WAV2TOK: DEEP SEQUENCE TOKENIZER FOR AUDIO RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Search over sequences is a fundamental problem. Very efficient solutions exist for text sequences, which are made up of discrete tokens chosen from a finite alphabet. Sequences, such as audio, video or sensor readings, are made up of continuous-valued samples with a large sampling rate, making similarity search inefficient. This paper proposes Wav2Tok, a deep sequence tokenizer that converts continuous-valued sequences to 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 learnt. Wav2Tok compresses the query and target sequences into short sequences of tokens that are faster to match. Experiments show consistent performance of Wav2Tok across various audio retrieval tasks, namely, music search (query by humming) and speech keyword 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. The examples 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 the sequences using Dynamic Time Warping (DTW) . But those can be very inefficient if the sequences are continuous valued and have high sampling rates, as in audio. Also, the high variability of query sequences makes the hand-made metrics cumbersome and ineffective . A more effective way of sequence retrieval is by mapping them to discrete tokens. For instance, for linguistic content based speech retrieval, audio is mapped to discrete linguistic tokens, such as phonemes, syllables or words, and search is performed over these tokens. For query-by-humming based music search, audio is mapped to discrete melody-related tokens, such as notes. For query-by-example based music search, audio is mapped to rule-based landmarks in spectrograms [Wang (2003)]. Hence, each problem uses a domain-specific hand-made mapping to tokens defined by a domain expert.

In this paper, we propose an audio tokenizer which maps the raw audio feature sequence to a sequence of tokens. The tokenizer is trained from pairs of query and target sequences, without any expert-defined tokens. The proposed algorithm makes use of the fact that relevant landmarks appear in the same order in both the query and the target sequences. This reduces the dimensionality of data and preserves the information relevant for effective retrieval. Existing text search methods, namely, locally sensitive hashing, edit distance, etc., can then be used to query over these tokens.

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 Recognition. Modern Representation learning techniques such as Contrastive Predictive Coding (CPC) [van den Oord et al. (2018)] and Autoregressive Predictive Coding (APC) [Chung & Glass (2020)] are able to generate state-of-the-art continuous representations which encode the slowly varying phoneme features in raw speech. Wav2Vec [Schneider et al. (2019)] generated continuous representations with an encoder pretrained to distinguish future timesteps from a set of distractors sampled from a proposal distribution by optimizing a CPC-based contrastive loss. After pretraining the models, the representations are mapped to phoneme tokens via minimization of the Connectionist Temporal Classification (CTC) Loss [Graves et al. (2006)]. With inspiration from VQ-VAE [van den Oord et al. (2017)] which marked the beginning of generation of discrete representations, VQ-Wav2Vec [Baevski et al. (2019)] discretized the representations generated by Wav2Vec [Schneider et al. (2019)] with either a K-Means Vector Quantizer [Baevski et al. (2019)] or a Gumble-Softmax based Vector Quantizer [Baevski et al. (2019)] to generate quantized speech representation used to pretrain a BERT [Devlin et al. (2018)]. Wav2Vec 2.0 [Baevski et al. (2020)] discretized and masked the representations to solve a CPC-based contrastive task over the quantized representations which are learned jointly. VQ-APC [Chung et al. (2020)] discretized the representations to solve APC task over the quantized representations which are learned jointly. Note, the main motivation to discretize raw audio in a latent space for all aforementioned works are to explicitly control information content encoded in the representations rather than achieve sequence tokenization. The codes or tokens generated by the models aren't constrained to be interpretable and hence initialised in large numbers. Performing CTC fine-tuning in essence groups codes in the vector quantizer module of each model together and achieves codebook compression to the number of phonemes or linguistic units. The discretization of representations do result in performance increase on various downstream tasks, with a pretrained Wav2Vec 2.0 [Baevski et al. (2020)] achieving state of the art results on speech recognition by being fine-tuned on only 10 minutes of transcribed audio. SeqRQ-AE [Liu et al. (2019)] is the closest to our work which uses a VQ-VAE [van den Oord et al. (2017)] to estimate representations for each phoneme in speech. The VQ-VAE has a codebook containing the same number of members as the number of phonemes. The codebook is trained via phonetic clustering, temporal segmentation and CTC finetuning on a small amount of transcribed audio.

Neural Audio Fingerprinting. 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.

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. Generation of representations for graphs can be done by Graph Neural Networks [Gori et al. (2005)]. Graph matching is used 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 runtime 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 METHODOLOGY

3.1 PROBLEM STATEMENT

Given a pair of semantically similar feature sequences $(\mathcal{X}_i, \mathcal{X}_j)$ of length T_i and T_j respectively and feature sequence $\mathcal{X}_k$ of length T_k with no similarity to either of the feature sequences $\mathcal{X}_i$ or $\mathcal{X}_j$, we aim to map sequence $\mathcal{X}_i$ to sequence of tokens $\mathcal{T}_i$ of length $T_i' \leq T_i$, sequence $\mathcal{X}_j$ to sequence of tokens $\mathcal{T}_j$ of length $T_j' \leq T_j$, and sequence $\mathcal{X}_k$ to sequence of tokens $\mathcal{T}_k$ of length $T_k' \leq T_k$ where all the tokens belong to a finite alphabet $\mathbb{A}$ such that, the edit distance between $\mathcal{T}_i$ and $\mathcal{T}_j$ is minimum while that between $\mathcal{T}_k$ and $\mathcal{T}_i$ as well as $\mathcal{T}_k$ and $\mathcal{T}_j$ is maximum.

3.2 MODEL ARCHITECTURE

Wav2Tok is comprised of an encoder, which, given input $\mathbf{x} \in R^n$, outputs a representation $\mathbf{z} \in R^m$, and a tokenizer, which, given the encoder output $\mathbf{z}$, outputs a token τ which belongs to a finite K-element alphabet $\mathbb{A} = \{1, ..., K\}$. Given, a temporal sequence $\mathcal{X} = [\mathbf{x}_1, ..., \mathbf{x}_T]$ of length T where $\mathbf{x}_t$ is the feature vector at time t. Our model performs frame-wise labelling of $\mathcal{X}$ to output a sequence of tokens $\mathcal{T} = [\tau_1, ..., \tau_T]$ where $\tau_t \in \mathbb{A}$ is the token assigned to input $\mathbf{x}_t$. The encoder $f : \mathbb{X} \mapsto \mathbb{Z}$ encodes $\mathbf{x}_t$ in a hyper-spherical space $\mathbb{Z}$ using a BiLSTM network [Schuster & Paliwal (1997)] followed by an L-2 normalization layer. The BiLSTM allows $\mathbf{z}_t$ to summarise information in both directions, and hence, encode surrounding context. The tokenizer $g : \mathbb{Z} \mapsto \mathbb{T}$ is a K-means vector quantizer which vector quantizes encoder output $\mathbf{z}_t$ with K quantizers or cluster centroids $\mathbf{w}_k^{tok} \in \mathbb{Z}; k = 1, ..., K$ and outputs token τ_t as, $\tau_t = \arg\max_k \{\mathbf{z}_t \cdot \mathbf{w}_k^{tok}\}$. The centroids are estimated prior to training via spherical K-means clustering in the spherical space $\mathbb{Z}$ generated by the freshly initialised encoder. The centroids are further tuned with training iterations.

