INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Attention Forcing for Speech Synthesis

Qingyun Dou, Joshua Efiong & Mark J. F. Gales

University of Cambridge
{qd212,je369,mjfg100}@cam.ac.uk

Abstract

Auto-regressive sequence-to-sequence models with attention mechanisms have achieved state-of-the-art performance in various tasks including speech synthesis. Training these models can be difficult. The standard approach guides a model with the reference output history during training. However during synthesis the generated output history must be used. This mismatch can impact performance. Several approaches have been proposed to handle this, normally by selectively using the generated output history. To make training stable, these approaches often require a heuristic schedule or an auxiliary classifier. This paper introduces attention forcing, which guides the model with the generated output history and reference attention. This approach reduces the training-evaluation mismatch without the need for a schedule or a classifier. Additionally, for standard training approaches, the frame rate is often reduced to prevent models from copying the output history. As attention forcing does not feed the reference output history to the model, it allows using a higher frame rate, which improves the speech quality. Finally, attention forcing allows the model to generate output sequences aligned with the references, which is important for some down-stream tasks such as training neural vocoders. Experiments show that attention forcing allows doubling the frame rate, and yields significant gain in speech quality.

Index Terms: sequence-to-sequence model, attention mechanism, training, speech synthesis

1. Introduction

Auto-regressive sequence-to-sequence (seq2seq) models with attention mechanisms are used in a variety of areas including Neural Machine Translation (NMT) [1, 2] and speech synthesis [3, 4], also known as Text-To-Speech (TTS). These models excel at connecting sequences of different length, but can be difficult to train. A standard approach is teacher forcing, which guides a model with reference output history during training. This makes the model unlikely to recover from its mistakes during inference, where the reference output is replaced by generated output. One alternative is to train the model in free running mode, where the model is guided by generated output history. This approach often struggles to converge, especially for attention-based models, which need to infer the correct output and align it with the input at the same time.

Several approaches are introduced to tackle the above problem, namely scheduled sampling [5] and professor forcing [6]. Scheduled sampling randomly decides, for each time step, whether the reference or generated output token is added to the output history. The probability of choosing the reference output token decays with a heuristic schedule. Professor forcing views the seq2seq model as a generator. During training, the generator operates in both teacher forcing mode and free running mode. In teacher forcing mode, it tries to maximize the standard likelihood. In free running mode, it tries to fool a discriminator, which is trained to tell the mode of the generator. To make training stable, these approaches require a well tuned schedule or discriminator, and recent research [7, 8] shows that their application to TTS can be challenging.

This paper introduces attention forcing, which guides the model with generated output history and reference attention. This approach makes training stable by decoupling the learning of the output and the alignment. There is no need for a schedule or a discriminator. Additionally, for standard training approaches, the frame rate is often reduced to prevent models from copying the output history. As attention forcing does not feed the reference output history to the model, it allows a higher frame rate, which can improve performance. Finally, attention forcing allows the model to generate outputs aligned with the references, which is important for some down-stream tasks such as training neural vocoders. Experiments show that attention forcing allows doubling the frame rate, and yields significant gain in expressiveness and overall speech quality.

2. Sequence-to-sequence generation

Sequence-to-sequence generation can be defined as mapping an input sequence $\boldsymbol{x}_{1:L}$ to an output sequence $\boldsymbol{y}_{1:T}$. From a probabilistic perspective, a model $\boldsymbol{\theta}$ estimates the distribution of $\boldsymbol{y}_{1:T}$ given $\boldsymbol{x}_{1:L}$, typically as a product of conditional distributions: $p(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:L};\boldsymbol{\theta}) = \prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:L};\boldsymbol{\theta})$.

Ideally, the model is trained through minimizing the KL-divergence between the true distribution $p(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:L})$ and the estimated distribution. In practice, this is approximated by minimizing the Negative Log-Likelihood (NLL) over some training data $\{\boldsymbol{y}_{1:T}^{(n)}, \boldsymbol{x}_{1:L}^{(n)}\}_1^N$, sampled from the true distribution:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\boldsymbol{x}_{1:L} \sim p(\boldsymbol{x}_{1:L})} \mathrm{KL}\big(p(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:L})||p(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:L};\boldsymbol{\theta})\big) \quad (1)$$

$$\propto -\textstyle\sum_{n=1}^{N} \log p(\boldsymbol{y}_{1:T}^{(n)}|\boldsymbol{x}_{1:L}^{(n)};\boldsymbol{\theta}) \quad (2)$$

$\mathcal{L}(\boldsymbol{\theta})$ denotes the loss. During inference, given an input $\boldsymbol{x}_{1:L}^*$, the output can be obtained through searching for the most probable sequence from $p(\boldsymbol{y}_{1:T}|\boldsymbol{x}_{1:L}^*;\boldsymbol{\theta})$. The exact search is expensive and is often approximated by greedy search for continuous output, or beam search for discrete output [5].

