SEMI-SUPERVISED TRAINING FOR IMPROVING DATA EFFICIENCY IN END-TO-END SPEECH SYNTHESIS

Yu-An Chung[*1] *Yuxuan Wang*[2] *Wei-Ning Hsu*[*1] *Yu Zhang*[2] *RJ Skerry-Ryan*[2]

[1]Massachusetts Institute of Technology [2]Google Inc.

ABSTRACT

Although end-to-end text-to-speech (TTS) models such as Tacotron have shown excellent results, they typically require a sizable set of high-quality <text, audio> pairs for training, which are expensive to collect. In this paper, we propose a semi-supervised training framework to improve the data efficiency of Tacotron. The idea is to allow Tacotron to utilize textual and acoustic knowledge contained in large, publicly-available text and speech corpora. Importantly, these external data are unpaired and potentially noisy. Specifically, first we embed each word in the input text into word vectors and condition the Tacotron encoder on them. We then use an unpaired speech corpus to pre-train the Tacotron decoder in the acoustic domain. Finally, we fine-tune the model using available paired data. We demonstrate that the proposed framework enables Tacotron to generate intelligible speech using less than half an hour of paired training data.

Index Terms— Tacotron, text-to-speech, semi-supervised learning, pre-training, data efficiency

1. INTRODUCTION

Recent advances in end-to-end text-to-speech (TTS) have shown great promise. We are now able to produce natural prosody with high audio fidelity using a much simplified voice building pipeline [1, 2, 3]. However, such models typically require a sizable dataset consisting of high-quality <text, audio> training pairs, which are expensive and time-consuming to collect. Requiring large amounts of data also hinders their applicability in low-resource settings.

This work aims to improve the data efficiency for end-to-end TTS training by leveraging large-scale, publicly available, and *unpaired* text and speech data. Unpaired data are plentiful and relatively easy to collect. Specifically, we propose a simple yet effective semi-supervised framework for training Tacotron [1], a recently proposed end-to-end TTS model. We propose to transfer the textual and acoustic representations learned from unpaired data to Tacotron in an unsupervised manner. This is then followed by a fine-tuning step using only a small amount of paired data to learn the alignment between the two representation domains.

In this preliminary study, we first identify the data requirement of a baseline Tacotron, i.e., the least amount of training data needed for a baseline Tacotron to produce intelligible speech. We then show that a Tacotron enhanced with the proposed framework is able to produce intelligible speech using less amount of data. Finally, we study different configurations for incorporating the framework. For evaluation, we perform both objective and subjective tests.

There exists previous work studying the application of unsupervised and weakly supervised learning for TTS [4, 5, 6, 7]. Related to our work, for example, [7] uses pre-trained word vectors in a LSTM-based acoustic model in parametric TTS [7]. These studies consider learning methods within the traditional TTS paradigm, however. This work, by contrast, examines them within end-to-end TTS, and specifically targets the data efficiency problem.

2. PROPOSED APPROACH

We use a baseline Tacotron architecture specified in [8], where we use a GMM attention [9], LSTM-based decoder with zoneout regularization [10] and phoneme inputs derived from normalized text. We use Griffin-Lim [11] as the inversion algorithm to convert the predicted spectrograms to waveforms, as our main focus is to enable Tacotron training on small data instead of producing high-fidelity audio. Using Griffin-Lim allows much faster experiment cycles.

The two main building blocks of Tacotron are the encoder and the attention-based decoder. At a high level, the encoder takes a source text as input and produces sequential representations of it; the decoder is then conditioned on the text representations to generate corresponding acoustic representations (spectrogram frames), which are then converted to waveforms. In the baseline Tacotron, the model is trained from scratch where all network weights are randomly initialized, and both the text and acoustic representations are learned from the given (parallel) training data. Below, we introduce our approach to inject external textual and acoustic knowledge to bootstrap the encoder and decoder, respectively.

[*]Work done while at Google.

Sound demos can be found at `https://google.github.io/tacotron/publications/semisupervised`.

