2020 54th Asilomar Conference on Signals, Systems, and Computers | 978-0-7381-3126-9/20/$31.00 ©2020 IEEE | DOI: 10.1109/IEEECONF51394.2020.9443419

WaveNetEQ — Packet Loss Concealment with WaveRNN

Florian Stimberg[1], Alex Narest[2], Alessio Bazzica[2], Lennart Kolmodin[2], Pablo Barrera González[2], Olga Sharonova[2], Henrik Lundin[2], and Thomas C. Walters[1]

[1]DeepMind, UK
[2]Google, Sweden

November 2020

Abstract

We present WaveNetEQ, a novel packet loss concealment method based on a WaveRNN architecture. The model is conditioned on a log-mel spectrogram of the past signal to extract slow moving features, like voice characteristics and prosody and achieves significantly better quality than pattern based methods for medium and long term packet loss. Through aggressive sparsification the model is efficient enough to run on a phone.

1 Introduction

With the recent increase in online call volume, good audio quality, despite an unstable internet connection, has become an even more important topic. Online calls can suffer from the issue of missing audio due to lost or mistimed data packets. The process of concealing these losses during a real-time voice call, known as packet-loss concealment (PLC), can be performed either at the codec level or as a post-processing step on the decoded audio. In this paper we present a new post-processing approach to packet-loss concealment, named WaveNetEQ, which is based on the use of a WaveRNN [1] neural network to predict missing segments of an audio stream. The commonly used post-processing methods are based on classic signal processing, often copying and repeating the previous pitch period (e.g. [2]). This yields good results for short packet losses but produces robot-like sounds for longer losses. We compare WaveNetEQ against NetEQ, the standard PLC algorithm in use in the widely-deployed WebRTC framework. NetEQ uses an LPC filter to estimate the voiced and unvoiced components of the signal. It then uses linear combinations of the last two pitch periods to continue the voiced component (see figure 1) and generated noise for the unvoiced.

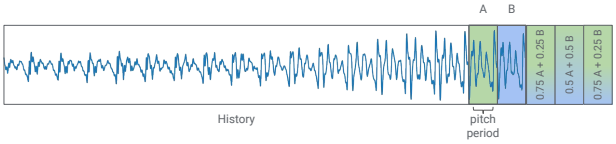

Figure 1: Packet loss concealment approach for voiced segments in NetEQ. The strong pitch-period is estimated through autocorrelation and then linear combinations of the last two pitch-period segments are use to continue the waveform.

2 Related Work

Packet loss concealment for speech has a long history in signal processing research but we'll focus on recent work using neural networks for waveform PLC. For an overview of classical PLC in signal processing see [3]. In [4] two feed forward neural networks are used to predict the log-power spectra and phase of missing frames given the same feature of the previous frame. They show quality improvements on previous methods but it is not clear if the approach would work well in a real time setting. Mul-

tiple different architectures are compared for long term ($> 200ms$) PLC in [5]. A major difference to our approach is that it uses both the audio information before and *after* the packet loss as input. Using the information of the waveform after the packet loss has obvious advantages but in a real time setting is only possible in very limited circumstances. Additionally, we specifically refrained from filling in too long segments as we want to avoid at all costs changing the content of the speech. A very different approach is used in [6]. They train an LSTM to predict the raw waveform but then *during inference* fine-tune it on the incoming packets. This approach is very interesting but the question remains if it is fast enough to run in real time on limited hardware.

3 Architecture

WaveRNN [1] is a text-to-speech (TTS) model that achieves audio quality on the level of the original WaveNet [7], but can be sampled from in real time on standard hardware. It consists of an autoregressive and a conditioning network. The former uses a single-layer recurrent neural network with a modified gated recurrent unit (GRU)[8] as its core. As input the autoregressive layer receives the last sample it produced and the output of the conditioning network that are combined using convolutional layers. The output is a dual-softmax layer that allows it to efficiently predict audio samples with 16-bit resolution. The conditioning network consists of a 1x1 convolution, followed by an RNN, using the same GRU cell as the autoregressive network, and three transposed convolutions which up-sample by a factor of 8 from the conditioning rate of 50Hz to 400Hz. The activations from the conditioning network are then broadcast to each time-step in the autoregressive network at its native sample rate. This means that there is a distinct new conditioning activation vector generated by the conditioning network every 2.5ms. The sampling runs at a frequency of 8kHz. Figure 2 shows the detailed architecture of WaveNetEQ.

