WAV2TOK: DEEP SEQUENCE TOKENIZER FOR AUDIO RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Search over audio sequences is a fundamental problem. Very efficient retrieval solutions exist for text sequences, which consist of discrete tokens chosen from a finite alphabet. These discrete tokens may correspond to words or characters depending on the problem. Audio sequences are composed of continuous-valued samples with a large sampling rate, making similarity search inefficient. This paper proposes Wav2Tok, a deep sequence tokenizer for audio that converts continuous-valued audio sequences to sequences of discrete tokens that are easier to retrieve via sequence queries. The only information available for training Wav2Tok is pairs of similar sequences, i.e., depending on how we form the pairs, the similarity semantics are learned. Wav2Tok compresses the query and target sequences into shorter sequences of tokens that are faster to match. Experiments show the consistent performance of Wav2Tok across various audio retrieval tasks, namely, music search (query by humming) and speech search via audio query.

1 INTRODUCTION

Sequence Retrieval aims at retrieving sequences similar to a query sequence, with the constraint that an ordered alignment exists between the query and the target sequence. In this paper, we address the following problem: Can we extract discrete tokens from any continuous signal for the purpose of retrieval of similar signals? This problem has deep connections with tasks such as child language acquisition, music cognition and learning languages without written forms. Some direct applications of the proposed task include speech search, where the order of constituent units, such as phonemes, syllables or words, remains same; and music search – query by humming or query by example – where the order of constituent units, such as relative notes or phrases, remains same. Apart from audio, the problem also extends to tasks such as handwritten word search and gesture search.

One can define similarity metrics over sequences using methods based on Dynamic Time Warping (DTW). These methods are inefficient if the sequences are continuous valued and have high sampling rates. Moreover, they depend on matching hand-made features, which are ineffective in the face of high variability of query sequences.

Problems such as *spoken term detection* involve detection of a query utterance in a long speech audio. The search space is huge, and performing DTW based search of query takes long time. A more efficient way of sequence retrieval is by mapping them to sequences of discrete tokens. Automatic speech recognition (ASR) can be employed for this purpose. However, ASR training requires knowledge of basic units of transcription. The popularly used units are phonemes and graphemes. This method thus becomes language dependent. Non-linguistic sounds, such as cough and sneeze, could be mapped to certain tokens defined for them. This approach could not be used when precise tokens are not defined, for example, for music search.

In *query by humming* based music search, audio is mapped to discrete melody-related tokens, such as notes, and these token sequence are matched for search. However, several music traditions do not have precise transcription systems. There, one can tell if two pieces, or motifs, are similar but cannot precisely transcribe them to tokens. The embellishments used in music could be too dynamic to be transcribed precisely. Moreover, when a musically untrained user sings a query, s/he cannot hit the right notes matching the target song. So the matching could rely on several factors other than notes, such as phonemes of lyrics, onset times (rhythm), and note transitions. Hence, the tokens to be used may not be derived from notes alone.

In this way, each tokenizer - for speech, music or other signals, in general - uses a domain-specific hand-made tokens defined by a domain expert. In this paper, we propose a tokenizer to map audio sequences to sequences of discrete tokens with an aim of retrieval. The mapping is learnt only from pairs of similar audio sequences. The tokens are not defined manually, but correspond to distinct semantic units learned from pairs of similar audio sequences. The method is general and can be applied to signals other than audio. In this paper, we apply the proposed method to speech and music audio search, for the problems of spoken term detection and query by humming, respectively.

The proposed method, named Wav2Tok, encodes audio via a BiLSTM neural network. The encoder generated representations are then mapped to discrete tokens via a K-means vector quantizer network. Each discrete token corresponds to a discrete representation in the vector quantizer's codebook.

Wav2Tok is trained with pairs of similar audio sequences in an self-supervised fashion without any transcription using a novel training algorithm. For each pair, we average the encoder generated representations, which map to the same token, by the K-means vector quantizer network to generate a prototype for that token. We then perform a contrastive learning task to increase similarity between the generated prototype for a particular token and the quantizer codebook discrete representation corresponding to the same token. A Connectionist temporal classification (CTC) [Graves et al. (2006)] based alignment is performed simultaneously to explicitly minimize the edit distance between the token sequences generated from each sequence in the pair.

We compare Wav2Tok to state of the art methods for discrete representation learning , such as Wav2Vec 2.0, and state of the art ASR models finetuned to perform phonetic tokenization. We evaluate the generalization capability of the tokens generated by the models on search experiments, namely, query-by-humming and spoken term detection. Wav2Tok outperforms the baselines in performance and uses much lesser trainable parameters, ensuring faster inference and deployment.

2 Related Work

Sequence Labelling. With expert-defined tokens, various methods are popularly used for mapping sequences to tokens. In conventional methods, Hidden Markov Models [Rabiner & Juang (1986)] and Conditional Random Fields [Lafferty et al. (2001)] have been popularly used for sequence labelling. These methods involve a significant amount of domain knowledge and many assumptions to make tractable models, which are avoided by End-to-End learning models such as Recurrent Neural Networks (RNNs) using Connectionist Temporal Classification framework [Graves et al. (2006)]. Sequence labeling can be used for sequence retrieval by converting the sequences to tokens, which are easy to search over. But this approach inevitably depends upon expert-defined tokens.

Unsupervised Speech Representation Learning. Automatic Speech Recognition systems are pretrained on large amounts of untranscribed speech data to generate state-of-the-art continuous representations which encodes the slowly varying phoneme features in raw speech. The representations are then mapped to phoneme tokens via Connectionist Temporal Classification (CTC) [Graves et al. (2006)] finetuning on small amount of transcribed audio. Works like Contrastive Predictive Coding (CPC) [van den Oord et al. (2018)], Autoregressive Predictive Coding (APC) [Chung & Glass (2020)], and Wav2Vec [Schneider et al. (2019)] generate continuous representations with powerful autoregressive models pretrained to predict future time-step representations.Further works started discretizing the continuous representations with VQ-VAE [van den Oord et al. (2017)] to generate discrete representations for speech. Works like VQ-Wav2Vec [Baevski et al. (2019)] and VQ-APC [Chung et al. (2020)] discretize the representations and perform the same prediction tasks as in Wav2Vec [Schneider et al. (2019)] and APC [Chung & Glass (2020)] respectively but over discrete representations. In VQ-Wav2Vec, the discrete representations are generated with either a K-Means Vector Quantizers [Baevski et al. (2019)] or Gumbel-Softmax based Vector Quantizers [Baevski et al. (2019)]. The learnt discrete representations are used to pretrain a BERT [Devlin et al. (2018)] to generate stronger continuous representations much in correspondence to BERT pretraining in Natural Language Processing. Wav2Vec 2.0 [Baevski et al. (2020)] generates strong discrete representations via a Gumble Softmax based Vector Quantizer [Baevski et al. (2019)] by pretraining huge transformer networks in identifying the correct discrete representation at a masked time step corresponding to the encoder generated latent representation at that time step. The main motivation of these methods for discretizing raw audio in a latent space was to explicitly control information con-