2.1. Attention-based seq2seq model

Attention mechanisms [9, 10] are commonly used to connect sequences of different length. This paper focuses on attention-based encoder-decoder models. For these models, the probability $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:L};\boldsymbol{\theta})$ is estimated as:

$$\begin{aligned} p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:L};\boldsymbol{\theta}) &\approx p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\alpha}_t, \boldsymbol{x}_{1:L};\boldsymbol{\theta}) \\ &\approx p(\boldsymbol{y}_t|\boldsymbol{s}_t, \boldsymbol{c}_t;\boldsymbol{\theta}_y) \end{aligned} \quad (3)$$

This research is jointly supported by China Scholarship Council & Cambridge Commonwealth, European and International Trust

http://dx.doi.org/10.21437/Interspeech.2020-2520

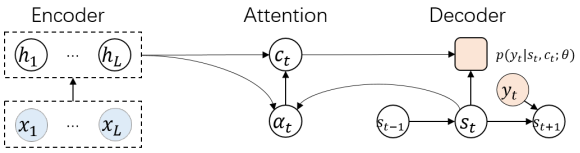

Figure 1: *Illustration of an attention-based encoder-decoder*

$\boldsymbol{\theta} = \{\boldsymbol{\theta}_y, \boldsymbol{\theta}_s, \boldsymbol{\theta}_c\}$. $\boldsymbol{\alpha}_t$ is an alignment vector (a set of attention weights). $\boldsymbol{s}_t$ is a state vector representing the output history $\boldsymbol{y}_{1:t-1}$, and $\boldsymbol{c}_t$ is a context vector summarizing $\boldsymbol{x}_{1:L}$ for the prediction of $\boldsymbol{y}_t$. Figure 1 and the following equations give a more detailed illustration of how $\boldsymbol{\alpha}_t$, $\boldsymbol{s}_t$ and $\boldsymbol{c}_t$ can be computed:

$$\boldsymbol{h}_{1:L} = f(\boldsymbol{x}_{1:L}; \boldsymbol{\theta}_h) \quad (4)$$

$$\boldsymbol{s}_t = f(\boldsymbol{s}_{t-1}, \boldsymbol{y}_{t-1}; \boldsymbol{\theta}_s) \quad (5)$$

$$\boldsymbol{\alpha}_t = f(\boldsymbol{s}_t, \boldsymbol{h}_{1:L}; \boldsymbol{\theta}_\alpha) \quad (6)$$

$$\boldsymbol{c}_t = \textstyle\sum_{l=1}^{L} \alpha_{t,l} \boldsymbol{h}_l \quad (7)$$

$$\hat{\boldsymbol{y}}_t \sim p(\boldsymbol{y}_t | \boldsymbol{s}_t, \boldsymbol{c}_t; \boldsymbol{\theta}_y) \quad (8)$$

First the encoder maps $\boldsymbol{x}_{1:L}$ to an encoding sequence $\boldsymbol{h}_{1:L}$. For each decoder time step, $\boldsymbol{s}_t$ is updated with $\boldsymbol{y}_{t-1}$. Based on $\boldsymbol{h}_{1:L}$ and $\boldsymbol{s}_t$, the attention mechanism computes $\boldsymbol{\alpha}_t$, and then $\boldsymbol{c}_t$. Finally, the decoder estimates a distribution based on $\boldsymbol{s}_t$ and $\boldsymbol{c}_t$, and optionally generates an output token $\hat{\boldsymbol{y}}_t$. Note that while illustrated with this form of attention, attention forcing is not limited to it.

2.2. Training approaches

As shown in equations 1 and 2, minimizing the KL-divergence can be approximated by minimizing the NLL. This motivates teacher forcing, where the reference output history is given to the model, and the loss can be written as:

$$\mathcal{L}_y^{(\mathrm{T})}(\boldsymbol{\theta}) = -\textstyle\sum_{n=1}^{N} \sum_{t=1}^{T} \log p(\boldsymbol{y}_t^{(n)} | \boldsymbol{y}_{1:t-1}^{(n)}, \boldsymbol{x}_{1:L}^{(n)}; \boldsymbol{\theta}) \quad (9)$$

This approach yields the correct model (zero KL-divergence) if the following assumptions hold: 1) the model is powerful enough ; 2) the model is optimized correctly; 3) there is enough training data to approximate the expectation shown in equation 1. However, these assumptions are often not true, hence the model is prone to mistakes that can accumulate across time.

