BIDIRECTIONAL VARIATIONAL INFERENCE FOR NON-AUTOREGRESSIVE TEXT-TO-SPEECH

Anonymous authors
Paper under double-blind review

ABSTRACT

Although early text-to-speech (TTS) models such as Tacotron 2 have succeeded in generating human-like speech, their autoregressive (AR) architectures have a limitation that they require a lot of time to generate a mel-spectrogram consisting of hundreds of steps. In this paper, we propose a novel non-autoregressive TTS model called BVAE-TTS, which eliminates the architectural limitation and generates a mel-spectrogram in parallel. BVAE-TTS adopts bidirectional-inference variational autoencoder (BVAE) that learns hierarchical latent representations using both bottom-up and top-down paths to increase its expressiveness. To apply BVAE to TTS, we design our model to utilize text information via an attention mechanism. By using attention maps that BVAE-TTS generates, we train a duration predictor so that the model uses the predicted length of each phoneme at inference. In experiments conducted on LJSpeech dataset, we show that our model generates a mel-spectrogram 27 times faster than Tacotron 2 with similar speech quality. Furthermore, our BVAE-TTS outperforms Glow-TTS, which is one of the state-of-the-art non-autoregressive TTS models, in terms of both speech quality and inference speed while having 58% fewer parameters.

1 INTRODUCTION

End-to-end text-to-speech (TTS) systems have recently attracted much attention as neural TTS models such as Tacotron 1, 2 (Wang et al., 2017; Shen et al., 2018), DeepVoice 3 (Ping et al., 2018) and Transformer-TTS (Li et al., 2019) began to generate high-quality speech that is very similar to the human voice. Typically, those TTS systems first generate a mel-spectrogram from a text using a sequence-to-sequence (seq2seq) model (Sutskever et al., 2014), and then synthesize speech from the mel-spectrogram using a neural vocoder like WaveGlow (Prenger et al., 2019).

Early neural TTS systems have used an autoregressive (AR) architecture to generate a mel-spectrogram mainly due to its two benefits. First, the autoregressive (AR) generation eases the difficulty of modeling mel-spectrogram distribution by factorizing the distribution into the product of homogeneous conditional factors in sequential order. Second, the seq2seq based AR architecture helps the models predict the length of the target mel-spectrogram from an input text, which is a non-trivial task because there are no pre-defined rules between the lengths of text and mel-spectrogram. Although they facilitate a high-quality speech synthesis, AR TTS models have a fundamental limitation that they cannot generate a mel-spectrogram in parallel, and so the inference time increases linearly with mel-spectrogram timesteps.

To overcome the limitation of the AR TTS models, several non-AR TTS models have been proposed based on knowledge distillation or a generative flow. While existing non-AR TTS models have achieved significant improvements in the inference speed, they still suffer from several problems. For one example, distillation-based non-AR TTS models such as FastSpeech (Ren et al., 2019) use an AR TTS model as a teacher model to guide their training, resulting in a more complex training process and unwanted dependency on the performance of the teacher model. For another example, flow-based non-AR TTS models such as Glow-TTS (Kim et al., 2020) require a lot of model parameters caused by its bijective transformation property.

In this work, we present a novel non-AR TTS model called BVAE-TTS that achieves high-quality speech synthesis in parallel. Contrary to the previous non-AR TTS models, BVAE-TTS does not

require a lot of model parameters or any teacher model for training. First, our BVAE-TTS generates a mel-spectrogram in parallel by adopting a bidirectional-inference variational autoencoder (BVAE) (Sønderby et al., 2016; Kingma et al., 2016; Maaløe et al., 2019; Vahdat & Kautz, 2020) with the 1-D convolutional network. For the high-quality speech synthesis, BVAE-TTS learns the mel-spectrogram distribution jointly with hierarchical latent variables in a bidirectional manner, which is the property of BVAE that uses both bottom-up and top-down paths. In this framework, we can manage the model size by alternating the hidden dimension of each layer. During inference, BVAE-TTS utilizes text information via a duration predictor, which is essential for our non-AR TTS model to predict how many steps of a mel-spectrogram will be generated from each phoneme. To train the duration predictor without any help of a teacher model or human-labels, we employ an attention mechanism in BVAE-TTS to learn attention maps between text and mel-spectrogram, and use the mapping information as duration labels.

In experiments conducted on LJSpeech dataset, we first compare our BVAE-TTS with Tacotron 2 and Glow-TTS in respect of speech quality, inference speed, and model size. The results show that our model achieves 27 times speed improvement on average over Tacotron 2 in generating a mel-spectrogram from a text with a similar level of speech quality. Furthermore, BVAE-TTS outperforms the state-of-the-art flow-based non-AR TTS model, Glow-TTS, in both speech quality and inference time, while having 58% fewer model parameters. Additionally, we conduct an analysis on BVAE-TTS to observe how the hierarchies is contained in the latent representations. In this analysis, we confirm that the bottom part of BVAE-TTS captures the variation of mel-spectrograms that can occur from the same text. The source code and the audio samples of our model will be available.

2 Related work

Since Tacotron 1 (Wang et al., 2017), which is the first end-to-end TTS system, was proposed, there has been much progress in seq2seq based TTS models. First, improving the speech quality of neural TTS models has been one of the major development directions. To this end, Tacotron 2 (Shen et al., 2018) was developed and it achieved high-quality speech synthesis that is similar to the human voice. Also, accelerating the training and inference speed of TTS models has been another major development direction. For example, to enable parallel training, Deep Voice 3 (Ping et al., 2018) and DCTTS (Tachibana et al., 2018) use CNN-based seq2seq models, and Transformer-TTS (Li et al., 2019) uses transformer-based seq2seq model. However, those neural TTS models still have slow inference speed due to their autoregressive (AR) generations.

To overcome the slow inference speed of AR-based models, non-AR TTS models have recently been studied. On one hand, there have been knowledge distillation based non-AR TTS models such as FastSpeech (Ren et al., 2019) and ParaNet (Peng et al., 2020), which use well-trained AR teacher models to guide their training. For example, FastSpeech (Ren et al., 2019) extracts attention maps between text and mel-spectrogram from the teacher model, and use the information to train its phoneme duration predictor. Despite their fast inference, these non-AR TTS models significantly rely on external teacher models, resulting in a complex two-step training process and unwanted dependency on the performance of the teacher model. In contrast with the distillation-based models, our BVAE-TTS does not require any pre-trained model even for the training of a phoneme duration predictor.

On the other hand, there have been generative flow based non-AR TTS models such as Flow-TTS (Miao et al., 2020) and Glow-TTS (Kim et al., 2020). These non-AR TTS models remove the dependency on the teacher model by utilizing the flow-based generative framework. While their speech quality is comparable to that of AR-based TTS models, flow-based generative models usually suffer from the problem that they require a lot of model parameters since the dimension of the flow network should be same through the whole network. Contrary to flow-based TTS models, our BVAE-TTS is free from the above issue while showing competitive speech synthesis quality to that of AR-based TTS models.

In the meanwhile, several TTS systems have utilized VAE to improve the controllability of end-to-end TTS models. For example, (Hsu et al., 2018) and (Zhang et al., 2019) incorporated VAE to Tacotron 2 to learn the style or prosody of the input speech. However, previous uses of VAE have been limited to an auxiliary network in TTS based on the AR generation framework. To the

best of our knowledge, our BVAE-TTS is the first parallel TTS model that directly uses the VAE architecture to the task of TTS.

