FPUAS : Fully Parallel UFANS-based End-to-End Acoustic System with 10x Speed Up

Dabiao Ma, Zhiba Su, Yuhao Lu
Turing Robot co.ltd
Multi-modal Group
Beijing, China

arXiv:1812.05710v1 [eess.AS] 12 Dec 2018

Abstract

A lightweight end-to-end acoustic system is crucial in the deployment of text-to-speech tasks. Finding one that produces good audios with small time latency and fewer errors remains a problem. In this paper, we propose a new non-autoregressive, fully parallel acoustic system that utilizes a new attention structure and a recently proposed convolutional structure. Compared with the most popular end-to-end text-to-speech systems, our acoustic system can produce equal quality or better quality audios of with fewer errors and reach at least 10 times speed up of inference.

1 Introduction

Text-to-speech (TTS) systems aim to convert texts to human-like speeches. A typical modern end-to-end TTS system that highly utilizes deep learning to predict acoustic intermediate features from texts firstly, and then synthesizes speeches with those features by a vocoder e.g. Griffin-Lim (Griffin et al. 1984), WORLD (MORISE, YOKOMORI, and OZAWA 2016), WaveNet (van den Oord et al. 2016b). The difficulties to predict acoustic intermediate features from texts are to map each character or phoneme to acoustic feature frames and to find a neutral structure that captures long-term dependencies. Tacotron (Wang et al. 2017) uses an autoregressive attention (Bahdanau, Cho, and Bengio 2014) structure that requires computation results from previous steps to predict alignment, and utilizes combination of Gated Recurrent Unit (GRU) (Cho et al. 2014) and convolutions as encoder and decoder. Deep voice 3 (Ping et al. 2018) also uses an autoregressive attention but only uses convolutions to speed up training and inference. DCTTS (Tachibana, Uenoyama, and Aihara 2017) greatly speeds up the training of attention module by introducing guided attention which remains to be autoregressive, and uses convolutions with larger receptive field than Deep Voice 3.

The autoregressive attention structure greatly limits the inference speed of these systems (in the context of GPU computation) and may lead to serious error modes e.g. repeated words, mispronunciations, or skipped words (Ping et al. 2018). With larger receptive field DCTTS produces audios of better quality than Deep Voice 3 but at the cost of slower inference speed. The questions are that if we can find a fully parallel, non autoregressive attention structure that can perfectly determine the alignment with fewer errors and if we can find a fully parallel neutral structure with capacity comparable to recurrent structures to be encoder and decoder.

In this paper, we try to answer the first question by introducing a new alignment determination approach specified for monotonic attention, and the second question by using a recently proposed fully convolutional structure that performs well in TTS tasks with given alignment.

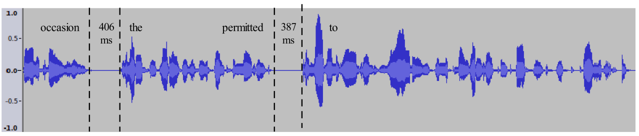
2 Overall system

2.1 Dataset and data preprocessing

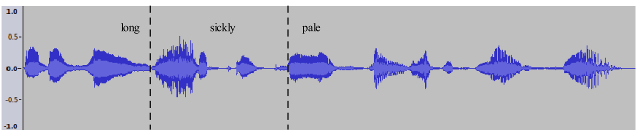
LJ speech (Ito 2017) is a public domain speech dataset consisting of 13100 pairs of text and audio clip of sample rate 22050 HZ without phoneme level alignment. The clips vary from 1 to 10 seconds and have a total length of about 24 hours. This dataset is used since DCTTS did experiments on it while Tacotron and Deep Voice 3 did experiments on their internal datasets that are not publicly available.

This dataset was examined and serious mislabeling was found that is particularly harmful to our method. Figure 1 shows two examples of the mislabeling. The two long silence clips within the audio of the first example is not labeled (e.g. as 'comma') in the text; It sounds like that no pause exists between 'long' and 'sickly', 'sickly' and 'pale' but 'comma' exists in the text in the second example. The first kind of mislabeling is much more common than the second one. To deal with the problem, very long silence (e.g. 300 ms) within an audio are replaced with a shorter one (e.g. 50 ms) by a simple script.

The textual features used are phonemes. There are two reasons for this. First, the alignment structure designed is specific for monotonic attention, and it is clear the mapping from phonemes to acoustic features is monotonic; Second, it is hard to correct when character based model makes mistakes, e.g., the word 'synthesized' never appears in the LJ speech dataset and DCTTS pronounces 'synthesized' as ['sʌntəsaɪzd]. With a phoneme based model, we could just modify the grapheme-to-phoneme system to correct mispronunciations. Here we used CMU US English



(a) LJ049-0026, text : on occasion the secret service has been permitted to have an agent riding in the passenger compartment with the president.



(b) LJ010-0266, text : and a long, sickly, pale face, with light hair,

Figure 1: Two examples of mislabeling

Dictionary (cmudict) to convert texts to phonemes.

In character based system, The 'blank' between words is taken as a special character. The 'blank' indication helps the model to distinguish the end of the previous phoneme from the start of the next phoneme. We did this differently by recording the relative positions of the phoneme. For example, 'text to speech' is converted to phonemes [T EH K S T; T UW; S P TY CH] with relative position (float number) [0/5, 1/5, 2/5, 3/5, 4/5; 0/2, 1/2; 0/4, 1/4, 2/4, 3/4] and inverse relative position [5/5, 4/5, 3/5, 2/5, 1/5; 2/2, 1/2; 4/4, 3/4, 2/4, 1/4]. During training, the relative positions are concatenated with the embeddings of each phoneme.

We did experiments on two kinds of acoustic features. The first is based on WORLD vocoder that uses mel-frequency cepstral coefficients(MFCCs). The second acoustic feature is linear-scale log magnitude spectrograms and mel-band spectrograms that can be feed into Griffin-Lim or a trained WaveNet vocoder. These features are the targets of the system.

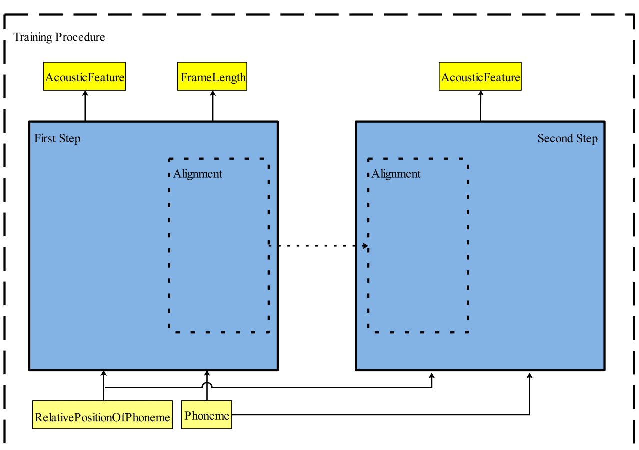


Figure 2: Training procedure

2.2 Training procedure

The training procedure consists of two steps. The first is to determine the alignment of phonemes to acoustic features, and get a well trained model to determine alignment during inference; The second is to train an encoder and a decoder with high capacity based on the alignment achieved from the first step which remains fixed during the second step. Please see Figure 2.

Step One, Alignment determination

This is the first step of the training procedure. The model consists of four parts. The encoder converts phonemes and relative position information to hidden states that are inputs to the decoder; The alignment module determines the alignment width of each phoneme from which the number of frames that should attend on that phoneme can be induced; After the alignment module completes, the decoder receives alignment information and converts the encoder hidden states to acoustic features. See Figure 3 (a) for the overall structure.