In practice, the model is often assessed by some distance $\mathcal{D}$ between the reference $\boldsymbol{y}_{1:T}$ and the prediction $\hat{\boldsymbol{y}}_{1:T}$. This motivates Minimun Bayes Risk training, which minimizes the expectation of $\mathcal{D}(\boldsymbol{y}_{1:T}, \hat{\boldsymbol{y}}_{1:T})$. This approach allows directly optimizing $\mathcal{D}$ [11, 12]. $\mathcal{D}$ does not need to be differentiable, and $\boldsymbol{y}_{1:T}$ and $\hat{\boldsymbol{y}}_{1:T}$ do not need to be aligned. However, for many tasks such as TTS, there is no gold-standard distance metric, and the alignment can be essential.

Although defined for sequences, $\mathcal{D}$ is usually computed at sub-sequence level, e.g. BLEU score for NMT and L_p distance for TTS. So training the model to predict the reference output, based on erroneous output history, indirectly reduces the Bayes risk. One example is to train the model in free running mode, where the generated output history is given to the model, and the probability term in equation 9 becomes $p(\boldsymbol{y}_t^{(n)} | \hat{\boldsymbol{y}}_{1:t-1}^{(n)}, \boldsymbol{x}_{1:L}^{(n)}; \boldsymbol{\theta})$. This approach often struggles to converge, and several approaches are proposed to tackle this problem, namely scheduled sampling and professor forcing.

Scheduled sampling [5] randomly decides whether the reference or generated output is added to the history. For this approach, the probability term in equation 9 becomes $p(\boldsymbol{y}_t^{(n)} | \tilde{\boldsymbol{y}}_{1:t-1}^{(n)}, \boldsymbol{x}_{1:L}^{(n)}; \boldsymbol{\theta})$; $\tilde{\boldsymbol{y}}_t = \boldsymbol{y}_t$ with probability ϵ, and $\hat{\boldsymbol{y}}_t$ otherwise. ϵ gradually decays from 1 to 0 with a heuristic schedule. This approach improves the results in many cases [5], but sometimes lead to worse results [5, 13, 7]. One concern is the decay schedule not fitting the learning pace of the model, another is that $\tilde{\boldsymbol{y}}_{1:t-1}$ is usually an inconsistent mixture of the reference and generated output. Professor forcing [6] is an alternative. During training, the model outputs two sequences for each input sequence, respectively in teacher forcing mode and free running mode[1]. The output and/or some hidden sequences are used to train a discriminator, which estimates the probability that a group of sequences is generated in teacher forcing mode. For the generator, there are two training objectives: 1) the standard NLL loss; 2) to fool the discriminator in free running mode, and optionally in teacher forcing mode. This approach regularizes the output and/or some hidden layers, encouraging them to behave as if in teacher forcing mode, at the expense of tuning the discriminator.

In terms of application to TTS, recent research has investigated scheduled sampling and variations of professor forcing. [7] and [8] both regularize the decoder states. As there is no standard distance metric, [7] designs a discriminator and uses the hinge version of adversarial loss. [8] uses L_1 distance. [8] finds scheduled sampling beneficial, while [7] finds the opposite, showing that the schedule can be hard to tune.

To our knowledge, teacher forcing is the most standard training approach for TTS, especially when neural vocoders are used. To train neural vocoders, it is beneficial to let the seq2seq model generate sequences aligned with the references, and teacher forcing is the standard option. Section 4 will elaborate on this issue. To make teacher forcing perform better, the frame rate is often reduced, even though it introduces noise to the waveform. The reason is as follows. Speech is inherently continuous, and the output sequences have such high frame rate that adjacent frames are similar. Hence teacher forcing is prone to local optima where the model tends to copy the output history. Reducing frame rate alleviates this problem. Note that this is required by the training approach, not the model.

3. Attention forcing

For attention-based seq2seq generation, we propose attention forcing. The basic idea is to use reference attention and generated output to guide the model during training. In attention forcing mode, the model does not need to learn to simultaneously infer the output and align it with the input. As the reference alignment is known, the decoder can focus on inferring the output, and the attention mechanism can focus on generating the correct alignment.