3 BACKGROUND

3.1 BIDIRECTIONAL-INFERENCE VARIATIONAL AUTOENCODER

Variational autoencoder (VAE) is a neural network generative model $p_\theta(\mathbf{x}, \mathbf{z})$ parameterized by θ, where $\mathbf{x}$ is an observed data and $\mathbf{z}$ is a latent vector. In practice, since we only have a dataset $X = \{\mathbf{x_1}, ..., \mathbf{x_N}\}$ without the knowledge about $\mathbf{z}$, θ is typically optimized by maximizing the likelihood:

$$\max_\theta \log p_\theta(\mathbf{X}) = \max_\theta \sum_{i=1}^{N} \log \int_{\mathbf{z}} p_\theta(\mathbf{x}_i, \mathbf{z}) d\mathbf{z}. \quad (1)$$

However, the integral over $\mathbf{z}$ is intractable to compute. Therefore, the VAE introduces an approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and does variational inference while maximizing the evidence lower bound (ELBO):

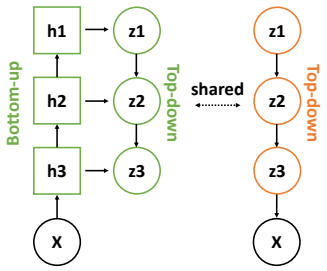

Figure 1: A Schematic diagram of bidirectional-inference variational autoencoder.

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right]. \quad (2)$$

In practice, for easy sampling and easy computation of the KL-divergence, the prior $p(\mathbf{z})$ and the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ is usually modeled as a multivariate normal distribution with a diagonal covariance matrix.

For a more expressive model, the latent vector $\mathbf{z}$ can be factorized into $\{\mathbf{z_1}, ..., \mathbf{z}_K\}$ with hierarchical dependency, where K is the number of hierarchies. Then, each of the prior and the approximate posterior is represented as $p_\theta(\mathbf{z}) = \Pi_k p_\theta(\mathbf{z}_k|\mathbf{z}_{<k})$ and $q_\phi(\mathbf{z}|\mathbf{x}) = \Pi_k q_\phi(\mathbf{z}_k|\mathbf{z}_{<k}, \mathbf{x})$, respectively. In (Sønderby et al., 2016; Kingma et al., 2016; Vahdat & Kautz, 2020), the variational inference is designed in a bidirectional way based on bottom-up path and top-down path, while letting the inference network (left) and generative network (right) share their parameters as shown in Figure 1. First, along the bottom-up path, BVAE extracts hierarchical features from $\mathbf{x}$ and stores them inside of it. Then, along the top-down path, BVAE does the variational inference and reconstructs the input data considering the stored hierarchical features together. This architecture helps the model effectively learn the hierarchies between the latent variables, and equation (2) is changed as follow:

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \sum_{k=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}_{<k}|\mathbf{x})}[D_{KL}\left[q_\phi(\mathbf{z}_k|\mathbf{x}, \mathbf{z}_{<k})||p(\mathbf{z}_k|\mathbf{z}_{<k})\right]]. \quad (3)$$

3.2 DURATION PREDICTOR IN NON-AUTOREGRESSIVE TEXT-TO-SPEECH

To achieve the non-autoregressive (non-AR) text-to-speech (TTS) model, it is essential for the model to predict the length of the target mel-spectrogram from a text. However, this is a challenging task considering that there are no pre-defined rules between the lengths of text and mel-spectrogram. Recently, several non-AR TTS models (Ren et al., 2019; Zeng et al., 2020; Miao et al., 2020; Kim et al., 2020) resolved the issue by introducing a module called duration predictor. The duration predictor is a module that predicts how many mel-spectrogram steps will be generated from each phoneme.

First, using the duration predictor, the non-AR TTS models compute durations $\hat{D} = \{\hat{d}_1, ..., \hat{d}_L\}$ corresponding to each phoneme based on phoneme representations $\mathbf{V} = \{\mathbf{v}_1, ..., \mathbf{v}_L\}$, where each $\hat{d}_i$ is a positive interger rounded from a positive real number and L is the number of phonemes. Then, $\mathbf{V}$ will be expanded to the length of the mel-spectrogram T, by expanding each $\mathbf{v}_i$ as many steps as $\hat{d}_i$. Finally, the non-AR TTS models generate a mel-spectrogram in parallel by decoding the expanded text representations. In practice, since there are no ground-truth labels for the durations, the duration predictor is trained using the duration labels obtained from well-aligned attention maps according to $d_i = \sum_{t=1}^{t=T}[\text{argmax}_s\ a_{s,t} == i]$, where $a_{s,t}$ represents an attention weight given from t-th mel-spectrogram step to s-th phoneme.

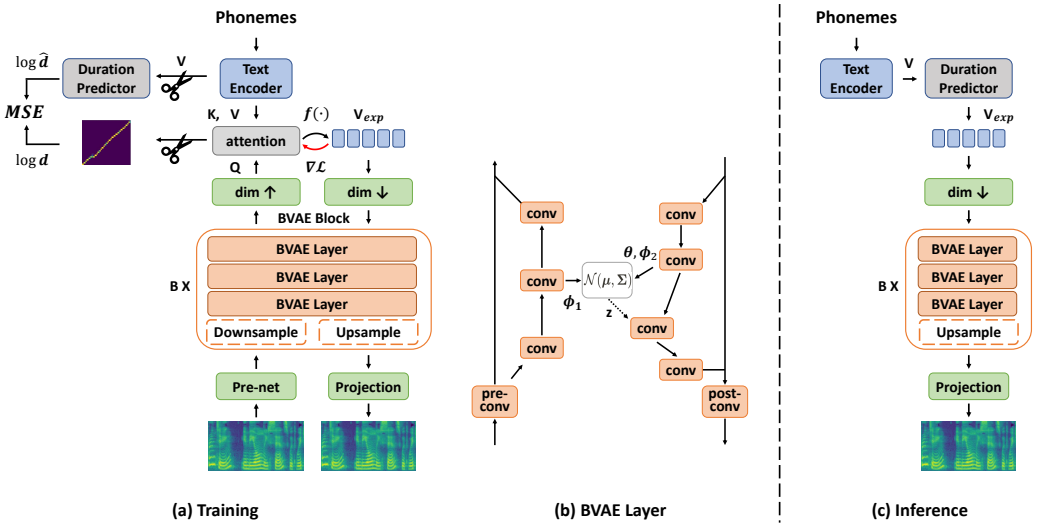

Figure 2: (a) The training procedure of BVAE-TTS. The scissors represent to detach the signal from the computational graph to block the gradient signal in backpropagation. The Downsample and Upsample layers are optionally included when they are needed. $\mathbf{V_{exp}}$ represents the expanded $\mathbf{V}$ to fit the length of the top-down path input. The $f(\cdot)$ represents the Straight-Through argmax with applying jitter, and $\nabla\mathcal{L}$ represents the gradient signal. (b) BVAE Layer. The dotted arrow represents sampling. (c) The inference procedure of BVAE-TTS that uses the top-down path only.