Encoder The encoder consists of one embedding operation and several dense layers. The encoder converts phonemes and relative position information to hidden states that is denoted as 'Value' in Figure 3 (a). Suppose the length of the phonemes is T_p, then 'Value' is a $T_p \times H_{enc}$ matrix where H_{enc} is the size of the hidden state.

Alignment Module This alignment module determines alignment width of the phoneme. This module consists of one embedding operation, one UFANS (Ma et al. 2018), several dense layers (or matrix multiplications) and one recurrent summation of a few scalars.

U-shaped Fast Acoustic Neutral Structure (UFANS) is recently proposed to speed up the inference of TTS systems (alignment given). UFANS has the following properties : fully parallel and thus fast (computing with GPU), receptive field increasing exponentially with the depth of layers, using combination of abstractions of different scales of the input. So among numerous variants of recurrent neutral networks (RNN) and convolutional neutral networks (CNN), UFANS was chosen as one of the components. Here the output of UFANS is scalars.

Suppose the length of phonemes of one utterance is T_p, and after the computation of UFANS we get a sequence of scalars of length T_p : $[u_0, u_1, ..., u_{T_p-1}]$; the length of frames of the true acoustic features is T_a. To compute the alignment of each phoneme, we need to use two prior information. The first is that the ratio of length of frames of acoustic features and length of phonemes $\hat{r}_a = T_a/T_p$ is known during training (during inference this ratio is set to be the average ratio of all training data and can be changed to control the speaking speed), denoted as 'AverageRatio' in Figure 3 (a). The second is that for any phoneme the alignment width is positive and has a minimum value $\hat{r}_{min}$, denoted as 'MinimumWidth' in Figure 3 (a). With these priors, define the alignment width of phoneme i as :

$$r_i = max(0, u_i + \hat{r}_a) + \hat{r}_{min} \tag{1}$$

denoted as 'AlignmentWidth' in Figure 3 (a).

Now we need to relate the alignment width $r_i, i = 0, ..., T_p - 1$ of each phoneme to the frame index $j, j = 0, ..., T_a - 1$. The intuition is that the acoustic frame $\sum_{k=0}^{i-1} r_k + \frac{1}{2} r_i$ should be the one that attends most on phoneme i. And we need a structure that satisfies the intuition.

Positional encoding (Gehring et al. 2017) is a method to embed a sequence of absolute positions to a sequence of vectors. (Vaswani et al. 2017) uses sine and cosine functions of different frequencies and adds the (constant) positional encoding vectors to input embeddings. Deep voice 3 uses sine and cosine positional encoding in a similar way. They both take positional encoding as a supplement to help the computation of attention and the positional encoding vectors remain constant (In Deep voice 3 different speakers have different positional encodings, but for one single speaker it is fixed). Here we improve its position by making it the only structure of attention as well as making it variable and trainable in phoneme level.

Define the absolute alignment position of phoneme i as :

$$s_i = \sum_{k=0}^{i-1} r_k + \frac{1}{2} r_i, i = 0, ..., T_p - 1, r_{-1} = 0 \tag{2}$$

Now choose L float numbers log uniformly from range $[1.0, 10000.0]$ and get a sequence of frequencies $[f_0, ..., f_{L-1}]$. For phoneme i, the positional encoding vector of this phoneme is defined as :

$$\begin{aligned} vp_i &= [vp_{i,sin}, vp_{i,cos}], \\ [vp_{i,sin}]_k &= sin(\frac{s_i}{f_k}), \\ [vp_{i,cos}]_k &= cos(\frac{s_i}{f_k}), k = 0, ..., L - 1 \end{aligned} \tag{3}$$

And if we concatenate $vp_i, i = 0, ..., T_p - 1$ together, we get a matrix that represents positional encoding of all the phonemes, denoted as 'Key' in Figure 3 (a) :

$$P = [vp_0^T, ..., vp_{T_p-1}^T] \tag{4}$$

And similarly, for the frame j of the acoustic feature, the positional encoding vector is defined as :

$$\begin{aligned} va_j &= [va_{j,sin}, va_{j,cos}], \\ [va_{j,sin}]_k &= sin(\frac{j}{f_k}), \\ [va_{i,cos}]_k &= cos(\frac{j}{f_k}), k = 0, ..., L - 1 \end{aligned} \tag{5}$$

And if we concatenate all the vectors, we get the matrix that represents positional encoding of all the acoustic frames, denoted as 'Query' in Figure 3 (a) :

$$F = [va_0^T, ..., va_{T_a-1}^T] \tag{6}$$

And now define the attention matrix as :

$$A = FP^T, A_{ji} = vp_i va_j^T, i = 0, ..., T_p - 1, j = 0, ..., T_a - 1 \tag{7}$$

That is, the attention of frame j on phoneme i is proportional to the inner product of positional encoding vector of phoneme i and positional encoding vector of frame j. This inner product can be rewritten as :

$$\begin{aligned} vp_i va_j^T &= \sum_f (cos(\frac{s_i}{f}) cos(\frac{j}{f}) + sin(\frac{s_i}{f}) sin(\frac{j}{f})) \\ &= \sum_f cos(\frac{s_i}{f} - \frac{j}{f}) \end{aligned} \tag{8}$$

It is clear when $j = s_i$, the frame j is the one that attends most on phoneme i. The attention A is normalized along the phoneme dimension, that is :

$$\hat{A}, \hat{A}_{ji} = \frac{A_{ji}}{\sum_i A_{ji}} \tag{9}$$

Now $\hat{A}_{ji}$ represents how much frame j attends on phoneme i.

Now define the number of frames that attend more on phoneme i than other phonemes to be its attention width w_i. Though the frame s_i attends most on phoneme i, the alignment width r_i and w_i are not the same. See Appendix A.

Sine and cosine positional encoding alignment attention has two very important properties that make it suitable for this task. In brief, function $g(x) = \sum_f cos(\frac{x-s}{f})$ has a heavy tail that makes one acoustic frame able to receive phoneme information far away; The gradient function $|\dot{g}(s)| = |\sum_f sin(\frac{x-s}{f})|$ is insensitive to the term $x - s$. See details in Appendix B.

Decoder The decoder consists of several convolutions with gated activation (van den Oord et al. 2016a), several Dropout (Srivastava et al. 2014) operations and one dense layer. The convolutions each has a very small receptive field (e.g. 5).

To determine the alignment, the decoder should have a small receptive field. If these Gated convolutions are replaced by another structure with high capacity (e.g. UFANS), the learning of alignment will be greatly disturbed. See appendix C.

The output of the last dense layer is the predicted acoustic features that are feed into the Loss module.

Loss Module The loss function consists of two parts.

The first $LOSS_1$ is the $L1$ or $L2$ mean error between predicted acoustic features and true features. Deep voice 3, DCTTS, Tacotron all use this loss.

The second $LOSS_2$ is to restrict the summation of alignment widths to be equal or close to T_a. Appendix A deduces that if the alignment is correct, then $\sum_{k=0}^{T_p-1} r_k = T_a$. We relax this restriction by using a threshold γ :

$$LOSS_2 = \begin{cases} \gamma, & \text{if } |\sum_{k=0}^{T_p-1} r_k - T_a| < \gamma \\ |\sum_{k=0}^{T_p-1} r_k - T_a|, & \text{otherwise} \end{cases} \tag{10}$$

The final loss $LOSS$ is a weighted addition of $LOSS_1$ and $LOSS_2$:

$$LOSS = LOSS_1 + \sigma LOSS_2 \tag{11}$$

Step Two, Training of new encoder and decoder

This step uses the well trained alignment module from step one to train a new encoder and a new decoder that has a high capacity. See Figure 3 (b).

