\pi}$ is continuous and monotonic across timesteps, completeness criteria is equivalent to the following boundary constraints:

$$\pi_0 = 0, \tag{4}$$

$$\pi_{T_2-1} = T_1 - 1. \tag{5}$$

This can be deduced from $\boldsymbol{\alpha}_0 = [1, 0, ..., 0]$ and $\boldsymbol{\alpha}_{T_2-1} = [0, 0, ..., 1]$.

3.3. Incorporate IMV into networks

We propose two strategies to incorporate IMV into sequence-to-sequence networks: Soft Monotonic Alignment (SMA) and Hard Monotonic Alignment (HMA).

Soft Monotonic Alignment (SMA). To let sequence-to-sequence models be trained with the constraints given by Eq. (3 - 5), a natural idea is to turn these constraints into training objectives. We formulate these constraints as a SMA loss which is computed as:

$$\begin{aligned}\mathcal{L}_{\text{SMA}} = &\lambda_0 \| |\Delta\boldsymbol{\pi}| - \Delta\boldsymbol{\pi} \|_1 \\ &+ \lambda_1 \| |\Delta\boldsymbol{\pi} - 1| + (\Delta\boldsymbol{\pi} - 1) \|_1 \\ &+ \lambda_2 (\frac{\pi_0}{T_1 - 1})^2 \\ &+ \lambda_3 (\frac{\pi_{T_2-1}}{T_1 - 1} - 1)^2,\end{aligned} \tag{6}$$

where $\| \cdot \|_1$ is ℓ^1 norm. $\lambda_0, \lambda_1, \lambda_2, \lambda_3$ are positive coefficients. As can be seen, $\mathcal{L}_{\text{SMA}}$ is non-negative, and it is zero only if $\boldsymbol{\pi}$ satisfies all the constraints. Computation of $\mathcal{L}_{\text{SMA}}$ requires alignment matrix $\boldsymbol{\alpha}$ only (index vector $\boldsymbol{p}$ is always known), therefore, it is quite easy to incorporate SMA loss into sequence-to-sequence networks without changing their network structures. In general, SMA plays a similar role as Guided Attention (Tachibana et al.) which speeds up the model convergence and improves robustness. However, SMA outperforms Guided Attention because SMA theoretically provides more accurate constraints on the alignments.

Hard Monotonic Alignment (HMA). While SMA allows sequence-to-sequence networks to produce monotonic alignments by incorporating with a SMA loss, the training of these networks may remain costly because the networks cannot produce monotonic alignments at the beginning phase of training. Instead, they learn this ability step by step. To address this limitation, we propose another monotonic strategy which we call HMA, for Hard Monotonic Alignment. The core idea of HMA is to build a network with a strategically designed structure, allowing for producing monotonic alignments without supervision.

First, we compute IMV $\boldsymbol{\pi}'$ from the alignment matrix $\boldsymbol{\alpha}$ according to Eq. (2). Although $\boldsymbol{\pi}'$ is not monotonic, it is then transformed to $\boldsymbol{\pi}$, a strictly monotonic IMV by enforcing $\Delta\boldsymbol{\pi} \geq 0$ using a ReLU activation.

$$\Delta\pi'_j = \pi'_j - \pi'_{j-1}, \qquad 0 < j \leq T_2 - 1, \tag{7}$$

$$\Delta\pi_j = \mathrm{ReLU}(\Delta\pi'_j), \qquad 0 < j \leq T_2 - 1, \tag{8}$$

$$\pi_j = \sum_{m=0}^{j} \Delta\pi_m, \qquad 0 \leq j \leq T_2 - 1, \Delta\pi_0 = 0. \tag{9}$$

Furthermore, to restrict the domain of $\boldsymbol{\pi}$ to the interval $[0, T_1 - 1]$ as given in Eq. (4 - 5), we multiply $\boldsymbol{\pi}$ by a positive scalar:

$$\pi^*_j = \pi_j * \frac{T_1 - 1}{\max(\boldsymbol{\pi})} = \pi_j * \frac{T_1 - 1}{\pi_{T_2-1}}, \tag{10}$$

where the maximum of $\boldsymbol{\pi}$ is π_{T_2-1} because $\boldsymbol{\pi}$ is monotonic increasing across timesteps.