3.3 MODEL TRAINING

Given a batch of tuples of similar sequences $(\mathcal{X}_i, \mathcal{X}_{i'})$ where $\mathcal{X}_i = [\mathbf{x}_t^i \in R^n; t = 1, ..., T_i]$ and $\mathcal{X}_{i'} = [\mathbf{x}_{t'}^{i'} \in R^n; t' = 1, ..., T_{i'}]$. The encoder f maps these sequences to representations $\mathcal{Z}_i = f(\mathcal{X}_i)$ and $\mathcal{Z}_{i'} = f(\mathcal{X}_{i'})$. The tokenizer g then labels $\mathcal{Z}_i$ framewise to token sequence $\mathcal{T}_i$ and $\mathcal{Z}_{i'}$ framewise to token sequence $\mathcal{T}_{i'}$.

For training, the tuple $(\mathcal{Z}_i, \mathcal{Z}_{i'})$ are concatenated to form the composed sequence $\mathcal{Z}_{ii'} = [\mathcal{Z}_i; \mathcal{Z}_{i'}]$ of length $T_{ii'} = T_i + T_{i'}$. The tuple $(\mathcal{T}_i, \mathcal{T}_{i'})$ are also concatenated to form the token sequence $\mathcal{T}_{ii'} = [\mathcal{T}_i; \mathcal{T}_{i'}]$ of length $T_{ii'} = T_i + T_{i'}$ corresponding to sequence $\mathcal{Z}_{ii'}$. Given $\mathcal{T}_{ii'} = [\tau_1^{ii'}, ..., \tau_{T_{ii'}}^{ii'}]$ and $\mathcal{Z}_{ii'} = [\mathbf{z}_1^{ii'}, ..., \mathbf{z}_{T_{ii'}}^{ii'}]$, we form a $K' \leq K$ length list of prototypes $\mathcal{P}_{ii'} = \{\mathbf{p}_{ii'}^{\tau_1}, ..., \mathbf{p}_{ii'}^{\tau_{K'}}\}$ where $\{\tau_1, ..., \tau_{K'}\} \in \mathbb{A}$ are all K' unique tokens that occur in $\mathcal{T}_{ii'}$ and $\mathbf{p}_{ii'}^{\tau_{k'}}$ is a prototype for token $\tau_{k'} \in \mathbb{A}$ generated from all vectors in $\mathcal{Z}_{ii'}$ mapping to $\tau_{k'}$ as,

$$\tilde{\mathbf{p}}_{ii'}^{\tau_{k'}} = \frac{1}{\sum_{t=1}^{T_{ii'}} \mathbf{1}_{\tau_t^{ii'} = \tau_{k'}}} \sum_{t=1}^{T_{ii'}} \mathbf{z}_t^{ii'} \mathbf{1}_{\tau_t^{ii'} = \tau_{k'}} \tag{1}$$

$$\mathbf{p}_{ii'}^{\tau_{k'}} = \frac{\tilde{\mathbf{p}}_{ii'}^{\tau_{k'}}}{||\tilde{\mathbf{p}}_{ii'}^{\tau_{k'}}||} \tag{2}$$

Let $\mathcal{P}_{ii'}$ and $\mathcal{P}_{jj'}$ be lists of prototypes generated given tuples $(\mathcal{X}_i, \mathcal{X}_{i'})$ and $(\mathcal{X}_j, \mathcal{X}_{j'})$, respectively. We sample tuples of prototypes $(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'})$ where $\mathbf{p}_{ii'}^{\tau}$ is sampled from $\mathcal{P}_{ii'}$ and $\mathbf{p}_{jj'}^{\tau'}$ is sampled from $\mathcal{P}_{jj'}$.

Approach A. Given $(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'})$, we optimize the following loss function $\mathcal{L}_{11}$ to increase the cosine similarity between $\mathbf{p}_{ii'}^{\tau}$ and $\mathbf{p}_{jj'}^{\tau'}$ to 1 if $\tau = \tau'$ and to decrease the cosine similarity to 0 if $\tau \neq \tau'$,

$$\mathcal{L}_{11}(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}, t_1) = BinXEnt(\mathbf{p}_{ii'}^{\tau} \cdot \mathbf{p}_{jj'}^{\tau'}, t_1) \tag{3}$$

$$BinXEnt(\mathbf{p}_{ii'}^{\tau} \cdot \mathbf{p}_{jj'}^{\tau'}, t_1) = t_1 \cdot \log(\mathbf{p}_{ii'}^{\tau} \cdot \mathbf{p}_{jj'}^{\tau'}) + (1 - t_1) \cdot \log(1 - \mathbf{p}_{ii'}^{\tau} \cdot \mathbf{p}_{jj'}^{\tau'}) \tag{4}$$

where $BinXEnt(.)$ is the binary cross-entropy function, $\mathbf{p}_{ii'}^{\tau} \cdot \mathbf{p}_{jj'}^{\tau'}$ gives a cosine similarity score as both the vectors are L-2 normalized, and target t_1 is 0 if $\tau \neq \tau'$ else 1. We also optimize the following loss function $\mathcal{L}_{12}$ to increase the cosine similarity of $\mathbf{p}_{ii'}^{\tau}$ or $\mathbf{p}_{jj'}^{\tau'}$ with respect to the k^{th} quantizer $\mathbf{w}_k^{tok}$ in our tokenizer $g : \mathbb{Z} \mapsto \mathbb{T}$ to 1 if τ or $\tau' = k$ repectively and to decrease the same to 0 if τ or $\tau' \neq k$,

$$\mathcal{L}_{12}(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}) = \sum_{k=1}^{K}(BinXEnt(\mathbf{p}_{ii'}^{\tau} \cdot \mathbf{w}_k^{tok}, t_{1k}) + BinXEnt(\mathbf{p}_{jj'}^{\tau'} \cdot \mathbf{w}_k^{tok}, t_{2k})) \tag{5}$$

where target t_{1k} is 0 if $\tau \neq k$ else 1 and target t_{2k} is 0 if $\tau' \neq k$ else 1. The constrastive loss function $\mathcal{L}_1$ for **Approach A** thus formed with the union of the above two loss functions is given as,

$$\mathcal{L}_1(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}, t_1) = \mathcal{L}_{11}(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}, t_1) + \mathcal{L}_{12}(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}) \tag{6}$$