Encoder The Encoder is the same with the encoder with that of the first step, but it is reinitializied and retrained.

Alignment Module The alignment module is the same with that of step one including all the parameters and remain fixed during training of step two. Except that instead of using $\hat{A}$ which takes combinations of 'Value' of phonemes as the input to the decoder, only the 'Value' of phoneme which occupies the most portion is used. The attention matrix $\widetilde{A}$ is obtained by :

$$\widetilde{A}_{ji} = \begin{cases} 1 & \text{if } i = \underset{k\in[0,\dots,T_p-1]}{\text{argmax}} \; A_{jk} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

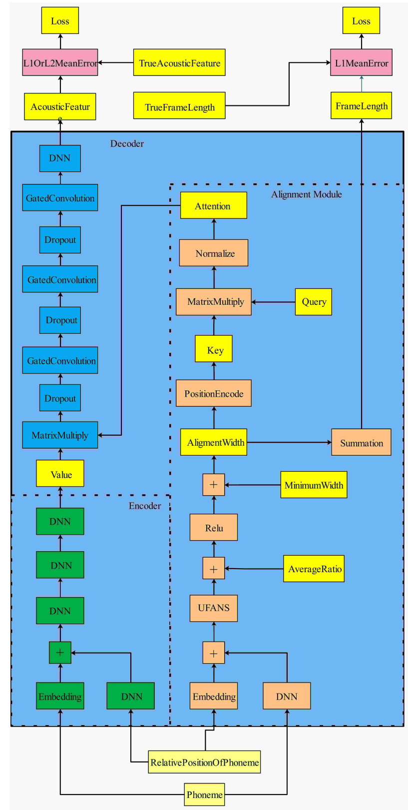
From the definition of attention width, $\widetilde{A}$ is actually the matrix of attention width.

Decoder UFANS is used as the decoder.

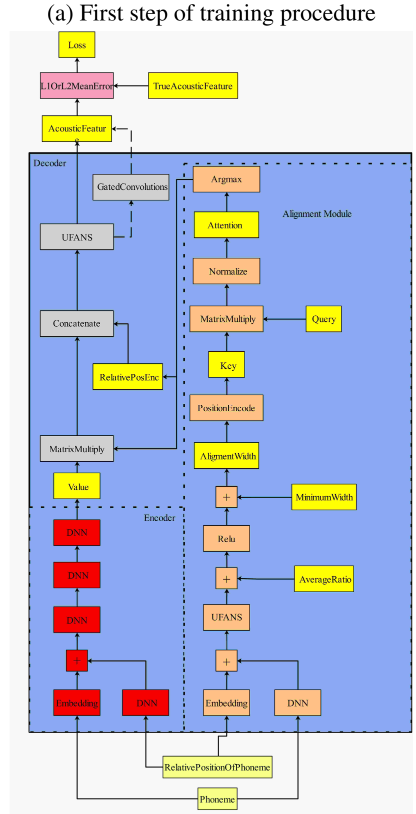
Before feeding attended hidden representations to UFANS, relative positional encoding is performed on $\widetilde{A}$ and concatenated with the hidden representations.

For phoneme i, there are w_i adjacent hidden representations that are encoding its information. To give the decoder more information, the relative position within the w_i representations (Wu, Watts, and King 2016) is created and concatenated with each representation. Here a three dimensional Gaussian positional encoding is used.

For the w_i representations $h_k, k = 0, ..., w_i - 1$ each is concatenated with :



(a) First step of training procedure



(b) Second step of training procedure

Figure 3: Detailed overall training procedure

$$
\begin{aligned}
&[\exp(-\frac{1}{2}(1.5 * \frac{k}{w_i} - 0.75)^2/0.16), \\
&exp(-\frac{1}{2}(0.75 * \frac{k}{w_i} - 0.75)^2/0.16), \\
&exp(-\frac{1}{2}(0.75 * \frac{k}{w_i})^2/0.16)]
\end{aligned} \tag{13}
$$

The computation of this relative positional encoding is fully parallel with operations on $\widetilde{A}$ and $s_i, i = 0, ..., Tp-1$.

Loss Module The loss is the same with $LOSS_1$ in the step one.

3 Experiments and Results

3.1 Training step one

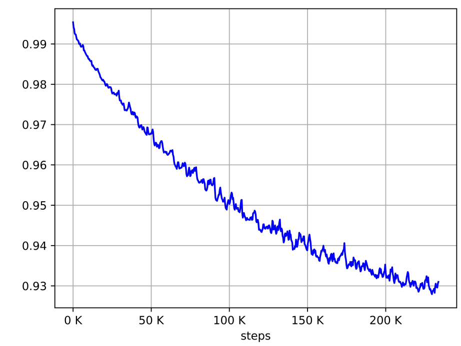
In this section, results of training step one are shown.

Model hyperparameters See Table 1 of main hyperparameters for MFCCs, see table 2 of main hyperparameters for spectrograms. Entry of column 'UFANS (alignment)' is the down-samplings/hidden layer size of UFANS. The entry of 'encoder' means the filter width/stride/dilation/hidden layer size of the gated convolutions of decoder. $\hat{r}_{min}$ and γ are different in two cases because the frame lengths are different.

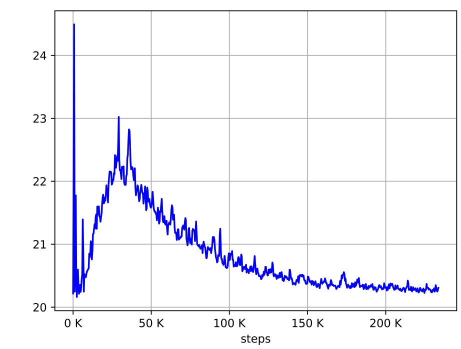
Loss curves Figure 4 is the loss curves of $LOSS_1$ and $LOSS_2$. To determine alignment only one of linear-scale and mel-band spectrograms is needed, here mel-band spectrograms and L_2 mean error are used here.

$LOSS_1$ converges but remains high. The reason is that step one uses a very simple decoder. But the purpose of this step is to determine alignment which is the most crucial. From figures of $LOSS_2$, it is known that the sum of alignment width stays in a reasonable level during training.

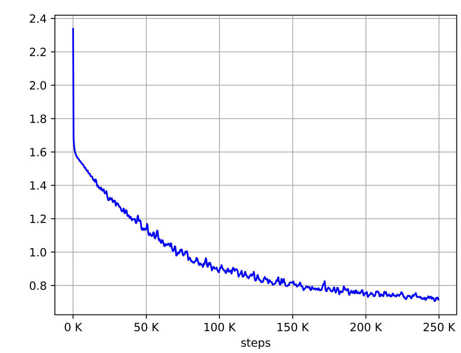
Performance evaluation Since the step of alignment determination is specific to this architecture, it can not be compared to Tacotron, DCTTS or Deep voice 3. To evaluate its performance, two methods are used. The first is to randomly pick a number of utterances from training data and resynthesize, then listen to both resynthesized and real audios to check if they have the same phoneme durations. The second is to compute attention width of synthesized wavs, and compare with phoneme durations of real waveforms. 100 utterances are randomly selected from training data and resynthesized. Table 3 is the statistical result of how they match the real phoneme durations. The attention width comparison matches the result of Table 3, see Appendix D.1 for an example.