The conditioning network usually has a wider receptive field, enabling it to model slow-changing, long-term features, while the autoregressive layer on its own is able to model the short term dynamics of realistic speech.

When WaveRNN is used for TTS, the conditioning layer gets the content and prosody of the target speech as input and influences the autoregressive network to produce the right waveforms. In a PLC setting we don't have access to this information. Instead, the conditioning network is fed a log-mel spectrogram window of the past audio, allowing it to extract the necessary information from the past audio and steering the autoregressive network to continue the audio in a way that fits the style and content of what was said just beforehand.

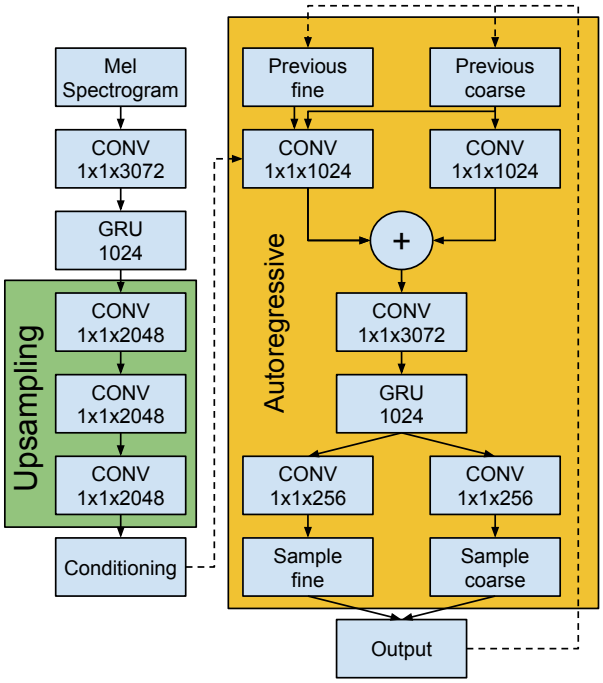

Figure 2: Detailed architecture of WaveNetEQ.

Compared to WaveRNN, for TTS we also change the conditioning network to only do a forward pass, otherwise the model would "see into the future" during training, which would not be possible during inference. For TTS this is not a concern as the text and prosody to synthesize is known at inference time.

4 Training

The model is trained on a dataset of 100 speakers in 48 different languages, containing over 1000 hours of speech overall. Training is done with teacher forcing [9]: at each

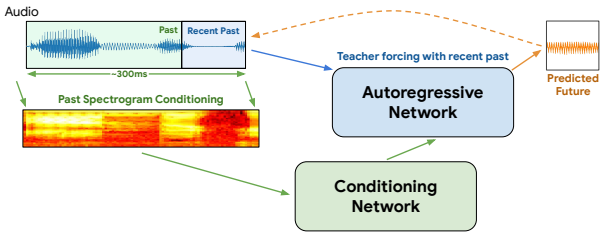

Figure 3: Inference-time signal pipeline for WaveNetEQ.

step it receives the spectrogram of the past and the last ground truth sample and then updates its weights through backpropagation. To make the model more robust we augment the dataset by creating synthetic mixes of the speech with background noise randomly selected from the MS-SNSD dataset[10]. This is done with signal-to-noise ratios from between 0 and 30dB and each speech sample is used to create 6 augmented samples this way.

A dense WaveRNN model would be too slow to run in real time on a mobile phone. Therefore we follow the original WaveRNN paper[1] and sparsify the weights of the model. The model is trained for 1000 steps normally and then weights are slowly sparsified in blocks of 4-by-4 until the overall sparsity reaches 96%, i.e. only 4% of the weights are non-zero. We then use custom inference code which allows real time sampling on a mobile CPU.

