Nix-TTS: An Incredibly Lightweight End-to-End Text-to-Speech Model via Non End-to-End Distillation

Rendi Chevi, Radityo Eko Prasoj, Alham Fikri Aj

Abstract

We propose Nix-TTS, a lightweight neural TTS (Text-to-Speech) model achieved by applying knowledge distillation to a powerful yet large-sized generative TTS teacher model. Distilling a TTS model might sound unintuitive due to the generative and disjointed nature of TTS architectures, but pre-trained TTS models can be simplified into encoder and decoder structures, where the former encodes text into some latent representation and the latter decodes the latent into speech data. We devise a framework to distill each component in a non end-to-end fashion. Nix-TTS is end-to-end (vocoder-free) with only 5.23M parameters or up to 82% reduction of the teacher model, end fashion. Nix-TTS is end-to-end (vocoder-free) with only 5.23M parameters or up to 82% reduction of the teacher model.

Index Terms: speech synthesis, knowledge distillation, lightweight text-to-speech, model compression

1. Introduction

Recent high-fidelity TTS models [1][2][3] are known to be quite large and suffer from slow CPU inference. Moreover, some of them are actually text-to-mel models which require an additional neural vocoder. This limitations hinder the possibility of deploying a real-time, highly natural, and intelligible voice-based interface in resource-constrained settings, in which lightweight TTS models are more desirable.

Though recent work on lightweight TTS with diverse approaches has shown some promising results [4][5][6][7][8][9], most of them, except for [10], are text-to-mel models which require a separate neural vocoder to be trained. Indeed, there exists universal vocoder models [11][12]. However, they can be quite large, which inflates the final model size.

In this work, we propose Nix-TTS [a] a lightweight end-to-end TTS model achieved via non end-to-end knowledge distillation (KD) [13]. Specifically, we separate a pretrained end-to-end teacher TTS model into encoder and decoder. The former encodes the input text into a latent representation, while the latter decodes the latent into raw speech. We perform KD on each teacher and student models, respectively. Given applicable loss functions $\mathcal{L}_E$ and $\mathcal{L}_D$, our goal is to design and train $\mathcal{F}_s$, so that $\mathcal{E}_s$ and $\mathcal{D}_s$ satisfy

$$\text{argmin}_{\mathcal{E}_s}, \text{argmin}_{\mathcal{D}_s} \mathcal{L}_D(x_w, x_w), |\omega_s| \ll |\omega_t|$$

that is, $\mathcal{F}_s$ generates $\hat{z}$ (via $\mathcal{E}_s$) and $\hat{x}_w$ close to its counterparts of $\mathcal{F}_t$, all the while being significantly smaller.

2. Method

2.1. Problem Formulation

Let $\mathcal{F}(-; \omega)$ be an end-to-end neural TTS model. We restrict the term “end-to-end” TTS model as a model that can generate speech data $x$ in raw waveform $x_w$ from text $c$ directly without the need of an external vocoder. Though the architecture of end-to-end TTS varies, during inference, $\mathcal{F}$ can typically be composed of encoder $\mathcal{E}$ and decoder $\mathcal{D}$ as follows:

$$\mathcal{F} = \mathcal{D} \circ \mathcal{E}, \quad x_w = \mathcal{F}(c) = \mathcal{D}(\mathcal{E}(c)), \quad z = \mathcal{E}(c) \quad (1)$$

where $\mathcal{E}$ encodes $c$ to latent representation $z$, then $\mathcal{D}$ decodes $z$ into $x_w$. Depending on the model, $z$ can be deterministic as in [1][2][3] or generative as in [4][5] such that $z \sim \mathcal{N}(\mu, \sigma)$ or any other probability distribution.