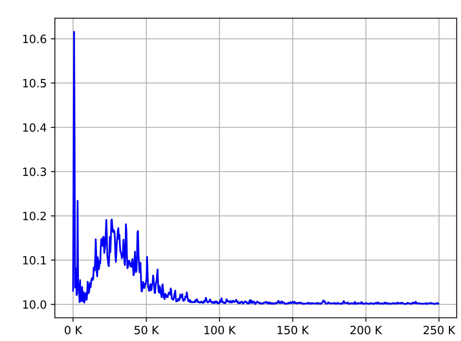
(a) $LOSS_1$ of MFCCs with L_2 loss function



(b) $LOSS_2$ of MFCCs with threshold 20



(c) $LOSS_1$ of mel-band spectrograms with L_2 loss function



(d) $LOSS_2$ of mel-band spectrograms with threshold 10

Figure 4: $LOSS_1$ and $LOSS_2$ of two acoustic features

H_{enc}	$\hat{r}_{min}$	L	UFANS (alignment)	decoder	dropout	γ	σ	optimizer/lr
512	2.0	512	4/512	5/1/1/512	0.5	20.0	0.01	Adam(Kingma and Ba 2014)/0.0004

Table 1: Hyperparameters for MFCCs

H_{enc}	$\hat{r}_{min}$	L	UFANS (alignment)	decoder	dropout	γ	σ	optimizer/lr
512	1.0	512	4/512	5/1/1/512	0.5	10.0	0.01	Adam/0.0004

Table 2: Hyperparameters for spectrograms

Though audios synthesized by step one are blurring, it is still compared with other models in sections below. Appendix D.2 is an attention plot (before and after operation $argmax$) of an utterance selected randomly from the Internet.

Mislabeling harms performance The mislabeling problem of the dataset is harmful to the determination of the alignment. Figure 5 shows resynthesized audios of 'LJ049-0026' from experiments with and without fixing the mislabeling problem. The unlabeled pause (see section 'Dataset and data preprocessing') causes phonemes 'ER M T UW' pronounce abnormally.

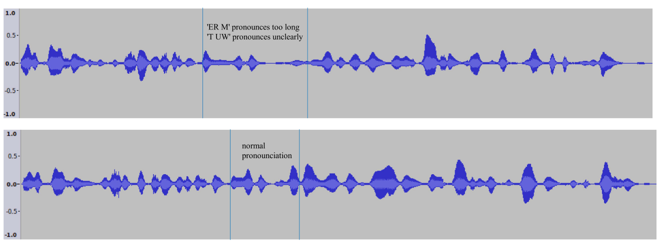


Figure 5: Resynthesized audios of 'LJ049-0026', mentioned in the section 'Dataset and data preprocessing'. The upper does not fix mislabeling problem

Gaussian function can not learn alignment If the sine and cosine positional encoding alignment attention is replaced by Gaussian function mentioned in appendix B, the model can not learn correct alignment. Figure 6 is the loss curve of $LOSS_1$ when using Gaussian function. Compared with Figure 4, it is clear that Gaussian function is not suitable for this task. See Appendix D.3 for the computed attention width.

Decoder with large receptive field can not learn alignment well To prove the theory that decoder with large receptive field can not learn alignment well, the decoder is replaced by a UFANS with a down-sampling number 6. Figure 7 is the curve of $LOSS_1$. Though the loss converges to a very low level (due to the high capacity of UFANS), the attention width computed is much worse than that computed with a simple decoder, see Appendix D.4. The synthesized audios often suffer from error modes like repeated words and skipped words. See Appendix D.4 for examples.

3.2 Training step two

In this section, results of training step two are shown. The inference speed, loss curves, quality and error mode of Tacotron, Deep Voice 3, DCTTS, and our method FPUAS (Fully Parallel UFANS-based Acoustic System) are compared.

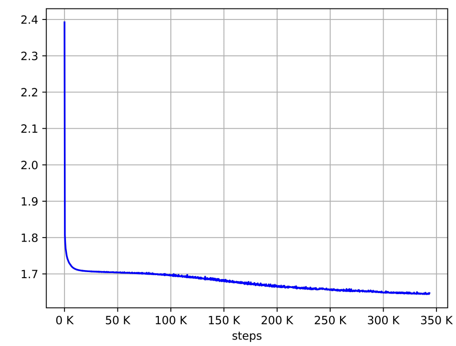


Figure 6: Loss curves of $LOSS_1$ when using Gaussian function, with mel-band spectrograms and L_2 loss function

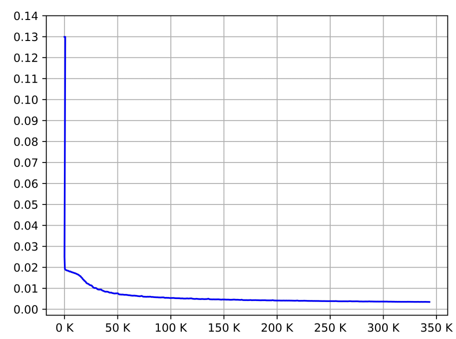


Figure 7: Loss curves of $LOSS_1$ when using UFANS as decoder, with mel-band spectrograms and L_2 loss function

Acoustic features The WORLD vocoder uses 60 dimensional mel-frequency cepstral coefficients, 2 dimensional band aperiodicity, 1 dimensional logarithmic fundamental frequency and their delta, delta-delta dynamic features (Tokuda et al. 2000). With one additional 1 dimensional voice/unvoiced feature it is 190 dimensions in total. This WORLD vocoder based feature uses FFT window size 2048 and has a frame length 5 ms which means features of a 5

	mismatch	weakly match	match	perfectly match
MFCCs	0	4	14	82
mel-band spectrograms	0	1	17	82

Table 3: Statistical result of how resynthesized audios match the real ones in two cases

second audio have a frame length of 1000.

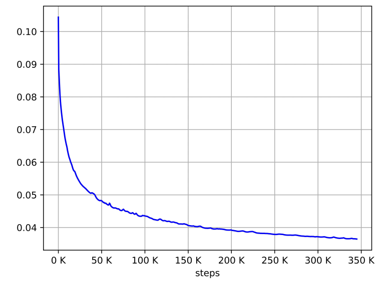
The spectrograms are obtained with FFT size 2048 and hop size 275. The dimensions of linear-scale log magnitude spectrograms and mel-band spectrograms are 1025 and 80. For a 5 second audio the frame length of features is about 400.

Vocoders The open source of WORLD vocoder can be found in github. Griffin Lim uses the linear-scale log magnitude spectrograms with sharpening factor 1.5 and 50 inversion iterations. The WaveNet vocoder consists of 2 stacks, and each stack consists of 10 layers with dilations $[1, 2, 4, 8, 16, 32, 64, 128, 256, 512]$. It uses a kernel size 3, residual channels size 128, gate channels size 256 and skip channels size 128. The target is the mixture of logistic distribution (Salimans et al. 2017) of 8-bit μ-law audios. It is trained with real linear-scale log magnitude spectrograms as local conditions.

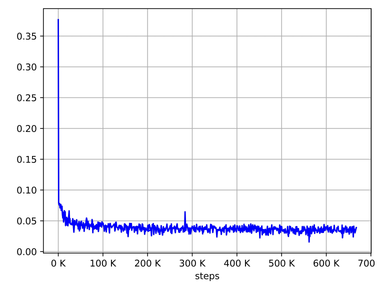
Tacotron, DCTTS and Deep Voice 3 See Appendix E for hyperparameters of the three systems. Since DCTTS and Tacotron only use spectrograms in their papers, comparison of audios by MFCCs are not performed.