Recall that our goal is to construct a monotonic alignment. To achieve this, we introduce the following transformation to reconstruct the alignment by leveraging a Gaussian kernel centered on $\boldsymbol{\pi}^*$:

$$\alpha'_{i,j} = \frac{\exp\left(-\sigma^{-2}(p_i - \pi^*_j)^2\right)}{\sum_{m=0}^{T_1-1} \exp\left(-\sigma^{-2}(p_m - \pi^*_j)^2\right)}, \tag{11}$$

where, σ^2 denotes the hyper-parameter representing alignment variation. $\boldsymbol{\alpha'}$ serves as a replacement of original alignment $\boldsymbol{\alpha}$. The difference between $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha'}$ is that $\boldsymbol{\alpha'}$ is guaranteed to be monotonic while $\boldsymbol{\alpha}$ has no constraint on monotonicity. HMA reduces the difficulty of learning monotonic

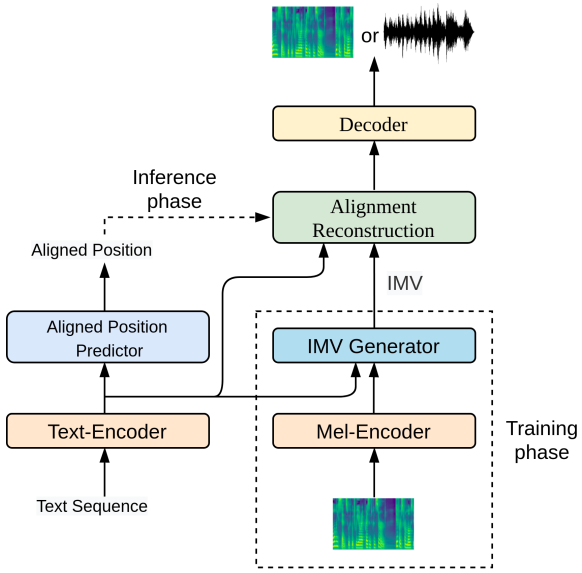

Figure 2. Overall model architecture.

alignments, thus improves the training efficiency. Similar to SMA, HMA can be employed to any sequence-to-sequence networks.

In next section, we propose a new TTS architecture - EfficientTTS that uses the monotonic strategy HMA as the internal aligner. The use of HMA allows the network training with strict monotonic constraints, and thus makes EfficientTTS fast and efficient for training.

4. EfficientTTS Architecture

The overall architecture design of EfficientTTS is shown in Fig. 2. In the training phase we compute the IMV from the hidden representations of text sequence and melspectrogram through an IMV generator. The hidden representations of text sequence and melspectrogram are learned from a text-encoder and a mel-encoder respectively. IMV is then converted to a 2-dimensional alignment matrix which is used to generate the time-aligned representation through an alignment reconstruction layer. The time-aligned representation is passed through a decoder producing the output melspectrograms or waveforms. We concurrently train an aligned position predictor which learns to predict aligned position for each input text token. In the inference phase, we reconstruct the alignment matrix from predicted aligned positions. We show the implementation of each components in the following subsections and more details including the pseudocode in Appendix B.

4.1. Text-Encoder and Mel-Encoder

We use a text-encoder and a mel-encoder to convert text symbols and melspectrograms to powerful hidden representations respectively.

We follow FastSpeech (Ren et al., 2019) in implementing the text-encoder, which consists of a text embedding layer and a stack of transformer FFT blocks. The use of transformer FFT structure enables text-encoder to learn both the local and global information, which is very important to alignment predictions.

In the implementation of the mel-encoder, we first convert melspectrograms to high-dimensional vectors through a linear projection. The linear projection is followed by a stack of convolutions interspersed with weight normalization, Leaky ReLU activation, and residual connection. Note that mel-encoder is only used in the training phase.

4.2. IMV generator

In order to generate a monotonic IMV in the training phase, we first learn the alignment $\boldsymbol{\alpha}$ between the input and output through a scaled dot-product attention (Vaswani et al., 2017) as given in Eq. (12), and then compute IMV from $\boldsymbol{\alpha}$.

$$\alpha_{i,j} = \frac{\exp\left(-D^{-0.5}(\boldsymbol{q_j} \cdot \boldsymbol{k_i})\right)}{\sum_{m=0}^{T_1-1} \exp\left(-D^{-0.5}(\boldsymbol{q_j} \cdot \boldsymbol{k_m})\right)}, \tag{12}$$

where, $\boldsymbol{q}$ and $\boldsymbol{k}$ are the outputs of mel-encoder and text-encoder, and D is the dimensionality of $\boldsymbol{q}$ and $\boldsymbol{k}$.

In our preliminary experiments, we follow Eq. (7 - 10) to generate IMV. However, we observe some alignment errors during training phase, especially for long sequences generation. It seems that the cumulative sum operation in Eq. (9) tends to accumulate alignment errors. To address this limitation, we introduce a bi-directional cumulative sum operation which allows the models to learn accurate alignment for long sequences.