Then, in a KD setting, let $\mathcal{F}_t$ and $\mathcal{F}_s$ be a teacher and a student end-to-end TTS model, respectively, following formulation 1 above. And, let $\{z, x_w\}$ and $\{\hat{z}, \hat{x}_w\}$ be the outputs generated by the teacher and student models, respectively. Given applicable loss functions $\mathcal{L}_E$ and $\mathcal{L}_D$, our goal is to design and train $\mathcal{F}_s$, so that $\mathcal{E}_s$ and $\mathcal{D}_s$ satisfy

$$\text{argmin}_{\mathcal{E}_s} \mathcal{L}_E(z, \tilde{z}), \text{argmin}_{\mathcal{D}_s} \mathcal{L}_D(x_w, \hat{x}_w), |\omega_s| \ll |\omega_t|$$

2.2. End-to-end TTS Teacher

We choose VITS [3] to be our teacher model $\mathcal{F}_t$. It is one of the few fully end-to-end, non-autoregressive, and high-fidelity TTS models. VITS has proven to beat two of the popular non end-to-end TTS models, Tacotron-2 [15] and Glow-TTS [3] in terms of speech quality while being around the same size ($\sim$30M parameters). Besides the speech quality, the large size and the model complexity make VITS a suitable teacher model for our KD.

2.2.1. Model Formulation

VITS is formulated as a conditional variational autoencoder (cVAE) augmented with normalizing flow and generative adversarial network (GAN). Following the cVAE framework proposed in VITS, let $q_\phi(z|x)$ and $p_\psi(x|z)$ be the posterior and data distributions respectively, parameterized by neural network’s parameters $\theta$ and $\phi$, where $x$ is the speech data variables and $z$ is the latent variables. The prior of which $z$ will be sampled from is defined as $p_\psi(z|c)$, where the latents are conditioned on input texts $c$, the prior distribution is parameterized by a neural network’s parameters $\psi$. VITS aims to learn the underlying distribution of $x$ given $c$, denoted as $p(x|c)$ by...
maximizing its evidence lower bound (ELBO).
\[
\log p(x|c) \geq -D_{KL}(q_theta(z|x)||p(z|c)) + \mathbb{E}_{q_theta(z|x)} \log p_theta(x|z) \tag{3}
\]

In practice, the reconstruction term is implemented as L1 loss between the ground truth and predicted speech data in mel-spectrogram form, \(x_m\), that is:
\[
\mathcal{L}_{recon} = \|x_m - \hat{x}_m\|_1 \tag{4}
\]

where \(\hat{x}_m \sim p_{\theta}(x|z)\).

Architecture-wise, VITS can be broken down into 3 modules, each encoding the distributions \(q_{\theta}(z|x)\), \(p_{\theta}(z|x)\), and \(p_{\theta}(z|c)\). We describe the role of each module in encoding and decoding the relevant features to be potentially distilled below.

**Posterior Encoder**

The module consists of non-causal WaveNet residual blocks [17] to encode \(x\) in linear spectorgram form, \(x_s\), into \(\{\mu_s, \sigma_s\}\), the parameters of \(q_{\theta}(z|x) = \mathcal{N}(\mu_s, \sigma_s)\). The module infers the latent samples \(z_s \sim \mathcal{N}(\mu_s, \sigma_s)\) which will then be passed to the Decoder to be reconstructed back to \(x\) in raw waveform \(x_w\).

**Prior Encoder**

The module consists of Transformer encoder blocks [18] and a normalizing flow \(f\), with affine coupling layers [19]. It encodes \(c\) into \(\{\mu_p, \sigma_p\}\), the parameters of \(p_{\theta}(z|c) = \mathcal{N}(\mu_p, \sigma_p)\), and prior latent samples, \(z_p \sim f(z_c)\).

The alignment between \(\{\mu_p, \sigma_p\}\) and \(z_p\) is then performed with Monotonic Alignment Search (MAS) [3]. The aligned prior’s parameters is denoted as \(\{\mu_p', \sigma_p'\}\). During inference, the network directly infers \(z_p\) from \(f^{-1}(\mu_p', \sigma_p')\), without needing \(x_s\).

