FeatherTTS: Robust and Efficient attention based Neural TTS

Qiao Tian[1], *Chao Liu*[2*], *Zewang Zhang*[1], *Heng Lu*[1], *Linghui Chen*[1]
Bin Wei[3], *Pujiang He*[3], *Shan Liu*[1]

[1]Tencent
[2]Harbin Institute of Technology(Shenzhen)
[3]Intel Corporation

{briantian, zewangzhang, bearlu, nedchen}@tencent.com

Abstract

Attention based neural TTS is elegant speech synthesis pipeline and has shown a powerful ability to generate natural speech. However, it is still not robust enough to meet the stability requirements for industrial products. Besides, it suffers from slow inference speed owning to the autoregressive generation process. In this work, we propose FeatherTTS, a robust and efficient attention-based neural TTS system. Firstly, we propose a novel Gaussian attention which utilizes interpretability of Gaussian attention and the strict monotonic property in TTS. By this method, we replace the commonly used stop token prediction architecture with attentive stop prediction. Secondly, we apply block sparsity on the autoregressive decoder to speed up speech synthesis. The experimental results show that our proposed FeatherTTS not only nearly eliminates the problem of word skipping, repeating in particularly hard texts and keep the naturalness of generated speech, but also speeds up acoustic feature generation by 3.5 times over Tacotron. Overall, the proposed FeatherTTS can be 35x faster than real-time on a single CPU.

Index Terms: acoustic model, attention, text-to-speech

1. Introduction

In recent years, with the rapid development of deep learning, neural text-to-speech (TTS) can synthesize speech which is more natural and expressive than traditional TTS pipeline. Neural TTS is usually divided into two parts: an acoustic model and a neural vocoder. First, the input text (phoneme) sequence is converted into an intermediate acoustic feature sequence(linear spectrogram or mel-spectrogram) through an acoustic model such as Tacotron [1], Tacotron2 [2], Transformer TTS [3], FastSpeech [4], etc. Then, the Griffin-Lim algorithm [5] or neural vocoder such as WaveNet [6] and WaveRNN [7] is used to generate the final waveform according to the acoustic features. Sequence-to-sequence models with an attention mechanism are currently the predominant paradigm in neural acoustic model and have shown a powerful ability to generate expressive and high-quality speech. Those models learn the alignment between text sequence and frame-level acoustic features through the attention mechanism, and then predict spectral features that contain information such as pronunciation and prosody. The speech quality synthesized by the neural TTS is limited by the alignment generated by the attention mechanism. Although attention-based neural TTS has achieved great success, it is difficult to deploy in the industry due to its accidental alignment errors.

Tacotron [1] with content-based attention mechanism does not take into account the monotonicity and locality of TTS alignment, an improved hybrid location-sensitive attention (LSA) mechanism proposed in Tacotron2 [2] combines content-based and location-based features to achieve the synthesis of longer utterances. However, such hybrid mechanism also causes alignment issues occasionally. The LSA mechanism is borrowed from neural machine translation (NMT) and is not completely applicable TTS. Because the pronunciation is monotonous, for TTS, the alignment process is required to monotonous forward. For machine translation, the alignment process is not necessarily monotonous, It is possible that the last word of the target language corresponds to the first word of the source language. Therefore, many studies have adopted many techniques in the attention mechanism to ensure monotonicity. Such as [8] proposed the forward attention, which only considers the alignment paths that satisfy the monotonic condition at each decoder time step. And [8] further proposed a transition agent for monotonous attention, which achieves faster convergence speed and higher stability. [9] proposed a guided attention loss, which adds the prior knowledge of alignment monotinicity to the training process to help TTS models converge faster. Even, many researches use hard alignment based on duration expansion instead of attention mechanism, such as FastSpeech [4], DurIAN [10]. This type model usually requires an auxiliary model to help training.

Recently, inspired by the purely location-based GMM attention mechanism [11], an improved location-based GMM attention mechanism called GMMv2b is proposed in Google's work [12], which shows that the GMMv2b-based mechanism is able to generalize to long utterances, and can also improve speed and consistency of alignment during training. However, such GMM attention is unnormalized and not strictly monotonic, which leads to unstable performance. In addition, the commonly used stop token architecture in Tacotron often causes early stop phenomenon for complex texts and long sentences.

