ON THE INTERPLAY BETWEEN SPARSITY, NATURALNESS, INTELLIGIBILITY, AND PROSODY IN SPEECH SYNTHESIS

Cheng-I Jeff Lai¹², Erica Cooper*³, Yang Zhang*², Shiyu Chang², Kaizhi Qian², Yi-Lun Liao¹
Yung-Sung Chuang¹, Alexander H. Liu¹, Junichi Yamagishi³, David Cox², James Glass¹

¹MIT CSAIL, USA  ²MIT-IBM Watson AI Lab, USA  ³National Institute of Informatics, Japan
clai24@mit.edu

ABSTRACT

Are end-to-end text-to-speech (TTS) models over-parametrized? To what extent can these models be pruned, and what happens to their synthesis capabilities? This work serves as a starting point to explore pruning both spectrogram prediction networks and vocoders. We thoroughly investigate the tradeoffs between sparsity and its subsequent effects on synthetic speech. Additionally, we explore several aspects of TTS pruning: amount of finetuning data versus sparsity, TTS-Augmentation to utilize unspoken text, and combining knowledge distillation and pruning. Our findings suggest that not only are end-to-end TTS models highly prunable, but also, perhaps surprisingly, pruned TTS models can produce synthetic speech with equal or higher naturalness and intelligibility, with similar prosody. All of our experiments are conducted on publicly available models, and findings in this work are backed by large-scale subjective tests and objective measures. Code and 200 pruned models are made available to facilitate future research on efficiency in TTS.

Index Terms— text-to-speech, vocoder, speech synthesis, pruning, efficiency

1. INTRODUCTION

End-to-end text-to-speech (TTS)¹ research has focused heavily on modeling techniques and architectures, aiming to produce more natural, adaptive, and expressive speech in robust, low-resource, controllable, or online conditions [1]. We argue that an overlooked orthogonal research direction in end-to-end TTS is architectural efficiency, and in particular, there has not been any established study on pruning end-to-end TTS in a principled manner. As the body of TTS research moves toward the mature end of the spectrum, we expect a myriad of effort delving into developing efficient TTS, with direct implications such as on-device TTS or a better rudimentary understanding of training TTS models from scratch [2].

To this end, we provide analyses on the effects of pruning end-to-end TTS, utilizing basic unstructured magnitude-based weight pruning³. The overarching message we aim to deliver is two-fold:

• End-to-end TTS models are over-parameterized; their weights can be pruned with unstructured magnitude-based methods.
• Pruned models can produce synthetic speech at equal or even better naturalness and intelligibility with similar prosody.

To introduce our work, we first review two areas of related work: Efficiency in TTS One line of work is on small-footprint, fast, and parallelizable versions of WaveNet [5] and WaveGlow [6] vocoders; prominent examples are WaveRNN [7], WaveFlow [8], Clarinet [9], HiFi-GAN [10], Parallel WaveNet [11], SqueezeWave [12], DiffWave [13], WaveGrad 1 [14], Parallel WaveGAN [15] etc. Another is acoustic models based on non-autoregressive generation (ParaNet [16], Flow-TTS [17], MelGAN [18], EfficientTTS [19], FastSpeech [20, 21]), neural architecture search (LightSpeech [22]), diffusion (WaveGrad 2 [23]), etc. Noticeably, efficient music generation has gathered attention too, e.g. NEWT [24] and DDSF [25].

ASR Pruning Earlier work on ASR pruning reduces the FST search space, such as [26]. More recently, the focus has shifted to pruning end-to-end ASR models [27, 28, 29, 30]. Generally speaking, pruning techniques proposed for vision models [3, 4] work decently well in prior ASR pruning work, which leads us to ask, how effective are simple pruning techniques for TTS?

This work thus builds upon a recent ASR pruning technique termed PARP [30], with the intention of not only reducing architectural complexity for end-to-end TTS, but also demonstrating the surprising efficacy and simplicity of pruning in contrast to prior TTS efficiency work. We first review PARP in Section 2. In Section 3, we describe our experimental and listening test setups, and in Section 4 we present results with several visualizations. Our contributions are:

• We present the first comprehensive study on pruning end-to-end acoustic models (Transformer-TTS [31], Tacotron2 [32]) and vocoders (Parallel WaveGAN [15]) with an unstructured magnitude based pruning method PARP [30].
• We extend PARP with knowledge distillation (KD) and TTS-Augmentation [33] for TTS pruning, demonstrating PARP’s applicability and effectiveness regardless of network architectures or input/output pairs.
• We show that end-to-end TTS models are over-parameterized. Pruned models produce speech with similar levels of naturalness, intelligibility, and prosody to that of unpruned models.
• For instance, with large-scale subjective tests and objective measures, Tacotron2 at 30% sparsity has statistically better naturalness than its original version; for another, small footprint CNN-based vocoder has little to no synthesis degradation at up to 88% sparsity.