tent encoded in the representations rather than achieve phonetic tokenization. The codes or tokens generated by the models aren't constrained to be interpretable and hence initialised in large numbers. Optimization of the pretraining loss function converges a subset of these codes or tokens to represent acoustic units. CTC based finetuning over transcribed audio groups these acoustic units to K distinct phonemes (linguistic units) as present in the transcriptions. HuBERT [Hsu et al. (2021)] learns continuous representations which are generated with huge transformer networks by utilizing a few iterations of offline clustering to provide aligned target labels for a BERT-like prediction loss. The clustering is performed with an ensemble of K-means clusterers with $K = \{100, 500\}$ clusters on transformer-generated latent representations. Wav2Vec-Unsupervised [Baevski et al. (2021)] presents an approach for completely unsupervised speech recognition. Representations generated by a pretrained Wav2Vec 2.0 model are mapped to phoneme sequences using a generator network. A discriminator then identifies if the phoneme sequence is real or fake based on the knowledge extracted from unlabeled text data which are also converted to phoneme sequences.

Note, all aforementioned approaches generate representations for speech using industry-grade models with large number of parameters pretrained on large amount of unlabeled audio and finetuned on transcribed audio. Our work aims to perform phoneme level tokenization of input audio using models having a minute number of parameters compared to the aforementioned models. Our learning approach can generate semantic tokens which outperform phonetic tokens generated by the baseline models with large number of parameters while training pairwise on small amount of unlabeled audio data.

Audio Representations for Retrieval. Now Playing [Arcas et al. (2017)] and [Chang et al. (2020)] use a Neural Network Fingerprinter (NNFP) module outputting representations which are efficient for search in query-by-example tasks where the difference between query and actual song is pretty minute in comparison to humming where only the melody is sung. Now Playing [Arcas et al. (2017)] trains representations by optimizing the Triplet loss [Schroff et al. (2015)] and [Chang et al. (2020)] trains representations by simulating the Maximum Inner Product Search on mini-batches of representations. For Query by Humming task, [Mostafa et al. (2016)] and [Mostafa & Fung (2017)] use deep learning models like DNNs and CNNs to generate representations which they map to midi-numbers or note tokens. Such works require note transcribed data to train the models. For Spoken Term Detection task, approaches like [Zhang & Glass (2009)], [Rodriguez-Fuentes et al. (2014)], [Lee et al. (2015)], [Ram et al. (2018)] convert audio to sequences of feature vectors and apply different variations of DTW based template matching to detect query in long utterances of speech which is time-consuming.

Cross Domain Alignment. Given a pair of semantically similar inputs for training, tasks such as visual question answering (text and image) and machine translation (text) involve learning of an alignment. The alignment here is not ordered and the inputs may be from different modalities. Attention models have been used to find alignment between output entities and input regions [Yao et al. (2018)]. [Chen et al. (2020)] use Gromov-Wasserstein distance between output and input entities to match them. However, there is no notion of tokens there, rather the salient entities in the input are represented as vectors in a graph.

Graph Matching. Graph Neural Networks [Gori et al. (2005)] are used to generate embeddings for graphs. These embeddings are used to perform graph matching to find similarity of structured graphs [Li et al. (2019)]. However, they perform the matching jointly on the pair of inputs, rather than representing each input independently. This makes them unsuitable for the search problem at hand due to large run-time complexity. The distance metrics used for graph matching are based on edit distance [Li et al. (2019)] and Wasserstein distance [Chen et al. (2020)].

3 Problem Statement

We aim to map $\mathcal{X}$, a sequence of vectors, to $\widetilde{\mathcal{T}}$, a sequence of discrete tokens from a finite alphabet $\mathbb{A}$, such that the similarity of sequences is preserved in the sense of edit distance. The length of sequence $\widetilde{\mathcal{T}}$ may be less than or equal to that of the sequence $\mathcal{X}$. In other words, given a pair of similar sequences $(\mathcal{X}_i, \mathcal{X}_j)$ and sequence $\mathcal{X}_k$ which is not similar to either sequences in the pair, we want to map them to token sequences such that $ED(\widetilde{\mathcal{T}}_i, \widetilde{\mathcal{T}}_j)$ should be less than $\min\{ED(\widetilde{\mathcal{T}}_i, \widetilde{\mathcal{T}}_k), ED(\widetilde{\mathcal{T}}_j, \widetilde{\mathcal{T}}_k)\}$, where $ED(\cdot, \cdot)$ is the edit distance between two sequences.

4 MODEL ARCHITECTURE

Wav2Tok is comprised of an encoder $f : \mathbb{X} \mapsto \mathbb{Z}$ which takes as input a temporal sequence of audio features $\mathcal{X} = [\mathbf{x}_t \in R^n; t = 1, ..., T]$ of length T, where $\mathbf{x}_t$ is the feature vector at time step t, and outputs a sequence of L-2 normalised latent representations $\mathcal{Z} = [\mathbf{z}_t = f(\mathbf{x}_t) \in R^m; t = 1, ..., T]$. The encoder network is implemented as a 2-layer BiLSTM [Schuster & Paliwal (1997)] with 3.6 million trainable parameters followed by a L-2 normalization layer. The BiLSTM network generates latent representations which summarise information in both directions, and hence, encodes surrounding context.

A K-means vector quantizer network $g : \mathbb{Z} \mapsto \mathbb{T}$ then labels sequence $\mathcal{Z}$ at each time-step with tokens belonging to a finite K-element alphabet $\mathbb{A} = \{1, ..., K\}$ and generates sequence of tokens $\mathcal{T} = [\tau_t = g(\mathbf{z}_t) \in \mathbb{A}; t = 1, ..., T]$. Network g vector quantizes input $\mathbf{z}_t$ with a codebook $E = \{\mathbf{e}_k \in \mathbb{Z}; k = 1, ..., K\}$ comprised of $|\mathbb{A}| = K$ discrete representations which are cluster centroids in representation space $\mathbb{Z}$ and outputs token $\tau_t = \arg\max_k \mathbf{z}_t \cdot \mathbf{e}_k$. Note, here the dot product gives a cosine similarity score since both the vectors are L-2 normalised, as a result, $\mathbf{e}_k \in E$ closest to $\mathbf{z}_t$ is chosen as it's discrete representation and index k as it's token τ_t. The K discrete representations in network g are trainable parameters.