In this paper, we propose a novel attention-based neural TTS model named FeatherTTS, which can perform stable, fast and high-quality synthesis. Our major contributions are as follows: (1) We introduce the Gaussian attention for acoustic modeling, a monotonic, normalized and stable attention mechanism, which is very interpretable for end to end speech synthesis. (2) To solve the stop early issue, we remove the widely adopted stop token architecture in Tacotron2 and propose the attentive stop loss (ATL), which can determine whether to stop directly based on alignment and fast convergence for Gaussian attention. (3) To improve the inference speed and reduce the number of parameters without sacrificing the speech quality, we propose to adopt block sparse strategy to prune the weights of decoder .

[*] This work was done during internship in Tencent.

10.21437/SSW.2021-35

2. Related work

2.1. Hybrid attention based Tacotron2

Sequence-to-Sequence models with an attention mechanism are currently the predominant paradigm in neural TTS. Attention-based neural TTS such as Tacotron2 [2] generally uses an encoder to encode input sequence $x_{1:J}$ into hidden representation $h_{1:J}$ as

$$\{\boldsymbol{h}_{1:J}\} = Encoder(\{\boldsymbol{x}_{1:J}\}), \tag{1}$$

where J is the length of input phoneme sequence. Then, the attention RNN generates a state vector s_i, which is used as the query vector of the attention mechanism to generate alignment α_i at decode time i. According to the alignment α_i, a weighted average of the encoder output is calculated, which is the context vector c_i.

$$s_i = RNN_{Att}(s_{i-1}, c_{i-1}, y_{i-1}) \tag{2}$$

$$\alpha_i = Attention(s_i, ...) \qquad c_i = \sum_i \alpha_{i,j} h_j \tag{3}$$

Finally, the context vector c_i is fed into the decoder, and the final acoustic feature sequence $y_{1:T}$ is computed through post-net as

$$d_i = RNN_{Dec}(d_{i-1}, c_i, s_i) \qquad y_i = f_o(d_i), \tag{4}$$

where T is the length of output mel-spectrogram sequence.

Recently, many works have proposed various attention mechanism. Such as Tacotron [1] uses the purely content-based attention mechanism introduced in [13], Tacotron2 [2] uses an improved hybrid location-sensitive mechanism introduced in [14], some works [8, 15, 16] explore the use of monotonic attention mechanisms, and some authors [17, 18] use the location-based GMM attention.

2.2. Location based GMMv2b

Recently, Google's work [12] proposed a modified location-based attention mechanism which is called GMMv2b, has achieved great success. The GMMv2b mechanism is inspired by the location-based GMM attention mechanism introduced in [11]. The GMMv2b attention mechanism uses K Gaussian components to compute the alignment α_i as (5), where $\alpha_{i,j}$ is the weight of j-th element of encoder outputs, K is the number of Gaussian kernels, $\omega_{i,k}$ is the weight of k-th Gaussian component and $\mu_{i,k}, \sigma_{i,k}$ is the mean and standard deviation of k-th Gaussian component at decoding time i, respectively. The mean of each Gaussian component is computed following the recurrence relation in (6). The monotonicity of GMM attention is guaranteed by making Δ_i non-negative.

$$\alpha_{i,j} = \sum_{k=1}^{K} \frac{\omega_{i,k}}{Z_{i,k}} \exp\left(-\frac{(j-\mu_{i,k})^2}{2(\sigma_{i,k})^2}\right) \tag{5}$$

$$\mu_i = \mu_{i-1} + \Delta_i. \tag{6}$$

GMM attention usually calculates the intermediate variables ($\hat{\omega}_i$, $\hat{\Delta}_i$, $\hat{\sigma}_i$) first, and then uses the exponential function to obtain the final variables. In order to stabilize GMM attention, GMMv2b-based attention uses the softmax and the softplus functions to compute the final mixture parameters as

$$\begin{cases} Z_i = \sqrt{2\pi\sigma_i^2}, \\ \omega_i = S_{max}(\hat{\omega}_i), \\ \Delta_i = S_+(\hat{\Delta}_i), \\ \sigma_i = S_+(\hat{\sigma}_i), \end{cases} \tag{7}$$

where S_{max} and S_+ are the softmax function and the softplus function respectively. Besides, GMMv2b-based attention adds initial biases to the the intermediate parameters $\hat{\Delta}_i$ and $\hat{\sigma}_i$, which can encourage the final parameters to take on useful values at initialization.

As shown in [12], the GMMv2b-based mechanism is able to generalize to long utterances and maintains good naturalness, which makes the synthesis of the entire paragraph possible.