Approach B. Given $(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'})$, a relation network $\mathcal{R} : R^m \mapsto R$ generates relation score $\mathcal{R}(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'})$ in the range $[0, 1]$ which is closer to 0 if $\tau \neq \tau'$ and closer to 1 if $\tau = \tau'$. The relation score acts as a non-linear similarity metric and is given as,

$$\mathcal{R}(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}) = \sigma(\mathcal{W}_r^T |\mathbf{p}_{ii'}^{\tau} - \mathbf{p}_{jj'}^{\tau'}|) \tag{7}$$

where $\sigma(.)$ is the sigmoid function, $|.|$ is the element-wise absolute value function, and $\mathcal{W}_r \in R^m$ is the weight matrix of the relation network $\mathcal{R} : R^m \mapsto R$. Given $\mathcal{R}(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'})$, we optimize the following loss function $\mathcal{L}_{21}$ to increase the relation score between $\mathbf{p}_{ii'}^{\tau}$ and $\mathbf{p}_{jj'}^{\tau'}$ to 1 if $\tau = \tau'$ and to decrease the relation score to 0 if $\tau \neq \tau'$,

$$\mathcal{L}_{21}(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}, t_2) = BinXEnt(\mathcal{R}(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}), t_2) \tag{8}$$

where target t_2 is 0 if $\tau \neq \tau'$ else 1. We also optimize the following loss function $\mathcal{L}_{22}$ to increase the relation score of $\mathbf{p}_{ii'}^{\tau}$ or $\mathbf{p}_{jj'}^{\tau'}$ with respect to the k^{th} quantizer $\mathbf{w}_k^{tok}$ in our tokenizer $g : \mathbb{Z} \mapsto \mathbb{T}$ to 1 if τ or $\tau' = k$ respectively and to decrease the same to 0 if τ or $\tau' \neq k$,

$$\mathcal{L}_{22}(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}) = \sum_{k=1}^{K}(BinXEnt(\mathcal{R}(\mathbf{p}_{ii'}^{\tau}, \mathbf{w}_k^{tok}), t_{1k}) + BinXEnt(\mathcal{R}(\mathbf{p}_{jj'}^{\tau'}, \mathbf{w}_k^{tok}), t_{2k})) \tag{9}$$

where target t_{3k} is 0 if $\tau \neq k$ else 1 and target t_{4k} is 0 if $\tau' \neq k$ else 1. The contrastive loss function $\mathcal{L}_2$ for **Approach B** thus formed with the union of the above two loss functions is given as,

$$\mathcal{L}_2(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}, t_2) = \mathcal{L}_{22}(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}, t_2) + \mathcal{L}_{22}(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}) \tag{10}$$

A spherical K-means vector quantization loss as inspired from the K-means vector quantization loss in VQ-Wav2Vec [Baevski et al. (2019)] may be iterated to add more movement to the centroids and the representations. Given prototype $\mathbf{p}_{ii'}^{\tau}$ for token $\tau \in \mathbb{A}$ generated from sequence of representations $\mathcal{Z}_{ii'} = [\mathbf{z}_1^{ii'}, ..., \mathbf{z}_{T_{ii'}}^{ii'}]$ and sequence of tokens $\mathcal{T}_{ii'} = [\tau_1^{ii'}, ..., \tau_{T_{ii'}}^{ii'}]$, our vector quantization loss is given as,

$$\mathcal{L}_{SK}(\mathbf{p}_{ii'}^{\tau}) = \frac{1}{\sum_{t=1}^{T_{ii'}} \mathbf{1}_{\tau_t^{ii'}=\tau}} \sum_{t=1}^{T_{ii'}}(2 - (\mathbf{sg}(\mathbf{z}_t^{ii'}) \cdot \mathbf{w}_{\tau}^{tok} + \gamma \cdot \mathbf{z}_t^{ii'} \cdot \mathbf{sg}(\mathbf{w}_{\tau}^{tok})))\mathbf{1}_{\tau_t^{ii'}=\tau} \tag{11}$$

where γ is a positive constant, and $\mathbf{sg}(.)$ is the stop-gradient operator. The stop-gradient operator converts the parameter input to a constant during gradient computation. Increasing $\mathbf{sg}(\mathbf{z}_t^{ii'}) \cdot \mathbf{w}_{\tau}^{tok}$ pushes $\mathbf{w}_{\tau}^{tok}$ towards $\mathbf{z}_t^{ii'}$ keeping $\mathbf{z}_t^{ii'}$ static and increasing $\mathbf{z}_t^{ii'} \cdot \mathbf{sg}(\mathbf{w}_{\tau}^{tok})$ pushes $\mathbf{z}_t^{ii'}$ towards $\mathbf{w}_{\tau}^{tok}$ keeping $\mathbf{w}_{\tau}^{tok}$ static. With γ set to 0, iterating the loss only affects $\mathbf{w}_{\tau}^{tok}$.

Given $(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau})$, the final loss function $\mathcal{L}$ which we minimize to optimize our model parameters is given as,

$$\mathcal{L}(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}, t_a) = \mathcal{L}_a(\mathbf{p}_{ii'}^{\tau}, \mathbf{p}_{jj'}^{\tau'}, t_a) + \alpha \cdot (\mathcal{L}_{SK}(\mathbf{p}_{ii'}^{\tau}) + \mathcal{L}_{SK}(\mathbf{p}_{jj'}^{\tau'})) \tag{12}$$

where α is a positive constant, t_a is the target, and $a = 1$ corresponds to **Approach A** while $a = 2$ corresponds to **Approach B**.

3.4 SEQUENCE SEGMENTATION AND COMPRESSION

Given a sequence of features $\mathcal{X} = [\mathbf{x}_1, ..., \mathbf{x}_T]$ where T is the sequence length, the encoder f outputs a sequence of L-2 normalised representations $\mathcal{Z} = [\mathbf{z}_1, ..., \mathbf{z}_T]$ and the tokenizer g outputs a sequence of tokens $\mathcal{T} = [\tau_1, ..., \tau_T]$. The consecutive repetition of tokens in $\mathcal{T}$ highlights a subsequence in $\mathcal{Z}$ whose constituent vectors are similar. Given $\mathcal{T}$, we delete all consecutive token repetitions to form compressed token sequence $\tilde{\mathcal{T}} = [\tau_{t_{s_1}:t_{e_1}}, ..., \tau_{t_{s_{T'}}:t_{e_{T'}}}]$ of length $T' \leq T$ where $t_{s_1} = 1$, $t_{e_{T'}} = T$, and $\tau_{t_{s_i}:t_{e_i}}$ is the token which occurs in the interval $[t_{s_i} : t_{e_i}]$ of sequence $\mathcal{T}$. Given the set of token onset and offset times $\{(t_{s_i}, t_{e_i})|i \in \{1, ..., T'\}\}$ generated via compression of $\mathcal{T}$ to $\tilde{\mathcal{T}}$ with t_{s_i} and t_{e_i} denoting the onset and offset time of the i-th token $\tau_{t_{s_i}:t_{e_i}}$ in $\tilde{\mathcal{T}}$, we generate compressed sequence $\tilde{\mathcal{Z}} = [\tilde{\mathbf{z}}_{t_{s_1}:t_{e_1}}, ..., \tilde{\mathbf{z}}_{t_{s_{T'}}:t_{e_{T'}}}]$ of length $T' \leq T$ from $\mathcal{Z}$ where $\tilde{\mathbf{z}}_{t_{s_i}:t_{e_i}}$ is a representation for all consecutive vectors $\mathbf{z}_t$ in the interval $[t_{s_i}, t_{e_i}]$ of sequence $\mathcal{Z}$ mapping to the same token $\tau_{t_{s_i}:t_{e_i}}$ and is given as,

