GRUs may be built one over the other and this time the output of one GRU is conveyed to the input of the next GRU. Another alternative could be to also connect the states between the cascaded recurrent units with mechanisms such an attention.

The relationships may be governed, for example, by formulas such as at least one of the following:

$$z_t = \sigma\left(W_z \cdot (h_{t-1}, x_t)\right)$$
$$r_t = \sigma\left(W_r \cdot (h_{t-1}, x_t)\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot (r_t * h_{t-1}, x_t)\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

where:

t refers to the time instant/step, and in case of unrolled GRUs, refers also to the particular module in the unrolled structure (e.g. t=0 is the first module, first time instant, t=1 the second, and so on);

x_t refers to the input vector of the recurrent module at instant t (e.g. to the frame at the time t, e.g. with or without the samples taken from the (e.g. immediately) preceding frame and/or with or without the samples taken from the (e.g. immediately) preceding frame);

h_t refers to the state and output at instant t of the recurrent unit, which will be inherited by the $(t+1)^{th}$ feedforward module in the unrolled case (with reference to Fig. 11, h_t is reintroduced in feedback as h_{t-1}, see below; with reference to Fig. 12, h_t is provided to the immediately subsequent feedforward module);

h_{t-1} refers to the state and output at time step t-1, which is the input of the unit at instant t. In case of unrolled GRU (Fig. 12), h_{t-1} is an input of t^{th} feedforward module (i.e., either the output of the immediately preceding recurrent module, or the input of the GRU) (if the t^{th} module is the first module, then h_{t-1} will be a default value);

$\widetilde{h_t}$ refers to a candidate state and/or output of the recurrent module;

z_t refers to an update gate vector;

r_t refers to a reset gate vector;

We report the computational complexity estimates in the following table.

Model	Encoder Complexity	Decoder Complexity
SSMGAN	0.05 GMACs	4.56 GMACs
SoundStream	5 GMACs	5 GMACs
NESC	0.5 GMACs	7 GMACs

Table 1: *Complexity Estimation*

Our implementation runs faster than real time on a single thread of an Intel(R) Core(TM) i7-6700 CPU at 3.40Ghz

3.3. Qualitative statistical analysis of the latent

We provide a qualitative analysis of the distribution of the latent in order to give a better understanding of its behaviour in practice. The quantized latent frames are embedded in a space of dimension 256 hence in order to plot their distribution we use their t-SNE projections. For each experiment we first encode 10 s of audio with different recording conditions and we label each frame depending on a priori information regarding its acoustic and linguistic characteristics. Each subplot represent a different set of audio randomly selected from the LibriTTS, VCTK and NTT Datasets. Afterwards we look for clusterings in the low dimensional projections. Notice that the model is not trained with any clustering objective, hence any such behaviour shown at inference time is an emergent aspect of the training set up.

We test both speaker characteristics, such as language and gender, and acoustic aspects like voicing and noisiness. In our first experiment (Fig. 2a) we test voicing information using a VAD algorithm to label each frame automatically. We notice a clear clustering of voiced, unvoiced and silent frames with the boundary consisting of transition frames. We similarly label voiced frames based on their quantized pitch values, but this shows no significant clustering behaviour. We do not show the picture because of lack of space.

Codec	POLQA	STOI	ViSQOL v3
OPUS 6 kbps	1.681	0.480	2.273
EVS 5.9 kbps	**3.308**	**0.553**	**3.036**
SSMGAN 1.6 kbps	2.213	0.536	2.505
NESC 1 kbps	1.534	0.612	2.109
NESC 2 kbps	2.382	0.641	2.615
NESC 3 kbps	**2.548**	**0.643**	**2.841**

Table 2: Average objective scores for neural decoders on the In-formalSet. For all metrics higher scores are better. Confidence intervals are negligible for POLQA and ViSQOL v3, while for STOI they are smaller than 0.025

Codec	POLQA	STOI	ViSQOL v3
OPUS 6 kbps	1.833	0.613	2.357
EVS 5.9 kbps	**3.486**	**0.736**	**3.071**
SSMGAN 1.6 kbps	2.267	0.647	2.476
NESC 1 kbps	1.659	0.745	2.074
NESC 2 kbps	2.492	0.802	2.595
NESC 3kbps	**2.707**	**0.817**	**2.822**

3.5. Subjective Evaluation

We test the model only on challenging unseen conditions in order to assess its robustness. For this we select a test set of speech samples from the NTT Dataset comprising unseen speakers, languages and recording conditions. In the test set "m" stands for male, "f" for female, "ar" for Arabic, "en" for English, "fr" for French, "ge" for German, "ko" for Korean, and "th" for Thai.