A compressor $\mathcal{C}$ compresses sequence of tokens $\mathcal{T}$ to sequence $\widetilde{\mathcal{T}}$ of length $\widetilde{T} \leq T$ by deleting all consecutive repetitions of tokens. $\mathcal{C}$ also generates the corresponding compressed sequence $\widetilde{\mathcal{Z}}$ of length $\widetilde{T}$ by averaging representations $\mathbf{z}_t \in \mathcal{Z}$ over the consecutive tokens and L-2 normalising the averaged representation. Figure 1 presents an illustration demonstrating our model architecture.

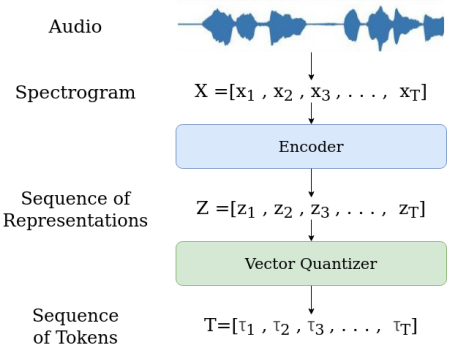

Figure 1: Model Architecture. Audio is converted to spectrogram $\mathcal{X}$ and passed as input to Wav2Tok. An encoder block generates a sequence of representations $\mathcal{Z}$ of length T. A K-means Vector Quantizer then maps each representation in the sequence to a token to generate a sequence of tokens $\mathcal{T}$. Compressor $\mathcal{C}$ compresses $\mathcal{T}$ to sequence $\widetilde{\mathcal{T}}$.

5 TRAINING

Wav2Tok is trained on pairs of sequences of audio features $(\mathcal{X}, \mathcal{X}')$ where the raw audio corresponding to $\mathcal{X}'$ is an augmented replica of that corresponding to $\mathcal{X}$. We apply either pitch shift or time stretch or both augmentations to raw audio to generate it's augmented replica. $\mathcal{X}$ and $\mathcal{X}'$ may differ in sources as well, i.e. a different person may sing the recording corresponding to $\mathcal{X}'$.

Given $(\mathcal{X}, \mathcal{X}')$, encoder f generates sequence of latent representations $\mathcal{Z}$ from input $\mathcal{X}$ and $\mathcal{Z}'$ from $\mathcal{X}'$. Quantizer g generates sequence of tokens $\mathcal{T}$ from input $\mathcal{Z}$ and $\mathcal{T}'$ from $\mathcal{Z}'$. Compressor $\mathcal{C}$ compresses sequence of tokens $\mathcal{T}$ to sequence $\widetilde{\mathcal{T}}$ and $\mathcal{T}'$ to $\widetilde{\mathcal{T}'}$.

We average all encoder generated representations in pair $(\mathcal{Z}, \mathcal{Z}')$ which map to the same token, say τ, to generate a prototype for τ. We then perform a contrastive task where we compare the prototype with each of of the K discrete representations in codebook E and increase it's similarity with the discrete representation corresponding to τ. We also align $(\mathcal{X}, \mathcal{X}')$ via a CTC based framework to constrain Wav2Tok to map both sequences to the same token sequence.

Our loss function is defined as,

$$\mathcal{L} = \mathcal{L}_m + \alpha \cdot \mathcal{L}_{CTC} \tag{1}$$

where $\mathcal{L}_m$ is the loss defined for the contrastive task, $\mathcal{L}_{CTC}$ is the alignment loss, and α is a positive constant.

Contrastive Loss. Let the set of $K' \leq K$ unique tokens occurring in pair $(\widetilde{\mathcal{T}}, \widetilde{\mathcal{T}'})$ be $\mathcal{U} = \{\tau_i; i = 1, .., K'\}$. We generate a list of token prototypes $\mathcal{P} = \{\mathbf{p}_\tau; \tau \in \mathcal{U}\}$ where $\mathbf{p}_\tau$ is L-2 normalised mean of representations in $\{\mathbf{z} \in \{\mathcal{Z}; \mathcal{Z}'\} : g(\mathbf{z}) = \tau\}$. Figure 2 presents an illustration demonstrating how we generate list of token prototypes $\mathcal{P}$.

Given $\mathbf{p}_\tau \in \mathcal{P}$, we perform a contrastive task to increase its similarity with discrete representation $\mathbf{e}_\tau \in E_{tok}$. To compare $\mathbf{p}_\tau$ with the codebook, metrics such as cosine similarity and Euclidean distance could be used. However, we find that using the following parameterized score for this purpose gives better performance,

$$\mathbf{s}_{\tau,k} = \sigma(W \cdot (\mathbf{p}_\tau - \mathbf{e}_k)) \in [0, 1] \tag{2}$$

where $\sigma(.)$ is the sigmoid function generating a score in the range $[0, 1]$ and $W \in R^{1\times d}$ is a parameter matrix. $\mathbf{s}_{\tau,k}$ acts as a parameterized similarity score between $\mathbf{p}_\tau$ and discrete representation $\mathbf{e}_k \in E$. We define our contrastive loss $\mathcal{L}_m$ as,

$$\mathcal{L}_m = -\sum_{\tau \in \mathcal{U}} \log \frac{\exp(\mathbf{s}_{\tau,\tau})}{\sum_{k=1}^{K} \exp(\mathbf{s}_{\tau,k})} \tag{3}$$

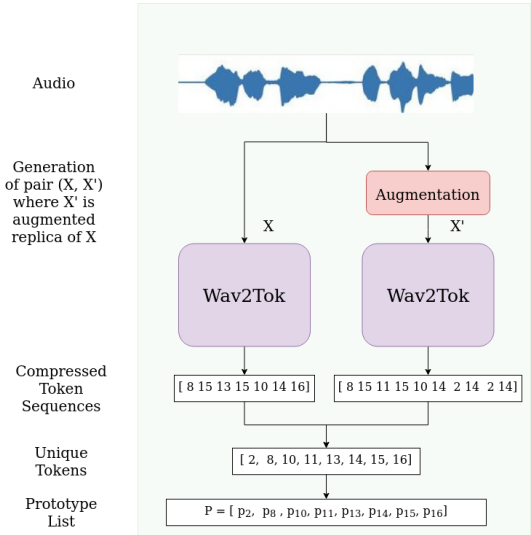

Figure 2: Generation of list of token prototypes $\mathcal{P}$. Input audio $\mathcal{X}$ is augmented in the Augmentation block to generate augmented replica $\mathcal{X}'$. The augmentation block may also output $\mathcal{X}'$ as the same audio corresponding to $\mathcal{X}$ but with different source i.e. uttered by a different speaker. Wav2Tok maps $\mathcal{X}$ and $\mathcal{X}'$ to compressed sequences of tokens $\widetilde{\mathcal{T}}$ and $\widetilde{\mathcal{T}'}$ respectively. We generate a token prototype for each of the unique tokens in $\widetilde{\mathcal{T}}$ and $\widetilde{\mathcal{T}'}$ by averaging all representations generated by Wav2Tok encoder network from inputs $\mathcal{X}$ and $\mathcal{X}'$ mapping to the same token. This results in a list of token prototypes $\mathcal{P}$ which we use to perform our contrastive task.