$$\tilde{\mathbf{z}}'_{t_{s_i}:t_{e_i}} = \frac{1}{t_{e_i} - t_{s_i}} \sum_{t=t_{s_i}}^{t_{e_i}} \mathbf{z}_t \tag{13}$$

$$\tilde{\mathbf{z}}_{t_{s_i}:t_{e_i}} = \frac{\tilde{\mathbf{z}}'_{t_{s_i}:t_{e_i}}}{||\tilde{\mathbf{z}}'_{t_{s_i}:t_{e_i}}||} \tag{14}$$

A compressor network $\mathcal{C}$ performs the aforementioned compression on sequences $\mathcal{Z}$ and $\mathcal{T}$ of length T to output sequences $\tilde{\mathcal{Z}}$ and $\tilde{\mathcal{T}}$ of length $T' \leq T$ respectively.

4 EXPERIMENTS

We test the performance of Wav2Tok for three tasks, viz., Query by Humming, Speech Search and Keyword Spotting.

4.1 DATASETS

We employ the **MIR-QBSH** dataset for query by humming tasks and **TIMIT** [Garofolo et al. (1993)] dataset for speech search and keyword spotting tasks. MIR-QBSH is composed of 4431 singing or humming recordings (8KHz, 8-bit, monophonic) submitted by 195 subjects. The dataset is further partitioned into 48 partitions, each composed of recordings corresponding to one of the 48 ground truth songs. TIMIT is composed of spoken text audio (16KHz, 16-bit, monophonic). We choose 59 words with reasonable number of occurrences as keywords and all other words as non-keywords, resulting in a dataset of 2169 keyword recordings.

4.2 BASELINES

STFT. This method performs DTW over raw STFT features.

Triplet. Representations generated by a BiLSTM encoder [Schuster & Paliwal (1997)] are tuned via optimizing the Triplet Loss [Schroff et al. (2015)] as done in optimizing neural audio fingerprints in Now Playing [Arcas et al. (2017)]. Let **a** be the reference or anchor sample, **p** be a positive sample (same sample but augmented), and **n** be a negative sample (sample with no similarity to **a**). Constraining all vectors in the triplet $(\mathbf{a},\ \mathbf{p}, \mathbf{n})$ to be L-2 normalized, the loss function for each triplet of is given as,

$$\mathcal{L}_{Triplet}(\mathbf{a},\ \mathbf{p}, \mathbf{n}) = max\{d(\mathbf{a}, \mathbf{p}) - d(\mathbf{a}, \mathbf{n}) + m, 0\} \tag{15}$$

$$d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}|| \tag{16}$$

where $max\{.\}$ represents the maximum operator,**x** and **y** are vectors with same dimension, and m is the margin which adds the meaning of similarity to the function. The representations were not constrained to be L-2 normalized.

MIPS. Representations generated by a BiLSTM encoder [Schuster & Paliwal (1997)] are constrained to be L-2 normalized and are trained with divergences taken via simulation of Maximum

Inner Product Search [Mussmann & Ermon (2016)] on mini-batches of representation vectors as proposed by [Chang et al. (2020)]. A minibatch $\{\mathbf{z}_i | i \in \{1, ..., N\}\}$ of size N is composed of $\frac{N}{2}$ pairs $\{\mathbf{z}^{org}, \mathbf{z}^{rep}\}$ where representation vector $\mathbf{z}^{rep}$ is an augmented replica of the representation vector $\mathbf{z}^{org}$. With $\mathbf{z}_j$ as the augmented replica of $\mathbf{z}_i$ in a minibatch of representation vectors, the loss function to be minimized is given as,

$$\mathcal{L}_{MIPS}(\mathbf{z}_i, \mathbf{z}_j) = -\log \frac{\exp(Sim(\mathbf{z}_i, \mathbf{z}_j))}{\sum_{k \neq i} \exp(Sim(\mathbf{z}_i, \mathbf{z}_k))} \tag{17}$$

Wav2Vec 2.0. A scaled down and modified Wav2Vec 2.0 [Baevski et al. (2020)] model where the latent Representations generated by a BiLSTM encoder [Schuster & Paliwal (1997)] are masked and Product Quantized (PQ) with a Gumble Softmax based Vector Quantizer [Baevski et al. (2019)] (see Appendix B). For a masked time step t, the latent representation $\mathbf{z}_t$ is fed to a Transformer encoder [Vaswani et al. (2017)] which outputs a contextualised representation $\mathbf{c}_t$ used to distinguish the true quantized representation $\mathbf{q}_t$ output by the vector quantizer given $\mathbf{z}_t$ from a set of $D + 1$ candidate quantized representations $\tilde{\mathbf{q}} \in \mathcal{Q}_t$ containing q_t and D distractors. The loss function iterated to train the model is given as,

$$\mathcal{L}_{Wav2Vec2} = -\log \frac{\exp(sim(\mathbf{c}_t, \mathbf{q}_t)/\kappa)}{\sum_{\tilde{\mathbf{q}} \sim \mathcal{Q}_t} \exp(sim(\mathbf{c}_t, \tilde{\mathbf{q}})/\kappa)} + \beta \cdot \mathcal{L}_d \tag{18}$$

where $sim(a, b) = \frac{a^T b}{||a|| ||b||}$ gives cosine similarity score, β is a positive constant, and $\mathcal{L}_d$ is a codebook diversity loss for the Gumble Softmax based Vector Quantizer module encouraging equal use of the codebook entries (see Appendix B).

Wav2Vec Tok. The same architecture as **Wav2Vec 2.0** but here the Gumble Softmax Based Vector Quantizer is used as a Tokenizer which labels the latent representations with K tokens forming alphabet $\mathbb{A} = \{1, ..., K\}$. The representations are also trained optimizing loss function [18].

MIPS and Triplet 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. Wav2Vec 2.0 and Wav2Vec Tok use a 2-layer BiLSTM encoder with 3.6 million trainable parameters to generate latent space and a 2-layer Transformer with 3 attention heads with 8.5 million trainable parameters to generate context space amounting to models with 12.1 million trainable parameters. Both are trained using the ADAM [Kingma & Ba (2017)] optimizer and a linear learning schedule warming-up a learning rate of 0.001 in the first 8% of the training steps.