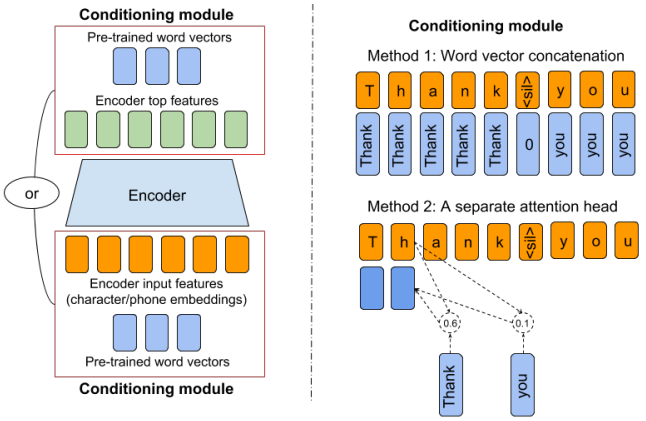

Fig. 1: Illustration of conditioning encoder on pre-trained word vectors. The left side shows the locations of encoder input and encoder top, where the word vectors can be incorporated via a conditioning module. The right side illustrates the conditioning module, where we show two methods of conditioning. '<sil>' denotes silence (e.g. space). In method 2, the dash lines correspond to the attention mechanism.

2.1. Conditioning the encoder on pre-trained word vectors

The goal of the encoder is to extract robust sequential representations of text. However, for a baseline Tacotron, the only training signal comes from the text data in the <text, audio> pairs, and the extracted representations are usually not rich enough when there's only a small amount of text.

We propose to exploit the textual knowledge contained in large text corpora, which typically contain millions to billions of words. From these large text corpora, one can train real-valued word vectors that contain the meanings of the words [12, 13] or language models that model grammatical and semantic context [14]. These word vectors can be added as auxiliary inputs to a TTS model to convey additional textual knowledge not learnable from the original text data.

To expose this additional knowledge to the encoder, we first embed each word in the input text into a word vector, and add the word vector sequence on one of two locations (illustrated in the left side of Figure 1): "encoder input" representing the phoneme embedding sequence, or "encoder top" representing the final encoder output sequence. While both conditioning locations allow the encoder to access the pre-trained word vectors, the choice of the conditioning location is an important design choice, which we study in experiments. For convenience, we call both encoder input and encoder top features *conditioning location features*. Due to the fact that the word vectors and conditioning location features might have different time resolutions (different sequence lengths), below we propose two ways of combining them (illustrated in the right side of Figure 1).

2.1.1. Word vectors concatenation

The first conditioning approach concatenates the word vector at the first phoneme of the corresponding word and replicates the word vector across all phonemes in the word. Take input text "Thank you" as an example. The phoneme inputs are 'th', 'a', 'ng', 'k', '<sil>', 'y', 'uu' (same rule applies to character inputs), where '<sil>' denotes silence (e.g. space). The word vector of "Thank" is appended to the phoneme embeddings of phonemes 'th', 'a', 'ng', and 'k'. This approach can be thought of as an hard attention mechanism where the alignment is pre-determined by the position mapping between words and their phonemes in the input text.

2.1.2. A conditioning attention head

If we consider the phrase "Thank you", it's possible that the semantics of "Thank" can help the encoder to generate more robust representation for "you". However, the first approach never exposes the word vector of "Thank" to the encoder when it's processing the phoneme embeddings of "you". Our second approach attempts to resolve this by applying a separate attention head between word vectors and conditioning location features. It takes each conditioning location feature as the attention query to generate the corresponding context vector, which is a weighted sum of the word vectors. The context vector and the conditioning location feature are then concatenated together for further processing. This enables each conditioning location feature to extract and gather information it needs from all word vectors. In this work, we use a simple tanh based additive attention [15].

Since the encoder weights are still trained from scratch with random initialization, we refer to this approach as *encoder conditioning* for the rest of the paper.