* EC and YZ contribute equally.
1 Project webpage: https://people.csail.mit.edu/clai24/prune-tts/
² We refer to end-to-end TTS systems as those composed of an acoustic model (also known as text-to-spectrogram prediction network) and a separate vocoder, as there are relatively few direct text-to-waveform models; see [1].
³ Given that there has not been a dedicated TTS pruning study in the past, we resort to the most basic form of pruning. For more advanced pruning techniques, please refer to [3, 4].

Structured pruning was in fact employed in WaveRNN, but merely for reducing memory overhead for the vocoder. What sets this work apart is our pursuit of the scientific aspects of pruning end-to-end TTS holistically.
2. METHOD

2.1. Problem Formulation

Consider a sequence-to-sequence learning problem, where \( X \) and \( Y \) represent the input and output sequences respectively. For ASR, \( X \) is waveforms and \( Y \) is character/phone sequences; for a TTS acoustic model, \( X \) is character/phone sequences and \( Y \) is spectrogram sequences; for a vocoder, \( X \) is spectrogram sequences and \( Y \) is waveforms. A mapping function \( f(X; \theta) \) parametrized by a neural network is learned, where \( \theta \in \mathcal{R}^d \) represents the network parameters and \( d \) represents the number of parameters. Sequence-level log-likelihood \( \mathbb{E} \ln P(Y | X; \theta) \) on target dataset \( D \) is maximized.

Our goal is to find a subnetwork \( m \odot \theta \), where \( \odot \) is the element-wise product and a binary pruning mask \( m \in \{0, 1\}^d \) is applied on the model weights \( \theta \). The ideal pruning method would learn \( m \) at target sparsity such that \( f(X; m \odot \theta) \) achieves similar loss as \( f(X; \theta) \) after training on \( D \).

2.2. Pruning Sequence-to-Sequence Models with PARP

Unstructured Magnitude Pruning (UMP) [2, 3] sorts the model’s weights according to their magnitudes across layers regardless of the network structure, and removes the smallest ones to meet a predefined sparsity level. Weights that are pruned out (specified by \( m \)) are zeroed out and do not receive gradient updates during training.

Iterative Magnitude Pruning (IMP) [2, 3] is based on UMP and assumes an initial model weight \( \theta_0 \) and a target dataset \( D \) are given. IMP can be described as:

1. Directly prune \( \theta_0 \) at target sparsity, and obtain an initial pruning mask \( m_0 \). Zero out weights in \( \theta_0 \) given by \( m_0 \).
2. Train \( f(X; m_0 \odot \theta_0) \) on \( D \) until convergence. Zeroed-out weights do not receive gradient updates via backpropagation.

The above procedure can be iterated multiple times by updating \( \theta_0 \) with the finetuned model weight \( \theta_D \) from Step 2.

Prune-Adjust-Re-Prune (PARP) [30] is a simple modified version of IMP recently proposed for self-supervised speech recognition, showing that pruned wav2vec 2.0 [34] attains lower WERs than the full model under low-resource conditions. Given its simplicity, here we show that PARP can be applied to any sequence-to-sequence learning scenario. Similarly, given an initial model weight \( \theta_0 \) and \( D \), PARP can be described as (See Fig 1 for visualization):

1. Same as IMP’s Step 1.
2. Train \( f(X; \theta_0) \) on \( D \). Zeroed-out weights in \( \theta_0 \) receive gradient updates via backprop. After \( N \) model updates, obtain the trained model \( f(X; \theta_D) \), and apply UMP on \( \theta_D \) to obtain mask \( m_D \). Return subnetwork \( m_D \odot \theta_D \).