Let $\hat{\boldsymbol{\theta}}$ denote the model trained with attention forcing, and later used for inference. In attention forcing mode, the probability $p(\boldsymbol{y}_t | \boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}})$ is estimated with the generated output $\hat{\boldsymbol{y}}_{1:t-1}$ and the reference alignment $\boldsymbol{\alpha}_t$, so equation 3 becomes:

$$\begin{aligned} p(\boldsymbol{y}_t | \boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}}) &\approx p(\boldsymbol{y}_t | \hat{\boldsymbol{y}}_{1:t-1}, \boldsymbol{\alpha}_t, \boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}}) \\ &\approx p(\boldsymbol{y}_t | \hat{\boldsymbol{s}}_t, \hat{\boldsymbol{c}}_t; \hat{\boldsymbol{\theta}}_y) \end{aligned} \quad (10)$$

$\hat{\boldsymbol{s}}_t$ and $\hat{\boldsymbol{c}}_t$ denote the state vector and context vector generated by $\hat{\boldsymbol{\theta}}$. Details of attention forcing can be illustrated by figure 2,

[1]The term "teacher forcing", as well as "attention forcing", can refer to either an operation mode, or the approach to train a model in that operation mode. An operation mode can be used not only to train a model, but also to generate from it.

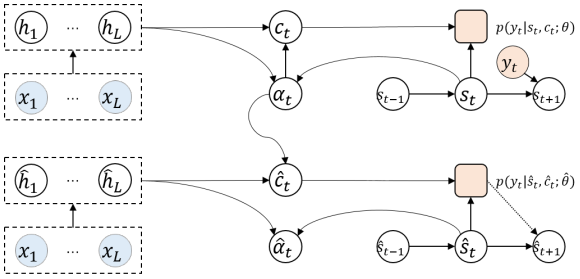

Figure 2: *Illustration of attention forcing*

as well as the following equations:

$$\boldsymbol{h}_{1:L} = f(\boldsymbol{x}_{1:L}; \boldsymbol{\theta}_h) \qquad \hat{\boldsymbol{h}}_{1:L} = f(\boldsymbol{x}_{1:L}; \hat{\boldsymbol{\theta}}_h) \tag{11}$$

$$\boldsymbol{s}_t = f(\boldsymbol{s}_{t-1}, \boldsymbol{y}_{t-1}; \boldsymbol{\theta}_s) \qquad \hat{\boldsymbol{s}}_t = f(\hat{\boldsymbol{s}}_{t-1}, \hat{\boldsymbol{y}}_{t-1}; \hat{\boldsymbol{\theta}}_s) \tag{12}$$

$$\boldsymbol{\alpha}_t = f(\boldsymbol{s}_t, \boldsymbol{h}_{1:L}; \boldsymbol{\theta}_\alpha) \qquad \hat{\boldsymbol{\alpha}}_t = f(\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{h}}_{1:L}; \hat{\boldsymbol{\theta}}_\alpha) \tag{13}$$

$$\hat{\boldsymbol{c}}_t = \textstyle\sum_{l=1}^{L} \alpha_{t,l} \hat{\boldsymbol{h}}_l \tag{14}$$

$$\hat{\boldsymbol{y}}_t \sim p(\boldsymbol{y}_t | \hat{\boldsymbol{s}}_t, \hat{\boldsymbol{c}}_t; \hat{\boldsymbol{\theta}}_y) \tag{15}$$

The right side of the equations 11 to 13, as well as equations 14 and 15, show how the attention forcing model $\hat{\boldsymbol{\theta}}$ operates. $\hat{\boldsymbol{s}}_t$ is computed with $\hat{\boldsymbol{y}}_{1:t-1}$. While an alignment $\hat{\boldsymbol{\alpha}}_t$ is generated by $\hat{\boldsymbol{\theta}}$, it is not used by the decoder, because $\hat{\boldsymbol{c}}_t$ is computed with the reference alignment $\boldsymbol{\alpha}_t$. In most cases, $\boldsymbol{\alpha}_t$ is not available. One option of obtaining it is shown by the left side of equations 11 to 13: to generate $\boldsymbol{\alpha}_t$ from a teacher forcing model $\boldsymbol{\theta}$. $\boldsymbol{\theta}$ is trained in teacher forcing mode, as shown in equation 9, and generates $\boldsymbol{\alpha}_t$, also in teacher forcing mode.