FPUAS Table 4 and Table 5 are hyperparameters for training step two of FPUAS. The entry 'GatedConv (linear)' is the number of layers and hidden state size of gated convolutions that are specific to predict linear-scale log magnitude spectrograms. Tacotron, DCTTS, Deep Voice 3 take mel-band spectrograms as an intermediate feature whose length is reduced by a reduction factor to speed up inference. FPUAS does not use this trick.

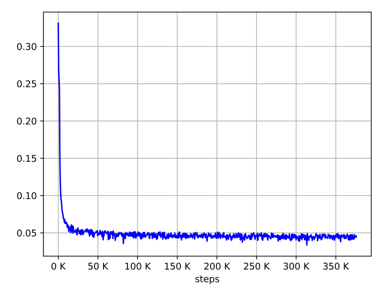
Loss curves See Figure 8 and 9 for the loss curves. All systems use L_1 loss function. For DCTTS, training of Text2Mel and SSRN are separated. Note that DCTTS, Tacotron, Deep Voice 3 are all autoregressive structures. During training, real spectrograms of previous step are feed to train spectrograms of the next step, but during inference, predicted spectrograms are feed to predict next spectrograms (van den Oord et al. 2017). FPAUS is non-autoregressive which means all spectrograms are predicted at the same time during both training and inference. It makes sense that during training FPUAS has a slightly higher loss (when predicting linear-scale log magnitude spectrograms) than the other three systems.



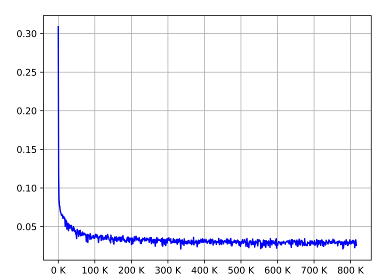
(a) mel-band spectrograms, FPUAS



(b) mel-band spectrograms, Tacotron



(c) mel-band spectrograms, DCTTS



(d) mel-band spectrograms, Deep Voice 3

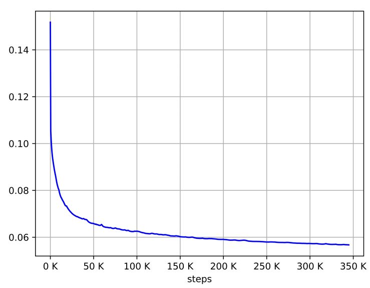
Figure 8: Loss curves of mel-band spectrograms

H_{enc}	$\hat{r}_{min}$	L	UFANS (alignment)	UFANS (decoder)	γ	σ	optimizer/lr
512	2.0	512	4/512	6/512	20.0	0.01	Adam/0.0004

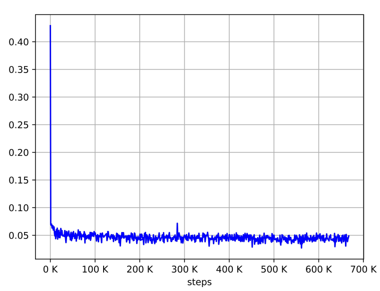
Table 4: Hyperparameters for MFCCs

H_{enc}	$\hat{r}_{min}$	L	UFANS (alignment)	UFANS (decoder)	GatedConv (linear)	γ	σ	optimizer/lr
512	1.0	512	4/512	6/512	3/512	10.0	0.01	Adam/0.0004

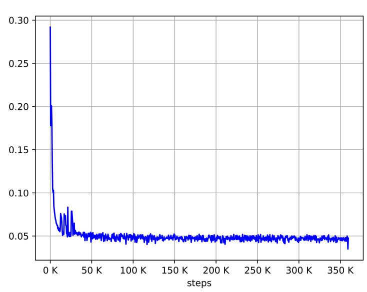
Table 5: Hyperparameters for spectrograms



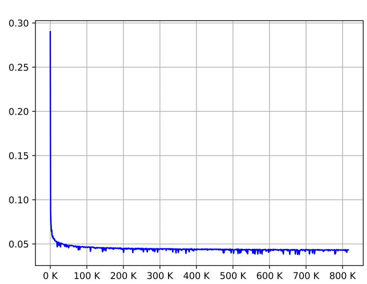
(a) linear log magnitude spectrograms, FPUAS



(b) linear log magnitude spectrograms, Tacotron



(c) linear log magnitude spectrograms, DCTTS



(d) linear log magnitude spectrograms, Deep Voice 3

Figure 9: Loss curves of linear log spectrograms

Inference speed The inference speed is evaluated as the time latency to synthesize a one-second speech, which includes data transfer from main memory to GPU global memory, GPU calculations and data transfer back to main memory. The estimation is performed on a GTX 1080Ti graphics card. See Table 6 for the estimation result. FPUAS is able to greatly take advantage of parallel computations.

Using reduction is a trick that can significantly reduce computation of inference. FPAUS does not use this trick but theoretically this trick can also reduce its computation.

MOS Harvard Sentences List 1 and List 2 are used to evaluate the MOS (mean opinion score) for all the methods. The synthesized audios are evaluated on Amazon Mechanical Turk using crowdMOS method (Protasio Ribeiro et al. 2011). The score ranges from 1 (Bad) to 5 (Excellent). See Table 7 for MOS results.(MOS of WaveNet-based audios are much lower than expected since large noise exists in these audios, we are currently training a new WaveNet model to overcome this problem)

Error mode To test the possible error modes of the four methods, 100 sentences are randomly selected from Los Angeles Times, Washington Post and some fairy tales. Tacotron, DCTTS and Deep Voice 3 all use tricks mentioned in their papers to reduce errors. FPUAS uses a trick that limits alignment width of each phoneme below a constant (10 for spectrograms) to prevent too long pronunciation. See Table 8 for the evaluation results. Repetition means repeated pronunciation of one or more phonemes. Mispronunciation means wrong pronunciation of one or more phonemes. Skip means one or more phonemes are skipped.

Tacotron and DCTTS both use only characters which makes more mispronunciation errors. Deep Voice 3 uses both characters and phonemes but it still suffers from the three error modes. FPUAS is more robust by using only phonemes but is not immune to errors. Possible reasons are that the transition from text to phonemes of training dataset is not perfect (mispronunciation) and that the reception field of decoder in training step one is still not small enough (repetition, skip). It may be better to use training dataset with phoneme-audio pairs. For deployment, using phonemes is more flexible than using characters since the mispronunciation errors are not able to be corrected when using characters.

	Tacotron	DCTTS	Deep Voice 3	FPUAS
Autoregressive	Yes	Yes	Yes	No
Reduction factor	5	4	4	No
Inference speed (ms)	6157	494.3	105.4	9.9

Table 6: Inference speed comparison

Methods	Training step (K)	Vocoder	MOS
Tacotron	650	Griffin Lim	3.51 ± 0.070
DCTTS	375/360	Griffin Lim	3.55 ± 0.107
Deep Voice 3	650	Griffin Lim	2.79 ± 0.096
FPUAS	250/350	Griffin Lim	3.65 ± 0.082
Tacotron	650	WaveNet	3.04 ± 0.103
DCTTS	375/360	WaveNet	3.43 ± 0.109
FPUAS	250/350	WaveNet	3.27 ± 0.108
FPUAS (step one)	250	Griffin Lim	1.12 ± 0.027
FPUAS	230/180	WORLD	3.81 ± 0.122

Table 7: MOS results

	Input	Repetition	Mispronunciation	Skip
Tacotron	characters	0	5	4
DCTTS	characters	0	10	1
Deep Voice 3	characters and phonemes	1	5	3
FPUAS	phonemes	1	2	1

Table 8: Error modes comparison

4 Discussion and Conclusion

In this paper, a new non-autoregressive, fully parallel (except a recurrent addition of dozens of scalars) TTS acoustic system is proposed. It can fully utilize the power of parallel computation (e.g. GPU computation) and has reached at least 10 times speed up of inference compared with most popular end-to-end TTS systems. This phoneme-based system can also produce audios of equal or better quality with fewer errors. But this system is not perfect: Errors may appear during the transition from texts to phonemes when preprocessing the training dataset (and thus we believe this system will produce even better audios with Mandarin datasets); It requires a well labeled dataset; The explicit alignment determination module needs further study and improvement.