Setting Initial Model Weight \( \theta_0 \) In [30], PARP’s \( \theta_0 \) can be self-supervised pretrained initializations, or any trained model weight \( \theta_0 \) (\( P \) needs not be the target task \( D \)). On the other hand, IMP’s \( \theta_0 \) is target-task dependent i.e. \( \theta_0 \) is set to a trained weight on \( D \), denoted as \( \theta_D \). However, since the focus in this work is on the final pruning performance only, we set \( \theta_0 \) to \( \theta_D \) by default for both PARP and IMP.

Progressive Pruning with PARP-P Following [30], we also experiment with progressive pruning (PARP-P), where PARP-P’s Step 1 prunes \( \theta_0 \) at a lower sparsity, and its Step 2 progressively prunes to the target sparsity every \( N \) model updates. We show later that PARP-P is especially effective in higher sparsity regions.

3. EXPERIMENTAL SETUP

3.1. TTS Models and Data

Model Configs Our end-to-end TTS is based on an acoustic model (phone to melspec) and a vocoder (melspec to wav). To ensure reproducibility, we used publicly available and widely adopted implementations: Transformer-TTS [31] and Tacotron2 [32] as the acoustic models, and Parallel WaveGAN [15] as the vocoder. Transformer-TTS and Tacotron2 have the same high-level structure (encoder, decoder, pre-net, post-net) and loss (12 reconstructions before and after post-nets and stop token cross-entropy). Transformer-TTS consists of a 6-layer encoder and a 6-layer decoder. Tacotron2’s encoder consists of 3-layer convolutions and a BLSTM, and its decoder is a 2-layer LSTM with attention. Both use a standard G2P for converting text to phone sequences as the model input. Parallel WaveGAN consists of convolution-based generator \( G \) and discriminator \( D \).

Datasets LJSpeech [35] is used for training acoustic models and vocoders. It is a female single-speaker read speech corpus with 13k text-audio pairs, totaling 24h of recordings. We also used the transcription of Librispeech’s train-clean-100 partition [36] as additional unseen text\(^5\) used in TTS-Augmentation.

3.2. PARP Implementation

UMP is based on PyTorch’s API\(^7\). For all models, \( \theta_0 \) is set to pre-trained checkpoints on LJSpeech, and \( N \) is set to 1 epoch of model updates. We jointly prune encoder, decoder, pre-nets, and post-nets for the acoustic model; for vocoder, since only \( G \) is needed during test-time synthesis, only \( G \) is pruned (\( D \) is still trainable).

3.3. Complementary Techniques for PARP

TTS-Augmentation for unseen transcriptions The first technique is based on TTS-Augmentation [33]. It is a form of self-training, where we take \( f(\theta_D) \) to label additional unseen text \( X_u \). The newly synthesized paired data, denoted \( D_u = (X_u, f(X_u; \theta_D)) \), is used together with \( D \) in PARP’s Step 2.

\(^5\)Checkpoints are also available at ESPNet and ParallelWaveGAN.
\(^6\)Both LJSpeech and Librispeech are based on audiobooks.
\(^7\)PyTorch Pruning API
Combining Knowledge-Distillation (KD) and PARP, with a teacher model denoted as $f(\theta_D)$. The training objective in PARP’s Step 2 is set to reconstructing both ground truth melspec and melspec synthesized by an (unpruned) teacher acoustic model $f(\theta_D)$.

3.4. Subjective and Objective Evaluations

We examine the following three aspects of the synthetic speech:

- **Naturalness** is quantified by the 5-point (1-point increment) scale Mean Opinion Score (MOS). 20 unique utterances (with 5 repetitions) are synthesized and compared across pruned models, for a total of 100 HITs (crowdsourced tasks) per MOS test. In each HIT, the input texts to all models are the same to minimize variability.

- **Intelligibility** is measured with Google’s ASR API.

- **Prosody** via mean and standard deviation (std) fundamental frequency ($F_0$) estimations and utterance duration, averaged over dev and eval utterances.

We also perform pairwise comparison (A/B) testings for naturalness and intelligibility (separately). Similar to our MOS test, we release 20 unique utterances (with 10 repetitions), for a total of 200 HITs per A/B test. In each HIT, input text to models are the same. MOS and A/B tests are conducted in Amazon Mechanical Turk (AMT).

Statistical Testing. To ensure our MOS results are statistically significant, we run Mann-Whitney U test for each MOS test, and pairwise t-test for each A/B test, both at significance level of $p \leq 0.05$.