2.2. Decoder pre-training

In a baseline Tacotron system, the decoder needs to simultaneously learn acoustic representations and their alignments with the text representations extracted by the encoder. To reduce the workload of the decoder, we propose using an independent speech data source to pre-train the decoder, such that it is initialized by a pre-learned acoustic representation. During pre-training, the decoder acts as a next-step frame predictor with teacher forcing. Since the only objective is to predict an acoustic frame from the previous one, this step does not require text transcripts. In this stage, we simply keep the encoder weights frozen and replace the attention context vectors by zero vectors. This forces the decoder to learn an autoregressive model of acoustics at the frame level.

After the decoder is pre-trained, we fine-tune the entire model (including both encoder and decoder) using paired data. By pre-training, the decoder no longer needs to learn the acoustic representations from scratch and can thus focus

more on learning the alignment between text and acoustic representations.

A potential source of error of our simple approach is that there is a model mismatch between decoder pre-training and model fine-tuning: during pre-training, the decoder is only conditioned on the previous frame; while during fine-tuning, it is additionally conditioned on the text representations from the encoder. Despite such a mismatch, we found decoder pre-training still helpful. In addition, we found that the pre-trained Tacotron converges much faster than the baseline.

3. EXPERIMENTS

We conduct experiments to demonstrate the effectiveness of our framework. We use an internal single-speaker US English dataset for training (fine-tuning).

3.1. Data requirements of the baseline Tacotron

To improve Tacotron's data efficiency, first we need to understand its limit. We'd like to answer the following question: what is the maximum amount of data N that could almost never successfully train a baseline Tacotron to produce intelligible speech? To find out N, we gradually decrease the amount of data used for training a baseline Tacotron from about 40 hours to about 12 minutes and listen to the synthesized speech on unseen phrases. As can be heard on our demo page, we estimated that using between 10 and 40 hours of data produces almost equally good synthesis, and using between 3 and 10 hours of data causes minor degradation but still sounds very good. However, when there are only about 24 minutes of data, the model fails to produce intelligible speech. When there are only 12 minutes of data, the model outputs gibberish that is impossible to understand. It's important to note that the transcripts in the 12 minutes data already cover all phonemes, therefore the failure is not simply due to phoneme coverage.

Therefore, in the next section, we focus on demonstrating the effectiveness of our semi-supervised framework using only 24 minutes of paired data.

3.2. Results on small data

Our encoder conditioning and decoder pre-training approaches can be applied to Tacotron independently or jointly. We denote the model that only incorporates encoder conditioning as *T-Enc*, model that only incorporates decoder pre-training as *T-Dec*, model that incorporates both as *T-Enc-Dec*, and the baseline Tacotron as *T-Base*.

We measure the synthesis quality using both objective and subjective tests. For the objective metric, we use mel cepstral distortions (MCD) [16], which measures the distance between synthesis and ground truth in the mel cepstrum space—the smaller the better. We use an evaluation set containing about 30 minutes (631 sentences) of unseen data. We found that our MCD results correlate well with our subjective perception. For subjective measurements, we ran a series of side-by-side preference tests using 1000 unseen phrases of different lengths.

Table 1: MCD between ground-truth audio and synthesis from 7 Tacotron variants (lower is better). For T-Enc, we include both the results of using NNLM (1st row) and W2V (2nd row) as the word embedding module; concatenation/attention and input/top denote the conditioning method and location, respectively. The best result is marked in bold.

T-Base	T-Enc				T-Dec	T-Enc-Dec
	concatenation		attention			
	input	top	input	top		
18.06	12.89	12.46	13.03	12.71	**12.09**	12.27
	13.72	13.14	13.86	13.51		

For encoder conditioning, we used a neural network language model (NNLM) [14] trained on English Google News 200B corpus from TensorFlow Hub as the word embedding module. The module maps each word to a 128-dimensional vector. We also tried word2vec (W2V) [17] trained on the same corpus as the word embedding module.

