Variable Bitrate Discrete Neural Representations via Causal Self-Attention

Shuyang Li[*]
UC San Diego
shl008@ucsd.edu

Huanru Henry Mao[*]
Altum
henry@altum.io

Julian McAuley
UC San Diego
jmcauley@ucsd.edu

Abstract

Generative modeling across text, video, and image domains has benefited from the introduction of discrete (quantized) representations for continuous data. Existing techniques are limited to learning fixed-size quantized codes, which requires training a new model for each desired compression rate. We aim to learn a single model able to produce discrete representations at different granularities (bitrates). We propose a study to answer the two main questions required to build such a model: 1) How do we produce a variable number of latent codes (variability)? 2) How do we allocate a given code budget across our input data (allocation)? We propose a framework based on the Perceiver IO architecture, incorporating causal attention to learn ordered latent codes that can then be adaptively pruned to a target compression rate. We will conduct extensive experiments on a variety of datasets, benchmarking our proposed approach for learning a single VBR quantizer against prior techniques for quantized audio encoding.

1 Introduction

Learning compressed discrete codes from continuous domain data has been shown to enable efficient latent generative modeling [24, 5, 23, 27, 26]. Latent generative modeling involves two stages: first, an autoencoder is trained to compress the continuous input data (e.g., raw images, audio) into discrete latent codes [31]; next, an auto-regressive language model is trained to model these codes [32]. This enables generating from the data distribution in latent space via sampling the auto-regressive model and decoding through the autoencoder. Here, quantization eliminates semantically irrelevant information via compression, enabling auto-regressive modeling of important information only.

In this proposal, we attempt to further optimize the compression step by exploiting variability in semantically meaningful content across dimensions. For example, periods of silence in speech require fewer bits to encode [36]. Taking advantage of variable bitrate (VBR) is a common technique in compression methods such as MP3 [28] and JPEG [33]. The core challenges of VBR compression in neural networks is solving the *allocation problem* (how many codes are needed?) and *variability problem* (how do we produce a variable number of latent dimensions?).

We propose an end-to-end domain-agnostic framework for training autoencoders to 1) produce a variable number of discrete latent codes from raw continuous data; and 2) effectively allocate the number of latent codes for a window of input data that yields a strong reconstruction quality for a given average bitrate. The key to our method is leveraging the recently proposed Perceiver architecture [10] with causal attention mechanisms [3] to learn ordered latent representations (Section 4).

We design experiments to measure the effectiveness of our approach across three increasingly challenging audio datasets in Section 5.3. We will investigate the correlation between compression

[*] denotes equal contribution

2nd Pre-registration workshop (NeurIPS 2021), Remote.

rate, reconstruction quality, and modeling efficiency by comparing models trained with our framework against baseline techniques for multi-model VBR [5, 34] compression, as well as established technologies for lossy audio compression.

Although the technique we propose is broadly applicable, our experiments will focus on the audio domain, where speech and music are widely consumed yet require substantial space for storage and transmission [19]. Much like hand-engineered VBR schemes for lossy compression (e.g., MP3), we aim to create compact, discrete representations of continuous data while maintaining high reconstruction fidelity.

2 Related Work

Discrete latent codes offer a potentially more efficient representation for many common data modalities like video, images and speech compared to continuous representations [18, 31]. Discrete latent representations have seen recent success for image [24, 23] and audio generation [5], speech recognition [2], and has downstream applications for pre-training on continuous-domain data [16]. By quantizing continuous-domain data into a discrete format, one can apply off-the-shelf language modeling techniques by treating raw data as text [12]. Existing approaches for learning discrete codes typically use a fixed-sized code-book and compresses data at a constant bitrate regardless of the semantic information contained in its data.

In **MP3** encoding, VBR is achieved by computing the maximum allowable noise within a frame of audio and iteratively increasing the granularity of quantization until it falls within the allowable reconstruction noise as determined by a psycho-acoustic model hand-tuned on several "difficult" music tracks [4].

Multi-model approaches try to learn a hierarchical stack of autoencoders [34] or multiple autoencoders [5]. Each autoencoder learns at a different bitrate (temporal resolution), enabling the selection of latent codes at different fixed granularities. Multi-model methods require multiple rounds of training or multiple models for each bitrate. At inference time, there is no automatic way to determine how many codes are required to maximize the rate-distortion trade-off. Add-on techniques may be used to determine code allocation. For example, Yang et al. [35] inspect the entropy of the latent variable and apply post-hoc techniques to decide how many codes to allocate.