For each timestep j, we accumulate $\Delta\pi$ in both the forward and backward direction as Eq. (13 - 14).

$$\pi_j^f = \sum_{m=0}^{j} \Delta\pi_m, \quad \pi_j^b = \sum_{m=j}^{T_2-1} \Delta\pi_m, \tag{13}$$

$$\pi_j = \pi_j^f - \pi_j^b. \tag{14}$$

Because $\boldsymbol{\pi}$ in Eq. (14) is monotonically increasing across timesteps, the minimum and maximum of $\boldsymbol{\pi}$ is π_0 and π_{T_2-1} respectively. Therefore, we are able to restrict $\boldsymbol{\pi}$ to the interval $[0, T_1 - 1]$ through a linear transformation:

$$\pi_j^* = \frac{\pi_j - \pi_0}{\pi_{T_2-1} - \pi_0} * (T_1 - 1). \tag{15}$$

4.3. Aligned position predictor

In the inference phase, the model needs to predict the IMV $\boldsymbol{\pi}$ from the hidden representation of text sequence $\boldsymbol{h}$, which

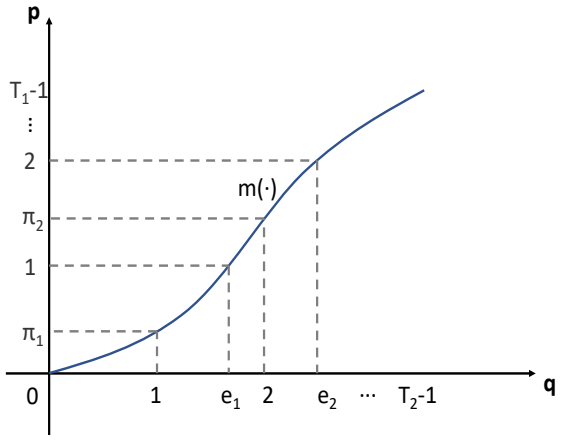

Figure 3. Schematics of $m(\cdot)$. $\boldsymbol{p}, \boldsymbol{q}$ are indexes of input sequence and output sequence respectively. $\boldsymbol{\pi}$ is IMV, $\boldsymbol{e}$ is the aligned position in output timestep for each input token.

is challenging in practice. There are two limitations: (1) $\boldsymbol{\pi}$ is time-aligned, which is in high resolution but $\boldsymbol{h}$ is in low resolution; (2) Each prediction of π_i affects later prediction of π_j $(j > i)$ due to cumulative sum operation introduced in Eq. (9), making it difficult to predict $\boldsymbol{\pi}$ in parallel. Fortunately, the limitations can be alleviated by predicting the aligned positions $\boldsymbol{e}$ of each input token instead. Let $\boldsymbol{q} = [0, 1, ..., T_2 - 1]$ be the index vector of $\boldsymbol{\pi}$. We define transformation $m(\cdot)$ as the mapping between $\boldsymbol{\pi}$ and $\boldsymbol{q}$: $\boldsymbol{\pi} = m(\boldsymbol{q})$. Since $\boldsymbol{\pi}$ is monotonically increasing with respect to $\boldsymbol{q}$, therefore, $m(\cdot)$ is a monotonic transformation thus invertible:

$$\boldsymbol{q} = m^{-1}(\boldsymbol{\pi}), \tag{16}$$

The aligned positions $\boldsymbol{e}$ in output timestep for each input token can be computed as:

$$\boldsymbol{e} = m^{-1}(\boldsymbol{p}), \quad \boldsymbol{p} = [0, 1, ..., T_1 - 1].$$

We illustrate the relations of $m(\cdot), \boldsymbol{e}, \boldsymbol{\pi}$ in Fig. 3. In order to compute $\boldsymbol{e}$, we first compute the probability density matrix $\boldsymbol{\gamma}$ utilizing a similar transformation as Eq. (11). The only difference is that the probability density is computed on different dimensions.

$$\gamma_{i,j} = \frac{\exp\left(-\sigma^{-2}(p_i - \pi_j)^2\right)}{\sum_{n=0}^{T_2-1} \exp\left(-\sigma^{-2}(p_i - \pi_n)^2\right)}. \tag{17}$$

The aligned position $\boldsymbol{e}$ is the weighted sum of the output index vector $\boldsymbol{q}$ weighted by $\boldsymbol{\gamma}$.

$$e_i = \sum_{n=0}^{T_2-1} \gamma_{i,n} * q_n. \tag{18}$$

As can be seen, the computation of $\boldsymbol{e}$ is differentiable which allows for training by gradients methods, thus can be used in both training and inference. Besides, $\boldsymbol{e}$ is predictable, because: (1) The resolution of $\boldsymbol{e}$ is the same as $\boldsymbol{h}$; (2) we can learn relative position $\Delta\boldsymbol{e}$, $(\Delta e_i = e_i - e_{i-1},\ 1 \leq i \leq T_1 - 1)$ instead of directly learning $\boldsymbol{e}$ to overcome the second limitation.