4 METHODOLOGY

In this section, we explain a novel non-autoregressive (non-AR) TTS model, BVAE-TTS, which is based on the bidirectional-inference variational autoencoder (BVAE). Firstly, we show how we extend the BVAE architecture to text-to-speech (TTS) in Section 4.1. Unlike the previous BVAE models (Sønderby et al., 2016; Kingma et al., 2016; Vahdat & Kautz, 2020) trained to generate natural images, our model learns to generate a mel-spectrogram that is not only natural but also corresponding to input text. Then, we describe the architecture of BVAE-TTS with several techniques in Section 4.2 and 4.3.

4.1 USING BVAE FOR TEXT-TO-SPEECH

To extend the BVAE architecture to TTS, we add an dot-product attention network (Bahdanau et al., 2015) on top of the BVAE. Via the attention network, text representations obtained from a text encoder are input into the top-down path of BVAE, where query, $\mathbf{Q}$, is obtained from the bottom-up path of BVAE, and key, $\mathbf{K}$, and value, $\mathbf{V}$, are obtained from the text encoder. Then, the BVAE-TTS does both the variational inference and mel-spectrogram reconstruction along the top-down path using the text representations with the following objectives:

$$\mathcal{L}_{recon} = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}\left[\log p_\theta(\mathbf{x}|\mathbf{z},\mathbf{y})\right], \tag{4}$$

$$\mathcal{L}_{KL} = \sum_{k=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}_{<k}|\mathbf{x},\mathbf{y})}\left[D_{KL}\left[q_\phi(\mathbf{z}_k|\mathbf{x},\mathbf{z}_{<k},\mathbf{y})||p(\mathbf{z}_k|\mathbf{z}_{<k},\mathbf{y})\right]\right], \tag{5}$$

where $\mathbf{y}$ represents text, and mean absolute error (MAE) loss is used for $\mathcal{L}_{recon}$.

In addition to that, a duration predictor is trained to predict durations corresponding to each phoneme in the logarithmic domain using mean square error (MSE) loss, $\mathcal{L}_{dur} = \mathbb{E}[(\log d_i - \log \hat{d}_i)^2]$. The duration predictor takes $\mathbf{V}$ obtained from the text encoder as input, but it does not affect the BVAE training by detaching this $\mathbf{V}$ from the computational graph.

4.2 BVAE-TTS ARCHITECTURE

In this section, we describe the architecture of BVAE-TTS that hierarchically learns the latent representations based on BVAE blocks consisting of BVAE layers.

BVAE block: As shown in Figure 2, the main part of BVAE-TTS is the stacked BVAE blocks, each consisting of 3 BVAE layers. To guide the multi-scale fine-to-coarse latent features to be contained in the latent hierarchies, the time dimension of input mel-spectogram is downsampled using bilinear downsampling operations (Zhang, 2019) in even-numbered BVAE blocks along the bottom-up path. Here, the numbering of BVAE blocks starts with one, and increases from bottom to top. On the contrary, its time dimension is upsampled again by repeating the signals in the same BVAE blocks along the top-down path. In odd-numbered BVAE-blocks, the channel dimension is decreased along the bottom-up path and is increased along the top-down path. It is done at the pre- or post-convolutional network of the first BVAE layer in each BVAE block shown in Figure 2b.

BVAE layer: The main element of the BVAE block is the BVAE layer. As shown in Figure 2b, at each bottom-up and top-down path, the parameters of the prior or the approximate posterior distribution $\theta = \{\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\},\ \phi_1 = \{\Delta\boldsymbol{\mu}_{l_1}, \Delta\boldsymbol{\Sigma}_{l_1}\},\ \phi_2 = \{\Delta\boldsymbol{\mu}_{l_2}, \Delta\boldsymbol{\Sigma}_{l_2}\}$ are obtained. Then, the distributions $p_\theta(\mathbf{z}_l|\mathbf{z}_{<l}, \mathbf{y})$ and $q_\phi(\mathbf{z}_l|\mathbf{z}_{<l}, \mathbf{x}, \mathbf{y})$ are defined as follow:

$$p_\theta(\mathbf{z}_l|\mathbf{z}_{<l}, \mathbf{y}) := \mathcal{N}(\boldsymbol{\mu}_l,\ \boldsymbol{\Sigma}_l), \tag{6}$$

$$q_\phi(\mathbf{z}_l|\mathbf{z}_{<l}, \mathbf{x}, \mathbf{y}) := \mathcal{N}(\boldsymbol{\mu}_l + \Delta\boldsymbol{\mu}_{l_1} + \Delta\boldsymbol{\mu}_{l_2},\ \boldsymbol{\Sigma}_l \cdot \Delta\boldsymbol{\Sigma}_{l_1} \cdot \Delta\boldsymbol{\Sigma}_{l_2}), \tag{7}$$

where the diagonal covariance matrices $\boldsymbol{\Sigma}$ are obtained from a softplus layer to guarantee that they are positive. This parameterization follows (Vahdat & Kautz, 2020), where the approximate posterior $q_\phi(\mathbf{z}_l|\mathbf{z}_{<l}, \mathbf{x}, \mathbf{y})$ is relative to the prior $p_\theta(\mathbf{z}_l|\mathbf{z}_{<l}, \mathbf{y})$. With this parameterization, when the prior moves, the approximate posterior moves accordingly while making the BVAE training easier and more stable. During training, the latent representation $\mathbf{z}$ is sampled from $q_\phi(\mathbf{z}_l|\mathbf{z}_{<l}, \mathbf{x}, \mathbf{y})$, and sampled from $p_\theta(\mathbf{z}_l|\mathbf{z}_{<l}, \mathbf{y})$ at inference. Also, following (Vahdat & Kautz, 2020), we apply the spectral normalization (Miyato et al., 2018) to the convolutional networks located on the residual path to stabilize the training.

Other details on the BVAE-TTS architecture such as text encoder or duration predictor are in Appendix A.1.

4.3 BRIDGE THE GAP BETWEEN TRAINING AND INFERENCE

When BVAE-TTS reconstructs a mel-spectrogram during training, text representations are expanded via the attention network, while they are expanded via duration predictor at inference. Therefore, to bridge the gap between the attention-based mel-spectrogram generation and the duration-based mel-spectrogram generation, we use the following techniques in this work.

Straight-Through argmax: In duration-based generation, the predicted duration of each phoneme is rounded. It means that there is a corresponding phoneme for every time step of a mel-spectrogram. Therefore, during training, we use a trick called Straight-Through (ST) argmax, where the text representations are passed to the top-down path via $\arg\max$ operation instead of the weighted sum in the attention mechanism. However, in the backward pass, the parameter update is conducted as if the signal obtained from the $\arg\max$ was a result of the weighted sum.

Jitter: To make the model more robust to the errors of the duration predictor, we apply jitter to the text representations, where the text representations obtained from the $\arg\max$ operation are replaced with neighboring text representations with each probability of 25% during training. We also experimentally observe that applying jitter makes the learning of the attention more stable, so that the attention is not be defused throughout the training.

Positional encoding biasing & Guided attention: In order to reduce the gap between the attention-based generation and the duration-based generation, it is important for the learned attention maps to have a diagonal shape. Therefore, we use two additional techniques to directly help BVAE-TTS learn the diagonal attention maps. First, we add positional encoding vectors with different angular speeds to query and key (Ping et al., 2018) as an inductive bias. Second, we use an additional guided attention loss $\mathcal{L}_{guide}$ that gives penalties for attention weights deviating from the diagonal following (Tachibana et al., 2018).