3. The proposed method

Although the GMMv2b-based mechanism has good performance, it also has many problems. First, this model still use stop token architecture which can lead to early stop. Second, GMM attention isn't completely monotonic because it uses a mixture of distributions with infinite support. Finally, GMMv2b attention is unnormalized because the attention weights are sampled from a continuous probability density function, this can lead to occasional spikes or dropouts in the alignment. Especially, there are repetition problems for the synthesis of short sentences, such as monophone and vowel. Therefore, we propose FeatherTTS, a more robust attention-based acoustic model, as shown in Fig. 1. Our model is based on the Tacotron2 [2] architecture and consists of a CBHG encoder, Gaussian attention and a block sparse decoder.

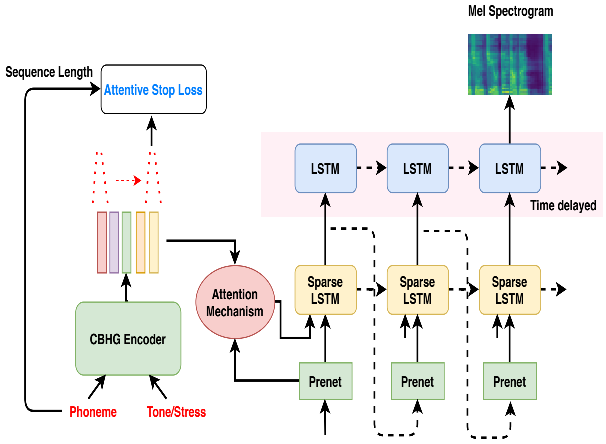

Figure 1: *The architecture of FeatherTTS*

3.1. Gaussian attention

In order to solve the incomplete monotonic and unnormalized problem in GMM attention, we propose to use Gaussian attention mechanism to model alignment, as shown in (8). Unlike the k Gaussian components used in GMM attention, we only use a single Gaussian function to calculate the alignment $\alpha_{i,j}$. Since the Gaussian function is naturally normalized, as long as the mean value of Gaussian attention at each decoding time step is monotonously forward, the monotonicity of the alignment can be guaranteed. We also calculate the intermediate variables ($\hat{\sigma}_i$, $\hat{\Delta}_i$) first, and then get the final parameters(σ_i, Δ_i) through the softplus function simlir to GMM attention.

$$\alpha_{i,j} = \exp\left(-\frac{(j-\mu_i)^2}{2(\sigma_i)^2}\right) \tag{8}$$

$$\mu_i = \mu_{i-1} + \Delta_i \tag{9}$$

We use such simple and normalized Gaussian attention function to calculate the alignment between the input phoneme sequences and the spectrogram frames. The mean μ_i and the variance $(\sigma_i)^2$ of the Gaussian attention mechanism control the position and width of the attention window, respectively. Δ_i is non-negative, so the mean μ_i is monotonically increasing, which guarantees the alignment process of the Gaussian attention mechanism is completely monotonic.

3.2. Attentive stop loss

The stop token architecture used in Tacotron2 [2] will cause stop early problems. Compared with the GMM attention of K Gaussian components, the single Gaussian attention mechanism has a weaker fitting ability and it will difficult to converge. Therefore, we need to add constraints to ensure that it can be aligned to the end of the input sequence at the end of decoding. In order to solve the above problems, we remove the stop token architecture, and propose the attentive stop loss, which directly judges the stop based on alignment. It is calculated as

$$L_{stop} = \left|\mu_T - (J+1)\right|, \tag{10}$$

where μ_T is the mean value of Gaussian attention function at last step, and J is the length of input phoneme sequence.

During training, the attentive stop loss forces the mean μ_i of Gaussian attention to go forward to the end of the phoneme sequence to ensure accurate alignment. In the inference stage, FeatherTTS will stop to predict when $\mu_i \geq (J+1)$.

3.3. Sparse autoregressive decoder

It has been demonstrated that, with the same computational complexity, a larger sparse network behaves better than a smaller dense network [7, 19]. In this work, to reduce the amount of computation of LSTM layers in decoder without a significant loss in quality, we reduce the number of non-zero values in each LSTM kernel weight. Inspired by [20, 21], we adopt the weight pruning scheme based on the weight magnitude.

We start to perform weight pruning after 20K steps and every 500 steps, we sort the weights of sparsified LSTM layers and zero out certain number of weights with the smallest magnitudes until the target sparsity 90% is reached at 200K step. After block sparsity, the number of main operations in every sparsified LSTM layer is

$$C = 4(1 - S)(I * H + H^2), \tag{11}$$

where I and H are the dimensions of input and hidden state of the LSTM cell, respectively, and S is the target sparsity.