**Decoder**

The module follows HiFi-GAN V1 generator architecture [11]. It learns to reconstruct \(z_s\) into \(x_w\) with the help of multi-period discriminator [3] in an adversarial fashion.

### 2.2. Available Knowledge to be Distilled

Assuming the teacher VITS is already trained, we can frame the model in encoder-decoder structure (Section 2.1). The natural configuration would be that the Prior Encoder serves as \(E_s\) that models the latent distribution \(q_{\theta}(z|x)\), and the Decoder serves as \(D_s\) that decodes \(x_s\) from latent samples \(z_s \sim q_{\theta}(z|x)\).

However, the Prior Encoder also encodes the same latent samples as the Prior Encoder, making both encoders suitable as \(E_s\). This means that the student \(E_s\) can actually be distilled from either, although the Prior Encoder is more complex due to the presence of \(f\) that only provides the stochastic samples of \(q_{\theta}(z|x)\), instead of its deterministic parameters. Thus, we opt to distil \(\hat{q}_{\theta}(z|x)\) from the Prior Encoder as it is arguably easier given the right student model.

### 2.3. End-to-end TTS Student

This section describes our proposed model, Nix-TTS. Nix-TTS serves as the end-to-end TTS student model \(F_s\) to be distilled from VITS as the end-to-end TTS teacher \(F_t\).

#### 2.3.1. Encoder Architecture

Nix-TTS’s encoder, \(E_s\), aims to model \(q_{\theta}(z|x) = \mathcal{N}(\mu_s, \sigma_s)\), by predicting its parameters, \(\{\mu_s, \sigma_s\}\). Because \(q_{\theta}(z|x)\) is conditioned on \(x\) instead of \(c\) which has different modality and dimensions, we need to encode \(c\) so that it is meaningfully aligned with the corresponding \(x\). To achieve this, we compose \(E_s\) out of 4 main modules: (1) Text Encoder, (2) Text Aligner, (3) Duration Predictor, and (4) Latent Encoder. We detail each of those modules below.

**Figure 1:** Nix-TTS distillation framework. The upper part illustrates the encoder-decoder structure (Section 2.1), while the lower part shows the student’s encoder model (Section 2.3.1).

**Text Encoder**

The module encodes \(c\) into text hidden representation \(c_{\text{hidden}}\). It feeds \(c\) through an embedding layer followed by an absolute positional encoding [18] and stacks of dilated residual 1D convolutions blocks. Each convolution block is followed by a SiLU (Sigmoid Linear Unit) activation [20] and a layer normalization [18]. We apply alternating dilation rates on each block [5] to increase the network’s receptive field.

**Text Aligner**

To learn the \(c\) and \(x_s\) alignment, we adopt the framework proposed by [21][22]. The model first encodes \(c_{\text{hidden}}\) and \(x_s\) into \(c_{\text{enc}}\) and \(x_{\text{enc}}\) with a series of convolutions. The “soft” alignment \(A_{\text{soft}}\) is then generated by taking the normalized pairwise affinity [22] between the two. As a non-autoregressive TTS requires “hard” durations to be defined per token, MAS [3] is applied to \(A_{\text{soft}}\) to generate \(A_{\text{hard}}\). The aligned text representation \(c_{\text{aligned}}\) can then be generated by applying batch matrix-matrix product between \(c_{\text{hidden}}\) and \(A_{\text{hard}}\). The generated alignments are defined as follows:

\[
A_{\text{soft}} = \text{softmax} \left( \sum_{k=1}^{K} \sum_{k'=1}^{K} (c_{\text{enc}}^{k} - x_{\text{enc}}^{k'})^2 \right) \tag{5}
\]

\[
A_{\text{hard}} = \text{MAS}(A_{\text{soft}}) \tag{6}
\]

**Duration Predictor**

The module serves to predict \(A_{\text{hard}}\) without the need of \(x_s\) during inference. We develop two variants of this module: deterministic and stochastic [4]. The latter increases the model expressiveness by being able to utter raw speeches with different rhythm and prosody for the same input text but with the cost of a slightly higher model size.