For more details on the techniques in this section, see Appendix A.2.

With the above techniques, BVAE-TTS is trained with the following objective:

$$\mathcal{L}_{total} = \mathcal{L}_{recon} + \alpha * \mathcal{L}_{KL} + \mathcal{L}_{dur} + \mathcal{L}_{guide}, \tag{8}$$

where the α is a warm-up constant that is linearly increasing from 0 to 1 over the first 20% of training. This technique is proposed in (Sønderby et al., 2016) to weaken the variational regularization in the early stages of training.

5 EXPERIMENTS

In this section, we describe the experimental setup and the results obtained from the quantitative and qualitative experiments to evaluate our BVAE-TTS. For comparison, we use two state-of-the-art TTS models: Tacotron 2[1] for an AR TTS model and Glow-TTS[2] for a non-AR TTS model. Here, we use pre-trained weights of the models that are publicly available.

5.1 EXPERIMENTAL SETUP

In the experiments, we mainly use the LJSpeech dataset (Ito & Johnson, 2017) consisting of 12,500 / 100 / 500 samples for training / validation / test splits, respectively. For speech data, we convert raw waveforms into log-mel-spectrograms with 1024 window length and 256 hop length, and use them as the target sequence of our BVAE-TTS model. For text data, we convert raw texts into phoneme sequences using grapheme-to-phoneme library (Park, 2019), and use them as the input sequence of BVAE-TTS.

We train the BVAE-TTS consisting of 4 BVAE blocks for 300K iterations with batch size of 128. For an optimizer, we use the Adamax Optimizer (Kingma & Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$ using the learning rate scheduling used in (Vaswani et al., 2017), where initial learning rate of 1e-3 and warm-up step of 4,000 are used. Training of BVAE-TTS takes about 48 hours on Intel(R) Xeon(R) Gold 5120 CPU (2.2GHz) and NVIDIA V100 GPU on the Pytorch 1.16.0 library with Python 3.6.10 over the Ubuntu 16.04 LTS. For more details on the hyperparameters, see Appendix A.3.

5.2 EXPERIMENTAL RESULTS

In this section, we compare BVAE-TTS with Tacotron 2 and Glow-TTS in terms of speech quality, inference time, and model size. For the quality evaluation, we use a pre-trained WaveGlow[3] as a vocoder that converts mel-spectrograms to waveforms. When we sample latent representations in Glow-TTS and BVAE-TTS, we use the temperature of 0.333 for the models for better speech quality. (Kingma & Dhariwal, 2018).

Table 1: Experimental results. The MOS-ID and MOS-OOD are written with 95% confidence intervals. The parenthesis represents the number of parameters of BVAE-TTS used at inference.

Method	MOS-ID	MOS-OOD	Inference Time (ms)	# of Parameters
GT Audio	4.68 ± 0.06	-	-	-
GT Mel-spectrogram	4.41 ± 0.07	-	-	-
Tacotron 2	4.35 ± 0.07	4.16 ± 0.07	658.5	28.2M
Glow-TTS	3.96 ± 0.08	3.89 ± 0.10	43.07	28.6M
BVAE-TTS	4.14 ± 0.07	4.21 ± 0.07	24.20	16.0M (12.0M)

Speech quality: In this experiment, we measure the Mean-Opinion-Score (MOS) for audio samples generated by each TTS model using fifty sentences randomly sampled from the in-domain LJSpeech test set (MOS-ID). In addition, we measure another MOS on fifty sentences randomly sampled from

[1] https://github.com/NVIDIA/tacotron2
[2] https://github.com/jaywalnut310/glow-tts
[3] http://github.com/NVIDIA/waveglow

the test-clean set of LibriTTS (Zen et al., 2019) to see the generalization ability on the out-of-domain text (MOS-OOD). Via Amazon Mechanical Turk (AMT), we assign five testers living in the United States to each audio sample, and ask them to listen to the audio sample and give a score between 1 to 5 in 9-scale based on its naturalness.

The MOS results shown in Table 1 demonstrate the superiority of our BVAE-TTS, where it outperforms the state-of-the-art non-AR TTS model, Glow-TTS, on both MOS-ID and MOS-OOD. Although BVAE-TTS does not surpass Tacotron 2 in MOS-ID, our model achieves better results on MOS-OOD, showing that the proposed model is generalized better than the others. For better understanding of the speech quality generated by the models, we strongly encourage the readers to listen to the audio samples in the supplementary material.

Inference time: We measure inference times taken to generate a mel-spectrogram from a text on the 500 sentences of LJSpeech test set. The average inference time of each TTS model is shown in Table 1. As can be seen in the table, our BVAE-TTS shows 27.2 times faster inference speed on average compared with Tacotron 2, and it is also 1.78 times faster than Glow-TTS. Due to the sequential generation property of the AR TTS model, the gap between the inference speed of BVAE-TTS and Tacotron 2 increases with the longer length of the input text. See Appendix B for more details.

Model size: As can be seen in the last column of Table 1, BVAE-TTS has the smallest number of parameters of 16.0M while maintaining high-quality speech synthesis. Furthermore, BVAE-TTS gets smaller (to 12.0M) at inference because the layers belonging to the bottom-up path are not used to generate a mel-spectrogram, and this is 58% fewer parameters compared to Glow-TTS. This shows that the training principle of BVAE-TTS that hierarchically learns the latent features with downsamplings enables our model to have small parameters. It is contrary to the flow-based TTS models such as Flow-TTS Miao et al. (2020) and Glow-TTS Kim et al. (2020), where the input dimension should be the same through the whole network.

5.3 Model analysis

As our BVAE-TTS is the first VAE-based parallel TTS model, we conduct several analyses on it. First, we analyze BVAE-TTS to see how the hierarchies is contained in the latent representations. Then, we verify the effectiveness of the techniques used in BVAE-TTS through ablation studies.

5.3.1 Analysis on hierarchy

In this experiment, we conduct an analysis on the hierarchical latent representation learning of BVAE-TTS. To see how the features of mel-spectrogram are contained in the hierarchies, we observe the variations of the mel-spectrograms sampled from the same text, while using different temperatures for different BVAE blocks. Specifically, we set a target BVAE block among the four BVAE blocks, and increase the variance of the target BVAE block by using the temperature of 1.0 or 2.0 or 5.0. On the contrary, we lower the variance of the BVAE blocks that are not target using a temperature of 0.333. Then, we sample 100 different mel-spectrograms from the same text input each, by varying the target BVAE block and its temperature.

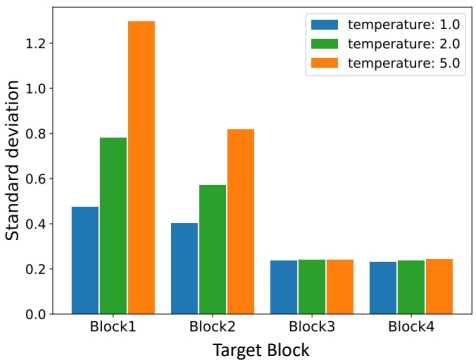

Figure 3: Averages of pixel-by-pixel standard deviations measured on randomly sampled 100 mel-spectrograms.