The aligned position predictor consists of 2 convolutions, each followed by the layer normalization and ReLU activation. We regard $\Delta\boldsymbol{e}$ computed from $\boldsymbol{\pi}$ as the training target. The loss function between the estimated position $\Delta\hat{\boldsymbol{e}}$ and the target one $\Delta\boldsymbol{e}$ is computed as:

$$\mathcal{L}_{ap} = \| \log\left(\Delta\hat{\boldsymbol{e}} + \epsilon\right) - \log\left(\Delta\boldsymbol{e} + \epsilon\right) \|_1, \tag{19}$$

Where, ϵ is a small number to avoid numerical instabilities. The goal with log-scale loss is to accurately fit small values, which tends to be more important towards the later phases of training. Aligned position predictor is learned jointly with the rest of the model. Because we generate alignments by leveraging the aligned positions, as a side benefit, EfficientTTS inherits the ability of speech rate control as duration-based non-autoregressive TTS models.

4.4. Alignment reconstruction

In order to map input hidden representations $\boldsymbol{h}$ to time-aligned representations, an alignment matrix is needed, for both training and inference. We can alternatively construct alignment from IMV $\boldsymbol{\pi}$ or the aligned positions $\boldsymbol{e}$. For most situations, Eq. (11) is an effective way to reconstruct alignment matrix from $\boldsymbol{\pi}$. But because we have to use aligned positions rather than $\boldsymbol{\pi}$ during inference, to be consistent, we reconstruct alignment matrix from the aligned positions $\boldsymbol{e}$ for training as well. Specifically, we take the aligned positions $\boldsymbol{e}$ computed from Eq. (18) for training, and the predicted one from aligned position predictor for inference.

We follow similar idea of EATS (Donahue et al., 2021) in reconstructing the alignment matrix $\boldsymbol{\alpha'}$ by introducing a Gaussian kernel centered on aligned position $\boldsymbol{e}$.

$$\alpha'_{i,j} = \frac{\exp\left(-\sigma^{-2}(e_i - q_j)^2\right)}{\sum_{m=0}^{T_1-1} \exp\left(-\sigma^{-2}(e_m - q_j)^2\right)}, \tag{20}$$

where $\boldsymbol{q}$ is the index vector of output sequence. The length of output sequence T_2 is known in training and computed from $\boldsymbol{e}$ in inference:

$$T_2 = e_{T_1-1} + \eta * \Delta e_{T_1-1}, \tag{21}$$

where, η is a hyper-parameter which we set to 1.2 for all experiments.

Although the reconstructed alignment maybe not as accurate as the one computed by Eq. (11) (due to the low resolution of $\boldsymbol{e}$), the effect on the output is small because the network is able to compensate. As a result, we enjoy improvement

in speech quality caused by the increasing consistency of training and inference. We map the output of text-encoder $\boldsymbol{h}$ to a time-aligned representation by making use of $\boldsymbol{\alpha}'$ following Eq. (1). The time-aligned representation is then fed as input to decoder.

4.5. Decoder

Since both the input and the output of the decoder are time-aligned, it is easy to implement decoder with parallel structures. We develop three models based on EfficientTTS as shown in Fig. 4. We give a brief introduction in this section and show more implementation details in Appendix B.

EFTS-CNN. We first parameterize the decoder by a stack of convolutions. Mean square error (MSE) is used as the reconstruction loss.

EFTS-Flow. To let TTS models have the ability to control the variations of generated speech, we implement a flow-based decoder. In the training phase, we learn a transformation f from melspectrogram to a high dimensional Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$ by directly maximizing the likelihood. To improve the diversity of generated speech, we sample the latent variable $\boldsymbol{z}$ from Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$ during inference, and interpret $\boldsymbol{z}$ with a zero vector $\boldsymbol{o}$ using temperature factor t to get a new latent vector $\boldsymbol{z}'$. We further use $\boldsymbol{z}'$ as the input of the model and inverse the transformation f to produce the melspectrogram . For sake of simplicity we follow the decoder structure of Flow-TTS (Miao et al.) in implementing our flow-based decoder.