Alignment Loss. We align sequence $\mathcal{Z}$ generated from $\mathcal{X}$ with token sequence $\widetilde{\mathcal{T}'}$, which corresponds to sequence $\mathcal{Z}'$ generated from $\mathcal{X}'$, via the CTC framework. The loss is defined as ,

$$\mathcal{L}_{CTC} = -\log Prob(\widetilde{\mathcal{T}'}|\mathcal{Z}) \tag{4}$$

This loss explicitly minimizes the edit distance between $\widetilde{\mathcal{T}}$ and $\widetilde{\mathcal{T}'}$. It puts a constraint on Wav2Tok to generate the same token sequence for both $\mathcal{X}$ and $\mathcal{X}'$.

Clustering. We cluster on a subset of encoder generated representations during initialisation of our network and then at regular intervals during training to set the discrete representations in our K-means Vector Quantizer network g. Initialising the clusters in this way prevents Wav2Tok from converging to a local optimum during the matching task, as is the case we found with random initialization of centroids. The intermittent clustering during training prevents Wav2Tok from collapsing a single token output for every input.

We optimize a K-Means loss function to estimate optimal cluster centroids which is given as,

$$\mathcal{L}_{Kmeans} = ||sg(z) - q|| \tag{5}$$

where $sg(x) \equiv x, \frac{d}{dx} sg(x) \equiv 0$ is the stop-gradient operator, z is an encoder generated latent representation and q is the discrete representation in network g closest to z. The stop gradient operator freezes the encoder f parameters from getting trained during clustering.

We train Wav2Tok using the ADAM [Kingma & Ba (2017)] optimizer and a linear learning schedule with a learning rate of 0.001 and 8% of the training steps as warm-up steps.

6 EXPERIMENTS

We test the performance of tokens and encoder generated continuous representations of Wav2Tok in audio retrieval. We perform Query by Humming (QbH) and Spoken Term Detection experiments to evaluate performance of Wav2Tok in comparison to the baselines.

6.1 MUSIC MELODY SEARCH: QUERY BY HUMMING

Task. Given a test query audio, we are to find in the search audio database the audio with the most similar melody.

Experiment Details. We use the MIR-QbSH dataset which is composed of 4431 humming audio recordings of 30 second duration corresponding to 48 songs. Each song is sung by several individuals. Not all individuals sing the same part of the song. The recordings have variation in the environments they were recorded in, tonal qualities, voices, pitch and time stretch. We train our models on audios corresponding to 40 songs in MIR-QbSH dataset and evaluate search performance on audios corresponding to the remaining 8 songs. The training dataset is composed of 1970 audio recordings for training and 676 audio recordings for validation. The test dataset is composed of 225 audio recordings as search database and 659 query audio recordings. We evaluate the performance of our models in identifying which song a given query corresponds to via comparison with all sequences in search database. All the audios in our test dataset are converted to sequences of tokens or representations by each of the models. Each query sequence is compared to all sequences in search database via Edit Distance (ED) based similarity score (if tokens) or DTW based similarity score (if representations). The song id of the most similar sequence in the search database is then selected as query song id. We calculate Mean Reciprocal Ranking (MRR) score with the ground truth song id of the queries to evaluate our models. The Reciprocal Ranking (RR) score is given as $1/r$ if the rth most similar sequence in search database has the same song id as query.

All the audio recordings are converted to Short Term Fourier Transform (STFT) matrices before being passed as inputs to our models. The STFT matrices are computed with 513 frequency bins, a window length of 1024 samples (summarising 128 ms of audio), and hop length of 512 samples.

6.2 SPOKEN TERM DETECTION

Task. Given a test query audio, we are to detect it's occurrence in a long utterance.

Experiment Details. We use the TIMIT dataset which is composed of 6300 utterances of English speech with time aligned word transcriptions. We form a dataset of multiple utterances of 59 keywords sampled from TIMIT. The models are trained on utterances of random sentences formed with a subset of 25 keywords and evaluated on Spoken Term Detection experiments conducted with the remaining 34 keywords as queries. The test dataset is composed of 337 utterances corresponding to the 34 query keywords and 100 long utterances per query keyword with half of them containing a single occurrence of query and other half containing no occurrence of query. Given a query and a long utterance, we convert both to sequences of tokens using the each of the audio tokenizers. We perform approximate string matching to detect the occurrence of query token sequence in the utterance token sequence. The STFT matrix inputs to the models are computed with 185 frequency bins, a window length of 368 samples (summarising 23 ms of audio), and hop length of 92 samples.

6.3 BASELINES

Triplet. We train encoder $f : \mathbb{X} \mapsto \mathbb{Z}$ to generate L-2 normalised continuous representations for audio retrieval. Encoder f is trained via optimizing the Triplet Loss [Schroff et al. (2015)] as done in training neural audio fingerprinter network in Now Playing [Arcas et al. (2017)]. Given pair of similar sequences $(\mathcal{X}, \mathcal{X}')$, encoder f generates sequences $\mathcal{Z}$ and $\mathcal{Z}'$. We form a mini-batch of size N of triplets $\{\mathbf{z}, \mathbf{z}^+, \mathbf{z}^-\}$ where representation $\mathbf{z}$ is sampled from sequence $\mathcal{Z}$, $\mathbf{z}^+$ and $\mathbf{z}^-$ are positive and negative samples respectively for $\mathbf{z}$ sampled from sequence $\mathcal{Z}'$. The loss is defined as, $\mathcal{L}_{Triplet} = \sum_{i=1}^{N} max\{||\mathbf{z}_i - \mathbf{z}_i^+|| - ||\mathbf{z}_i - \mathbf{z}_i^-|| + m, 0\}$, where m is a margin for similarity.

MIPS. We train encoder $f : \mathbb{X} \mapsto \mathbb{Z}$ to generate L-2 normalised continuous representations for audio retrieval. Encoder f is trained via simulation of Maximum Inner Product Search [Muss-

mann & Ermon (2016)] on mini-batches of representations as proposed by [Chang et al. (2020)]. Given pair of similar sequences $(\mathcal{X}, \mathcal{X}')$, encoder f generates sequences $\mathcal{Z}$ and $\mathcal{Z}'$. We form a mini-batch of size N of pairs of $\{\mathbf{z}, \mathbf{z}^+\}$ where encoder generated representation $\mathbf{z}$ is sampled from sequence $\mathcal{Z}$ and $\mathbf{z}^+$ is a positive for $\mathbf{z}$ sampled from $\mathcal{Z}'$. The loss is defined as, $\mathcal{L}_{MIPS} = -\sum_{i=1}^{N} \log \frac{\exp(\mathbf{z}_i, \mathbf{z}_i^+)}{\sum_{j \neq i}(\exp(\mathbf{z}_i \cdot \mathbf{z}_j^+) + \exp(\mathbf{z}_i \cdot \mathbf{z}_j))}$.