In FeatherTTS, we used the time-delayed post-net as in [22], which is a vanilla LSTM layer with 256 units. Overall, FeatherTTS is trained to minimize the total loss as

$$Loss = \frac{1}{T}\sum_{i=1}^{T}\left|y_i' - y_i\right| + \frac{1}{T-d}\sum_{i=1}^{T-d}\left|y_{i+d}'' - y_i\right| + \lambda L_{stop}, \tag{12}$$

where d is the number of frames of time delay and λ is a scaling factor. On the right hand side of Eq. 12, the first two items of the loss function are L1 loss between reference mel-spectrogram y_i and the predicted both before and after mel-spectrogram y_i', y_i''. The last item is the attentive stop loss.

Table 1: *Mean Opinion Score (MOS) with* 95% *confidence intervals for different models.*

Model	**MOS on speech quality**
Tacotron2(GMMv2b)	4.31 ± 0.03
FeatherTTS w/o Block sparsity	4.32 ± 0.04
FeatherTTS	**4.33 ± 0.04**

4. Experiments

4.1. Data Set

We used a corpus containing 20 hours of Mandarin recordings by a professional broadcaster for all experiments. The corpus was split into a training set of approximately 18 hours and a test set of 2 hours. All the recordings were down-sampled to 24KHz sampling rate with 16-bit format. We used 80-band mel-scale spectrogram as training target, and then the mel-scale spectrogram was converted into waveforms by FeatherWave neural vocoder [23].

4.2. Experimental Setup

For comparison, we implemented two models including GMMv2b-based Tacotron2 [12] and FeatherTTS. As the baseline model, the GMMv2b-based model is composed of five mixture components. In order to reduce the model size, training and inference time, two consecutive frames were predicted at each decoding time step. For FeatherTTS, we delayed 5 frames and the rate of attentive stop loss λ was set to 0.001. All models were trained 300k steps with batch size 32 on a single GPU. Other experimental setups are the same as AdaDurIAN [22] if not specified.

4.3. Evaluations

In this section, we evaluated the proposed FeatherTTS and Tacotron2 (GMMv2b) [12] in term of naturalness and robustness, and compared the synthesis speed of the above two models with FastSpeech [4].

4.3.1. Mean Opinion Score

We used the Mean Opinion Score (MOS) to measure the naturalness of the synthesized speech[1]. Through crowdsourcing, we conducted the MOS evaluation on 20 synthesized audios which are unseen during training.The results of subjective MOS evaluation are presented in Table 1. The results show that, under the same vocoder configuration, both FeatherTTS and Tacotron2(GMMv2b) have similar MOS values. In addition, we compared the effect of block sparsity on the sound quality. It can be seen from the experimental results that FeatherTTS with block sparsity outperforms FeatherTTS without block sparsity with a gap of 0.01 in MOS, which is basically in line with our expectations.

4.3.2. Word Error Rate

FeatherTTS is designed to keep the naturalness as Tacotron2(GMMv2b) while avoiding the mispronunciations observed in the Tacotron2(GMMv2b). Wrod error rate (WER) is a general indicator for evaluating ASR and NMT

[1] Part of synthesized samples could be found at this URL: `https://wavecoder.github.io/FeatherTTS/`

Table 2: *The Word Error Rate (WER) for different models.*

Model	Word error rate
Tacotron2(GMMv2b)	4.1%
FeatherTTS	**0.9%**

Table 3: *The inference speed of different models.*

Model	Speed
FastSpeech	13.3x
Tacotron2(GMMv2b)	10.4x
FeatherTTS	**35.0x**
FeatherTTS BF16	**60.0x**

systems, and it can be used in TTS to measure the robustness of TTS synthesized speech. Therefore, we compared the robustness of two systems in terms of generated speech. We used manual listening and checking methods to perform fine-grained error checks on the synthesized speeches, such as pronunciation errors, word skipping, repeating, etc. The synthesized sentences are from different fields and are very hard for TTS, such as website links, alphanumeric combination, etc. There are a total of 50 test sentences and 10 participants, and each sentence is checked by at least 5 different participants. The final experiment results as shown in Table 2. We can see that Tacotron2(GMMv2b) has an error rate of 4.1%, while FeatherTTS is more robust, with an error rate of only 0.9%. This strongly proves the role of Gaussian attention and attentive stop loss in improving model stability.