We also test the model on noisy speech. For this we select the same speech samples as for the clean speech test and apply a similar augmentation policy as in Section 3.1. We add ambient noise samples (e.g. airport noises, typing noises, ...) at SNR between 10 dB and 30 dB and then convolve with room impulse responses (RIR) coming from small, medium and big sized rectangular rooms. More precisely, "ar/f", "en/f", "fr/m", "ko/m", and "th/f" are convolved with RIRs from small rooms,

[19] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, “Neural discrete representation learning,” in Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 6309–6318.

[20] H. Zen, V. Dang, R. Clark, Y. Zhang, R. Weiss, Y. Jia, Z. Chen, and Y. Wu, “Libritts: A corpus derived from librispeech for text-to-speech,” arXiv preprint arXiv:1904.02882, 2019.

[21] C. Reddy, H. Dubey, V. Gopal, R. Cutler, S. Braun, H. Gamper, R. Aichner, and S. Srinivasan, “Icassp 2021 deep noise suppression challenge,” in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2021, pp. 6623–6627.

[22] D. Povey, “OpenSLR 28 Dataset,” https://www.openslr.org/28//.

[23] R. Yamamoto, E. Song, and J. Kim, “Parallel WaveGAN: A Fast Waveform Generation Model Based on Generative Adversarial Networks with Multi-Resolution Spectrogram,” in ICASSP 2020, IEEE International Conference on Acoustics, Speech and Signal Processing, 2020, pp. 6199–6203.

[24] K. Kumar, R. Kumar, de T. Boissiere, L. Gestin et al., “MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis,” in Advances in NeurIPS 32, 2019, pp. 14 910–14 921.

[25] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” ICLR, 2015.

[29] M. Chinen, F. S. Lim, J. Skoglund, N. Gureev, F. O'Gorman, and A. Hines, "ViSQOL v3: An open source production ready objective speech and audio metric," in *2020 twelfth international conference on quality of multimedia experience (QoMEX).* IEEE, 2020, pp. 1- 6.

[30] J. Beerends, C. Schmidmer, J. Berger, M, Obermann, R. Ullmann, J. Pomy, and M. Keyhl, "Perceptual Objective Listening Quality Assessment (POLQA), *the* third generation ITU-T standard for end-to-end speech quality measurement part I - temporal alignment," *Journal of the audio engineering society,* vol. 61, no. 6 , pp. 1116-184, june. 2013.

[31] C. H. Taal, R. C. Hendriks. R, Heusdens, and J. Jensen, "Algorithm for intelligibility prediction of time frequency weighted noisy speech," *IEEE Trans Audio Speech Lang. Process., vol,* pp.2125- 2 136. 201 I.

Further characterization of the figures

Figures 1b and 8: Neural End-2-End Speech Codec high level architecture.

Claims

1. Audio generator, configured to generate an audio signal from a bitstream, the bitstream representing the audio signal, the audio signal being subdivided in a sequence of frames, the audio generator comprising:

a first data provisioner configured to provide, for a given frame, first data derived from an input signal;

a first processing block, configured, for the given frame, to receive the first data and to output first output data in the given frame,

wherein the first processing block comprises:

at least one preconditioning learnable layer configured to receive the bitstream, or a processed version thereof, and, for the given frame, output target data representing the audio signal in the given frame;

at least one conditioning learnable layer configured, for the given frame, to process the target data to obtain conditioning feature parameters for the given frame; and

a styling element, configured to apply the conditioning feature parameters to the first data or normalized first data;

wherein the at least one preconditioning learnable layer includes at least one recurrent learnable layer.

2. The audio generator of claim 1, configured to obtain the audio signal from the first output data or a processed version of the first output data.

3. The audio generator of claim 1 or 2, wherein the first data have multiple channels, wherein the first output data comprise a plurality of channels, the audio generator also comprising a second processing block, configured, for the given frame, to receive, as second data, the first output data or data derived from the first output data, the output target data being with multiple channels and multiple samples for the given frame, wherein the second processing block is configured to combine the plurality of channels of the second data to obtain the audio signal.

4. The audio generator of any of the preceding claims, wherein the recurrent learnable layer includes at least one gated recurrent unit, GRU.

5. The audio generator of any of the preceding claims, wherein the recurrent learnable layer includes at least one long short term memory, LSTM, recurrent learnable layer.