\[
d_{\text{hard}}^{ij} = \sum_{k=1}^{K} A_{\text{hard}}^{ijk} \tag{7}
\]

Regardless of which variant, stacks of 1D convolutions are used to predict the per-token durations \(d_{\text{hard}}\) extracted from \(A_{\text{hard}}\).
The main tasks of we break down the training objectives for both. We distill Nix-TTS in a non end-to-end fashion, where the en-

2.4. Non End-to-end Distillation Objective

respectively, making the training and inference much faster.

multi-period discriminator architecture \cite{4}. 

parameters. The decoder follows Hifi-GAN’s generator archi-

lows the same architecture as \( D \) whereas \( s \) is reduced capac-

Note that \( \mathcal{C}_n \) is the feature maps in the discriminator’s \( n \)th layer, \( n_t \) is the number of feature maps in \( t \)th layer, and \( L \) is the number of layers of \( \mathcal{C}_n \).

We also augment our decoder objective with generalized energy distance (GED) loss \cite{27} in order to accelerates the training convergence and improves the audio quality, defined as

\[
L_{\text{ged}} = \mathbb{E}_{x_w} \left[ 2d_{\text{spec}}(x_w, \hat{x}_w^*) - d_{\text{spec}}(\hat{x}_w^*, \hat{x}_w^t) \right] 
\]  

(13)

where \( d_{\text{spec}}(\cdot) \) is a multi-scale spectrogram distance as seen in \cite{28} whereas \( \hat{x}_w^* \) and \( \hat{x}_w^t \) are audio generated from \( D_s \) conditioned on the same \( \mu_s, \sigma_s \) but different noise samples drawn from \( N(0, 1) \). The final decoder objective is then computed as:

\[
L_D = L_{\text{adv, disc}} + L_{\text{adv, gen}} + L_{\text{match}} + L_{\text{recog}} + L_{\text{ged}} 
\]  

(14)

3. Experiments

In this section, we describe the configurations of the models, dataset, and training settings along with the evaluation metrics.

Teacher and Dataset Configuration

We use the VITS model released by the authors \cite{4}, which was trained on LIJSpeech \cite{29}. We use the same dataset in our KD experiments following the data processing protocol described in \cite{4}.

Student Configuration

The student follows the definition in Section 2.3. We use kernel size of 5, hidden size of 192, and alternating dilation rates of \{1, 2, 4\} for the text and latent encoder. We adopt the implementation of \cite{30} for the text aligner with the hyperparameters described in \cite{21}. For the duration predictor, we use two 1D convolution layers with kernel size of 3 and hidden size of 192 for the deterministic variant and follow \cite{31} with 2 layers of flow for the stochastic one. For the decoder, we use the modified HiFi-GAN v1 following Section 2.3.2 with upsample channel of 256.

MiniVITS: A Smaller VITS

In addition to the VITS and Nix-TTS models, we implement and test another model we call MiniVITS, a smaller version of the teacher VITS with a similar model size to Nix-TTS. This is achieved by cutting the number of layers of each VITS’ modules by half, while keeping the training configuration and the rest of hyperparameters the same as VITS’ default. MiniVITS serves as a naive end-to-end distillation baseline to be compared with the more thoughtfully-designed Nix-TTS.

Training Configuration

We train Nix-TTS and MiniVITS on LIJSpeech, following the split and configuration used on the VITS teacher model \cite{4}. We use AdamW \cite{31} with an initial LR of \( 2 \times 10^{-3} \), \( \beta_1 = 0.8 \), \( \beta_2 = 0.99 \), and \( \lambda = 0.1 \). We also apply LR decay exponentially by a factor of 0.999875 per epoch. All models are trained on a single RTX 3090 GPU with a total batch size of 32.