EFTS-Wav. To simplify the 2-staged training pipeline and train TTS models in a fully end-to-end manner, we also develop a text-to-wav model by incorporating EfficientTTS with a dilated convolutional adversarial decoder. we follow MelGAN (Kumar et al., 2019) in implementing convolutional adversarial decoder.

5. Experiments

In this section, we first compare the proposed models with their counterparts in terms of speech fidelity, training and inference efficiency. We then analyze the effectiveness of proposed monotonic alignment approach on both EFTS-CNN and Tacotron 2. We also demonstrate that proposed models can generate speech in great diversity at the end of this section. [1]

[1]Audio samples of the proposed models are available at: https://mcf330.github.io/EfficientTTSAudioSamples/

5.1. Experimental setup

Datasets. We conduct most of our experiments on an open-source standard Mandarin dataset from DataBaker[2], which consists of $10,000$ Chinese clips from a single female speaker with a sampling rate of 22.05kHZ. The length of the clips varies from 1 to 10 seconds and the clips have a total length of about 12 hours. We also conduct some experiments using LJ-Speech dataset (Ito, 2017), which is a 24-hour waveform audio set of a single female speaker with 131,00 audio clips and a sample rate of 22.05kHZ.

Counterpart models. We compare proposed models with autoregressive Tacotron 2 and non-autoregressive Glow-TTS in the following experiments. We directly use the open-source implementations of Tacotron 2[3] and Glow-TTS[4] with default configurations. We use HiFi-GAN (Kong et al., 2020) vocoder to produce waveforms from melspectrograms. We use the open-source implementation of HiFi-GAN[5] with HiFi-GAN-V1 configuration.

5.2. Comparison with counterpart models

Speech quality. We conduct a 5-scale mean opinion score (MOS) evaluation on DataBaker dataset to measure the quality of synthesized audios. Each audio is listened by at least 15 testers, who are all native speakers. We compare the MOS of the audio samples generated by EfficientTTS families with ground truth audios, as well as audio samples generated by counterpart models. The MOS results with 95% confidence intervals is shown in Table 2. We draw the observation that EfficientTTS families outperform counterpart models. Tacotron 2 suffers from a declining speech quality caused by the inconsistency between teacher forcing training and autoregressive inference, and Glow-TTS replicates the hidden representations of text sequence, which corrupts the continuity of hidden representations. EfficientTTS reconstructs the alignments using IMV, which is more expressive than token duration, therefore achieves better speech quality. In addition, the alignment part of EfficientTTS is trained together with the rest of the model, which further improves the speech quality. As our training settings may be different with original settings for counterpart models, we further compare our model EFTS-CNN with pertained models of Tacotron 2 and Glow-TTS on LJ-Speech dataset. As shown in Table 3, EFTS-CNN outperforms counterpart models on LJ-Speech too.

Training and Inference speed. Being end-to-end and fully parallel, the proposed models are very efficient for both

[2]https://www.data-baker.com/open_source.html

[3]https://github.com/NVIDIA/tacotron2

[4]https://github.com/jaywalnut310/glow-tts

[5]https://github.com/jik876/hifi-gan

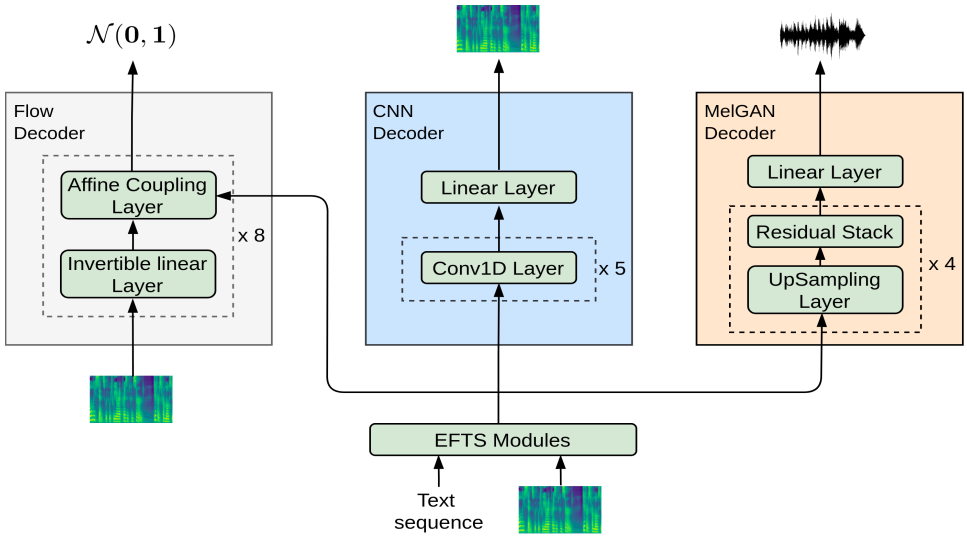

Figure 4. EfficientTTS families. From left to right are EFTS-Flow, EFTS-CNN and EFTS-Wav respectively.