For decoder pre-training, we used VCTK [18], a publicly available corpus containing 44 hours of speech from 109 speakers, the majority of which have British accents. Note that there is an accent mismatch between the decoder pre-training (multiple speakers with British accents) and fine-tuning (single speaker with US accent) datasets. As mentioned above, we only use the speech signals in VCTK but not their transcripts.

3.2.1. MCD objective tests

The MCD results are shown in Table 1. We first compare the four configurations of encoder conditioning. Here we include the results of both NNLM and W2V word embeddings. We can see that models using NNLM always outperform their W2V counterpart. We speculate that this is because W2V only conveys word meanings but not the contextual or structural information, which is modeled in NNLM.

In terms of conditioning locations, we see that conditioning at encoder top always outperforms conditioning at encoder input. While feeding word vectors to the early parts of the network seems intuitive (as in encoder input conditioning), we believe it is not the best choice in the low-resource setting. If the encoder weights learned from small data are noisy, for example, they may "distort" well-trained word vectors. Therefore, conditioning word vectors at a higher layer (e.g. encoder top) may lead to better generalization.

In terms of conditioning method, we find that the simple concatenation method always outperforms using a separate attention head. We also attribute this to the limited training

data: although a separate attention head offers more flexibility for learning the alignment, it also introduces more trainable parameters. In summary, the best configuration for encoder conditioning is to directly concatenate word vectors obtained from a pre-trained NNLM at the encoder top. We used this configuration for T-Enc for the rest of the experiments.

From Table 1 we can see that T-Enc, T-Dec, and T-Enc-Dec all achieve much lower MCD than T-Base. Among them, T-Dec achieves the best result. However, T-Enc, T-Dec, and T-Enc-Dec achieve similar MCD results (12.46, 12.09, 12.27, respectively). As shown in side-by-side comparisons below, the raters did not strongly prefer one over the other two, either.

3.2.2. *Side-by-side subjective tests*

Table 2: Results of SxS subjective tests based on a 7-point rating scale. We report both rater preferences (in percentage) and p-values for each comparison.

Competing pair	Preference (%)			p-value
	Former	Latter	Neutral	
T-Base vs. T-Enc	3.3	65.1	31.6	1.07e-84
T-Base vs. T-Dec	3.2	61.8	35.0	3.47e-83
T-Enc vs. T-Dec	16.1	18.2	65.7	0.256
T-Enc-Dec vs. T-Dec	17.0	17.9	65.1	0.630

Table 2 shows the results of the four side-by-side (SxS) preference tests, comparing T-Base against T-Enc, T-Base against T-Dec, T-Enc against T-Dec, and T-Enc-Dec against T-Dec. As we can see from the table, both T-Enc and T-Dec significantly outperform T-Base: in both tests, raters strongly preferred them over the baseline by more than 60%. Interestingly, the raters considered T-Dec, T-Enc, and T-Enc-Dec similarly preferable. The results of SxS tests are consistent to those of MCD objective tests, and both demonstrate the effectiveness of our semi-supervised framework.

3.3. Results on other amounts of data

We also compare T-Base, T-Enc, T-Dec, and T-Enc-Dec trained on other amounts of data. In Figure 2, each curve corresponds to a Tacotron variant, showing the relationship between the amount of paired data used for training that Tacotron variant and the MCD between ground-truth audio and synthesis from it. We can see that the largest gap between T-Base and the three semi-supervised systems occurs when using only 12 minutes (0.5 shards) of paired data. The gap keeps decreasing when the amount of data increases. This phenomenon is somewhat expected, because with more paired data, Tacotron relies less on external knowledge for learning representations and alignments. However, semi-supervised Tacotron consistently achieves lower MCD than the baseline, which may indicate benefits beyond better data efficiency (e.g. improved prosody).

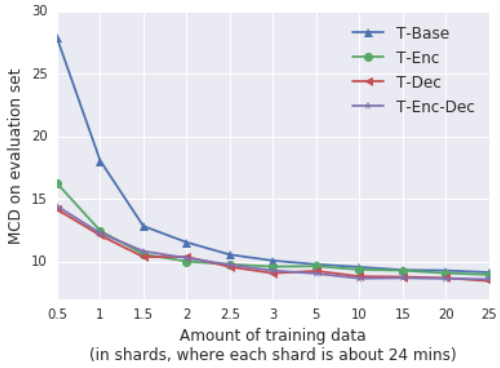

Fig. 2: MCD results by increasing the amount of paired data.