4.3.3. Synthesis Speed

In this experiment, we proved the effectiveness of the block sparse decoder for accelerating training and inference. We compared the real-time rate of FastSpeech, Tacotron2(GMMv2b) and FeatherTTS to generate mel-spectrograms on a single core CPU(Intel Xeon Platinum 8255C). The results of synthesis speed are presented in Table 3. Tacotron2(GMMv2b) can achieve an inference speed of 10.4 times faster than real time, while FeatherTTS can further be accelerated by 3.5 times over Tacotron2(GMMv2b). In addition, compared with non-autoregressive FastSpeech, FeatherTTS is also about 2.6 times faster . Furthermore, we truncated the parameters and ran them on the BF16 [24, 25] format to reduce the memory consumption, and finally achieve 60 times faster than real-time on a single CPU core (Cooper Lake, 3rd Gen Intel Xeon Scalable processors). The above experiments prove the accelerating performance of the proposed methods for inference, and makes it possible to deploy TTS on edge devices.

5. Conclusions

In this work, we proposed FeatherTTS, an improved neural TTS system with Gaussian attention, attentive stop loss and block sparse decoder. Experiments demonstrate that such attention mechanism is very efficient and would greatly improve robustness of attention-based neural TTS system. With block sparse decoder, our proposed FeatherTTS can speed up the mel-spectrogram generation by 3.5 times faster than Tacotron2 nearly without any performance degradation. The ideas introduced in FeatherTTS pave a new way for both efficient and robust speech synthesis, and could be also applied to other sequence-to-sequence task including automatic speech recognition.

For future work, we will continue to investigate the performance of FeatherTTS on edge-devices.

6. Acknowledgments

The authors would like to thank Yi Xie in IAGS, Intel Asia-Pacific Research & Development Co Ltd.. This member in Intel helped to optimized our algorithm with AVX512 and BF16 intrinsics to get good performance on the 3rd Gen Intel Xeon Scalable processors.

7. References

[1] 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.

[2] 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.

[3] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6706–6713.

[4] 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.

[5] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[6] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio." in *SSW*, 2016, p. 125.

[7] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[8] J.-X. Zhang, Z.-H. Ling, and L.-R. Dai, "Forward attention in sequence-to-sequence acoustic modeling for speech synthesis," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2018, pp. 4789–4793.

[9] H. Tachibana, K. Uenoyama, and S. 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).* IEEE, 2018, pp. 4784–4788.

[10] 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.

[11] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[12] E. Battenberg, R. Skerry-Ryan, S. Mariooryad, D. Stanton, D. Kao, M. Shannon, and T. Bagby, "Location-relative attention mechanisms for robust long-form speech synthesis," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2020, pp. 6194–6198.

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

[14] 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.

[15] C. Raffel, M.-T. Luong, P. J. Liu, R. J. Weiss, and D. Eck, "Online and linear-time attention by enforcing monotonic alignments," *arXiv preprint arXiv:1704.00784*, 2017.

[16] M. He, Y. Deng, and L. He, "Robust sequence-to-sequence acoustic modeling with stepwise monotonic attention for neural tts," *arXiv preprint arXiv:1906.00672*, 2019.

[17] K. Kastner, J. F. Santos, Y. Bengio, and A. Courville, "Representation mixing for tts synthesis," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2019, pp. 5906–5910.

[18] E. Battenberg, S. Mariooryad, D. Stanton, R. Skerry-Ryan, M. Shannon, D. Kao, and T. Bagby, "Effective use of variational embedding capacity in expressive end-to-end speech synthesis," *arXiv preprint arXiv:1906.03402*, 2019.

[19] J.-M. Valin and J. Skoglund, "Lpcnet: Improving neural speech synthesis through linear prediction," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2019, pp. 5891–5895.

[20] S. Narang, E. Undersander, and G. Diamos, "Block-sparse recurrent neural networks," *arXiv preprint arXiv:1711.02782*, 2017.

[21] S. Narang, E. Elsen, G. Diamos, and S. Sengupta, "Exploring sparsity in recurrent neural networks," *arXiv preprint arXiv:1704.05119*, 2017.

[22] Z. Zhang, Q. Tian, H. Lu, L.-H. Chen, and S. Liu, "Adadurian: Few-shot adaptation for neural text-to-speech with durian," *arXiv preprint arXiv:2005.05642*, 2020.

[23] Q. Tian, Z. Zhang, H. Lu, L.-H. Chen, and S. Liu, "Featherwave: An efficient high-fidelity neural vocoder with multi-band linear prediction," *arXiv preprint arXiv:2005.05551*, 2020.

[24] P. Teich, "Tearing apart google's tpu 3.0 ai coprocessor," *Retrieved June*, vol. 12, p. 2018, 2018.

[25] S. Wang and P. Kanwar, "Bfloat16: the secret to high performance on cloud tpus," *Google Cloud Blog*, 2019.