All efforts are made to find a lightweight TTS system for deployment with less inference latency that can produce good audios with fewer possible errors. This paper described and analyzed every component of FPUAS in detail and compared it with most popular end-to-end TTS systems (inference speed, quality and error modes). Future study focuses on how to reduce errors, how to produce audios of better quality, how to extend FPUAS to multi-speaker cases and how to use both characters and phonemes in FPUAS.

References

[Bahdanau, Cho, and Bengio 2014] Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv e-prints* abs/1409.0473.

[Cho et al. 2014] Cho, K.; van Merriënboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. Doha, Qatar: Association for Computational Linguistics.

[Gehring et al. 2017] Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. 2017. Convolutional sequence to sequence learning. In *ICML*.

[Griffin et al. 1984] Griffin, D. W.; Jae; Lim, S.; and Member, S. 1984. Signal estimation from modified short-time fourier transform. *IEEE Trans. Acoustics, Speech and Sig. Proc* 236–243.

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

[Kingma and Ba 2014] Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

[Ma et al. 2018] Ma, D.; Su, Z.; Lu, Y.; Wang, W.; and Li, Z. 2018. Ufans: U-shaped fully-parallel acoustic neural structure for statistical parametric speech synthesis with 20x faster. *arXiv preprint arXiv:1811.12208*.

[MORISE, YOKOMORI, and OZAWA 2016] MORISE, M.; YOKOMORI, F.; and OZAWA, K. 2016. World: A vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems* E99.D(7):1877–1884.

[Ping et al. 2018] Ping, W.; Peng, K.; Gibiansky, A.; Arik, S. O.; Kannan, A.; Narang, S.; Raiman, J.; and Miller, J. 2018. Deep voice 3: 2000-speaker neural text-to-speech. In *International Conference on Learning Representations*.

[Protasio Ribeiro et al. 2011] Protasio Ribeiro, F.; Florencio, D.; Zhang, C.; and Seltzer, M. 2011. Crowdmos: An approach for crowdsourcing mean opinion score studies. In *ICASSP*. IEEE.

[Salimans et al. 2017] Salimans, T.; Karpathy, A.; Chen, X.; and Kingma, D. P. 2017. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR* abs/1701.05517.

[Srivastava et al. 2014] Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

[Tachibana, Uenoyama, and Aihara 2017] Tachibana, H.; Uenoyama, K.; and Aihara, S. 2017. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. *CoRR* abs/1710.08969.

[Tokuda et al. 2000] Tokuda, K.; Yoshimura, T.; Masuko, T.; Kobayashi, T.; and Kitamura, T. 2000. Speech parameter generation algorithms for hmm-based speech synthesis. In *ICASSP*.

[van den Oord et al. 2016a] van den Oord, A.; Kalchbrenner, N.; Espeholt, L.; kavukcuoglu, k.; Vinyals, O.; and Graves, A. 2016a. Conditional image generation with pixelcnn decoders. In Lee, D. D.; Sugiyama, M.; Luxburg, U. V.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc. 4790–4798.

[van den Oord et al. 2016b] van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A. W.; and Kavukcuoglu, K. 2016b. Wavenet: A generative model for raw audio. *CoRR* abs/1609.03499.

[van den Oord et al. 2017] van den Oord, A.; Li, Y.; Babuschkin, I.; Simonyan, K.; Vinyals, O.; Kavukcuoglu, K.; van den Driessche, G.; Lockhart, E.; Cobo, L. C.; Stimberg, F.; Casagrande, N.; Grewe, D.; Noury, S.; Dieleman, S.; Elsen, E.; Kalchbrenner, N.; Zen, H.; Graves, A.; King, H.; Walters, T.; Belov, D.; and Hassabis, D. 2017. Parallel wavenet: Fast high-fidelity speech synthesis. *CoRR* abs/1711.10433.

[Vaswani et al. 2017] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*.

[Wang et al. 2017] Wang, Y.; Skerry-Ryan, R. J.; Stanton, D.; Wu, Y.; Weiss, R. J.; Jaitly, N.; Yang, Z.; Xiao, Y.; Chen, Z.; Bengio, S.; Le, Q. V.; Agiomyrgiannakis, Y.; Clark, R.; and Saurous, R. A. 2017. Tacotron: A fully end-to-end text-to-speech synthesis model. *CoRR* abs/1703.10135.

[Wu, Watts, and King 2016] Wu, Z.; Watts, O.; and King, S. 2016. Merlin: An open source neural network speech synthesis system. In *9th ISCA Speech Synthesis Workshop (2016)*, 202–207.

A Alignment width and Attention Width

For two adjacent absolute alignment positions s_i and s_{i+1}, consider the two functions $g_1(x) = \sum_f cos(\frac{x-s_i}{f})$ and $g_2(x) = \sum_f cos(\frac{x-s_{i+1}}{f})$. The values of the two functions only depend on the relative position of x to s_i, s_{i+1}. From appendix B, it is know the value of function g_1 decreases when x moves away from s_i (locally, but sufficiently). So when $x \in [s_i, \frac{1}{2}(s_i + s_{i+1}))$, $g_1(x) > g_2(x)$, when $x \in (\frac{1}{2}(s_i + s_{i+1}), s_{i+1}]$, $g_1(x) < g_2(x)$. Then $x = \frac{1}{2}(s_i + s_{i+1})$ is the right attention boundary of phoneme i, simiarly the left attention boundary is $x = \frac{1}{2}(s_{i-1} + s_i)$. It is deduced that :

$$w_i = \frac{1}{2}(s_i + s_{i+1}) - \frac{1}{2}(s_{i-1} + s_i) \tag{14}$$

$$= \frac{1}{4}(r_{i-1} + r_{i+1} + 2r_i) \tag{15}$$

$$i = 0, ..., T_p - 1, r_{-1} = r_0, r_{T_p} = r_{T_P - 1} \tag{16}$$

which means attention width and alignment width can be linear transformed to each other. And

$$\sum_{k=0}^{T_p-1} r_k = \sum_{k=0}^{T_p-1} w_k = T_a \tag{17}$$

B Properties of sine and cosine positional encoding alignment attention

Besides this alignment attention, other attentions also seem to work like Gaussian functions. But experiments show Gaussian function is not suitable for this task. Here the reason is revealed.

Suppose $L = 1024$, then the alignment attention of phoneme i is of the form $g(x) = \sum_{k=0}^{1023} cos(\frac{x-s_i}{f_k})$. And also consider a Gaussian function $G(x) = exp(-(\frac{x-s_i}{40})^2)$. Since the two functions only depend on $x - s_i$, it is convenient to set s_i to 0 and set $x \in [-400, 400]$.

After normalizing :

$$\hat{g}(x) = \frac{g(x)}{\sup_{x\in[-400,400]} g(x)}, \hat{G}(x) = \frac{G(x)}{\sup_{x\in[-400,400]} G(x)} \tag{18}$$

The normalized functions are drawn in Figure 10.