4. CONCLUSIONS AND DISCUSSIONS

We have proposed a semi-supervised training framework for improving data efficiency in end-to-end TTS. Our framework leverages large-scale, publicly available, and unpaired text and speech data to provide additional textual and acoustic knowledge to the Tacotron encoder and decoder, respectively. We have shown that our framework makes end-to-end TTS feasible in small-data regime. Specifically, a semi-supervised trained Tacotron can produce intelligible speech using just 24 minutes of paired training data. This promising result also provides some guiding principles for future data collection efforts for both single and multi-speaker TTS. While we used Tacotron as the TTS model in this study, we believe the framework is generally applicable to other end-to-end TTS models.

This is only a preliminary work, and there is still much to be investigated. For example, we've been using phoneme inputs in this work and we'd like to understand the performance tradeoffs on grapheme inputs. For leveraging textual knowledge, instead of simply conditioning with word vectors, a likely more effective method is to initialize the entire encoder with a pre-trained bidirectional NNLM [19]. For decoder pre-training, the model mismatch during pre-training and fine-tuning can be further studied. An analysis on what kind of information are extracted from external data and how they are actually used by Tacotron is also an important future work. Lastly, since the main focus of this work is to make end-to-end TTS feasible in small-data regime instead of producing high-fidelity audio, we only used Griffin-Lim as the waveform synthesizer. To produce high-fidelity speech with very little paired data, we still need to address the problem of adapting neural vocoders in the semi-supervised setting.

5. ACKNOWLEDGEMENTS

The authors thank Daisy Stanton, Eric Battenberg, Soroosh Mariooryad, Yinfei Yang, and the Machine Hearing and Google Brain teams for their helpful feedback and discussions.

6. REFERENCES

[1] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Zongheng Yang Jaitly, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous, "Tacotron: Towards end-to-end speech synthesis," in *INTERSPEECH*, 2017.

[2] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu, "Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions," in *ICASSP*, 2018.

[3] Wei Ping, Kainan Peng, and Jitong Chen, "Clarinet: Parallel wave generation in end-to-end text-to-speech," in *ICLR*, 2019.

[4] Oliver Samuel Watts, *Unsupervised learning for text-to-speech synthesis*, Ph.D. thesis, The University of Edinburgh, 2013.

[5] Heng Lu, Simon King, and Oliver Watts, "Combining a vector space representation of linguistic context with a deep neural network for text-to-speech synthesis," in *ISCA Workshop on Speech Synthesis*, 2013.

[6] Oliver Watts, Zhizheng Wu, and Simon King, "Sentence-level control vectors for deep neural network speech synthesis," in *INTERSPEECH*, 2015.

[7] Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao, "Word embedding for recurrent neural network based TTS synthesis," in *ICASSP*, 2015.

[8] Yuxuan Wang, Daisy Stanton, Yu Zhang, RJ Skerry-Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Fei Ren, Ye Jia, and Rif A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *ICML*, 2018.

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

[10] David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Aaron Courville, and Chris Pal, "Zoneout: Regularizing RNNs by randomly preserving hidden activations," in *ICLR*, 2017.

[11] Daniel Griffin and Jae 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.

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[13] Jeffrey Pennington, Richard Socher, and Christopher Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014.

[14] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, no. 2, pp. 1137–1155, 2003.

[15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[16] R. Kubichek, "Mel-cepstral distance measure for objective speech quality assessment," in *PacRim*, 1993.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, "Efficient estimation of word representations in vector space," in *ICLR Workshop*, 2013.

[18] Christophe Veaux, Junichi Yamagishi, and Kirsten MacDonald, "CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit," 2017.

[19] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer, "Deep contextualized word representations," in *NAACL-HLT*, 2018.