Model Evaluation

We employ crowd-sourced CMOS (Comparison Mean Opinion Score) \cite{32} to compare the relative speech quality between the three models. In addition, we assess the models’ duration stochasticity by evaluating the length variation of the predicted speeches, following the experiment settings in \cite{33, 4}. Finally, for model speedup and complexity, we compare the number of parameters, storage requirement, and RTF (Real Time Factor). We measure the RTF in two sce-
normal python-dependent model might not be able to run. To this end, we transform each model into an optimized ONNX version [14].

4. Results

4.1. Speech Quality - CMOS

We conducted two CMOS surveys, to compare Nix-TTS and MiniVITS against the teacher VITS. Our results (Table 1) show that our Nix-TTS model is only slightly less preferred than the teacher VITS model, while being significantly smaller (by 82%). On the other hand, MiniVITS is much less preferred than the teacher while having a similar number of parameters compared to Nix-TTS. This shows that our distillation technique works well and it is better than a naive end-to-end KD.

Table 1: Comparison Mean Opinion Score

| Model       | ↓ Params | ↑ CMOS |
|-------------|---------|--------|
| VITS        | 29.08M  | 0.00   |
| MiniVITS    | 5.20M   | −0.90  |
| Nix-TTS     | 5.23M   | −0.27  |

4.2. Speech Variation

Our experiments show that Nix-TTS manages to output speeches in meaningfully varying degrees of rhythm when equipped with a stochastic duration predictor. Fig. 2 indicates the sampled durations from Nix-TTS is varied, though it is 1.71× less spread-out than VITS, which is owed to its reduced learning capacity. Conversely, MiniVITS density is flatter with a high variance, even more than the teacher. This turns out to be not favorable, as it results in unnatural speech rhythms. We attribute this to the fact that MiniVITS training, unlike Nix-TTS, is unstable, which we observe throughout the experiment.

Figure 2: Comparison of durations sampled from each model.

4.3. Model Speedup and Complexity

Our results are summarized in Table 2. On a single thread Intel Core i7 CPU @ 1.10GHz, our Nix-TTS model is 2.66× faster than the teacher VITS model in terms of RTF. It is also faster than MiniVITS which was distilled end-to-end naively. On Raspberry Pi Model B, the speedup is even more pronounced than MiniVITS which was distilled end-to-end naively. On the other hand, MiniVITS is much less preferred than the teacher while having a similar number of parameters compared to Nix-TTS. This shows that our distillation technique works well and it is better than a naive end-to-end KD.

Table 2: Model Speedup and Complexity

| Intel Core i7 CPU @ 1.10GHz (Single Thread) |
|---------------------------------------------|
| Model                                      | ↓ Params | ↑ CompRat | ↑ RTF | ↑ SpdRat |
| FastSpeech-2⋆                              | 49.07M   | —         | 0.519 | —        |
| Glow-TTS*                                  | 42.51M   | 13.30%    | 0.423 | 1.22×    |
| VITS1           ⚫                          | 29.08M   | 40.73%    | 0.484 | 1.07×    |
| MiniVITS       ⚫                          | 5.20M    | 89.40%    | 0.256 | 2.02×    |
| Nix-TTS         ⚫                          | 5.23M    | 89.34%    | 0.159 | 3.26×    |
| Nix-TTS         ⚫                          | 6.00M    | 87.58%    | 0.187 | 2.77×    |
| Raspberry Pi Model B                        |
| Model                                      | ↓ Storage | ↑ CompRat | ↑ RTF | ↑ SpdRat |
| VITS1            ⚫                          | 113.5MB   | —         | 16.50 | —        |
| MiniVITS        ⚫                          | 17.8MB   | 84.31%    | 2.019 | 8.17×    |
| Nix-TTS         ⚫                          | 21.2MB   | 81.32%    | 1.974 | 8.36×    |
| Nix-TTS         ⚫                          | 21.7MB   | 80.88%    | 1.978 | 8.34×    |

*The model requires a vocoder for which we use HiFi-GAN v1. †: model equipped with stochastic duration predictor, ⚫: ONNX version of the model. Underlined models: reference to compute the compression (CompRat) and speedup ratios (SpdRat) for each setting. CompRat is based on number of parameters (Params) on Intel-i7 CPU results, but based on Storage on Raspberry Pi.