4. RESULTS

4.1. Does Sparsity improve Naturalness?

Fig 2 is the box plot of MOS scores of pruned end-to-end TTS models at 0%~99% sparsities with PARP. In each set of experiments, only one of the acoustic model or vocoder is pruned, while the other is kept intact. For either pruned Transformer-TTS or Tacotron2 acoustic models, their MOS scores are statistically not different from the unpruned ones at up to 90% sparsity. For pruned Parallel WaveGAN, pairing it with an unpruned Transformer-TTS reaches up to 88% sparsity without any statistical MOS decrease, and up to 85% if paired with an unpruned Tacotron2. Based on these results, we first conclude that end-to-end TTS models are over-parameterized across model architectures, and removing the majority of their weights does not significantly affect naturalness.

Secondly, we observe that the 30% pruned Tacotron2 has a statistically higher MOS score than unpruned Tacotron2. Although this phenomenon is not seen in Transformer-TTS, WaveGAN, or at other sparsities, it is nonetheless surprising given PARP’s simplicity. We can hypothesize that under the right conditions, pruned models train better, which results in higher naturalness over unpruned models.

4.2. Does Sparsity improve Intelligibility?

We examine the following three aspects of the synthetic speech:

- **Naturalness** is quantified by the 5-point (1-point increment) scale Mean Opinion Score (MOS). 20 unique utterances (with 5 repetitions) are synthesized and compared across pruned models, for a total of 100 HITs (crowdsourced tasks) per MOS test. In each HIT, the input texts to all models are the same to minimize variability.

- **Intelligibility** is measured with Google’s ASR API.

- **Prosody** via mean and standard deviation (std) fundamental frequency ($F_0$) estimations and utterance duration, averaged over dev and eval utterances.

We also perform pairwise comparison (A/B) testings for naturalness and intelligibility (separately). Similar to our MOS test, we release 20 unique utterances (with 10 repetitions), for a total of 200 HITs per A/B test. In each HIT, input text to models are the same. MOS and A/B tests are conducted in Amazon Mechanical Turk (AMT).

Statistical Testing. To ensure our MOS results are statistically significant, we run Mann-Whitney U test for each MOS test, and pairwise t-test for each A/B test, both at significance level of $p \leq 0.05$.

Fig 2. Box plots for four independent MOS tests across configurations (pruned/unpruned acoustic models + pruned/unpruned vocoders). At each sparsity, ■ is the mean and ◦ is the median MOS score over 100 HITs. Ground truth recordings (natural) are included as the topline.

Fig 3. Top plots the synthetic speech WERs over sparsities for all model combinations. Bottom compares the WERs for different pruning configurations.
4.3. Does Sparsity change Prosody?

![Figure 4](image.png)

**Fig. 4.** Top is utterance duration mismatch (in seconds), Middle is $F_0$ mean mismatch (in Hz), and Bottom is $F_0$ std (in Hz). Mismatches are calculated against ground truth recordings. Full model (0%) results are also included.

We used synthetic speech’s utterance duration and mean/std $F_0$ across time as three rough proxies for prosody. Fig 4 plots the prosody mismatch between pruned models and ground truth recordings across model combinations. Observe PARP on Tacotron2 and on Transformer-TTS result in visible differences in prosody changes over sparsities. In the top plot, pruned Transformer-TTS (yellow dotted line) have the same utterance duration (+0.2 seconds over ground truth) at 10%~75% sparsities, while in the same region, pruned Tacotron2 (purple dotted line) results in a linear decrease in duration (-0.2~0.8 seconds). Indeed, we confirmed by listening to synthesis samples that pruning Tacotron2 leads to shorter utterance duration as sparsity increases.

In the middle plot and up to 80% sparsity, pruned Tacotron2 models have a much large $F_0$ mean variation (-20~7.5 Hz) compared to that of Transformer-TTS (-10~15 Hz). We hypothesize that PARP on RNN-based models leads to unstable gradients through time during training, while Transformer-based models are easier to prune. Further, PARP on WaveGAN (red dotted line) has a minimal effect on both metrics across sparsities, which leads us to another hypothesis that vocoder is not responsible for prosody generation.

In the bottom plot and up to 80% sparsity, pruned models all have minimal $F_0$ std variations ($\leq$ 2 Hz) compared to 53Hz ground truth $F_0$ std. We infer that at reasonable sparsities, pruning does not hurt prosodic expressivity, due to lack of $F_0$ oversmoothing [1, 37].