Figure 3 shows the averages of pixel-by-pixel standard deviations measured on the randomly sampled 100 mel-spectrograms from the same text. The block numbers in the figure are given from 1 to 4 starting from the BVAE block at the bottom. In this experiment, we observe that the variance of the speech is mostly contained in latent representations of BVAE block 1, 2, which are close to the mel-spectrogram. However, there is not much variance in the generated mel-spectrograms when we increase the variance of BVAE block 3 or 4, which are close to the text representations. Therefore,

we can conclude that the global content is mostly contained in the expanded text representations obtained by the text encoder and the duration predictor, and the BVAE block 3,4 focus on building the content rather than its style. Note that while Figure 3 shows standard deviations measured using one examplar sentence, “One, Two, Three.”, the tendency is consistent regardless of the input sentence. Mel-spectrogram samples obtained in this experiment are in Appendix C.

5.3.2 Ablation study

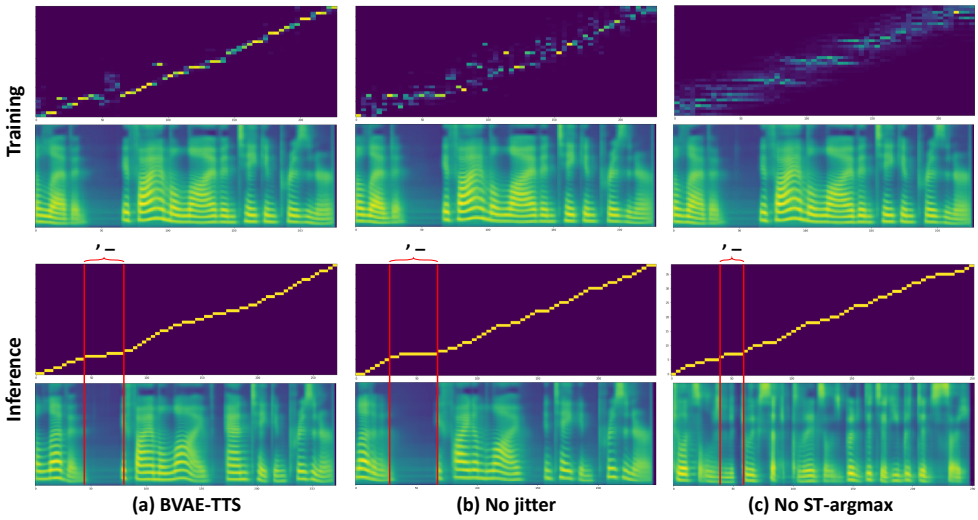

Figure 4: Examples of ablations. Each row represents (1) learned attention map, (2) reconstructed mel-spectrogram, (3) diagonally stacked predicted durations, (4) generated mel-spectrogram from a text. (1), (2) are obtained during training, and (3),(4) are obtained at inference. The red lines represent the section of the durations corresponding to a ‘comma’ and a ‘whitespace’ in text.

We conduct ablation studies to see the effects of the jitter and Straight-Through (ST) argmax in BVAE-TTS, and the results are shown in Figure 4. The most noticeable difference appears in the attention maps that they learn. As shown in the first row of Figure 4 (a) and (b), applying jitter shows the effectiveness for BVAE-TTS to learn well-aligned attention maps. It results in using more accurate duration labels to train the duration predictor, which leads to more natural speech. Actually, we observe that BVAE-TTS without applying jitter still generates a clear speech even though it is a little unnatural, where it obtains a 3.68 MOS. As shown in the bottom mel-spectrogram of Figure 4c, the BVAE-TTS without ST-argmax technique just generates a stuttering noise.

As shown in Figure 4a, although the BVAE-TTS also does not learn the perfect attention map, BVAE-TTS successfully generates a mel-spectrogram at inference. Since the text is forced to be used monotonically in the duration-based generation, it makes the model more robust to the attention errors while making fewer pronouncing mistakes. In addition, when using the duration predictor, it is possible to locally control the speed of speech by adjusting the predicted durations. The experiment on the speed control is included in Appendix D.

6 Conclusion

In this work, we propose BVAE-TTS, which is the first VAE-based non-AR TTS model that generates a mel-spectrogram from a text in parallel. To use the BVAE architecture in text-to-speech, we combine BVAE with the attention mechanism to utilize a text, and to extract duration labels for the training of the duration predictor. In experiments, BVAE-TTS achieves to generate speech 27x faster than Tacotron 2 with similar speech quality, and also outperforms Glow-TTS in terms of both speech quality and inference time with 58% fewer parameters.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Wei-Ning Hsu, Yu Zhang, Ron J Weiss, Heiga Zen, Yonghui Wu, Yuxuan Wang, Yuan Cao, Ye Jia, Zhifeng Chen, Jonathan Shen, et al. Hierarchical generative modeling for controllable speech synthesis. In *International Conference on Learning Representations*, 2018.

Keith Ito and Linda Johnson. The lj speech dataset. https://keithito.com/LJ-Speech-Dataset/, 2017.

Jaehyeon Kim, Sungwon Kim, Jungil Kong, and Sungroh Yoon. Glow-tts: A generative flow for text-to-speech via monotonic alignment search. *arXiv preprint arXiv:2005.11129*, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pp. 10215–10224, 2018.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pp. 4743–4751, 2016.

Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. Neural speech synthesis with transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6706–6713, 2019.

Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. Biva: A very deep hierarchy of latent variables for generative modeling. In *Advances in neural information processing systems*, pp. 6551–6562, 2019.

Chenfeng Miao, Shuang Liang, Minchuan Chen, Jun Ma, Shaojun Wang, and Jing Xiao. 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)*, pp. 7209–7213. IEEE, 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1QRgziT-.

Jongseok Park, Kyubyong & Kim. g2pe. https://github.com/Kyubyong/g2p, 2019.

Kainan Peng, Wei Ping, Zhao Song, and Kexin Zhao. Non-autoregressive neural text-to-speech. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 10192–10204. PMLR, 2020.

Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. Deep voice 3: 2000-speaker neural text-to-speech. *Proc. ICLR*, pp. 214–217, 2018.

Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3617–3621. IEEE, 2019.

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. In *Advances in Neural Information Processing Systems*, pp. 3171–3180, 2019.

Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj 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)*, pp. 4779–4783. IEEE, 2018.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pp. 3738–3746, 2016.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Hideyuki Tachibana, Katsuya Uenoyama, and Shunsuke Aihara. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4784–4788. IEEE, 2018.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Yuxuan Wang, R.J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis. In *Proc. Interspeech 2017*, pp. 4006–4010, 2017. doi: 10.21437/Interspeech.2017-1452. URL `http://dx.doi.org/10.21437/Interspeech.2017-1452`.

Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J. Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech. In *Proc. Interspeech 2019*, pp. 1526–1530, 2019. doi: 10.21437/Interspeech.2019-2441. URL `http://dx.doi.org/10.21437/Interspeech.2019-2441`.

Zhen Zeng, Jianzong Wang, Ning Cheng, Tian Xia, and Jing Xiao. Aligntts: Efficient feed-forward text-to-speech system without explicit alignment. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6714–6718. IEEE, 2020.

Richard Zhang. Making convolutional networks shift-invariant again. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7324–7334. PMLR, 2019. URL `http://proceedings.mlr.press/v97/zhang19a.html`.

Ya-Jie Zhang, Shifeng Pan, Lei He, and Zhen-Hua Ling. Learning latent representations for style control and transfer in end-to-end speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6945–6949. IEEE, 2019.