Wav2Vec2. We train our audio tokenizer via Wav2Vec 2.0 [Baevski et al. (2020)] learning framework. Quantizer g in our audio tokenizer is chosen to be a Gumbel Softmax based Vector Quantizer (See Appendix B for details) as used in [Baevski et al. (2020)]. Given sequence $\mathcal{X}$, encoder f outputs sequence of L-2 normalised representations $\mathcal{Z}$ of length T. Quantizer g outputs sequence of discrete representations $\mathcal{Q} = \{q_t = g(z_t \in \mathcal{Z}); t = 1, ..., T\}$. During training, each $q_t \in \mathcal{Q}$ is a Gumbel Softmax combination of the K codebook members in quantizer g codebook E. In inference, q_t is chosen as one of the codebook members in E from which we generate sequence of discrete tokens $\mathcal{T}$ with the codebook indices of each discrete representation in sequence $\mathcal{Q}$. A transformer network $h : \mathbb{Z} \mapsto \mathbb{O}$ is used to generate sequence of contextualised representations $\mathcal{O} = \{\mathbf{o}_t = h(z_t \in \mathcal{Z}); t = 1, ..., T\}$. For masked time step t, contextualised representation $\mathbf{o}_t$ aggregates the set of representations $\{\mathbf{o}_{\leq t} \in \mathcal{O}\}$. We use $\mathbf{o}_t$ to identify the true discrete representation $\mathbf{q}_t$ for $\mathbf{z}_t$ from a set $\mathcal{D}_t$ composed of $\mathbf{q}_t$ and D discrete representations sampled from other time steps. The loss is defined as, $\mathcal{L}_{Wav2Vec2} = -\log \frac{\exp(Sim(\mathbf{o}_t, \mathbf{q}_t))}{\sum_{\tilde{q} \in \mathcal{D}_t} \exp(Sim(\mathbf{o}_t, \tilde{q}))} + \mathcal{L}_d$, where $Sim(a, b) = \frac{a^T b}{||a|| ||b||}$ is the cosine similarity function, and $\mathcal{L}_d$ is a codebook diversity loss.

Wav2Vec2P. We train the same audio tokenizer used in aforementioned baseline Wav2Vec with our variation of Wav2Vec 2.0 [Baevski et al. (2020)] learning framework which learns discrete representations from pairs of similar sequences. Given pair $(\mathcal{X}, \mathcal{X}')$, encoder f outputs sequences $\mathcal{Z}$ of length T and $\mathcal{Z}'$ of length T' respectively. Assuming $T \leq T'$, we generate sequence $\mathcal{Z}^+$ of length T whose t time step element $\mathbf{z}_t^+$ is a positive for $\mathbf{z}_t \in \mathcal{Z}$ sampled from sequence $\mathcal{Z}'$. Gumbel Softmax based Vector Quantizer g quantizes each representation in sequence $\mathcal{Z}^+$ to generate sequence $\mathcal{Q}^+$. Transformer h inputs $\mathcal{Z}$ and generates sequence $\mathcal{O}$. For masked time-step t, we use $\mathbf{o}_t \in \mathcal{O}$ to identify the true discrete representation $\mathbf{q}_t^+ \in \mathcal{Q}^+$ for $\mathbf{z}_t^+ \in \mathcal{Z}^+$ from a set $\mathcal{D}_t^+$ composed of $\mathbf{q}_t^+$ and D discrete representations sampled from other time steps in $\mathcal{Q}^+$. The loss is defined as, $\mathcal{L}_{Wav2Vec2P} = -\log \frac{\exp(Sim(\mathbf{o}_t, \mathbf{q}_t^+))}{\sum_{\tilde{q} \in \mathcal{D}_t^+} \exp(Sim(\mathbf{o}_t, \tilde{q}))} + \mathcal{L}_d$.

Wav2Vec2-O. The original Wav2Vec 2.0 base model with 12 Transformer blocks and 95 million training parameters as proposed by [Baevski et al. (2020)]. It is pretrained on 960 hours of LibriSpeech data and fine tuned on TIMIT dataset. It uses $K = 32$ tokens for tokenization.

Wav2Vec2-Multi. A Wav2Vec 2.0 large model with 24 Transformer blocks and 317 million training parameters pretrained on 53 languages as proposed by [Conneau et al. (2020)]. It is finetuned on Common Voice Dataset to detect all possible phonemes in training languages to transcribe unseen languages with $K = 392$ tokens.

Triplet and MIPS use a 2-layer BiLSTM as encoder with 3.6 million trainable parameters. We use the LAMB optimizer [You et al. (2020)] and a Cosine Annealing Learning Schedule [Loshchilov & Hutter (2017)] with a learning rate restart of 0.0001 to train them. Wav2Vec2 and Wav2Vec2P use a 2-layer BiLSTM encoder with 3.6 million trainable parameters to generate latent representations and 3 Transformer blocks with 8.5 million trainable parameters to generate contextualized representations amounting to models with 12.1 million trainable parameters. Both are trained using the ADAM [Kingma & Ba (2017)] optimizer and a linear learning schedule with a learning rate of 0.001 and 8% of the training steps as warm-up steps. In comparison to the baselines, Wav2Tok uses only a 2-layer BiLSTM as encoder with 3.6 million trainable parameters.

7 Results

7.1 Music Melody Search: Query by Humming

We present search performances for 3 settings of query namely- Query with no augmentation or Vanilla Query (V), Time Stretched Query (TS), and Pitch Shifted Query (PS). Time stretch and pitch shift are the most common augmentations that may be faced in query by humming data. No

augmentations were applied to audio in search database. Evaluations are performed on sequences corresponding to songs not seen during training. The results present generalizability of the tokens or representations generated by the models. We set the number of tokens as $K = 25$ for Wav2Tok , Wav2Vec2, and Wav2Vec2P (See Appendix A.2 for experiments to support our choices).

