Masked Language Model for audio

February, 2025

Mattia Paolacci

Abstract

In this work, I delved into sound discretization techniques, which allow treating the audio waveform as a sequence of codes, enabling the use of NLP approaches such as Masked Language Modeling.

1. Introduction

In recent years, the field of natural language processing (NLP) has advanced rapidly with the introduction of transformer-based models, like (Devlin et al., 2019), which excel at capturing complex contexts and bidirectional relationships in text. At the same time, audio signal processing has seen increased interest in discrete representations and quantized autoencoders, such as (van den Oord et al., 2017).

Project goal. This project aimed to adapt the architecture proposed in (van den Oord et al., 2017), called VQ-VAE, which originally worked with images, to handle audio input. I took the source code from (Zalando-Research) and used it as a starting point. Therefore, after training the VQ-VAE, I was able to use its encoder to map an audio waveform— a sequence of real values ranging from [-1, 1]—into a sequence of indices, where each index corresponds to a vector from the VQ-VAE codebook. Such a sequence of indices can be treated as a sequence of tokens which we can process using NLP techniques. Furthermore, I tried to train the VQ-VAE model proposed by (Dhariwal et al., 2020) and added logic to produce the encoded sequence and decode it back into an audio waveform. Lastly, the next step was to train BERT on such a masked sequence, following the same methodology used in (Devlin et al., 2019). After training, I observed how the model reconstructed parts of the sequence. The repo of this project is available here.

Email: Mattia Paolacci <paolacci.166444@di.uniroma1.it>.

Deep Learning and Applied AI 2024, Sapienza University of Rome, 2nd semester a.y. 2023/2024.

2. Related work

Noteworthy, in (Dhariwal et al., 2020), the authors used a hierarchical VQ-VAE to discretize the audio waveform, obtaining a sequence of discrete vectors used to learn a prior and generate samples, while (Parker et al., 2024) generates new outputs using a masked-LM employing an iterative process.

3. Method

In this section, we will examine the model architectures and techniques used in this project more deeply.

3.1. Sound discretization with VQ-VAE

The model consists of an *Encoder*, a *Decoder* and a *Quantization* module; we are going to focus on each of these.

The **Encoder** maps a sequence of values representing the waveform, we denote it by $x = \langle x_t \rangle_{t=1}^{T}$ in which $x_t \in [-1, 1]$, to a sequence of *latent vectors* denoted by $h = \langle \mathbf{h_s} \rangle_{s=1}^{S}$. Note that T (*resp.* S) is the length of the sequence and x_t (*resp.* x_s) is the t-th (*resp.* *s-th*) element of the sequence. The *Encoder* is composed of multiple *convolution* blocks, having *stride=2*, so the input gets halved through each block. In my implementation, I parametrized the number of *convolution blocks*, therefore it is possible to specify the number of conv. blocks with k obtaining $S = \frac{T}{2^k}$.

Concerning the *Quantization module*, it is composed of a **codebook**, which contains the embeddings that will be learnt. Such *codebook* represents the *discrete latent embedding space* that is $R^{K \times D}$, in which K is the number of the *embedding vectors* (or codes) and D is the dimensionality of these *embedding vectors*. The aim of the *Encoder* is to map a portion (whose length depends on k) of the input waveform to its *latent representation*, denoted by $\mathbf{h_s}$. Therefore, $\mathbf{h_s}$ gets mapped to the most similar *discrete embedding vector*, denoted by $\mathbf{e_j}$, by the bottleneck as shown in 1.

$$j = \text{argmin}_i \|\mathbf{h_s} - \mathbf{e_i}\| \tag{1}$$

Hence, let Q denote the *quantization operation*, then $Q(\mathbf{h_s}) \mapsto \mathbf{e_j}$. Therefore, by using the *Encoder* and the

Quantization module we can map an audio waveform to a sequence of discrete embeddings (actually their indices). We refer to that as *quantized audio sequence*.

Loss function. In my implementation of the VQ-VAE, I used the loss function defined in 2.

$$\mathcal{L} = \mathcal{L}_{recon} + \mathcal{L}_{codebook} + \beta\mathcal{L}_{commit} + \mathcal{L}_{spectral} \quad (2)$$

$$\mathcal{L}_{recon} = \frac{1}{T}\sum_{t=1}^{T} \|x_t - \hat{x_t}\|_2^2 \quad (3)$$

$$\mathcal{L}_{codebook} = \frac{1}{S}\sum_{s=1}^{S} \|\text{sg}[\mathbf{h_s}] - Q(\mathbf{h_s})\|_2^2 \quad (4)$$

$$\mathcal{L}_{commit} = \frac{1}{S}\sum_{s=1}^{S} \|\mathbf{h_s} - \text{sg}[Q(\mathbf{h_s})]\|_2^2 \quad (5)$$

$$\mathcal{L}_{spectral} = \||\text{STFT}(x)| - |\text{STFT}(\hat{x})|\|_2^2 \quad (6)$$