A DETAILS ON BVAE-TTS

A.1 ARCHITECTURE DETAILS

Text encoder: To obtain key, $\mathbf{K}$, and value, $\mathbf{V}$, for attention mechanism, we use a text encoder used in DeepVoice 3 (Ping et al., 2018), which consists of a series of 1-D convolutional network with a gated linear unit and a residual connection; $y = \sqrt{0.5}(x + \mathrm{conv}_1(x) * \sigma(\mathrm{conv}_2(x)))$. The last hidden representations of the text encoder are used as $\mathbf{K}$ and $\mathbf{V}$ is computed according to $\mathbf{V} = \sqrt{0.5}(\mathbf{K} + \mathbf{E})$, where $\mathbf{E}$ represents phoneme embedding vectors.

Duration predictor: We use a duration predictor used in FastSpeech (Ren et al., 2019), which consists of two Conv1d-ELU-LayerNorm-Dropout blocks, followed by a linear projection layer. Then, we take exponential of the output values and add ones to them, in order to guarantee that the predicted durations are larger than one.

Pre-net & Projection layer & Dimension matching layers: To adjust dimensions properly, there are four additional layers; Pre-net, Projection layer, and two additional dimension matching layers. Pre-net consists of two Conv1d-Dropout-ELU blocks and Projection layer consists of two Conv1d-ELU-LayerNorm-Dropout blocks followed by a Conv1d layer with sigmoid activation. For the dimension matching layers, we use a 1-D convolutional network with kernel width 5 to increase the dimension, and use a linear layer to decrease the dimension.

BVAE block & layer: To increase the receptive field of the model, the pre- & post- convolutional networks use dilated convolution with the dilation factors 1, 2, 4 for each BVAE layer in each BVAE block. Every convolutional network except for the networks that output the distribution parameters are followed by an ELU layer. When the residual path is merged to the original path, we combine the signals according to $y = \sqrt{0.5}(x + f(x))$ to stabilize the training.

A.2 TECHNIQUES

Straight-Through argmax: Let $\mathbf{A} = \{a_{i,t}\}$ be the attention matrix obtained from attention mechanism. Then, the original attention mechanism expands the value $\mathbf{V}$ to $\mathbf{V_{exp}}$ according to $\mathbf{V_{exp}} = \mathbf{A}^\intercal \cdot \mathbf{V}$. However, with the ST-argmax, the attention matrix is firstly converted to $\mathbf{A_{onehot}} = \{a'_{i,t}\}$, where $a'_{i,t} = 1$ if $a_{i,t} == \max_i(a_{i,t})$, else 0. Then $\mathbf{V_{exp}}$ is obtained as follow:

$$\mathbf{V_{exp}} = (\mathbf{A} + (\mathbf{A_{onehot}} - \mathbf{A}).\text{detach}())^\intercal \cdot \mathbf{V}, \tag{9}$$

where by using the detach trick, the gradient flows to $\mathbf{A}$ instead of $\mathbf{A_{onehot}}$ during backpropagation.

Jitter: To apply jitter to the hidden states, we apply jitter to $\mathbf{A_{onehot}}$ obtained from ST-argmax as shown in Figure 5. By using a function '**random.choices**' that returns a element chosen from a list according to probabilities $\mathbf{p}$, applying jitter up to single step to $\mathbf{A_{onehot}}$ can be implemented according to $a'_{i,t} = \textbf{random.choices}([a'_{i-1,t},\ a'_{i,t},\ a'_{i+1,t}],\ \mathbf{p} = [0.25, 0.5, 0.25])$.

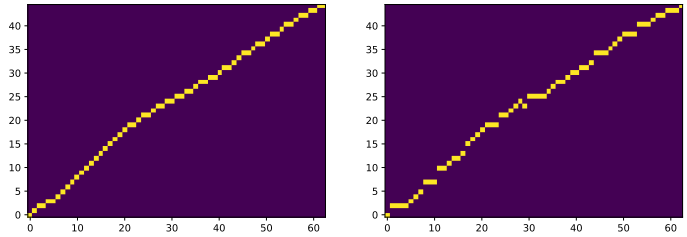

Figure 5: An example of applying jitter to ideal diagonal attention matrix $\mathbf{A_{onehot}}$

Positional encoding: We add positional encodings with different angular speed w_s to query $\mathbf{Q}$ and key $\mathbf{K}$ to introduce inductive bias of diagonal attention:

$$PE_{(pos,2i)} = sin(\frac{w_s * pos}{10000^{2i/d}}), \tag{10}$$

$$PE_{(pos,2i+1)} = cos(\frac{w_s * pos}{10000^{2i/d}}), \tag{11}$$

where pos is the position, i is the dimension, and d is the dimension of $\mathbf{Q}$ and $\mathbf{K}$.

For $\mathbf{Q}$, $w_s = 1$, and for $\mathbf{K}$, $w_s = T/L$, where T and L are the number of time steps of $\mathbf{Q}$ and $\mathbf{K}$. Then, the attention matrix $\mathbf{A}$ is calcualted according to $\mathbf{A} = (K + PE_{key}) \cdot (Q + PE_{query})^{\intercal}$.

Guided attention: To directly lead the attention map to be formed in diagonal, we additionally introduce 'guided attention loss' as follow:

$$\mathcal{L}_{guide} = \mathbb{E}[a_{i,t} * (1 - \exp(-(i/L - t/T)^2)/2g^2)],$$

where g is set to 0.2.

A.3 Hyper parameters

Table 2: Hyperparameters of BVAE-TTS.

Hyperparameter	**BVAE-TTS**
Phoneme Embedding Dimension	256
Text Encoder Layers	7
Text Encoder Hidden Dimension	256
Text Encoder Conv1D Kernel Width	5
Text Encoder Conv1D Filter Size	256
Text Encoder Dropout	0.1
Pre-net Layers	2
Pre-net Dropout	0.5
Pre-net Hidden Dimension	256
Downsampling Conv 1D kernel	[0.25, 0.5, 0.25]
Projection Layers	3
Projection Dropout	0.5
Projection Conv1D Kernel Width	5
Projection Conv1D Filter Size	256
Duration Predictor Conv1D Kernel Width	3
Duration Predictor Conv1D Filter Size	256
Duration Predictor Dropout	0.1
BVAE Blocks (**B**)	4
BVAE Layers per block	3
BVAE Conv1D Kernel Width	5
BVAE Blocks Hidden Dimensions	128, 128, 64, 64
Total Number of Parameters	**16.0M**

B PARALLEL SYNTHESIS

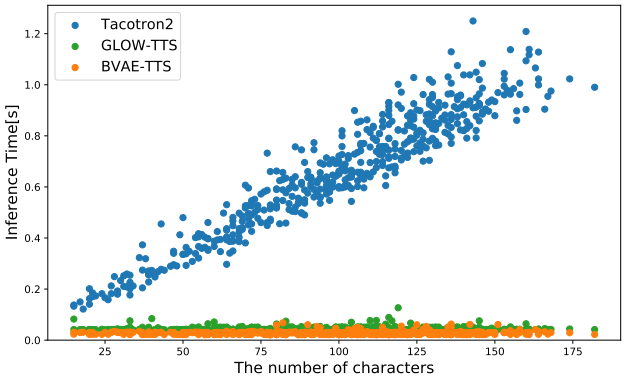

Figure 6: Inference time measured on 500 sentences of LJSpeech test set.