Table 1. Quantitative results of training time and inference latency. We run training and inference on a single V100 GPU. We select 20 sentence for inference speed evaluation and run inference of each sentence 20 times to get an average inference latency. The lengths of the generated melspectrograms range from 110 to 802, with an average of 531. We exclude the time cost of transferring data between CPU and GPU in inference speed evaluation. HiFi-GAN is used to produce waveform from melspectrograms.

Model family	Training Time(h)	Training Speedup	Inference Time text-to-mel(ms)	Inference Speedup text-to-mel	Inference Time text-to-wav(ms)	Inference Speedup text-to-wav
Tacotron 2	54	-	780	-	824	-
Glow-TTS	120	0.45×	42	18.6×	86	9.6×
EFTS-CNN	6	**9**×	**8**	**97.5**×	52	15.8×
EFTS-Flow	32	1.7×	21	37.1×	65	12.7×
EFTS-Wav	-	-	-	-	**18**	**45.8**×

Table 2. The MOS with 95% confidence intervals for different methods on DataBaker. The temperature of latent variable z is set to 0.667 for both Glow-TTS and EFTS-Flow.

Method	MOS
Ground Truth	4.64 ± 0.07
Ground Truth (Mel+HiFi-GAN)	4.58 ± 0.13
Tacotron 2 (Mel+HiFi-GAN)	4.20 ± 0.11
Glow-TTS (Mel+HiFi-GAN)	3.97 ± 0.21
EFTS-CNN (Mel+HiFi-GAN)	$\mathbf{4.41 \pm 0.13}$
EFTS-Flow (Mel+HiFi-GAN)	$\mathbf{4.35 \pm 0.17}$
EFTS-Wav	$\mathbf{4.40 \pm 0.21}$

Table 3. The MOS with 95% confidence intervals for different methods on LJ-Speech. The temperature of latent variable z is set to 0.667 for Glow-TTS.

Method	MOS
Ground Truth	4.75 ± 0.12
Ground Truth (Mel+HiFi-GAN)	4.51 ± 0.13
Tacotron 2 (Mel+HiFi-GAN)	4.08 ± 0.13
Glow-TTS (Mel+HiFi-GAN)	4.13 ± 0.18
EFTS-CNN (Mel+HiFi-GAN)	$\mathbf{4.27 \pm 0.14}$

training and inference. Quantitative results of training time and inference latency are shown in Table 1. As can be seen, EFTS-CNN requires the least amount of training time. Although EFTS-Flow requires comparable training time with Tacotron 2, it is significantly faster than Glow-TTS. As for inference latency, EfficientTTS models are faster than Tacotron 2 and Glow-TTS. In particular, the inference latency of EFTS-CNN is 8ms which is 97.5× faster than Tacotron 2, and significantly faster than Glow-TTS. Thanks to the removal of melspectrogram generation, EFTS-Wav is significantly faster than 2-staged models, taking only 18ms to synthesize test audios from text sequences, which is 45.8× faster than Tacotron 2.

Table 4. Comparison of different monotonic alignment approaches on EFTS-CNN.

Models	Training Steps	MSE Loss
EFTS-HMA	60k	0.095
EFTS-SMA	450k	0.33
EFTS-NM	not converge	-

Table 5. The comparison of robustness between EFTS-CNN and Tacotron 2. We implement Tacotron 2 with different settings, including vanilla Tacotron 2, Tacotron 2 with SMA (T2-SMA), Tacotron 2 with HMA (T2-HMA).

Models	Repeats	Skips	Mispron-unciations	Error Rate
Tacotron 2	13	7	5	50%
T2-SMA	3	1	3	14%
T2-HMA	0	1	3	8%
EFTS-CNN	0	0	2	4%

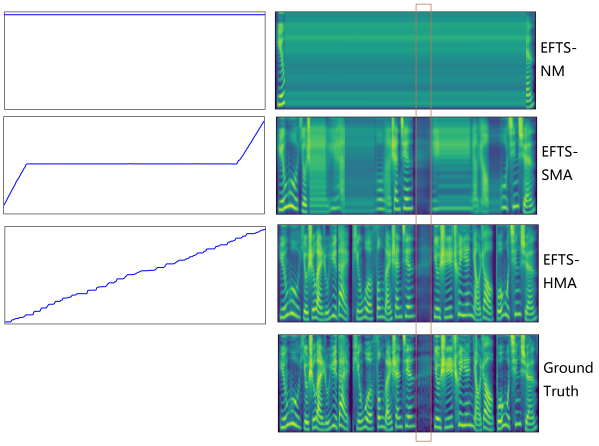

Figure 5. Comparisons of IMV and reconstructed melspectrograms. The plots are taken from the $150k^{th}$ training step of EFTS-NM and EFTS-SMA, and the $20k^{th}$ training step of EFTS-HMA. The left column illustrates the IMV while the right column plots the reconstructed melspectrograms.