4.3 Our Models

TokA. Model trained in accordance to **Approach A**. We do not iterate over the quantization loss $\mathcal{L}_{SK}$ by setting positive constant α in final loss function $\mathcal{L}$ to 0 since we are calculating the contrastive loss $\mathcal{L}_1$ with cosine similarity allowing it to give the same regularisation as $\mathcal{L}_{SK}$.

TokB. Model trained in accordance to **Approach B**.

TokB+Trans. Model trained in accordance to **Approach B** but with a Transformer encoder [Vaswani et al. (2017)] as encoder network instead of an BiLSTM and $\alpha = 0$.

TokB+NoSim. Model trained in accordance to **Approach B** but with batches of tuples of sequences which need not be semantically similar and $\alpha = 0$.

TokA, TokB, and TokB+NoSim use a 2-layer BiLSTM as encoder with 3.6 million trainable parameters. TokB+Trans uses a 2-layer Transformer with 3 attention heads as encoder with 8.5 million trainable parameters. We train then using the ADAM [Kingma & Ba (2017)] optimizer and a linear learning schedule, warming-up a learning rate of 0.001 in the first 8% of the training steps.

4.4 Music Melody Search: Query by Humming

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

Experiment Details. We divide the MIR-QBSH dataset into 3 splits of training, validation and test with split sizes in the ratio $2462 : 844 : 1125$. The recordings are all converted to Short Time Fourier Transform matrices with 513 frequency bins, a window length of 1024 samples (128 ms), and hop length of 512 samples. The training and the validation split consists of pairs of similar audio. The test set consists of a search database (with audio from validation split) and a set of queries (disjoint from both training and validation splits). All the audio in the search dataset are converted to sequences of representations or tokens, using the same tokenizer as used over the search database. Given a query sampled from the test split, we perform DTW based search (if representations) or Edit Distance based search (if tokens) over the search database.

Results. The proposed TokB representation with DTW search performs the best in terms of Mean Reciprocal Rank (MRR) scores (see Appendix D) as compared to other baselines and proposed methods, as shown in Table 1. Larger the MRR score, better is the method. Quantization of the representations to tokens (see TokB ED) makes the search faster, while slightly degrading the performance. To check the robustness of the representation or tokens, we also apply a series of transformations on the queries in the form of time stretch and pitch shift. Melodic semantics are not altered by these transformations. Along with the original query (V or vanilla) Table 1 also shows the results for Time Stretched (TS) query, Pitch Shifted (PS) query, and Time Stretched + Pitch Shifted (TS + PS) query. Sequence compression further speeds up the search while lowering the performance metrics (see Compressed columns in Table 1). ED is faster than DTW and compressed sequences are faster to search. So, ED based search over compressed tokens is the fastest.

Models	Normal				Compressed			
	V	TS	PS	TS+PS	V	TS	PS	TS+PS
STFT DTW	0.44	0.28	0.38	0.214	-	-	-	-
Triplet DTW	0.278	0.158	0.238	0.14	-	-	-	-
MIPS DTW	0.45	0.15	0.414	0.146	-	-	-	-
Wav2Vec 2.0 DTW	**0.75**	**0.631**	**0.714**	**0.581**	-	-	-	-
Wav2Vec Tok DTW	0.73	0.62	0.68	0.563	0.537	0.454	0.5	0.4
TokA DTW	0.804	0.75	0.78	0.701	0.6	0.48	0.54	0.4
TokB DTW	**0.84**	**0.8**	**0.825**	**0.753**	**0.722**	**0.64**	**0.68**	**0.57**
Wav2Vec Tok ED	0.69	0.62	0.67	0.56	0.414	0.353	0.4	0.336
TokA ED	0.66	0.61	0.645	0.5	0.45	0.36	0.425	0.3
TokB ED	**0.768**	**0.741**	**0.75**	**0.77**	**0.63**	**0.54**	**0.54**	**0.48**

Table 1: MRR scores for query by humming, with $K = 25$

Model	Search Time
Normal	
Wav2Vec Tok DTW	3.48
Wav2Vec Tok ED	0.2
TokB DTW	3.5
TokB ED	0.32
Compressed	
Wav2Vec Tok DTW	0.4
Wav2Vec Tok ED	0.02
TokB DTW	0.6
TokB ED	0.04

Table 2: Average Search Time (in s) per query

Effect of $\mathcal{L}_{SK}$. We study the effect of including $\mathcal{L}_{SK}$ to the loss function by setting different values of α and γ in Eq. 12. We observe an overall performance drop with inclusion of $\mathcal{L}_{SK}$, as shown in Table 3.

Models	Normal				Compressed			
	V	TS	PS	TS+PS	V	TS	PS	TS+PS
TokB DTW	**0.84**	**0.8**	**0.825**	**0.753**	**0.722**	**0.64**	**0.68**	**0.57**
TokB($\alpha = 1, \gamma = 0$) DTW	0.824	0.756	0.8	0.71	0.682	0.523	0.574	0.43
TokB($\alpha = 1, \gamma = 0.1$) DTW	0.81	0.742	0.78	0.676	0.702	0.55	0.621	0.474
TokB ED	**0.768**	**0.741**	**0.75**	**0.77**	**0.63**	**0.54**	**0.54**	**0.48**
TokB($\alpha = 1, \gamma = 0$) ED	0.685	0.49	0.641	0.42	0.57	0.41	0.46	0.323
TokB($\alpha = 1, \gamma = 0.1$) ED	0.663	0.459	0.6	0.46	0.474	0.35	0.39	0.3

Table 3: MRR scores for query by humming, with $K = 25$

Variation in number of Tokens. The effect of varying the size of alphabet $\mathbb{A}$ is shown in Table 4. $K = 25$ gives the best performance for all models. Although $K = 15$ and $K = 40$ gives a small performance drop in MRR for vanilla queries, the robustness of the tokens to transformed queries degrades a lot. TokB gave the best performance for almost all the settings of K while Wav2Vec Tok suffers a large drop in performance for $K = 40$.

Token Semantics. Query by humming involves similarity based on melody information, which is carried by the semantic pairing of the audio in training data. As an ablation study, we randomize this pairing; we call this model TokB+NoSim. The results are presented in Table 5. There is a drop in performance with TokB+NoSim as compared to TokB, which is more severe for transformed queries. These results support that the model TokB is learning the semantic information from the pairs.