During training, there are two objectives: to infer the reference output and to imitate the reference alignment. The respective loss functions are:

$$\mathcal{L}_y^{(\mathtt{A})}(\hat{\boldsymbol{\theta}}) = -\textstyle\sum_{n=1}^{N}\sum_{t=1}^{T} \log p(\boldsymbol{y}_t^{(n)} | \hat{\boldsymbol{y}}_{<t}^{(n)}, \boldsymbol{\alpha}_t^{(n)}, \boldsymbol{x}_{1:L}^{(n)}; \hat{\boldsymbol{\theta}}) \tag{16}$$

$$\mathcal{L}_\alpha^{(\mathtt{A})}(\hat{\boldsymbol{\theta}}) = \textstyle\sum_{n=1}^{N}\sum_{t=1}^{T} \mathrm{KL}(\boldsymbol{\alpha}_t^{(n)} || \hat{\boldsymbol{\alpha}}_t^{(n)}) \tag{17}$$

As an alignment corresponds to a categorical distribution, KL-divergence is a natural difference metric. The two losses can be jointly optimized as $\mathcal{L}_{y,\alpha}^{(\mathtt{A})} = \mathcal{L}_y^{(\mathtt{A})} + \gamma \mathcal{L}_\alpha^{(\mathtt{A})}$. γ is a scaling factor that should be set according to the dynamic range of the two losses. Our default optimization option is as follows. $\boldsymbol{\theta}$ is trained in teacher forcing mode, and then fixed to generate the reference attention. $\hat{\boldsymbol{\theta}}$ is trained with the joint loss $\mathcal{L}_{y,\alpha}^{(\mathtt{A})}$. This option makes training more stable, most probably because the reference attention is the same in each epoch. An alternative is to train $\boldsymbol{\theta}$ and $\hat{\boldsymbol{\theta}}$ simultaneously to save time. Another is to tie (parts of) $\boldsymbol{\theta}$ and $\hat{\boldsymbol{\theta}}$ to save memory.

At inference stage, the attention forcing model operates in free running mode. In this case, equation 14 becomes $\hat{\boldsymbol{c}}_t = \sum_{l=1}^{L} \hat{\alpha}_{t,l} \hat{\boldsymbol{h}}_l$. The decoder is guided by $\hat{\boldsymbol{\alpha}}_t$, instead of $\boldsymbol{\alpha}_t$.

Intuitively, attention forcing, as well as scheduled sampling and professor forcing, is in the middle of teacher forcing and free running. An advantage of attention forcing is that it does not require a schedule or a discriminator, which can be difficult to tune. In terms of regularization, attention forcing is similar to professor forcing. The output layer of the attention mechanism is regularized, and the KL-divergence is a well-established difference metric. [7] and [8] also perform hidden layer regularization, and both regularize the decoder states, for which there is not a natural difference metric. [7] introduces a specific discriminator; [8] experiments with L_1 loss, and one concern is the implicit assumption that the states are in L_1 space.

The effect of regularization on attention mechanisms has been studied in previous work [14, 15, 16], where alternative approaches of obtaining reference attention are introduced. [16] and [14] require collecting extra data for reference attention, and [15] uses a statistical machine translation model to estimate them. In contrast, we propose to generate the reference attention with a teacher forcing model, which can be trained simultaneously with the attention forcing model.

Our previous work [17] shows that it can be challenging to apply attention forcing to seq2seq tasks where the attention is complicated, e.g. non-monotonic attention in NMT. Methods tacking this issue have been investigated and will be presented in the future. Another challenge of attention forcing is that when applying it to models without an attention mechanism, attention needs to be defined first. For convolutional neural networks, for example, attention maps can be defined based on the activation or gradient [18]. Some recent work on TTS [19, 20, 21, 22] uses a duration model instead of attention. In this case, one-hot alignment vectors can be defined according to the duration of input tokens.

4. Application to speech synthesis

Attention forcing has a feature that is desirable for many tasks such as TTS: when the reference alignment is given, the generated output will be aligned with the reference. The goal for TTS is to map characters $\boldsymbol{x}_{1:L}$ to waveform $\boldsymbol{w}_{1:J}$. The direct mapping is difficult, largely due to the dramatic difference in length. In the state-of-the-art TTS pipeline, a frame-level model $\boldsymbol{\theta}$ maps $\boldsymbol{x}_{1:L}$ to vocoder features $\boldsymbol{y}_{1:T}$, and a waveform-level model $\boldsymbol{\phi}$ maps $\boldsymbol{y}_{1:T}$ to $\boldsymbol{w}_{1:J}$ [3]. $\boldsymbol{\theta}$ is an attention-based seq2seq model that handles alignment [13, 3]; $\boldsymbol{\phi}$ is a neural vocoder, which does not deal with alignment [23, 24, 25, 26].