This alignment attention has a much heavier tail than Gaussian function which is necessary to learn the alignment. In Figure 11, the frame j is currently attends most on phoneme m, but the right phoneme for j is n. In order for the model to learn the right alignment, frame j should receive information from n, that is $\hat{A}_{jn} << \hat{A}_{jm}$ is not allowed to happen. For Gaussian function, $\hat{A}_{jn}$ vanishes too fast as $|j - s_n|$ increases. The heavy tail of the alignment attention helps phoneme j receive information from the correct but distant phoneme.

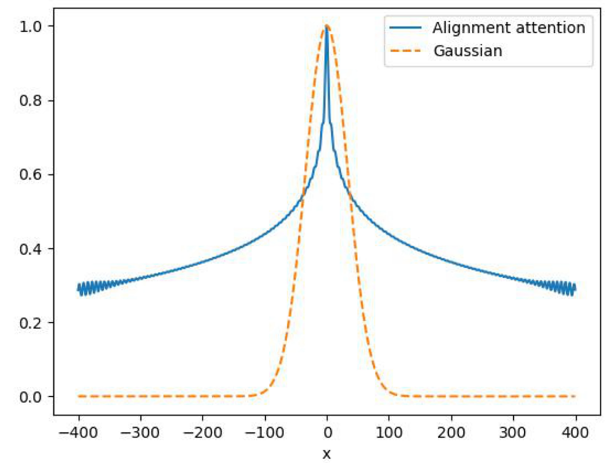


Figure 10: Function $\hat{g}(x)$ and $\hat{G}(x)$

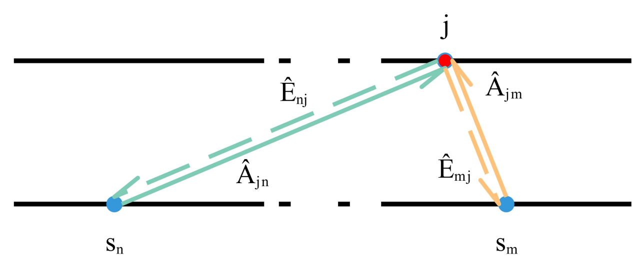


Figure 11: Frame j is currently attending most on phoneme m, but phoneme n is the correct phoneme. Gradients $\hat{E}_{nj}$ and $\hat{E}_{mj}$ are amplified by $\dot{g}(s)$ or $\dot{G}(s)$ before reaching phoneme n and phoneme m.

Now let $x = j$ fixed, make s the variable, and consider the two functions, $g(s) = \sum_{k=0}^{1023} cos(\frac{j-s}{f_k})$, $G(s) = exp(-(\frac{j-s}{40})^2)$. Suppose during training, frame j receives information from phonemes and realizes that phoneme n is more probable than m to be the correct phoneme. Then the backward information flow (gradient) from j to $g(s_n)$ (alignment attention) or $G(s_n)$ (Gaussian) is larger than the flow from j to $g(s_m)$ or $G(s_m)$, that is $|\hat{E}_{nj}| > |\hat{E}_{mj}|$. From the chain rule of gradient, the gradient that s_n receives is $\dot{g}(s_n) * \hat{E}_{nj}$ for alignment attention or $\dot{G}(s_n) * \hat{E}_{nj}$ for Gaussian function; the gradient that s_m receives is $\dot{g}(s_m) * \hat{E}_{mj}$ or $\dot{G}(s_m) * \hat{E}_{mj}$.

Consider the two functions, $|\dot{g}(s)| = |\sum_{k=0}^{1023} sin(\frac{j-s}{f_k})|$, $|\dot{G}(s)| = |exp(-(\frac{j-s}{40})^2) * \frac{j-s}{800}|$. The two functions (after normalization) are drawn in Figure 12.

It is obvious that for Gaussian function even if $|\hat{E}_{nj}| > |\hat{E}_{mj}|$, $|\dot{G}(s_n) * \hat{E}_{nj}|$ vanishes quickly as s_n moves away from j. Thus the backward flow vanishes quickly and the correct alignment is not learned; For alignment attention, though the function $|\dot{g}(s)|$ is oscillating, it is much more insensitive to the term $j - s$. So $|\dot{g}(s_n) * \hat{E}_{nj}|$ does not vanish and the relation $|\dot{g}(s_n) * \hat{E}_{nj}| > |\dot{g}(s_m) * \hat{E}_{mj}|$ may still hold.

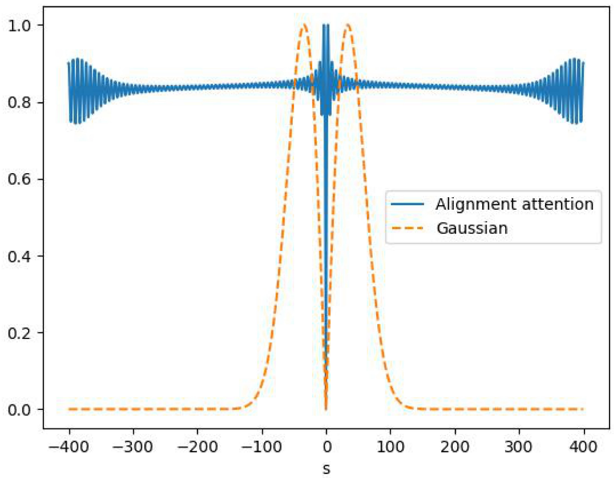


Figure 12: Normalized $|\dot{g}(s)|$ and $|\dot{G}(s)|$

C Decoder of alignment determination should be simple

In Figure 13, suppose phoneme j receives information from s_n with attention $\hat{E}_{jn}$ if using a decoder with large receptive field, and with attention $\hat{A}_{jn}$ if using a decoder with small receptive field. From appendix B, it is obvious $\hat{E}_{jn} > \hat{A}_{jn}$. If $\hat{E}_{jn}$ is large enough, then even if $|j - s_n|$ is large phoneme j can still attends most on phoneme n, which is a large disturbance to learn the alignment.

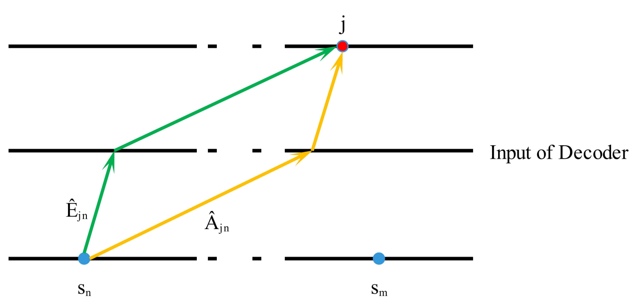


Figure 13: The green arrow is the forward information flow of a decoder with large receptive field, the yellow one is of a decoder with much smaller receptive field.

D Analysis of resynthesized waveforms of step one

Here only results of mel-band spectrograms using vocoder Griffin-Lim are shown. For MFCCs, results are similar.