6. The audio generator of any of the preceding claims, wherein the recurrent learnable layer is configured to generate the output, which is the target data or a predecessor thereof, for a given time instant by keeping into account the output and/or a state of a preceding time instant, wherein the relevance of the output and/or state of a preceding time instant is obtained by training.

7. The audio generator of any of the preceding claims, wherein the recurrent learnable layer operates along a series of time steps each having at least one state, in such a way that each time step is conditioned by the output and/or state of the preceding time step.

8. The audio generator of claim 7, further comprising a plurality of feedforward modules, each providing the state and/or output to the immediately subsequent module.

9. The audio generator of any of the preceding claims, wherein the recurrent learnable layer is configured to generate a state and/or output h_t for a particular t-th state or module by:

weighting a candidate state and/or output through an update gate vector z_t , to generate a first weighted addend; and

weighting the state and/or output h_{t-1} of the preceding time step through a vector which is complementary to 1 with the update gate vector z_t, to generate a second weighted addend; and

adding the first addend with the second addend.

10. The audio generator of any of claims7-9, wherein the recurrent learnable layer is configured to generate a state and/or output h_t by:

through reciprocally complementary weighting vectors, adding a weighted version of a candidate state and/or output with a weighted version of the state and/or output h_{t-1} of the preceding time step.

11. The audio generator of claim 9 or 10, wherein the recurrent learnable layer is configured to generate the candidate state and/or output by at least applying a weight parameter W, obtained by training, to:

an element-wise product between a reset gate vector r_t and the state and/or output h_{t-1} of the preceding time step, concatenated with the input x_t for the current time instant.

12. The audio generator of claim 11, further configured to apply an activation function after having applied the weight parameter W.

13. The audio generator of claim 12, wherein the activation function is TanH.

14. The audio generator of any of claims 10-13, wherein the recurrent learnable layer is configured to generate the candidate state and/or output by at least:

weighting, through weight parameter W obtained by training, a vector which is conditioned by both:

the input x_t for the current time instant and

the state and/or output h_{t-1} of the preceding time step weighted onto a reset gate vector r_t.

15. The audio generator of any of claims 9-14, wherein the recurrent learnable layer is configured to generate the update gate vector z_t by applying a parameter W_z to a concatenation of:

the input h_{t-1} of the recurrent module h_{t-1} concatenated with

the input x_t for the current time instant.

16. The audio generator of claim 15, configured, after having applied the parameter W_z, to apply an activation function.

17. The audio generator of claim 16, wherein the activation function is a sigmoid, σ.

18. The audio generator of any of claims 9-17, wherein the reset gate vector r_t is obtained by applying a weight parameter W_r to a concatenation of both:

the state and/or output h_{t-1} of the preceding time step and

the input x_t for the current time instant.

19. The audio generator of claim 18, configured, after having applied the parameter W_r, to apply an activation function.

20. The audio generator of claim 19, wherein the activation function is a sigmoid, σ.

21. Audio generator according to any of the preceding claims, comprising a quantization index converter configured to convert indexes of the bitstream onto codes.

22. The audio generator of claim 21, wherein the first data has a plurality of channels, the first output data comprises a plurality of channels, the target data being with multiple channels,

further comprising a second processing block configured, for the given frame, to receive, as second data, the first output data or data derived from the first output data, wherein the second processing block is configured to combine the plurality of channels of the first output data or of the second output data to obtain the audio signal.

23. Method for generating an audio signal from a bitstream, the bitstream representing the audio signal, the audio signal being subdivided in a sequence of frames, the method comprising:

providing, for a given frame, first data derived from an input signal;

through a first processing block, receiving the first data and outputting first output data in the given frame,

wherein the first processing block comprises:

at least one preconditioning learnable layer receiving the bitstream, or a processed version thereof, and, for the given frame, output target data representing the audio signal in the given frame;

at least one conditioning learnable layer processing, e.g. for the given frame, the target data to obtain conditioning feature parameters for the given frame; and

a styling element, applying the conditioning feature parameters to the first data or normalized first data;

wherein the at least one preconditioning learnable layer includes at least one recurrent learnable layer.

24. The method of claim 23, wherein the first data have multiple channels, wherein the first output data comprise a plurality of channels, the method also comprising through a second processing block receiving, as second data, the first output data or data derived from the first output data, wherein the second processing block combines the plurality of channels of the second data to obtain the audio signal.

25. The method of claim 23 or 24, further comprising obtaining the audio signal from the first output data or a processed version of the first output data.