The training dataset $\{\boldsymbol{w}_{1:J}^{(n)}, \boldsymbol{x}_{1:L}^{(n)}\}_1^N$ usually contains pairs of waveform $\boldsymbol{w}_{1:J}^{(n)}$ and text $\boldsymbol{x}_{1:L}^{(n)}$. To simplify notations, the superscript $^{(n)}$ is omitted by default in the following discussion. For each $\boldsymbol{w}_{1:J}$, $\boldsymbol{y}_{1:T}$ can be extracted. The frame-level model $\boldsymbol{\theta}$ is trained with $\{\boldsymbol{x}_{1:L}, \boldsymbol{y}_{1:T}\}$. The waveform-level model $\boldsymbol{\phi}$ can be trained with $\{\boldsymbol{y}_{1:T}, \boldsymbol{w}_{1:J}\}$, or $\{\hat{\boldsymbol{y}}_{1:T}, \boldsymbol{w}_{1:J}\}$, where $\hat{\boldsymbol{y}}_{1:T}$ is generated by $\boldsymbol{\theta}$. Training with $\hat{\boldsymbol{y}}_{1:T}$ allows $\boldsymbol{\phi}$ to fix some mistakes made by $\boldsymbol{\theta}$, but this is only possible when $\hat{\boldsymbol{y}}_{1:T}$ is aligned with $\boldsymbol{w}_{1:J}$. To ensure the alignment, the standard approach is to train $\boldsymbol{\theta}$ in teacher forcing mode, and then generate from it in the same mode. This paper proposes an alternative approach: to use attention forcing instead of teacher forcing. As analyzed in section 3, training $\boldsymbol{\theta}$ with attention forcing improves its performance. Furthermore, in attention forcing mode, each output $\hat{\boldsymbol{y}}_t$ is predicted based on $\hat{\boldsymbol{y}}_{1:t-1}$ instead of $\boldsymbol{y}_{1:t-1}$, hence $\hat{\boldsymbol{y}}_{1:T}$ is more likely than in teacher forcing mode to contain errors that $\boldsymbol{\theta}$ makes at inference stage. Note that if $\boldsymbol{\theta}$ is trained with scheduled sampling or professor forcing, it is often not possible to predict, based only on generated output history, a vocoder feature sequence aligned with the reference waveform.

TTS has an inherently continuous output space. Hence during training, it is often assumed that the output tokens follow a certain type of distribution, so that minimizing the loss $\mathcal{L}_y^{(\mathtt{A})}$ shown in equation 16 can be approximated by minimizing some simple distance metric between $\boldsymbol{y}_{1:T}$ and $\hat{\boldsymbol{y}}_{1:T}$. For example, assuming that the distribution shown in equation 10 is Gaussian, minimizing $\mathcal{L}_y^{(\mathtt{A})}$ is equivalent to minimizing the average Euclidean distance: $\mathcal{L}_y^{(\mathtt{A})}(\hat{\boldsymbol{\theta}}) \propto \sum_{n=1}^{N}\sum_{t=1}^{T} ||\boldsymbol{y}_t^{(n)} - \hat{\boldsymbol{y}}_t^{(n)}||$. At inference stage, the exact search from $p(\boldsymbol{y}_{1:T} | \boldsymbol{x}_{1:L}^*; \boldsymbol{\theta})$ is approximated by greedy search.

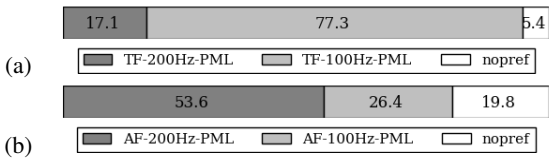

Figure 3: *Listening tests comparing 100Hz and 200Hz models; frame-level models trained with (a) teacher forcing, (b) attention forcing; waveform generated using a PML vocoder*

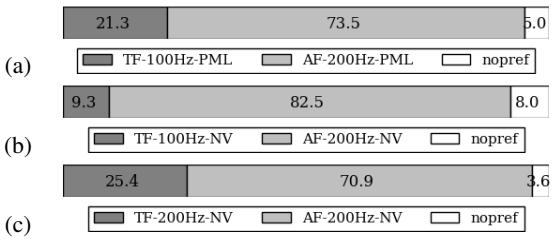

Figure 4: *Listening tests comparing teacher forcing and attention forcing; waveform generated using (a) PML vocoder (b,c) neural vocoder*

5. Experiments

The TTS experiments are conducted on LJ dataset [27], which contains 13100 utterances from a single speaker. The utterances vary in length from 1 to 10 seconds, totaling approximately 24 hours. The training-validation-test split is 13000-50-50. The speech is down-sampled to 16kHz for the neural vocoder. The default frame rate is 200Hz, which is common for TTS, and is sometimes reduced to 100Hz for comparison. The reference vocoder features are extracted with PML vocoder [28], as a study comparing various vocoders shows that PML has the best overall performance [29].