Quality of Tokenization. Table 1 presents the performance of the sequence of tokens $\widetilde{\mathcal{T}}$ generated by the audio tokenizers on ED based similarity search. Tokens generated by Wav2Tok present good generalization capabilities in terms of MRR and outperforms all the baselines. It generates time and pitch invariant tokens as we see no drop in performance when either augmentation is applied to query. Wav2Vec2-O is pretrained and finetuned on english speech only. The tokens generated by it do not contain much melodic information but are robust to augmentations. The multilingual pretraining of Wav2Vec2-Multi and finetuning to all possible phonemes in training languages infuse both melodic and phonetic information to its tokens and hence gives good performance. Wav2Tok outperforms both Wav2Vec2-O and Wav2Vec2-Multi given its pairwise training which allows it to infuse more melodic information to the tokens while also being trained on a small amount of unlabelled data. The gumbel softmax based quantizer in Wav2Vec2 and Wav2Vec2P isn't ideal for infusing melodic information to tokens but it does infuse phonetic information as will be seen in Section 7.2. We compare the tokens with representations learnt by neural audio fingerprinting baselines, MIPS and Triplet evaluated on standard DTW based similarity search. The continuous representations present sub-par generalization to unseen songs.

We present the performances of the uncompressed sequences $\mathcal{T}$ and $\mathcal{Z}$ and compressed sequence $\widetilde{\mathcal{Z}}$ generated by the audio tokenizers in Appendix A.1. We observe a drop in performance for all audio tokenizers when we apply sequence compression to sequences $\mathcal{T}$ and $\mathcal{Z}$. Wav2Tok outperforms all the baselines and generates superior quality of all aforementioned sequences.

Search Time. Table 1 presents the search time taken for similarity search over the tokens or representations generated by the models. The search time taken per query is 2 order of magnitude lesser for ED based Search over compressed sequence of tokens $\widetilde{\mathcal{T}}$ than standard DTW based Search over continuous representations $\mathcal{Z}$. The pretrained models being finetuned on transcribed audio give the best quality of tokenization in terms of compression and search time. Wav2Tok gives comparable quality of tokenization and outperforms the pretrained models in inference time.

Table 1: Quality of Tokenization: MRR scores for query by humming

Model	V	TS	PS	Search Time	Inference
Triplet DTW	0.5	0.48	0.5	3.5s	0.1s
MIPS DTW	0.6	0.55	0.58		
Wav2Vec2 ED	0.66	0.63	0.64	0.06s	0.17s
Wav2Vec2P ED	0.69	0.65	0.67		
Wav2Vec2-O ED	0.72	0.72	0.71	0.01s	0.43s
Wav2Vec2-Multi ED	0.82	0.82	0.82		1.2s
Wav2Tok ED	**0.84**	**0.84**	**0.84**	0.04s	0.14s

Table 2: Ablation Studies and Some Variations: MRR scores for query by humming, K = 25

Model	V	TS	PS
Wav2Tok+NoSim ED	0.73	0.73	0.72
Wav2Tok+NoRel ED	0.79	0.76	0.77
Wav2Tok+NewInit ED	0.77	0.76	0.78
Wav2Tok+Trans ED	0.7	0.66	0.67
Wav2Tok ED	**0.84**	**0.84**	**0.84**

Ablation Studies. Query by humming involves similarity based on melody information, which is carried by the semantic pairing of the audio in training data. We constrain this pairing to include sequences not semantically similar and call this model Wav2Tok+NoSim. The results are shown in Table 2. There is a significant drop in robustness and performance of the tokens generated by Wav2Tok+NoSim in comparison to Wav2Tok but the representations suffer a small drop. Hence, although the representation space may be well clustered, Wav2Tok is able to add more semantics to the tokens as it is being trained with pairs of similar sequences in comparison to Wav2Tok+NoSim.

We train Wav2Tok with cosine similarity scores instead of a parameterized score and call this model Wav2Tok+Cos. The significant drop in performance of both representations and tokens generated by Wav2Tok+Cos in comparison to Wav2Tok validates the enhancement brought about by using a parameterized score.

Some Variations. In Wav2Tok+Trans, we use 3 Transformer blocks amounting to 8.5 million parameters instead of a 3.6 million parameter BiLSTM as our encoder in Wav2Tok architecture. There is an overall drop in performance and robustness of the tokens generated by Wav2Tok+Trans in-comparison to Wav2Tok as shown in Table 2. Possible cause for such drop may be overfitting.

In Wav2Tok+NewInit, instead of initialising the discrete representations in our tokenizer network g as K cluster centroids in the space $\mathbb{Z}$ generated by a freshly initialised encoder network f, we asso-

ciate the discrete representations with K centroids in the input space $\mathbb{X}$. Such association does not initialize the discrete representations of our tokenizer with optimal centroids which cluster the space $\mathbb{Z}$ perfectly. This results in significant drop in performance and robustness of tokens in comparison to Wav2Tok as shown in Table 2.

7.2 Spoken Term Detection

Quality of Tokenization. Table 3 presents the quality of tokezination of the query keywords by the models evaluated in the Spoken Term Detection experiments. We present the performances of Wav2Vec2 , Wav2Vec2P, Wav2Vec2-O, Wav2Vec2-Multi and proposed Wav2Tok. We conduct search experiments on a test dataset composed of a search database of 337 utterances of the 34 keywords used as queries in the Spoken Term Detection experiments and 1289 query utterances. We identify the keyword to which each query corresponds to via comparison to all the 337 utterances in the search database via ED based similarity score. The word id of the most similar utterance is selected as the word to which the query corresponds to. We set $K = 40$ equivalent to the number of phonemes in English for speech experiments.

Wav2Tok gives the best performance in terms of MRR scores. It outperforms huge models like Wav2Vec2-O and Wav2Vec2-Multi which are designed specifically for the task of phonetic tokenization of speech audio by a large margin while using a minute number of parameters in comparison to them. Wav2Vec2 and Wav2Vec2P also outperform Wav2Vec2-Multi and Wav2Vec2-O while using smaller number of parameters. Wav2Vec2-O and Wav2Vec2-Multi use a blank token to handle consecutive occurrences of the same tokens and to label background noise. The utterances of each keyword in the test dataset are very small in time duration. This causes Wav2Vec2-O to confuse word utterances as background noise. It ends up generating a sequence of blank tokens and performs poorly in search. Wav2Vec2-Multi using a larger number of phonetic tokens does not suffer this issue. Wav2Tok , Wav2Vec2, and Wav2Vec2P have no such blank token. This brings a drop in search performance with sequence compression.