Eq.3 penalizes the distance between the input and the reconstructed output, but using it by itself the model only learns to reconstruct low frequencies, thus following (Dhariwal et al., 2020), I added Eq.6, which is the spectral loss, in which STFT stands for Short-Time Fourier transform, that helps to reconstruct the mid-to-high frequencies. Regarding Eq.4, it moves the *codebook embeddings* towards the encoder output, while Eq.5 ensures the encoder commits to an embedding; following (van den Oord et al., 2017) I used $\beta = 0.25$. The stop-gradient operator (sg) makes its operand treated as a constant during backpropagation, meaning its derivative is zero, and as a result, the operand does not contribute to the weight updates.

3.2. Masked Language Modeling

In this context we have a vocabulary of the codebook indices along with some special indices having the same meaning of MASK, PAD and SEP. The **masking procedure** is the same as the one defined in (Devlin et al., 2019), therefore, the training data generator chooses 15% of the index positions at random for prediction. If the i-th index is chosen, we replace the i-th index with (1) the MASK index 80% of the time (2) a random index 10% of the time (3) the unchanged i-th index 10% of the time.

Concerning the **generative task**, it is interesting to observe how *BERT* brings together two parts of a *quantized audio sequence*. Specifically, given a sequence of indices, the data generator masks its middle part, i.e. it masks the P central indices, and feeds that to *BERT*, which tries to reconstruct the masked part. Hence, given e_i, the i-th embedding from the *codebook*, we denote i_l as the *l-th* index in the *quantized audio sequence*. 7 describes the *BERT* masked input.

$$i_1, i_2, \ldots, \underbrace{i_{mask}, \ldots, i_{mask}}_{\text{P elements}}, \ldots, i_T \quad (7)$$

Concerning the **separation task**, its goal is to separate the stem of an instrument (e.g., guitar) of the audio mixture (i.e. the stems of all instruments together in the same waveform). To accomplish this task the training is carried out with the same *masking procedure* described previously, but also, the sequence to mask is the concatenation between the mixture sequence and the stem sequence, while the i_{sep} index separates them; 8 shows this. Note that i_j^{mix} denotes an index related to the mixture and i_j^{stem} denotes an index related to the stem (e.g. guitar or piano etc.).

$$i_1^{mix}, i_2^{mix}, \ldots, i_M^{mix}, i_{sep}, i_1^{stem}, \ldots, i_M^{stem} \quad (8)$$

Then, during inference, to make the model perform the separation task, we mask all the stem indices.

4. Experimental results

I performed the training of the models using Colab with NVIDIA Tesla T4. As a dataset, I used (Manilow et al., 2019) which contains mono-channel audio tracks, with a sample rate of 44.1kHz. Regarding my implementation of the VQ-VAE, I trained it for 8k epochs, about 8 hours. The codebook contains 512 codes each one of dimension 64. I observed that $k = 4$ doesn't lead to a noticeable quality loss in reconstruction. However, BERT was able to learn only the noise, since it has a window of 512 tokens, it can only "see" a chunk of $\frac{512\times2^4}{44.100} \approx 0.18$ sec of the audio. Due to this narrow window, it may not be able to model and capture long-range structures in the audio track, limiting itself to observing only very local features, which could have too high entropy. I tried to do better with the Jukebox VQ-VAE, trying to compress with a factor of 128x or 256x, but since it requires too many resources (GPU usage time, RAM etc.) to train, I could only train it for a few hours, resulting in a very poor reconstruction of the audio.

5. Conclusions and future works

Thanks to this project, I had the opportunity to gain an in-depth understanding of the functioning of one of the key models for discretizing continuous inputs, along with the associated challenges arising from compression. Additionally, I had the chance to experiment with whether it is possible to handle audio sequences encoded with one of the breakthrough models in the NLP field. As a next step, with the necessary resources available, it could be interesting to find a way to compress the input by a higher factor, such as 128x or 256x. This way, with a 512 token BERT window, it would be able to observe an audio chunk of $\frac{512\times2^8}{44.100} \approx 3sec.$

References

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for lan-

guage understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., and Sutskever, I. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

Manilow, E., Wichern, G., Seetharaman, P., and Le Roux, J. Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019.

Parker, J. D., Spijkervet, J., Kosta, K., Yesiler, F., Kuznetsov, B., Wang, J.-C., Avent, M., Chen, J., and Le, D. Stemgen: A music generation model that listens. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1116–1120. IEEE, 2024.

van den Oord, A., Vinyals, O., and kavukcuoglu, k. Neural discrete representation learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf.

Zalando-Research. URL https://github.com/zalandoresearch/pytorch-vq-vae/blob/master/vq-vae.ipynb. From Zalando Research GitHub: PyTorch implementation of VQ-VAE by Aäron van den Oord et al.