5 Inference

During inference we prime the GRU units' state through teacher forcing the model for 10ms on the recent past of the audio, as can be seen in figure 3.

We apply multiple measures to avoid the model coming up with speech in a mostly silent segments or continuing speech for so long that it could change the content of the conversation. WaveNetEQ's output layer is a softmax from which we sample during inference, therefore we can easily control the narrowness of the output distribution by tweaking the sampling temperature. A higher temperature will lead to more randomness, while lower temperature means a more peaked distribution, which in practice leads to the model tending towards silence. We employ a dynamic tempering strategy by starting out with a low temperature, which we then linearly increase up to

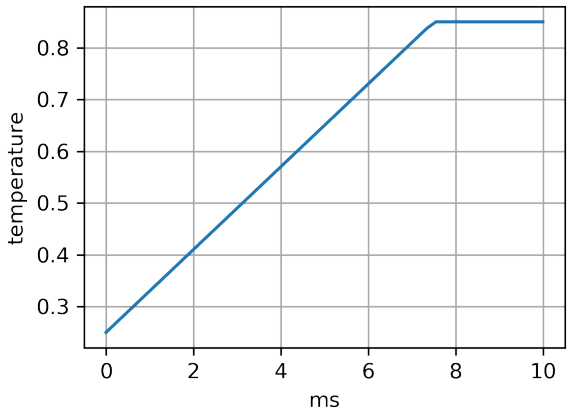

Figure 4: Dynamic temperature schedule used when sampling from WaveNetEQ.

a maximum value. This achieves a good balance between stopping the model from starting to "babble" during silent segments and avoiding the model to drop to silence too early. The temperature schedule can be seen in figure 4.

Additionally, we employ a level limiter by reducing the amplitude envelope of synthesized segments to be below the level of the past audio used for priming. This avoids the model suddenly increasing the volume of the speech.

As a final measure, we start to fade towards silence after 20ms, so that the model stops producing output after 120ms.

When the packet loss is over and we start receiving real audio again, the model needs to avoid a sudden jump from its synthetic output to the real waveform, which would result in unpleasant artifacts. To achieve this, we apply a simple merge-out, by first producing 25ms more audio than was lost in the network disruptions, and then finding the best alignment between the two signals by choosing the point of the highest cross correlation and cross-fading for 2ms around it.

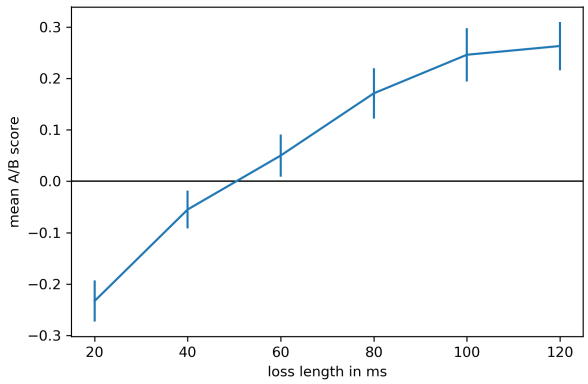

Figure 5: Mean A/B scores for different lengths of packet loss, with 95% confidence intervals. In each sample 10% of the overall audio was replaced by the PLC methods. The samples were rated on a scale from -3 to 3 where positive values meant WaveNetEQ sounded better.

method	loss length	A/B rating	
dynamic tempering	20ms	0.038	± 0.028
	120ms	0.015	± 0.031
noise augmentation	20ms	0.042	± 0.029
	120ms	-0.013	± 0.033
sparsity	20ms	-0.067	± 0.030
	120ms	-0.29	± 0.033

Table 1: A/B listening test results of ablation studies. Positive numbers mean raters preferred the full model while, negative numbers mean raters preferred the model where the specified method was not used.