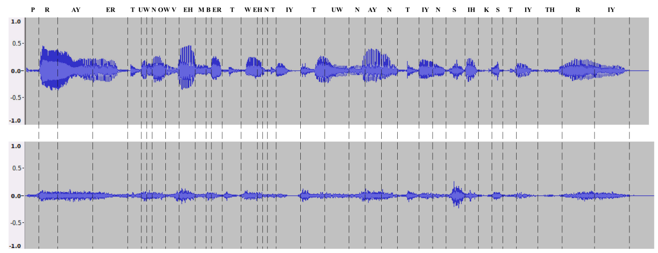
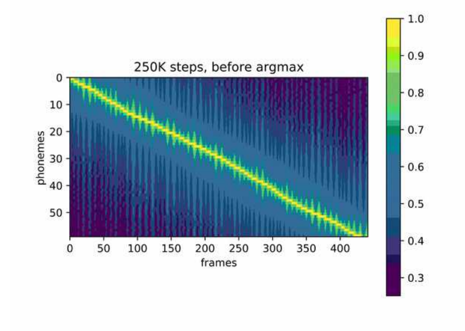
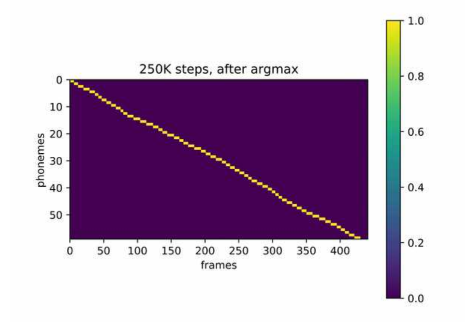


Figure 14: The upper is real audio of 'LJ048-0033', the lower is the resynthesized audio of step one.
text : prior to November twenty two nineteen sixty three
phoneme : P R AY ER T UW N OW V EH M B ER T W EH N T IY T UW N AY N T IY N S IH K S T IY TH R IY



(a) Attention plot before operation $argmax$

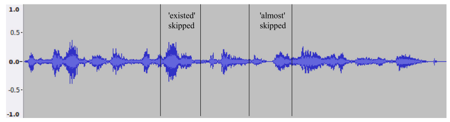


(b) Attention plot after operation $argmax$

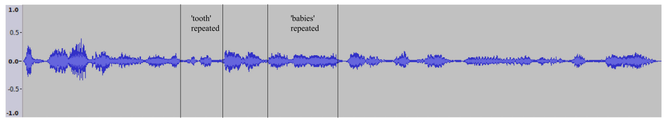
Figure 15: Attention plot of text : This is the destination for all things related to development at stack overflow.
Phoneme : DH IH S IH Z DH AH D EH S T AH N EY SH AH N F AO R AO L TH IH NG Z R IH L EY T IH D T UW D IH V EH L AH P M AH N T AE T S T AE K OW V ER F L OW .

D.1 Attention width comparison

The phoneme durations are labeled by hand and are not so precise. Figure 14 is the labeled phonemes of audio 'LJ048-0033' and resynthesized audios. Table 9 (a) is the comparison of phoneme durations of 'LJ048-0033' and computed attention width of the resynthesized audio.



(a) text : Artistic work have existed for almost as long as humankind.



(b) text : If the easter bunny and the tooth fairy had babies they would take your teeth and leave chocolate for you.

Figure 16: The upper audio suffers from skipped words and the lower suffers from repeated words

D.2 Attention plot

See Figure 15.

D.3 Attention width with Gaussian function replacing alignment attention

Table 9 (b) clearly shows that the model with Gaussian function is not able to learn alignment. The same utterance as Table 9 (a) is used.

D.4 Attention width with UFANS as decoder

Table 9 (c) is the attention width with UFANS as decoder in step one using the same utterance as in Table 9 (a). Clearly the alignment information is much worse than Table 9 (a). Figure 16 show two synthesized problematic audios from utterances randomly selected from the Internet.

E Hyperparameters of Tacotron, Deep Voice 3 and DCTTS

Table 10 shows main hyperparameters of these models.

	P	R	AY	ER	T	UW	N	OW	V	EH	M	B	ER	T	W	EH	N	T
real	5.35	7.28	15.48	13.43	4.96	3.44	3.36	5.44	4.72	7.20	4.56	1.92	7.12	5.36	3.36	3.84	3.20	4.16
resynthesized	3.55	7.97	13.28	11.37	4.88	4.00	6.19	5.27	5.46	6.39	3.56	2.08	6.13	5.69	4.34	3.03	2.50	5.92
	IY	T	UW	N	AY	N	T	IY	N	S	IH	K	S	T	IY	TH	R	IY
real	10.80	9.76	9.76	6.80	6.08	6.16	7.28	5.28	5.36	6.56	6.16	4.08	3.52	6.32	9.36	9.76	13.92	12.88
resynthesized	10.89	11.26	9.69	7.72	5.33	6.55	7.30	5.90	5.81	5.43	5.11	4.33	3.57	6.81	10.57	11.54	12.95	12.85

(a) Phoneme durations of real audio and computed attention width of resynthesized audio

	P	R	AY	ER	T	UW	N	OW	V	EH	M	B	ER	T	W	EH	N	T
real	5.35	7.28	15.48	13.43	4.96	3.44	3.36	5.44	4.72	7.20	4.56	1.92	7.12	5.36	3.36	3.84	3.20	4.16
resynthesized	6.31	6.03	5.78	6.11	6.59	6.73	6.74	6.76	6.75	6.75	6.77	6.80	6.84	6.82	6.79	6.78	6.77	6.77
	IY	T	UW	N	AY	N	T	IY	N	S	IH	K	S	T	IY	TH	R	IY
real	10.80	9.76	9.76	6.80	6.08	6.16	7.28	5.28	5.36	6.56	6.16	4.08	3.52	6.32	9.36	9.76	13.92	12.88
resynthesized	6.78	6.79	6.77	6.75	6.76	6.74	6.72	6.74	6.76	6.80	6.84	6.82	6.79	6.78	6.77	6.81	6.97	7.24

(b) Phoneme durations of real audio and computed attention width of resynthesized audio with Gaussian function replacing alignment attention

	P	R	AY	ER	T	UW	N	OW	V	EH	M	B	ER	T	W	EH	N	T
real	5.35	7.28	15.48	13.43	4.96	3.44	3.36	5.44	4.72	7.20	4.56	1.92	7.12	5.36	3.36	3.84	3.20	4.16
resynthesized	4.08	8.09	9.41	8.45	6.90	5.70	5.21	5.71	5.98	5.29	4.87	5.20	5.43	5.14	5.10	5.37	6.42	7.55
	IY	T	UW	N	AY	N	T	IY	N	S	IH	K	S	T	IY	TH	R	IY
real	10.80	9.76	9.76	6.80	6.08	6.16	7.28	5.28	5.36	6.56	6.16	4.08	3.52	6.32	9.36	9.76	13.92	12.88
resynthesized	7.14	7.38	8.96	8.20	5.39	5.54	7.31	6.34	5.56	6.42	5.84	4.76	5.62	7.61	8.26	8.17	9.02	8.97

(c) Phoneme durations of real audio and computed attention width of resynthesized audio with UFANS as decoder

Table 9: Phoneme durations of real audio and computed attention width of resynthesized audio

embedding size	256
number of banks (encoder)	16
channel size of Conv1D (encoder)	128
number/size of GRU (encoder)	1/128
number/size of GRU (attention)	1/256
number/size of GRU (decoder)	2/256
number of banks (post)	8
channel size of Conv1D (post)	128
number/size of GRU (post)	1/128
reduction factor	5

(a) Hyperparameters of Tacotron

embedding size	128
hidden state size (Text2Mel)	256
hidden state size (SSRN)	512
reduction factor	4

(b) Hyperparameters of DCTTS

embedding size	256
layers/Conv1D width/Conv1D channels (encoder)	7/5/64
layers/Conv1D width/ (decoder)	4/5
size of attention (decoder)	256
layers/Conv1D width/Conv1D channels (converter)	4/5/256
reduction factor	4

(c) Hyperparameters of Deep Voice 3

Table 10: Hyperparameters of Tacotron, DCTTS and Deep Voice 3