Table 3: Quality of Tokenization for speech

Model	Normal ($\mathcal{T}$)	Compressed ($\bar{\mathcal{T}}$)
Wav2Vec2 ED	0.68	0.63
Wav2Vec2P ED	0.7	0.65
Wav2Vec2-O ED	0.4	0.4
Wav2Vec2-Multi ED	0.67	0.67
Wav2Tok ED	**0.74**	0.66

Table 4: Spoken Term Detection

Models	F1 Score	Search Time
Wav2Vec2 ED	0.64	0.066
Wav2Vec2P ED	0.64	0.066
Wav2Vec2-O ED	0.61	0.29
Wav2Vec2-Multi ED	0.63	0.72
Wav2Tok ED	**0.65**	**0.064**

Spoken Term Detection. We convert the query word utterance and the long utterance where we are to detect the query in to sequences of tokens by all our models and detect the occurrence of the query via approximate string matching. Table 4 presents the performance of Wav2Vec2 , Wav2Vec2P , Wav2Vec2-O, Wav2Vec2-Multi, and proposed Wav2Tok in Spoken Term Detection. Wav2Tok, Wav2Vec2, Wav2Vec2P, and Wav2Vec2-Multi give comparable performance in terms of F1- score with Wav2Tok performing slightly better. The search time (in seconds) taken by Wav2Tok, Wav2Vec2, and Wav2Vec2P are comparable as the same encoder network architecture is used in the them. It is an order of magnitude smaller than that taken by Wav2Vec2-O and Wav2Vec2-Multi which use a huge encoder network. Wav2Vec2-Multi has the most number of parameters among all the models used in the experiment and hence takes the most search time.

8 Conclusion and Future Work

In this paper, we present an audio sequence tokenizer Wav2Tok that generates semantically meaningful ordered representations (or tokens) that can be used for efficient retrieval by query sequences. The tokens are learnt from pairs of semantically similar sequences, without any expert generated annotations. The tokenizer is tested for audio retrieval tasks involving a variety of semantics - melodic as well as phonetic. The performance is found to be consistent and better than state of the art models adapted for retrieval. We would like to apply more efficient search algorithms such as locally sensitive hashing and longest common subsequence search on the generated tokens to further speed up the search. We are working towards extending the proposed framework to image and video retrieval, as they also have spatial ordering. We would also like to investigate the domain specific, i.e., linguistic or musicological, aspects of the tokens so extracted.

9 REPRODUCIBILITY

The codes required to reproduce the results of this paper will be made available after the review. The experiments are performed using standard datasets.

REFERENCES

Blaise Agüera Arcas, Beat Gfeller, Ruiqi Guo, Kevin Kilgour, Sanjiv Kumar, James Lyon, Julian Odell, Marvin Ritter, Dominik Roblek, Matthew Sharifi, and Mihajlo Velimirovic. Now playing: Continuous low-power music recognition. *CoRR*, abs/1711.10958, 2017. URL http://arxiv.org/abs/1711.10958.

Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. *CoRR*, abs/1910.05453, 2019. URL http://arxiv.org/abs/1910.05453.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *CoRR*, abs/2006.11477, 2020. URL https://arxiv.org/abs/2006.11477.

Alexei Baevski, Wei-Ning Hsu, Alexis Conneau, and Michael Auli. Unsupervised speech recognition. *CoRR*, abs/2105.11084, 2021. URL https://arxiv.org/abs/2105.11084.

Sungkyun Chang, Donmoon Lee, Jeongsoo Park, Hyungui Lim, Kyogu Lee, Karam Ko, and Yoonchang Han. Neural audio fingerprint for high-specific audio retrieval based on contrastive learning. *CoRR*, abs/2010.11910, 2020. URL https://arxiv.org/abs/2010.11910.

Liqun Chen, Zhe Gan, Yu Cheng, Linjie Li, Lawrence Carin, and Jingjing Liu. Graph optimal transport for cross-domain alignment. *37th International Conference on Machine Learning, ICML 2020*, PartF16814:1520–1531, 2020.

Yu-An Chung and James Glass. Generative pre-training for speech with autoregressive predictive coding. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3497–3501, 2020. doi: 10.1109/ICASSP40776.2020.9054438.

Yu-An Chung, Hao Tang, and James Glass. Vector-quantized autoregressive predictive coding. *arXiv preprint arXiv:2005.08392*, 2020.

Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. Unsupervised cross-lingual representation learning for speech recognition. *CoRR*, abs/2006.13979, 2020. URL https://arxiv.org/abs/2006.13979.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 369–376, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143891. URL https://doi.org/10.1145/1143844.1143891.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *CoRR*, abs/2106.07447, 2021. URL https://arxiv.org/abs/2106.07447.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pp. 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558607781.

Lin-shan Lee, James Glass, Hung-yi Lee, and Chun-an Chan. Spoken content retrieval—beyond cascading speech recognition with text retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(9):1389–1420, 2015. doi: 10.1109/TASLP.2015.2438543.

Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*, pp. 3835–3845. PMLR, 2019.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

Naziba Mostafa and Pascale Fung. A note based query by humming system using convolutional neural network. In *INTERSPEECH*, pp. 3102–3106, 2017.

Naziba Mostafa, Yan Wan, Unnayan Amitabh, and Pascale Fung. A machine learning based music retrieval and recommendation system. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 1970–1977, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://aclanthology.org/L16-1312.

Stephen Mussmann and Stefano Ermon. Learning and inference via maximum inner product search. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2587–2596, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/mussmann16.html.

L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1): 4–16, 1986. doi: 10.1109/MASSP.1986.1165342.

Dhananjay Ram, Afsaneh Asaei, and Hervé Bourlard. Sparse subspace modeling for query by example spoken term detection. *IEEE ACM Trans. Audio Speech Lang. Process.*, 26(6):1126–1139, 2018. URL https://doi.org/10.1109/TASLP.2018.2815780.

Luis J. Rodriguez-Fuentes, Amparo Varona, Mikel Penagarikano, Germán Bordel, and Mireia Diez. High-performance query-by-example spoken term detection on the sws 2013 evaluation. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7819–7823, 2014. doi: 10.1109/ICASSP.2014.6855122.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *CoRR*, abs/1904.05862, 2019. URL http://arxiv.org/abs/1904.05862.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015. URL http://arxiv.org/abs/1503.03832.

M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. doi: 10.1109/78.650093.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *CoRR*, abs/1711.00937, 2017. URL http://arxiv.org/abs/1711.00937.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 684–699, 2018.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley J. Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *CoRR*, abs/1903.05662, 2019. URL http://arxiv.org/abs/1903.05662.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes, 2020.

Yaodong Zhang and James R. Glass. Unsupervised spoken keyword spotting via segmental dtw on gaussian posteriorgrams. In *2009 IEEE Workshop on Automatic Speech Recognition Understanding*, pp. 398–403, 2009. doi: 10.1109/ASRU.2009.5372931.

A Further Studies

A.1 Sequence Compression

We present the quality of sequence of tokens $\mathcal{T}$ and sequence of representations $\mathcal{Z}$ and their corresponding compressed versions sequences $\widetilde{\mathcal{T}}$ and $\widetilde{\mathcal{Z}}$ generated by the audio tokenizers in Table 5. In addition, we present the search performances of all the audio tokenizers when both time stretch and pitch shift augmentations are applied to query.