Models	Normal				Compressed			
	V	TS	PS	TS+PS	V	TS	PS	TS+PS
$K = 15$								
Wav2Vec Tok DTW	0.712	0.6	0.67	0.55	0.531	0.428	0.49	0.39
TokA DTW	0.75	0.62	0.7	0.56	0.523	0.375	0.471	0.35
TokB DTW	**0.831**	**0.751**	**0.8**	**0.7**	**0.7**	**0.54**	**0.6**	**0.47**
Wav2Vec Tok ED	0.66	0.6	0.63	0.53	0.4	0.35	0.37	0.315
TokA ED	0.71	0.545	0.65	0.5	0.44	**0.385**	0.41	**0.32**
TokB ED	**0.773**	**0.683**	**0.724**	**0.62**	**0.46**	0.31	**0.42**	0.3
$K = 25$								
Wav2Vec Tok DTW	0.73	0.62	0.68	0.563	0.537	0.454	0.5	0.4
TokA DTW	0.804	0.75	0.78	0.701	0.6	0.48	0.54	0.4
TokB DTW	**0.84**	**0.8**	**0.825**	**0.753**	**0.722**	**0.64**	**0.68**	**0.57**
Wav2Vec Tok ED	0.69	0.62	0.67	0.56	0.414	0.353	0.4	0.336
TokA ED	0.66	0.61	0.645	0.5	0.45	0.36	0.425	0.3
TokB ED	**0.768**	**0.741**	**0.75**	**0.77**	**0.63**	**0.54**	**0.54**	**0.48**
$K = 40$								
Wav2Vec Tok DTW	0.47	0.377	0.45	0.342	0.253	0.235	0.241	0.214
TokA DTW	**0.82**	0.75	**0.79**	0.7	0.6	0.45	0.52	0.371
TokB DTW	0.81	0.75	0.78	0.7	**0.642**	**0.534**	**0.57**	**0.48**
Wav2Vec Tok ED	0.35	0.27	0.33	0.24	0.16	0.15	0.143	0.131
TokA ED	0.66	0.56	0.621	0.48	0.46	0.35	0.4	0.283
TokB ED	**0.71**	**0.6**	**0.665**	**0.54**	0.46	0.35	0.4	**0.3**
V - Vanilla, TS - Time Stretched, PS - Pitch Shifted								

Table 4: Effect of varying K: MRR scores for query by humming

Models	Normal				Compressed			
	V	TS	PS	TS+PS	V	TS	PS	TS+PS
TokB DTW	**0.84**	**0.8**	**0.825**	**0.753**	**0.722**	**0.64**	**0.68**	**0.57**
TokB+Trans DTW	0.79	0.723	0.765	0.683	0.564	0.434	0.5	0.4
TokB+NoSim DTW	0.83	0.75	0.79	0.7	0.68	0.57	0.63	0.5
TokB ED	**0.768**	**0.741**	**0.75**	**0.77**	**0.63**	**0.54**	**0.54**	**0.48**
TokB+Trans ED	0.72	0.6	0.67	0.57	0.43	0.3	0.373	0.28
TokB+NoSim ED	0.724	0.6	0.674	0.53	0.554	0.44	0.473	0.404

Table 5: Ablation Analysis: MRR scores for query by humming

Transformer as Encoder. We use a Transformer with 8.5 million parameters instead of a 3.6 million parameter BiLSTM as our encoder. We call this model as TokB+Trans. There is an overall drop in performance and robustness of the tokens generated by TokB+Trans incomparison to TokB (Table 5). Possible causes for such drop could be overfitting by the larger transformer encoder or the number of attention heads (3, here). Decreasing the number of layers and increasing the number of attention heads might bring in some improvement in performance.

4.5 SPEECH SEARCH

Task. Given a query speech audio, we are to find in the search database the audio with similar speech content.

Experiment Details. We divide the keyword dataset, extracted from TIMIT, into three splits, viz., training, validation and test with split sizes corresponding to each keyword in the ratio $1414 : 188 : 567$. All audio are converted to spectrograms using Short Time Fourier Transform with 185 frequency bins, a window length of 368 samples (23 ms), and hop length of 92 samples. We train and test our models in the same fashion as done for query by humming experiments (Sec. 4.4).

Results. The representations generated by TokB and MIPS perform the best as shown in Table 6. There is only a slight drop of 0.02 MRR when sequence compression is applied to the representations generated by TokB (see Compressed columns in Table 6), implying a faster search with high MRR performance.

Models	Normal	Compressed
STFT DTW	0.49	-
Triplet DTW	0.5	-
MIPS DTW	**0.737**	-
Wav2Vec Tok DTW	0.64	0.5
TokA DTW	0.622	0.624
TokB DTW	**0.735**	**0.71**
Wav2Vec Tok ED	0.4	0.175
TokA ED	0.443	0.364
TokB ED	0.61	0.54

Table 6: TIMIT Similarity Search Results (MRR Scores)

Models	Accuracy	Precision	Recall	F1 Score	Time
STFT DTW	0.48	0.66	0.48	0.51	**0.11**
Triplet DTW	0.47	0.61	0.47	0.49	0.315
MIPS DTW	**0.55**	**0.68**	**0.55**	**0.58**	0.383
TokA DTW	0.52	0.67	0.52	0.55	0.352
TokB DTW	0.57	**0.69**	0.57	0.6	0.372
TokA ED	0.55	0.65	0.55	0.57	0.204
TokB ED	**0.63**	0.68	**0.63**	**0.62**	**0.041**

Table 7: TIMIT Keyword Spotting Results

4.6 KEYWORD SPOTTING

Task. Given a query audio containing an uttered keyword, spot the location of the keyword in a dataset of long speech recordings. It is considered a hit if the spotted segment in a long audio has a non-zero overlap with the ground truth segment of the keyword.

Experiment Details. All models are trained on the keyword dataset as done for speech search. We synthetically generate 564 long utterances each composed of either 60 or 80 non-keywords sampled randomly from TIMIT and 6 to 10 occurrences of keywords sampled from the keyword dataset extracted from TIMIT. The training set comprises 273 long utterances which are used to set the detection threshold for the keyword spotters. The test set consists of a query database (with audio from the validation split we generated from the keyword dataset) and a search database formed with the remaining 291 long utterances. The recordings are all converted to spectrograms using Short Time Fourier Transform with 185 frequency bins, a window length of 368 samples (23 ms), and hop length of 92 samples. We convert the search dataset as well as the query audio into representation or tokens using a tokenizer. For search, we chunk the long audio into smaller segments. Let $\mathcal{U}^1 = [u_1^1, ..., u_T^1]$ of length T represent the query keyword to be spotted. Here, u_t may either be a representation z_t or a token τ_t or raw **STFT** feature x_t. Let $\mathcal{U}^2 = [u_1^2, ..., u_{T'}^2]$ represent the long query of length $T' \gg T$. For any time step t in $\mathcal{U}^2$, a detection score s_t is generated as follows,

$$s_t = \min_{l \in \{0.5 \cdot T, T, 2 \cdot T\}} Dist(\mathcal{U}^1, \mathcal{U}^2_{t:t+l}) \tag{19}$$

where $\mathcal{U}^2_{t:t+l}$ is the segment of $\mathcal{U}^2$ in the time interval $(t, t+l)$ of length $l \in \{0.5 \cdot T, T, 2 \cdot T\}$, and $Dist(.)$ generates DTW distance when $u_t = \mathbf{z}_t$ or $\mathbf{x}_t$, and Edit Distance when $u_t = \tau_t$. We set detection threshold θ by training our keyword spotter with some training utterances (see Appendix C). A keyword is detected at time step t in given query iff $s_t \leq \theta$. We evaluate performance of all our keyword spotters based on Accuracy, Precision, Recall, F1 score and Search Time. Many efficient and faster algorithms exist for text search or genome sequence search but we chose to use aforementioned approach to test the tokenizer efficacy.