The frame-level seq2seq models and their training are the same as the Tacotron described by Table 1 in [13], except that: 1) the decoder target is PML features; 2) the attention mechanism is the hybrid (content-based + location-based) attention [10]; 3) each decoding step predicts 5 frames; 4) some models are trained with Attention Forcing (AF). The waveform-level models and their training are the same as the Hierarchical Recurrent Neural Network (HRNN) neural vocoder in [30], except that 1) the Gated Recurrent Unit (GRU) [31] dimension is 512; 2) each recurrent tier uses one layer of GRU; 3) the frequencies for tiers 0 to 3 are respectively 16, 8, 2 and 0.4kHz. The neural vocoders are trained with Teacher Forcing (TF). The seq2seq models are trained with TF or AF. The scaling factor γ is 50. During inference, all the models operate in free running mode.

For TTS, human perception is the best metric for overall speech quality. Hence the models are compared in subjective listening tests. Over 30 workers from Amazon Mechanical Turk are instructed to listen to pairs of utterances, and indicate which has better overall quality. Each comparison includes five pairs of utterances from the test set. So the test set is subjectively evaluated about three times. In addition, global variance is computed for each model, to objectively measure the expressiveness [32]; it is averaged over the test set and feature dimensions.

To see the impact of AF on frame rate, two pairs of seq2seq models are trained. The first pair, $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$, is trained with TF; $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ operate at 100Hz and 200Hz respectively. The second pair, $\{\hat{\boldsymbol{\theta}}_1, \hat{\boldsymbol{\theta}}_2\}$, has the same frame rates, but is trained with AF. PML vocoders map the features to speech. Figure 3 (a) shows the result of the listening test comparing $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$, and figure 3 (b) is the equivalent for $\{\hat{\boldsymbol{\theta}}_1, \hat{\boldsymbol{\theta}}_2\}$. Each number indicates a percentage of preference. The results show that reducing the frame rate is beneficial for TF, despite the introduction of some noise. In contrast, AF allows the use of a higher frame rate, which improves the speech quality. We strongly encourage listening to the samples, which clearly demonstrates the differences.[2]

To see the impact of AF for seq2seq models, the best TF model $\boldsymbol{\theta}_1$ is compared with the best AF model $\hat{\boldsymbol{\theta}}_2$. Figure 4 (a) shows the result: AF yields better performance. As for expressiveness, table 1 shows the global variances of all the models. It can be seen that AF yields more expressiveness than TF. Doubling the frame rate results in less expressiveness for TF, but not for AF. One likely reason is that AF prevents the model from copying the output history. Note the consistency between table 1 and figure 3, i.e. expressiveness and preference, indicating that expressiveness is important for overall quality. While speed is not the focus of this work, it can be important for TTS, and is reported here for information. For the 200Hz models, one iteration takes about 3.0s with TF, and 2.6s with AF; generating one second of feature sequence takes about 0.64s. For the 100Hz models, the time is halved thanks to shorter sequences.

Table 1: *Global variance of vocoder features generated by different models, computed over the test set, averaged over all sequences and dimensions*

	Global variance	
Training	200Hz	100Hz
Teacher forcing	0.39	0.54
Attention forcing	0.71	0.70

Next, completely neural TTS systems are built, to investigate the synergy between AF and neural vocoders. For a TF system, a neural vocoder $\boldsymbol{\phi}$ is trained with the vocoder features generated by $\boldsymbol{\theta}$ in TF mode. For an AF system, a neural vocoder $\hat{\boldsymbol{\phi}}$ is trained with the vocoder features generated by $\hat{\boldsymbol{\theta}}$ in AF mode. Figure 4 (a) and (b) show that AF works better with neural vocoders: when PML vocoders are replaced by neural vocoders, the AF system outperforms the TF system even further. Figure 4 (b) and (c) show that AF results in the best neural TTS system. An extra finding is that for TF systems, neural vocoders can fix issues caused by high frame rate. While figure 3 (a) shows that $\boldsymbol{\theta}_1$ outperforms $\boldsymbol{\theta}_2$, figure 4 (b) and (c) show that a neural vocoder can help $\boldsymbol{\theta}_2$ surpass $\boldsymbol{\theta}_1$. One likely reason is that the neural vocoder alleviates the loss of expressiveness caused by high frame rate.

6. Conclusion

This paper introduces attention forcing, which guides a seq2seq model with generated output history and reference attention. This approach can train the model to recover from its mistakes without the need for a schedule or a classifier. In addition, it allows the use of a higher frame rate, as it prevents the model from copying the output history. Finally, it allows the model to generate output sequences aligned with the references, which can be important for down-stream tasks such as training neural vocoders. The experiments show that attention forcing allows doubling the frame rate and yields significant gain in expressiveness and overall speech quality.