5.3. Evaluation of monotonic alignment approach

In order to evaluate the behaviour of proposed monotonic alignment approach, we conduct several experiments on EFTS-CNN and Tacotron 2. We first compare the training efficiency on EFTS-CNN, and then conduct a robustness test on both Tacotron 2 and EFTS-CNN.

Experiments on EFTS-CNN. We train EFTS-CNN with different settings, including: (1) EFTS-HMA, default implementation of EFTS-CNN, with a hard monotonic IMV generator. (2) EFTS-SMA, an EFTS-CNN model with a soft monotonic IMV generator. (3) EFTS-NM, an EFTS-CNN model with no constraints on monotonicity. The network structure of EFTS-NM is the same as EFTS-SMA, except that EFTS-SMA is trained with SMA loss while EFTS-NM is trained without SMA loss. We first find that EFTS-NM does not converge at all, its alignment matrix is not diagonal, while both EFTS-SMA and EFTS-HMA are able to produce reasonable alignment. We illustrate the IMV and reconstructed melspectrogram in training phase in Fig. 5 for all the models. As can be seen, EFTS-HMA achieves a significant speed-up over EFTS-SMA. Therefore, we can conclude that the monotonic alignment is quite essential for proposed models. Our approaches, for both SMA and HMA, succeed to learn monotonic alignments while the vanilla attention mechanism fails. Thanks to the strict monotonicity constraints, EFTS-HMA significantly improve the model performance. More training details is shown on Table 4.

Robustness. Many TTS models encounter misalignment at synthesis, especially for autoregressive models. We analyze the attention errors for EfficientTTS in this subsection, the errors are including: repeated words, skipped words and mispronunciations. We perform a robustness evaluation on a test set of 50 sentences, which includes particularly challenging cases for TTS systems, such as particularly long sentences, repeated letters etc. We compare EFTS-CNN with Tacotron 2. We also incorporate SMA and HMA into Tacotron 2 for a more detailed comparison (The detailed implementations of Tacotron2-SMA and Tacotron2-HMA and more experimental results are shown in Appendix D.). The experimental results of robustness test are shown in Table 5. It can be seen that EFTS-CNN effectively eliminate repeats errors and skips errors while Tacotron 2 encounters many errors. However, the synthesis errors are significantly reduced for Tacotron 2 by leveraging SMA or HMA, which indicates that the proposed monotonic alignment approach can improve the robustness for TTS models.

5.4. Diversity

To synthesize speech samples in great diversity, most of TTS models make use of external conditions such as style embedding or speaker embedding, or just rely on drop-out during inference. However, EfficientTTS is able to synthesize varieties of speech samples in several ways, including: (1) Synthesizing speech with different alignment scheme. The alignment scheme could either be an IMV which is extracted from existing audio by mel-encoder and IMV generator, or a sequence of aligned positions; (2) Synthesizing speech with different speech rate by multiplying a scalar across predicted aligned positions, which is similar to other duration-based non-autoregressive models; (3) Synthesizing speech with different speech variations for EFTS-Flow by changing the temperature t of latent variable z during inference. We plot varieties of melspectrograms generated from

the same text sequences in Appendix C.

6. Conclusions And Future Works

In this work, we propose a non-autoregressive architecture which enables high quality speech generation as well as efficient training and synthesis. We develop a family of models based on EfficientTTS covering text-to-melspectrogram and text-to-waveform generation. Through extensive experiments, we observe improved quantitative results including training efficiency, synthesis speed, robustness, as well as speech quality. We show that the proposed models are very competitive compared with existing TTS models.

There are many possible directions for future work. EfficientTTS enables not only generating speech at a given alignment and but also extracting alignment from given speech, making it an excellent candidate for voice conversion and singing synthesis. It is also a good choice to apply the proposed monotonic alignment approach to other sequence-to-sequence tasks where monotonic alignment matters, such as Automatic Speech Recognition (ASR), Neural Machine Translation (NMT), and Optical Character Recognition (OCR).

References

Bahdanau, D., Cho, K., and Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*, 2015.