Results. Tokens and representations generated by TokB give the best results in terms of F1 Score, as shown in Table 7. The tokenization performed by TokB also results in the best search time since TokB performs the best segmentation of the input utterances. DTW over raw STFT features is fast because of low dimensionality of the features.

5 CONCLUSION AND FUTURE WORK

In this paper, we present a 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. Also, we are working towards extending the proposed framework to 2D data retrieval.

6 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.

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.

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.

J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren. Darpa timit acoustic phonetic continuous speech corpus cdrom, 1993.

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.

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.

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.

Alexander H. Liu, Tao Tu, Hung-yi Lee, and Lin-Shan Lee. Towards unsupervised speech recognition and synthesis with quantized speech representation learning. *CoRR*, abs/1910.12729, 2019. URL http://arxiv.org/abs/1910.12729.

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

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.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Avery Wang. An industrial strength audio search algorithm. 01 2003.

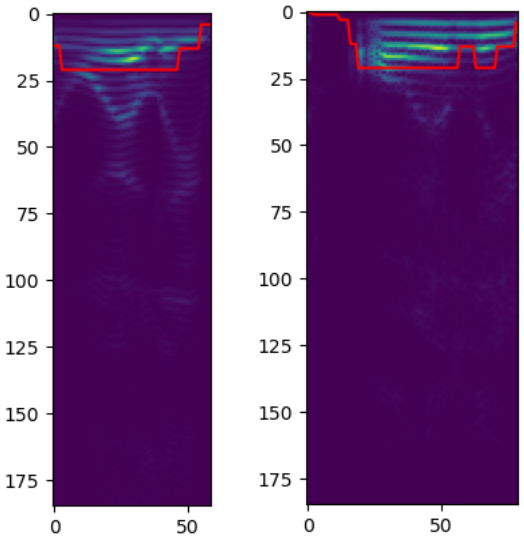
Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 684–699, 2018.

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.

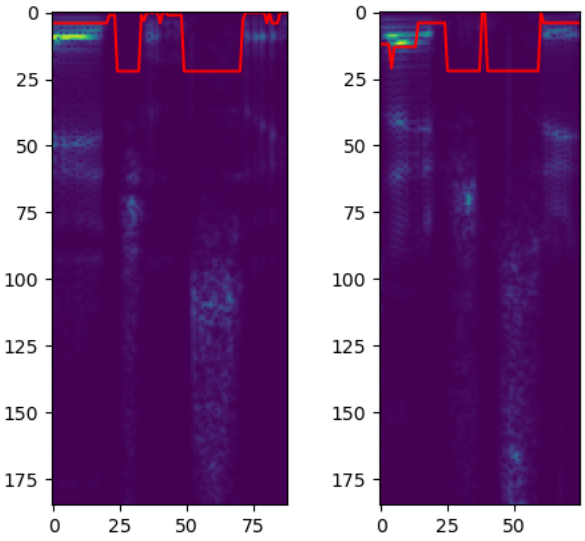
A VISUALIZATION OF TOKENIZATION

We present the segmentation of sequences brought about by our tokenizer TokB in the following figures. Figures 1, 2, 3, present tokenizations of 2 utterances of each of the keywords 'oily', 'agency', and 'carry' respectively. The red line marks the segments generated as a result of tokenization. The height of the red line at each time-step of the spectrogram denotes the token assigned to that time-step. Figures 4, 5, 6, present tokenizations of pairs of songs sharing the same id '00024', '00027', and '00042' respectively.



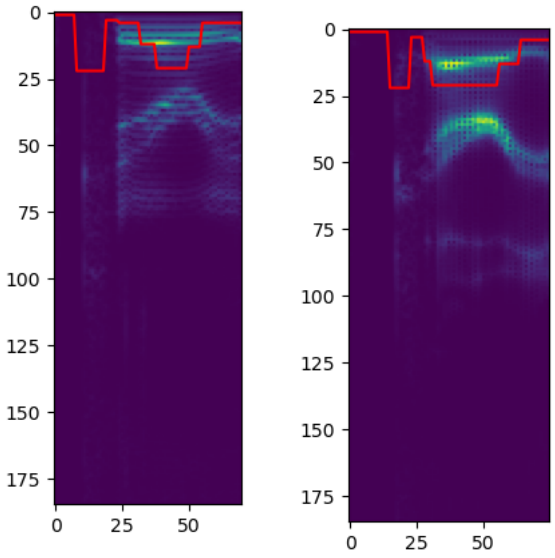
(a) (b)

Figure 1: Tokenization of 2 utterances of the keyword 'oily'



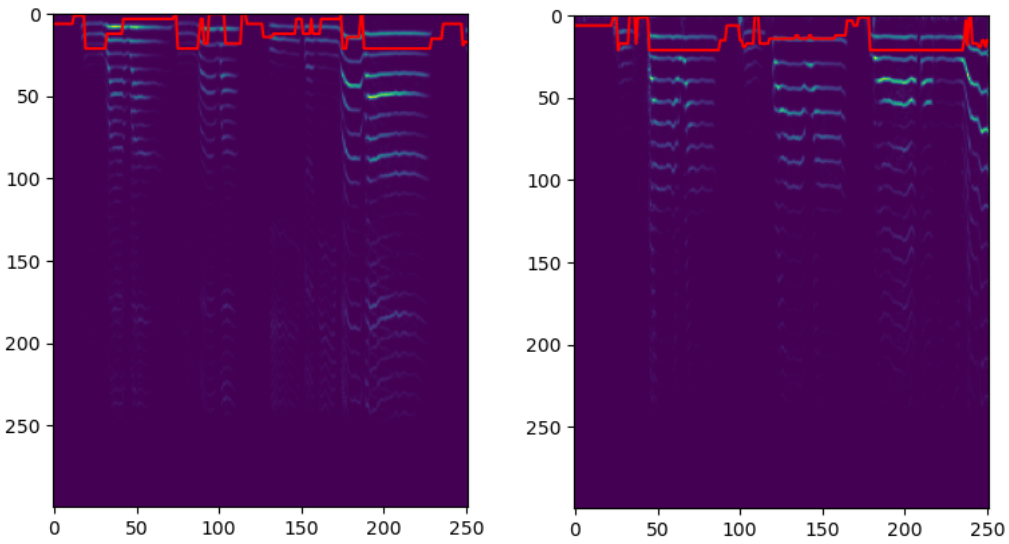
(a) (b)