To see the benefit of parallel mel-spectrogram synthesis, we measure the inference time of Tacotron 2, Glow-TTS, and BVAE-TTS on the 500 sentences of LJSpeech test set. Figure 6 shows that the inference times of the non-AR TTS models are almost constant even if the length of the input text gets longer. On the contrary, the inference time of Tacotron2 linearly increases.

C ANALYSIS ON HIERARCHY

While changing the target BVAE block and its temperature as described in 5.3.1, we observe the generated mel-spectrogram samples. As shown in Figure 7-9, the variance of the mel-spectrograms is clear when we increase the variance of the two bottom BVAE blocks. On the contrary, the mel-spectrograms are almost same when we increase the variance of the top two BVAE blocks. Especially, when we change the temperature of 'BVAE block 2' to 5.0, the mel-spectrograms are the most diverse with good speech quality.

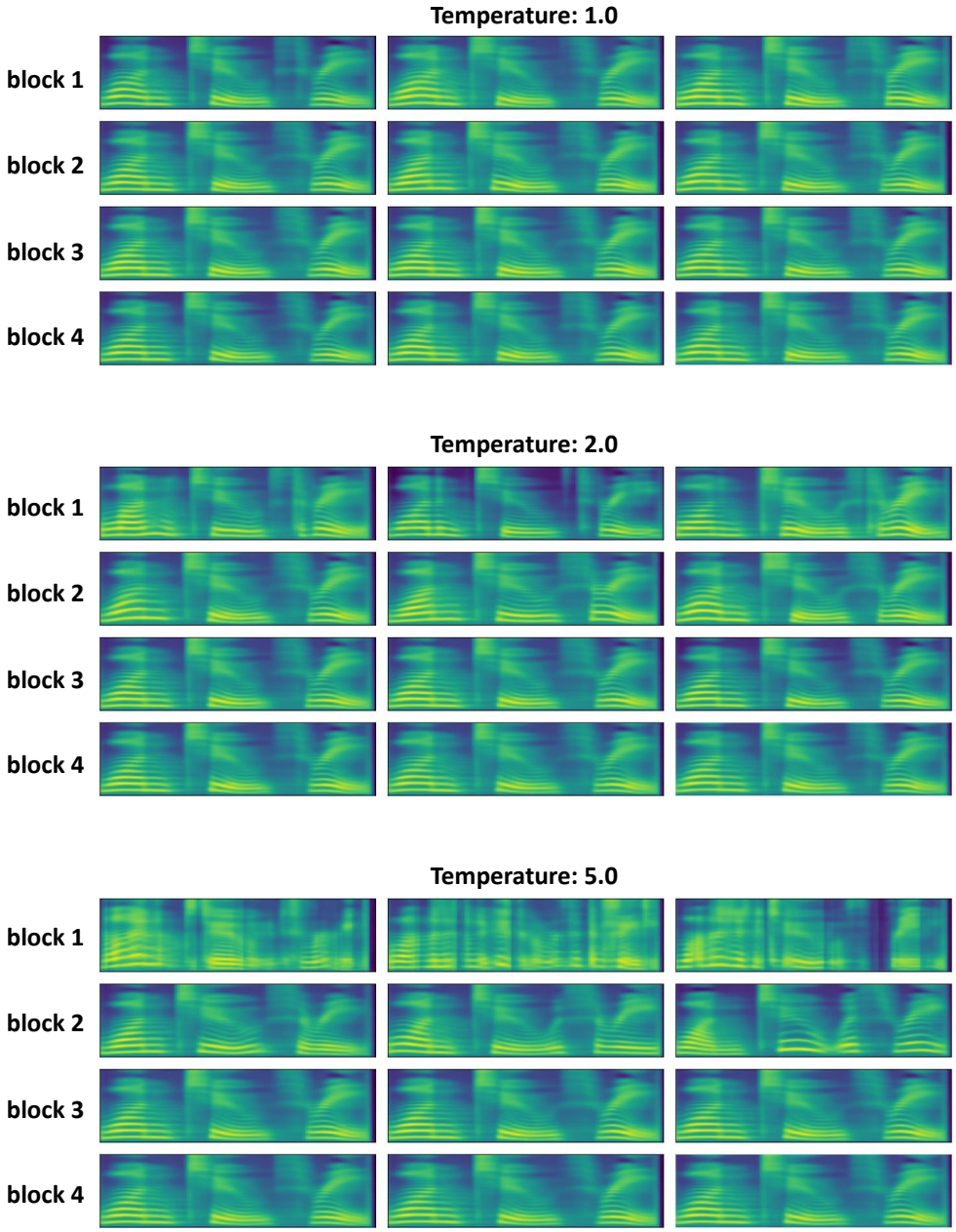

Figure 7: Mel-spectrograms generated from same text, "One, Two, Three.".