Chiu, C.-C. and Raffel, C. Monotonic Chunkwise Attention. In *International Conference on Learning Representations*, 2018.

Donahue, J., Dieleman, S., Binkowski, M., Elsen, E., and Simonyan, K. End-to-End Adversarial Text-to-Speech. In *International Conference on Learning Representations*, 2021.

Ito, K. The lj speech dataset. 2017. URL https://keithito.com/LJ-Speech-Dataset/.

Kim, J., Kim, S., Kong, J., and Yoon, S. Glow-TTS: A Generative Flow for Text-to-Speech via Monotonic Alignment Search. In *Advances in Neural Information Processing Systems*, 2020.

Kong, J., Kim, J., and Bae, J. HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis . In *Advances in Neural Information Processing Systems*, 2020.

Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., de Brebisson, A., Bengio, Y., and Courville., A. MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis. In *Advances in Neural Information Processing Systems*, 2019.

Li, N., Liu, S., Liu, Y., Zhao, S., Liu, M., and Zhou, M. Close to Human Quality TTS with Transformer. In *AAAI*, 2019.

Li, N., Liu, S., Liu, Y., Zhao, S., Liu, M., and Zhou, M. MoBoAligner: a Neural Alignment Model for Non-autoregressive TTS with Monotonic Boundary Search. In *Interspeech*, 2020.

McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., and Sonderegger, M. Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi. In *Interspeech*, 2017.

Miao, C., Liang, S., Chen, M., Ma, J., Wang, S., and Xiao, J. Flow-TTS: A Non-Autoregressive Network for Text to Speech Based On Flow. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Peng, K., Ping, W., Song, Z., and Zhao, K. Non-Autoregressive Neural Text-to-Speech. In *International Conference on Machine Learning*, 2020.

Ping, W., Peng, K., Gibiansky, A., Arik, S. O., Kannan, A., Narang, S., Raiman, J., and Miller, J. Deep Voice 3: 2000-Speaker Neural Text-to-Speech. In *International Conference on Learning Representations*, 2018.

Ping, W., Peng, K., and Chen, J. ClariNet: Parallel Wave Generation in End-to-End Text-to-Speech. In *International Conference on Learning Representations*, 2019.

Prenger, R., Valle, R., and Catanzaro, B. WaveGlow: A Flow-Based Generative Network For Speech Synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

Raffel, C., Luong, M.-T., Liu, P. J., Weiss, R. J., and Eck, D. Online And Linear-time Attention by Enforcing Monotonic Alignments. In *International Conference on Machine Learning*, 2017.

Ren, Y., Ruan, Y., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. FastSpeech: Fast, Robust and Controllable Text to Speech. In *Advances in Neural Information Processing Systems*, 2019.

Ren, Y., Hu, C., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. FastSpeech 2: Fast and High-Quality End-to-End Text to Speech . In *International Conference on Learning Representations*, 2021.

Sakoe, H. Dynamic-Programming Approach to Continuous Speech Recognition. In *1971 Proc. the International Congress of Acoustics, Budapest*, 1971.

Sakoe, H. and Chiba, S. Dynamic Programming Algorithm Optimization For Spoken Word Recognition. *IEEE Transactions On Acoustics, Speech, And Signal Processing*, 26 (1):43–49, 1978.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., and Ryan, R. S. Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. In *ICASSP 2018-2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Tachibana, H., Uenoyama, K., and Aihara, S. Efficiently Trainable Text-to-Speech System Based On Deep Convolutional Networks With Guided Attention. In *ICASSP 2018-2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Valin, J.-M. and Skoglund, J. LPCNet: Improving Neural Speech Synthesis Through Linear Prediction. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

Valle, R., Shih, K. J., Prenger, R., and Catanzaro, B. Flowtron: an Autoregressive Flow-based Generative Network for Text-to-Speech Synthesis. In *International Conference on Learning Representations*, 2021.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. WaveNet: A Generative Model for Raw Audio. In *9th ISCA Speech Synthesis Workshop*, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., N.Gomez, A., Kaiser, , and Polosukhin, I. Attention Is All You Need . In *Advances in Neural Information Processing Systems*, 2017.

Wang, Y., Skerry-Ryan, R., Stanton, D., Y. Wu, R. J. W., Jaitly, N., and Yang, Z. Tacotron: Towards End-to-End Speech Synthesis. In *Interspeech*, 2017.

Weiss, R. J., Skerry-Ryan, R., Battenberg, E., Mariooryad, S., and Kingma, D. P. Wave-Tacotron: Spectrogram-Free End-to-End Text-to-Speech Synthesis. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.