Figure 3: *Models with an external vocoder (HiFi-GAN v1). A→B indicates the performance comparison between model A and B according to B’s paper, unless if the label is bold which means the number comes from our own experiment in Table 1 or 2. Models in purple are other lightweight TTS models.

5. Conclusion

We proposed Nix-TTS, a lightweight end-to-end TTS model achieved by performing KD to a large teacher TTS model in a non-end-to-end fashion. Our experiments show that Nix-TTS retains a fair voice naturalness and intelligibility along with the duration stochasticity of the teacher. All the while being 82% smaller, 3.26× and 8.36× faster on Intel-i7 CPU and Raspberry Pi, and the smallest among several notable lightweight TTS.

3We use github.com/mingd24/FastSpeech2 [35].

4In speechresearch.github.io/lightspeech/ they use PWG [37].
6. References

[1] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, “Fastspeech 2: Fast and high-quality end-to-end text to speech,” in International Conference on Learning Representations, 2020.

[2] J. Donahue, S. Dulac-Arnold, M. Bindikowski, E. Elsen, and K. Simonian, “End-to-end adversarial text-to-speech,” in International Conference on Learning Representations, 2020.

[3] J. Kim, S. Kim, J. Kong, and S. Yoon, “Glow-tts: A generative flow for text-to-speech via monotonic alignment search,” Advances in Neural Information Processing Systems, vol. 33, pp. 8067–8077, 2020.

[4] J. Kim, J. Kong, and J. Son, “Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech,” in International Conference on Machine Learning, PMLR, 2021, pp. 5530–5540.

[5] J. Vainer and O. Dusek, “Speedyspeech: Efficient neural speech synthesis,” in Interspeech 2020. ISCA, 2020, pp. 3575–3579.

[6] Y. Lee, J. Shin, and K. Jung, “Bidirectional variational inference for non-autoregressive text-to-speech,” in International Conference on Learning Representations, 2021.

[7] S. Beliaev and B. Ginsburg, “Talknet 2: Non-autoregressive depth-wise separable convolutional model for speech synthesis with explicit pitch and duration prediction,” arXiv preprint arXiv:2104.08189, 2021.

[8] R. Luo, X. Tan, R. Wang, T. Qin, J. Li, S. Zhao, E. Chen, and T.-Y. Liu, “Lightspeech: Lightweight and fast text to speech with neural architecture search,” in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2021, pp. 5699–5703.

[9] C.-I. J. Lai, E. Cooper, Y. Zhang, S. Chang, K. Qian, Y.-L. Liao, Y.-S. Chuang, A. H. Liu, J. Yamagishi, D. Cox et al., “On the interplay between sparsity, naturalness, intelligibility, and prosody in speech synthesis,” arXiv preprint arXiv:2110.01147, 2021.

[10] H.-K. Nguyen, K. Jeong, S. Um, M.-J. Hwang, E. Song, and H.-G. Kang, “Litett: A lightweight mel-spectrogram-free text-to-wave synthesizer based on generative adversarial networks,” Proc. Interspeech 2021, pp. 3595–3599, 2021.

[11] J. Kong, J. Kim, and J. Bae, “Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis,” Advances in Neural Information Processing Systems, vol. 33, pp. 17022–17033, 2020.

[12] Y. Jiao, A. Gabryś, G. Tinchev, B. Putrycz, D. Korzekwa, and V. Klimkov, “Universal neural vocoding with parallel wavenet,” in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2021-2021. IEEE, 2021, pp. 6044–6048.

[13] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” stat., vol. 1050, p. 9, 2015.