[2]The code and samples generated by each model are available at `http://mi.eng.cam.ac.uk/~qd212/ispc2020`

7. References

[1] G. Neubig, "Neural machine translation and sequence-to-sequence models: A tutorial," *arXiv preprint arXiv:1703.01619*, 2017.

[2] P.-Y. Huang, F. Liu, S.-R. Shiang, J. Oh, and C. Dyer, "Attention-based multimodal neural machine translation," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, 2016, pp. 639–645.

[3] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[4] Y. Wang, D. Stanton, Y. Zhang, R. Skerry-Ryan, E. Battenberg, J. Shor, Y. Xiao, F. Ren, Y. Jia, and R. A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," *arXiv preprint arXiv:1803.09017*, 2018.

[5] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[6] A. M. Lamb, A. G. A. P. Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Advances In Neural Information Processing Systems*, 2016, pp. 4601–4609.

[7] H. Guo, F. K. Soong, L. He, and L. Xie, "A new gan-based end-to-end tts training algorithm," *arXiv preprint arXiv:1904.04775*, 2019.

[8] R. Liu, B. Sisman, J. Li, F. Bao, G. Gao, and H. Li, "Teacher-student training for robust tacotron-based tts," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6274–6278.

[9] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[10] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in Neural Information Processing Systems*, 2015, pp. 577–585.

[11] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *arXiv preprint arXiv:1511.06732*, 2015.

[12] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio, "An actor-critic algorithm for sequence prediction," *arXiv preprint arXiv:1607.07086*, 2016.

[13] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.

[14] Y. Yu, J. Choi, Y. Kim, K. Yoo, S.-H. Lee, and G. Kim, "Supervising neural attention models for video captioning by human gaze data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 490–498.

[15] L. Liu, M. Utiyama, A. Finch, and E. Sumita, "Neural machine translation with supervised attention," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 3093–3102. [Online]. Available: https://www.aclweb.org/anthology/C16-1291

[16] Y. Bao, S. Chang, M. Yu, and R. Barzilay, "Deriving machine attention from human rationales," *arXiv preprint arXiv:1808.09367*, 2018.

[17] Q. Dou, Y. Lu, J. Efiong, and M. J. Gales, "Attention forcing for sequence-to-sequence model training," *arXiv preprint arXiv:1909.12289*, 2019.

[18] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *arXiv preprint arXiv:1612.03928*, 2016.

[19] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech: Fast, robust and controllable text to speech," in *Advances in Neural Information Processing Systems*, 2019, pp. 3171–3180.

[20] Y. Ren, C. Hu, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech 2: Fast and high-quality end-to-end text-to-speech," *arXiv preprint arXiv:2006.04558*, 2020.

[21] C. Yu, H. Lu, N. Hu, M. Yu, C. Weng, K. Xu, P. Liu, D. Tuo, S. Kang, G. Lei *et al.*, "Durian: Duration informed attention network for multimodal synthesis," *arXiv preprint arXiv:1909.01700*, 2019.

[22] J. Donahue, S. Dieleman, M. Bińkowski, E. Elsen, and K. Simonyan, "End-to-end adversarial text-to-speech," *arXiv preprint arXiv:2006.03575*, 2020.

[23] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[24] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "Samplernn: An unconditional end-to-end neural audio generation model," *arXiv preprint arXiv:1612.07837*, 2016.

[25] J. Lorenzo-Trueba, T. Drugman, J. Latorre, T. Merritt, B. Putrycz, and R. Barra-Chicote, "Robust universal neural vocoding," *arXiv preprint arXiv:1811.06292*, 2018.

[26] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. v. d. Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," *arXiv preprint arXiv:1802.08435*, 2018.

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

[28] G. A. Degottex, P. K. Lanchantin, and M. J. Gales, "A pulse model in log-domain for a uniform synthesizer," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2016, pp. 230–236.

[29] M. Airaksinen, L. Juvela, B. Bollepalli, J. Yamagishi, and P. Alku, "A comparison between straight, glottal, and sinusoidal vocoding in statistical parametric speech synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 9, pp. 1658–1670, 2018.

[30] Q. Dou, M. Wan, G. Degottex, Z. Ma, and M. J. Gales, "Hierarchical rnns for waveform-level speech synthesis," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 618–625.

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

[32] T. Nose and T. Kobayashi, "An intuitive style control technique in hmm-based expressive speech synthesis using subjective style intensity and multiple-regression global variance model," *Speech Communication*, vol. 55, no. 2, pp. 347–357, 2013.