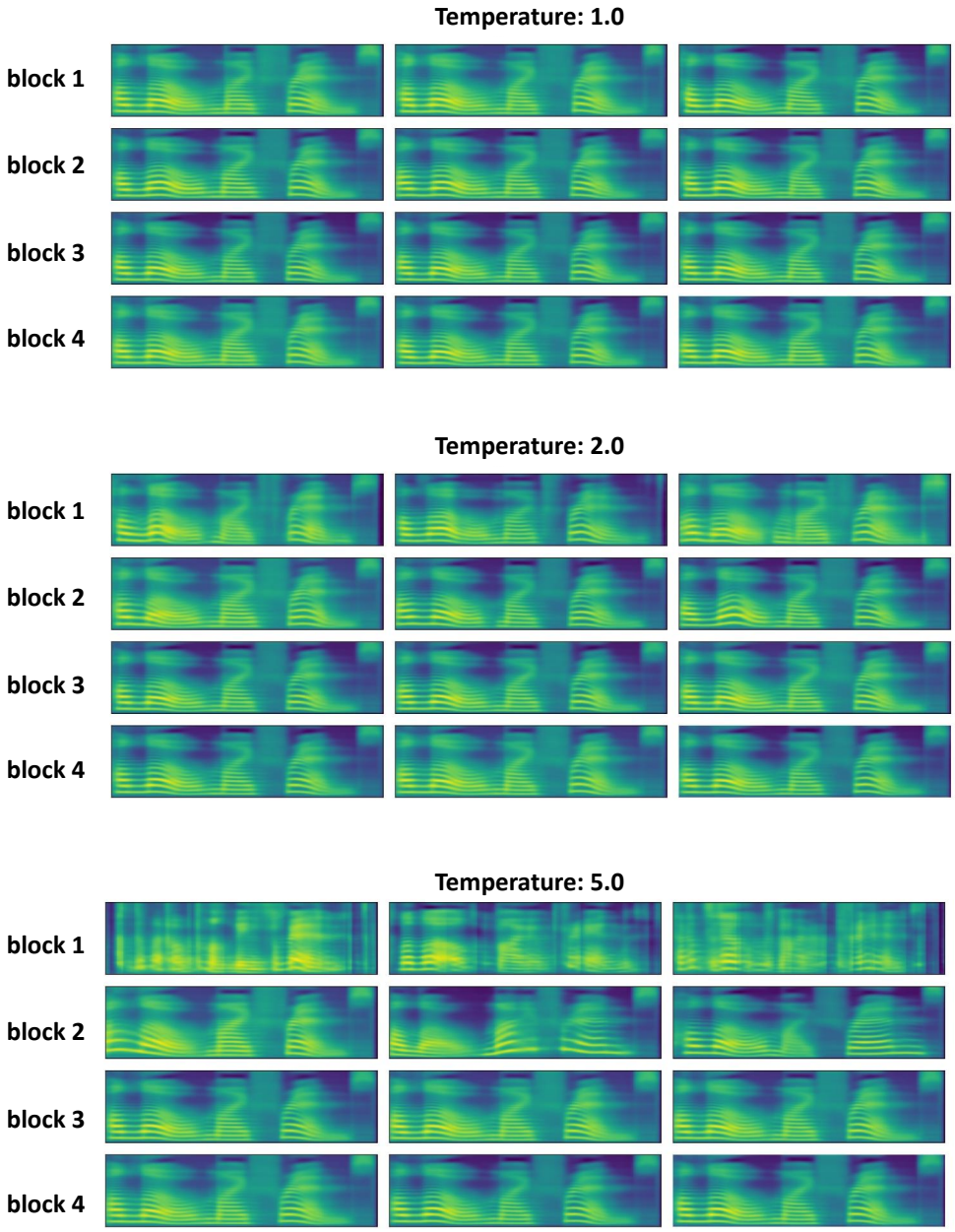

Figure 8: Mel-spectrograms generated from same text, “Hello, my friends.”.

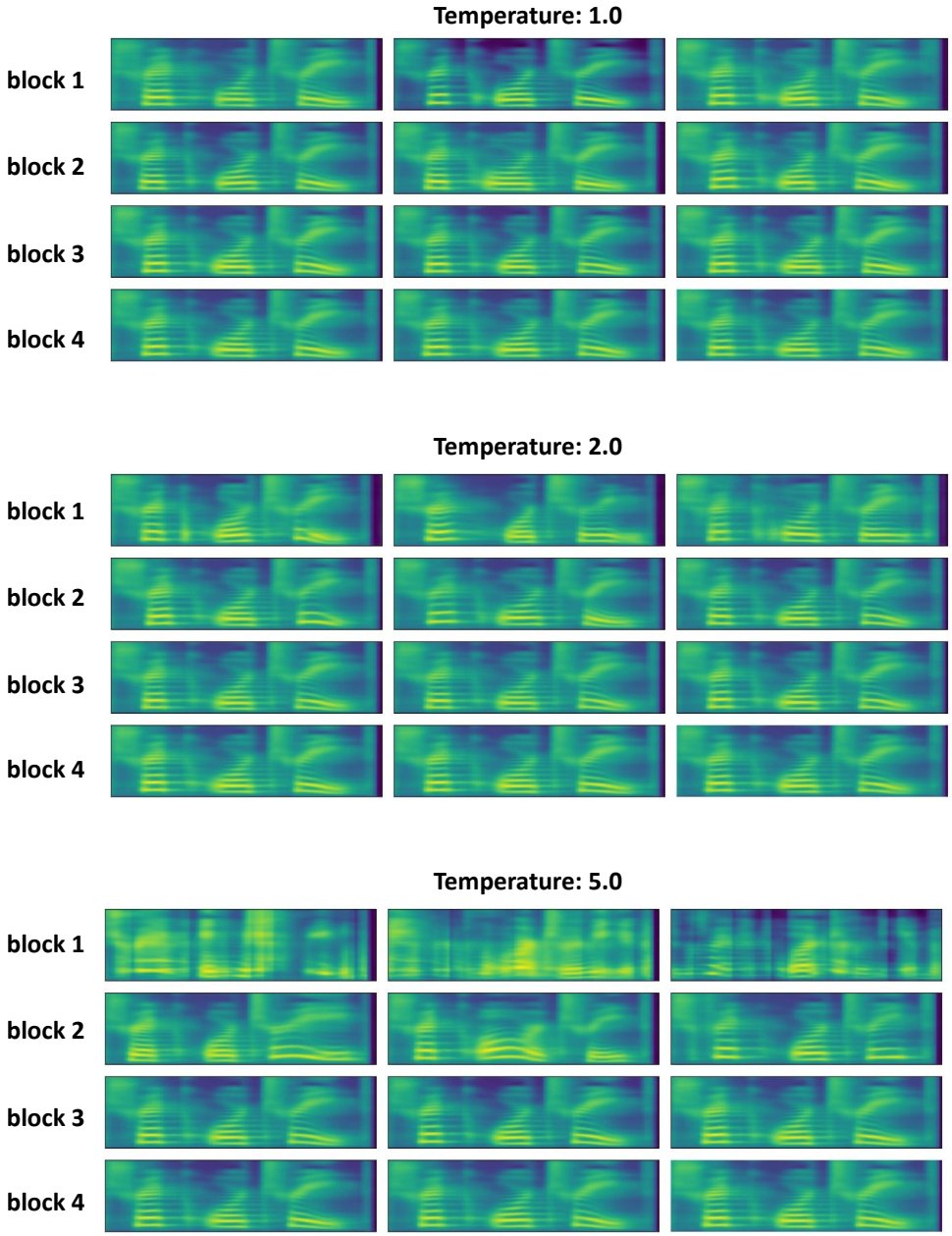

Figure 9: Mel-spectrograms generated from same text, “Trick or treat!”.

D SPEED CONTROL

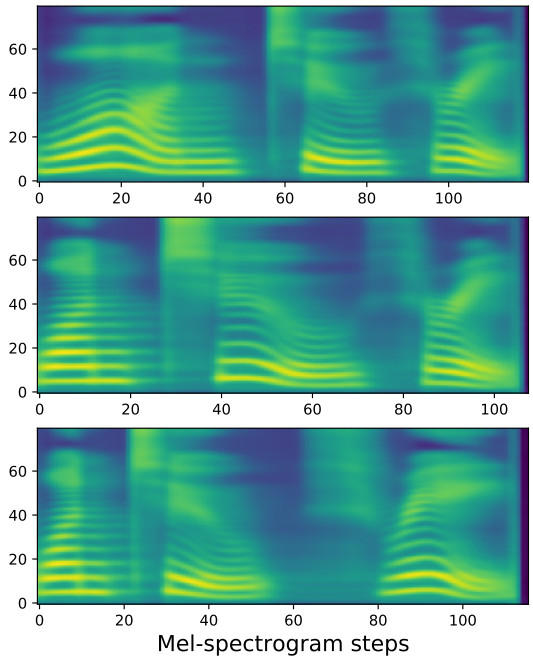

Figure 10: Speed controlled mel-spectrograms generated from same text "One, Two, Three."

While AR-based TTS models generally suffer from the lack of controllability, BVAE-TTS can control the fine-grained speed of speech by multiplying a positive constant to predicted durations from the duration predictor. Figure 10 shows three mel-spectrograms produced by BVAE-TTS using the same sentence, "One, Two, Three.". By changing the target word from "One" to "Three", we multiply 2.0 to the durations of phonemes belonging to the target word, and multiply 0.7 to the durations of phonemes belonging to the others. In this experiment, we observe that BVAE-TTS successfully generates speech while varying the pronouncing speed of each word in a single sentence. Interestingly, our model intensifies the pronunciation of the target word, showing the capability of properly adjusting the prosody according to the speed.