4.4. Does more finetuning data improve sparsity?

In [30], the authors attain pruned wav2vec 2.0 at much higher sparsity without WER increase given sufficient finetuning data (10h Librispeech split). Therefore, one question we had was, how much fine-tuning data is “good enough” for pruning end-to-end TTS? We did two sets of experiments, and for each, we modify the amount of data in PARP’s Step 2, while keeping $\theta_0$ as is (trained on full LJSpeech).

The first set of experiments result is Fig 5. Even at as high as 90% sparsity, 30% of finetuning data (~7.2h) is enough for PARP to reach the same level of naturalness as full data\(^{10}\). The other set of experiment is TTS-Augmentation for utilizing additional unspoken text (~100h, no domain mismatch) for PARP’s Step 2. In Fig 5’s bottom plot, we see TTS-Augmentations (dark & light green lines) bear minimal effect on the synthetic speech WERs. However, Table 1 indicates that TTS-Augmentation PARP+seq-aug does statistically improve PARP in naturalness and intelligibility subjective testings.

![Figure 5](image.png)

**Fig. 5.** Effect of amount of finetuning data in PARP’s Step 2 on MOS score. Model is 90% pruned Transformer-TTS.

4.5. Ablations

**Knowledge Distillation hurts PARP** Surprisingly, we found combining knowledge distillation from teacher model $f(\theta_s)$ with PARP significantly reduces the synthesis quality, see PARP+KD v.s. PARP in Table 1. Perhaps more careful tuning is required to make KD work.

**Importance of $\theta_0$** Bottom plot of Fig 3 (black dotted line) and Table 1 (PARP v.s. PARP-RI) demonstrate the importance of setting the initial model weight $\theta_0$. In both cases, we set $\theta_0$ to random initialization (RI) instead of $\theta_D$ on LJSpeech.

**Effectiveness of IMP** Table 1 shows the clear advantage of PARP-P over IMP at high sparsities, yet PARP is not strictly better than IMP.

Table 1. A/B testing results. Each comparison is over 200 HITs. Bold numbers are statistical significant under pairwise z test.

<table>
<thead>
<tr>
<th>Proposal</th>
<th>Baseline</th>
<th>Sparsity Level</th>
<th>Preference over Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>pruned Transformer-TTS</td>
<td>Baseline</td>
<td>Naturalness</td>
<td>Intelligibility</td>
</tr>
<tr>
<td>+ unpruned Parallel WaveGAN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARP-P</td>
<td>95%</td>
<td>57%</td>
<td>66%</td>
</tr>
<tr>
<td>PARP+KD</td>
<td>70%</td>
<td>46%</td>
<td>43%</td>
</tr>
<tr>
<td>PARP-P</td>
<td>90%</td>
<td>53%</td>
<td>51%</td>
</tr>
<tr>
<td>PARP+seq-aug PARP</td>
<td>10%</td>
<td>58%</td>
<td>56%</td>
</tr>
<tr>
<td>PARP</td>
<td>30%</td>
<td>54%</td>
<td>58%</td>
</tr>
<tr>
<td>PARP+R1</td>
<td>90%</td>
<td>42%</td>
<td>37%</td>
</tr>
<tr>
<td>PARP+seq-aug PARP+R1</td>
<td>10%</td>
<td>58%</td>
<td>58%</td>
</tr>
</tbody>
</table>

5. REMARKS

**Significance** This work is scientific in nature, as most of our results arose from analyzing pruned models via large-scale experimentation and testing. In fact, we are less interested in answering questions like “how much can Tacotron2 be reduced to?”, and more curious about inquiries along the line of, “what are patterns/properties unique to end-to-end TTS?” Continuing in this direction should allow us to understand more in depth how end-to-end TTS models are trained.

**Future Work** Possible extensions upon this work are 1) inclusion of an ASR loss term in PARP’s step 2 for melspec pruning, 2) multilingual/multi-speaker TTS pruning, 3) further study on why over-parameterization benefit end-to-end TTS, 4) showing the Lottery Ticket Hypothesis [2] exists in end-to-end TTS, and 5) determining if insights from pruning help design better TTS model architectures.

---

\(^{10}\)The effect of using less data to obtain $\theta_0$ remains unclear.
6. REFERENCES