Wav2Tok outperformed the baselines and generated the best quality of sequences $\mathcal{T}$, $\mathcal{Z}$, $\widetilde{\mathcal{T}}$ and $\widetilde{\mathcal{Z}}$. We notice an overall drop in performance for all the audio tokenizers when sequence compression is applied. Compression from $\mathcal{T}$ to $\widetilde{\mathcal{T}}$ increases the robustness of the token sequences generated by Wav2Tok to various augmentations with a trade-off in search performance. Sequence compression brings an order of magnitude drop in search time for all the audio tokenizers. Wav2Vec2P learnt better tokens and representations than Wav2Vec2 because of it's pairwise training on similar audio.

Table 5: Compression of Sequences: MRR scores for query by humming, K = 25

Model	V	TS	PS	TS+PS	Search Time
Without Compression					
Wav2Vec2 DTW	0.85	0.84	0.84	0.83	3.5s
Wav2Vec2P DTW	0.87	0.85	0.87	0.85	
Wav2Tok DTW	**0.92**	**0.89**	**0.93**	**0.89**	
Wav2Vec2 ED	0.72	0.69	0.73	0.68	0.68s
Wav2Vec2P ED	0.75	0.71	0.73	0.7	
Wav2Tok ED	**0.9**	**0.84**	**0.9**	**0.83**	**0.32s**
With Compression					
Wav2Vec2 DTW	0.76	0.72	0.74	0.7	0.8s
Wav2Vec2P DTW	0.81	0.77	0.79	0.75	
Wav2Tok DTW	**0.88**	**0.88**	**0.87**	**0.88**	**0.6s**
Wav2Vec2 ED	0.66	0.63	0.64	0.61	0.06s
Wav2Vec2P ED	0.69	0.65	0.67	0.64	
Wav2Tok ED	**0.84**	**0.84**	**0.84**	**0.82**	**0.04s**

A.2 Variation in number of tokens K

The effect of varying the size of alphabet $\mathbb{A}$ is shown in Table 6. We train Wav2Vec2, Wav2Vec2P, and proposed Wav2Tok with alphabets of size $K \in \{15, 25, 40\}$. Out of the three settings for K, $K = 25$ gives the best performance for all models. Wav2Tok gives best performance for all settings of K.

B Gumbel Softmax based Vector Quantizer

The Gumbel Softmax based Vector Quantizer [Baevski et al. (2019)] quantizes input latent representation $z_t \in R^m$ with C codebooks containing K quantizers $e \in R^{K \times \frac{m}{C}}$ each. For our experiments,

Table 6: Effect of varying K: MRR scores for query by humming

MODELS	WITHOUT COMPRESSION				WITH COMPRESSION			
	V	TS	PS	TS+PS	V	TS	PS	TS+PS
$K = 15$								
WAV2VEC2 DTW	0.85	0.83	0.84	0.82	0.7	0.66	0.67	0.65
WAV2VEC2P DTW	0.87	0.85	0.85	0.83	0.82	0.77	0.8	0.75
WAV2TOK DTW	**0.88**	**0.87**	**0.88**	**0.86**	**0.84**	**0.8**	**0.83**	**0.78**
WAV2VEC2 ED	0.79	0.77	0.78	0.75	0.58	0.56	0.57	0.54
WAV2VEC2P ED	0.8	0.77	0.79	0.75	0.71	0.68	0.7	0.66
WAV2TOK ED	**0.82**	**0.8**	**0.81**	**0.8**	**0.77**	**0.75**	**0.76**	**0.73**
$K = 25$								
WAV2VEC2 DTW	0.85	0.84	0.84	0.83	0.76	0.72	0.74	0.7
WAV2VEC2P DTW	0.87	0.85	0.87	0.85	0.81	0.77	0.79	0.75
WAV2TOK DTW	**0.92**	**0.89**	**0.93**	**0.89**	**0.88**	**0.88**	**0.87**	**0.88**
WAV2VEC2 ED	0.72	0.69	0.73	0.68	0.66	0.63	0.64	0.61
WAV2VEC2P ED	0.75	0.71	0.73	0.7	0.69	0.65	0.67	0.64
WAV2TOK ED	**0.9**	**0.84**	**0.9**	**0.83**	**0.84**	**0.84**	**0.84**	**0.82**
$K = 40$								
WAV2VEC2 DTW	0.84	0.82	0.83	0.81	0.72	0.68	0.7	0.67
WAV2VEC2P DTW	0.86	0.85	0.85	0.83	0.81	0.77	0.79	0.75
WAV2TOK DTW	**0.9**	**0.88**	**0.89**	**0.87**	**0.86**	**0.83**	**0.83**	**0.81**
WAV2VEC2 ED	0.71	0.66	0.69	0.65	0.6	0.58	0.58	0.56
WAV2VEC2P ED	0.73	0.7	0.73	0.69	0.68	0.65	0.67	0.63
WAV2TOK ED	**0.83**	**0.8**	**0.82**	**0.8**	**0.77**	**0.75**	**0.76**	**0.74**

we set $C = 1$ and $K \in \{15, 25, 40\}$. Given $\mathbf{z}_t$, one of the K quantizers from each of the C codebooks are chosen resulting in vectors $e_1, ..., e_C$. The codebook vectors are then concatenated and linearly transformed from R^m to R^d to output a discrete representation $q_t \in R^d$.

$\mathbf{z}_t$ is mapped to $\mathbf{l} \in R^{C \times K}$ logits to give probability scores for the choice of codeword. The probability $p_{c,k}$ of choosing k^{th} quantizer in c^{th} codebook is given as,

$$p_{c,k} = \frac{\exp\left(l_{c,k} + n_k\right)/\tau}{\sum_{i=1}^{K} \exp\left(l_{c,i} + n_i\right)/\tau} \tag{6}$$

where τ is a non-negative temperature, $n = -log(-log(u))$ and u are samples from the uniform distribution **Unif**$(0, 1)$.

During forward pass, the codeword is chosen as $\kappa = \arg\max_j p_{c,j}$. During backward pass, the loss is calculated over the gumble softmax distribution p. We use the straight-through gradient estimator [Yin et al. (2019)] to estimate the gradient.

Codebook Diversity Loss $\mathcal{L}_d$. This loss promotes equal use of all the entries in each of the C codebooks. Minimization of this loss maximizes the entropy of the averaged softmax distribution $\tilde{p}$ over the K entries for each codebook $\tilde{p}_c$ across a batch of utterances.

$$\mathcal{L}_d = \frac{1}{CK} \sum_{c=1}^{C} \sum_{k=1}^{K} \tilde{p}_{c,k} \log \tilde{p}_{c,k} \tag{7}$$