Figure 2: Tokenization of 2 utterances of the keyword 'agency'



(a) (b)

Figure 3: Tokenization of 2 utterances of the keyword 'carry'

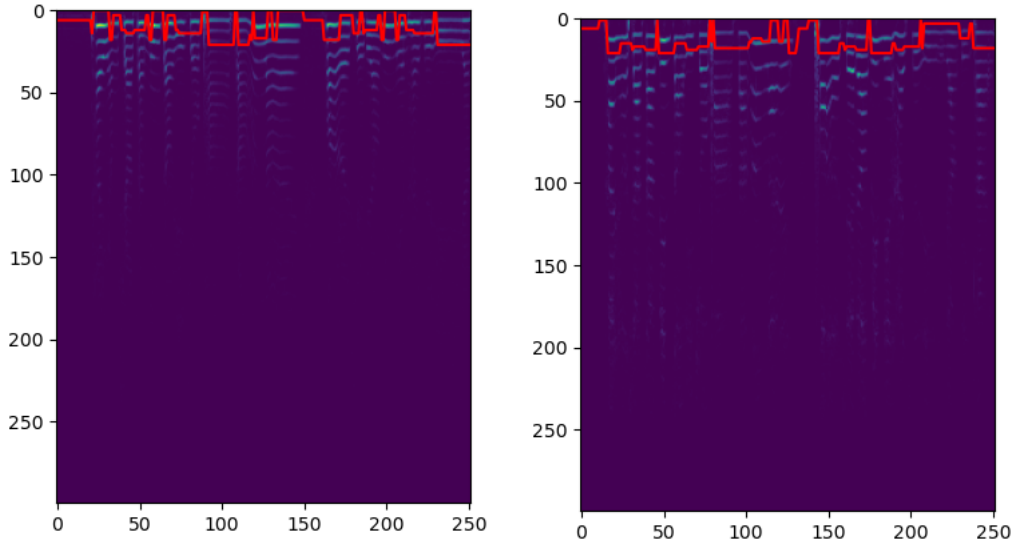


(a) (b)

Figure 4: Tokenization of 2 songs with id '00024'

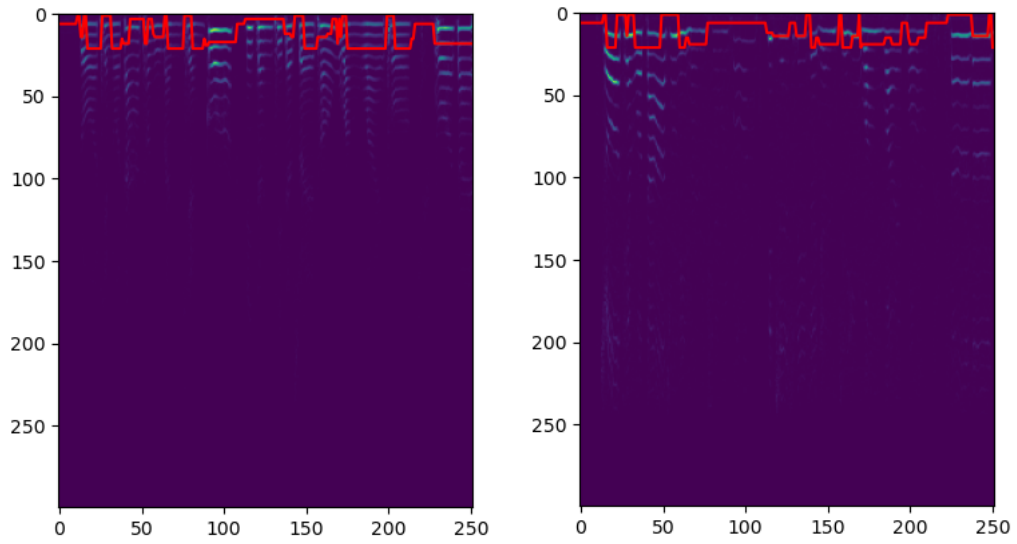
B GUMBLE SOFTMAX BASED VECTOR QUANTIZER

The Gumble Softmax based Vector Quantizer product quantizes input latent representation $z_t \in R^m$ with C codebooks each containing K quantizers $e \in R^{K \times \frac{m}{C}}$. Given $\mathbf{z}_t$, one of the K quantizers from each of the C codebooks are chosen resulting in vectors $e_1, ..., e_C$ which are concatenated and linearly transformed from R^m to R^d to output $q_t \in R^d$. $\mathbf{z}_t$ 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,



(a) (b)

Figure 5: Tokenization of 2 songs with id '00027'



(a) (b)

Figure 6: Tokenization of 2 songs with id '00042'

$$p_{c,k} = \frac{\exp\left(l_{c,k} + n_k\right)/\tau}{\sum_{i=1}^{K} \exp\left(l_{c,i} + n_i\right)/\tau} \tag{20}$$

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 $\kappa = \arg\max_j p_{c,j}$. The straight-through gradient estimator [Yin et al. (2019)] is utilized to estimate the gradient during backward pass.

Codebook Diversity Loss $\mathcal{L}_d$. This loss promotes equal use of all the entries in each of the C codebooks by maximizing the entropy of the averaged softmax distribution $\mathbf{l}$ over the K entries for

each codebook $\tilde{p}_c$ across a batch of utterances.

$$\mathcal{L}_d = \frac{1}{CK} \sum_{c=1}^{C} \sum_{k=1}^{K} \tilde{p}_{c,k} \log \tilde{p}_{c,k} \tag{21}$$

C SETTING OF DETECTION THRESHOLD FOR KEYWORD SPOTTING

Given the utterances are converted to sequences of tokens, we use Edit distance as detection score. The Edit distance being an integer, we find the mode m_{hit} of the set of detection scores s_t generated at time steps t when a keyword is present. We also find the mode m_{no-hit} of the set of detection scores s_t generated at time steps t when no keyword is present. The detection threshold is then set as $th = \frac{1}{2}(m_{no-hit} + m_{hit})$. If $s_t \leq th$, keyword is spotted else non-keyword.

Given the utterances are converted to sequences of representations, we use DTW distance as detection score. We find the average s_{hit}^{avg} of the set of detection scores s_t generated at time steps t when a keyword is present. We also find the average s_{no-hit}^{avg} of the set of detection scores s_t generated at time steps t when no keyword is present. The detection threshold is then set as $th = \frac{1}{2}(s_{no-hit}^{avg} + s_{hit}^{avg})$. If $s_t \leq th$, keyword is spotted else non-keyword.

D MEAN RECIPROCAL RANK

Mean Reciprocal Rank (MRR) is a metric used to evaluate search performances. Given a query, a search over the database outputs a ranking on the documents in a descending order of some similarity metric calculated with the query. Let $rank_i$ be the rank of the first relevant document for the i^{th} query sampled from a set of N queries. The MRR score is given as,

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank_i} \tag{22}$$