6 Results

We tested how our model compared to the current PLC component in the WebRTC framework, NetEQ[1], through an A/B test. Samples from the librispeech [11] dataset given to raters were corrupted by removing 10% of the audio in equally spaced chunks. We varied the length of the individual packet losses to see how it affects the quality for both methods.

For all listening tests experienced raters were asked to listen to samples from both WaveNetEQ and NetEQ and rate which one had the better audio quality on a scale from +3 to -3. Overall 300 samples from each algorithm were evaluated and we used ratings from 7 different raters for each sample.

The results, seen in figure 5, show that while WaveNetEQ is worse for short losses of 20ms, and the differences are minimal from 40ms to 60ms, it performs significantly better for longer packet losses[2].

To see the effect of different factors of our training and inference setup we carried out multiple ablation studies. For these listening tests we compared samples from the final models with samples from models not using noise augmented training data, dynamic tempering and sparsification, respectively. The listening tests used 4 ratings per sample pair. The results in table 1 show that dynamic tempering leads to a significant effect for short packet losses. For long packet losses this effect was not significant. More importantly, qualitative examination showed that dynamic tempering removed any instances of babbling that we previously observed. Training on the noise augmented data also has a significant effect for short packet losses. Additionally, we observed that models trained on the augmented data performed better when the samples contained background noise. Models that weren't sparsified performed better in listening tests. But the difference was relatively small and not significant for longer packet losses. This was to be expected as we apply a very aggressive sparsification leaving only 4% of the weights to be non-zero. This loss in quality is acceptable because it allows us to run the models on a mobile phone which wouldn't be possible for the dense models.

As listening tests like these can only be an approximation of the actual application in a production setting, we conducted experiments in the Duo video chat app, which showed that WaveNetEQ significantly improved the user experience compared to NetEQ.

7 Conclusion

We introduced WaveNetEQ an autoregressive neural network based on the WaveRNN architecture that can pro-

[1]NetEQ does more than PLC and we only replace the post-processing PLC with WaveNetEQ.

[2]Audio samples comparing NetEQ to WaveNetEQ can be found at ai.googleblog.com/2020/04/improving-audio-quality-in-duo-with.html

duce real time packet loss concealment on a mobile phone and improves on current production systems for medium and long packet losses. Further investigation is needed for WaveNetEQ's failure to beat the NetEQ baseline for short losses but our user tests have shown that WaveNetEQ improves the video chat experience overall in a production setting. We investigated the impact of several modifications to the inference and training process and showed that they significantly improved the quality of the produced samples. In recent years the number of generative models for audio has increased immensely and it will be interesting to see if they can be used to further improve real time packet loss concealment in the future.

References

[1] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pages 2415–2424, 2018.

[2] David Goodman, G Lockhart, O Wasem, and Wai-Choong Wong. Waveform substitution techniques for recovering missing speech segments in packet voice communications. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(6):1440–1448, 1986.

[3] Colin Perkins, Orion Hodson, and Vicky Hardman. A survey of packet loss recovery techniques for streaming audio. *IEEE network*, 12(5):40–48, 1998.

[4] Bong-Ki Lee and Joon-Hyuk Chang. Packet loss concealment based on deep neural networks for digital speech transmission. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(2):378–387, 2015.

[5] Ya-Liang Chang, Kuan-Ying Lee, Po-Yu Wu, Hung-yi Lee, and Winston Hsu. Deep long audio inpainting. *arXiv preprint arXiv:1911.06476*, 2019.

[6] Reza Lotfidereshgi and Philippe Gournay. Speech prediction using an adaptive recurrent neural network with application to packet loss concealment. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5394–5398. IEEE, 2018.

[7] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[8] J. Chung, C . Gulcehre, K. Cho, , and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv e-prints*, 2014.

[9] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[10] Chandan KA Reddy, Ebrahim Beyrami, Jamie Pool, Ross Cutler, Sriram Srinivasan, and Johannes Gehrke. A scalable noisy speech dataset and online subjective test framework. *Proc. Interspeech 2019*, pages 1816–1820, 2019.

[11] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.