[14] A. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. Grefe, E. Lockhart, L. Cubo, F. Sturmberg et al., “Parallel wavenet: Fast high-fidelity speech synthesis,” in International conference on machine learning. PMLR, 2018, pp. 3918–3926.

[15] W. Ping, K. Peng, and J. Chen, “Clarinet: Parallel wave generation in end-to-end text-to-speech,” in International Conference on Learning Representations, 2018.

[16] 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 ICASSP 2018. IEEE, 2018, pp. 4779–4783.

[17] R. Prenger, R. Valle, and B. Catanzaro, “Waveglow: A flow-based generative network for speech synthesis,” in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 3617–3621.

[18] A. Vasswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” Advances in neural information processing systems, vol. 30, 2017.

[19] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real NVP,” in 5th International Conference on Learning Representations, ICLR 2017. OpenReview.net, 2017.

[20] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for activation functions,” in 6th International Conference on Learning Representations, ICLR 2018. OpenReview.net, 2018.

[21] K. J. Shih, R. Valle, R. Badlani, A. Lancucki, W. Ping, and B. Catanzaro, “Rad-tts: Parallel flow-based tts with robust alignment learning and diverse synthesis,” in ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models, 2021.

[22] R. Badlani, A. Lancucki, K. J. Shih, R. Valle, W. Ping, and B. Catanzaro, “One tts alignment to rule them all,” arXiv preprint arXiv:2108.10447, 2021.

[23] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, “Neural spline flows,” Advances in neural information processing systems, vol. 32, 2019.

[24] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258.

[25] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, “Least squares generative adversarial networks,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2794–2802.

[26] A. B. Larsen, S. K. Sanderby, H. Larochelle, and O. Winther, “Autoencoding beyond pixels using a learned similarity metric,” in International conference on machine learning. PMLR, 2016, pp. 1558–1566.

[27] A. Gritsenko, T. Salimans, R. van den Berg, J. Snoek, and N. Kalchbrenner, “A spectral energy distance for parallel speech synthesis,” Advances in Neural Information Processing Systems, vol. 33, pp. 13062–13072, 2020.

[28] J. Engel, C. Gu, A. Roberts et al., “Ddsdp: Differentiable digital signal processing,” in International Conference on Learning Representations, 2019.

[29] K. Ito and L. Johnson, “The lj speech dataset,” https://keithito.com/LJ-Speech-Dataset/, 2017.

[30] K. Lee, “Comprehensive-transformer-tts,” Zenodo. [Online] Available: https://github.com/keonlee9420/Comprehensive-Transformer-TTS, DOI: 10.5281/zenodo.6137914, Zenodo, Feb. 2022.

[31] I. Loschilov and F. Hutter, “Decoupled weight decay regularization,” in International Conference on Learning Representations, 2018.

[32] P. C. Loizou, “Speech quality assessment,” in Multimedia analysis, processing and communications. Springer, 2011, pp. 623–654.

[33] R. Valle, K. J. Shih, R. Prenger, and B. Catanzaro, “Flowtron: an autoregressive flow-based generative network for text-to-speech synthesis,” in International Conference on Learning Representations, 2020.

[34] J. Bai, F. Lu, K. Zhang et al., “Onnx: Open neural network exchange,” https://github.com/onnx/onnx, 2019.

[35] C.-M. Chien, J.-H. Lin, C.-y. Huang, P.-c. Hsu, and H.-y. Lee, “Investigating on incorporating pretrained and learnable speaker representations for multi-speaker multi-style text-to-speech,” in ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 8588–8592.

[36] C. Miao, L. Shuang, Z. Liu, C. Minchuan, J. Ma, S. Wang, and J. Xiao, “EfficientTs: An efficient and high-quality text-to-speech architecture,” in International Conference on Machine Learning. PMLR, 2021, pp. 7700–7709.

[37] R. Yamamoto, E. Song, and J.-M. Kim, “Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram,” in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020, pp. 6199–6203.