Structured dropout approaches such as Nested Dropout [25] introduce noise during training to impose order in an autoencoder's latent space (similar in spirit to PCA [21]). They define a geometric distribution where dropping one unit causes all units to its right to also be dropped. Nested Dropout is used in the Soundstream audio codec [36], which reports positive subjective evaluations of low bitrate outputs. However, structured dropout does not inform us of how many codes we need to prune, and it is difficult to train—the right-most (most frequently dropped) latent codes are typically of poor quality as they receive few training gradients [14].

SlowAE [6] produces **event-based representations** by outputting features at each time-step t, then quantizing the features at each time-step and compressing via run-length encoding across the time dimension. This approach is not fully end-to-end and many additional post-processing steps (each with hand-tuned parameters) are required to determine the number of codes to generate. We instead allocate a fixed maximum number of codes (independent of t) for a single segment of audio, and then output a quantized distribution over these codes.

Our approach combines ideas from these previous work. In effect, our model is *multi-model* in that it can reduce its latent modeling capacity by having a smaller input latent array. Our causal attention mechanism imposes latent ordering similar to Nested Dropout.

3 Problem Statement

Given a continuous data sequence $x_{[1:T]} \in \mathbb{R}^C$ chunked into a set of non-overlapping windows $x_w \in \{x_1, ..., x_W\}$ (e.g. a flattened image patch or sequence of audio), we aim to train an autoencoder to encode x_w into a set of n discrete latent codes $z = q(x_w, n), z \in \mathbb{C}^n$ where $\mathbb{C}$ is the alphabet of the discrete code, typically represented as some integer. The encoder q must adapt the dimensionality of z on-demand. z can then be decoded back into the original input space via a learned decoder: $\hat{x_w} = d(z)$.

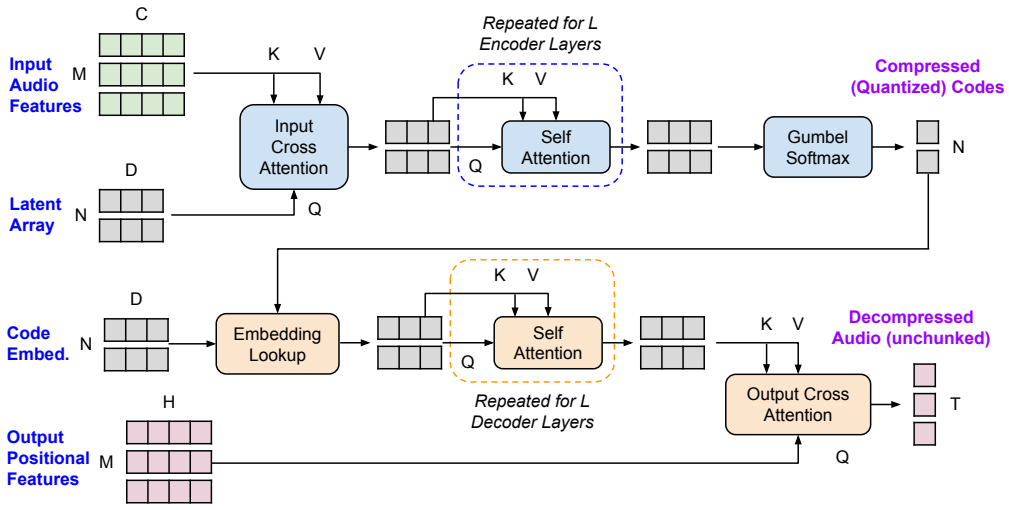

Figure 1: **Compression:** We encode input audio features into a fixed-size representation via a learned latent code array, imposing order via L rounds of causal self-attention. Gumbel softmax is used to quantize the array into a distribution over the code space. **Decompression:** Quantized codes are embedded via a learned lookup table, going through L additional rounds of causal self-attention. Attention outputs are attended over via 1D Fourier positional features to produce a decompressed T-length audio sequence.

To produce a variable number of codes to represent the input data, our framework must recommend the number of codes to allocate for a given window of input data x_w. We define a **code allocation scoring function** $s(x_w) \geq 0$ that represents the relative importance of a given window to other windows in terms of their semantic content. To choose n_w for each window of data, we can compute the scoring function on all W windows of the input, and allocate based on the percentage contributed by the score produced by each window:

$$n_w = \frac{s(x_w)}{\sum_{i=1}^{W} s(x_i)} \times N$$

where N is the total number of codes the user wishes to allocate for the x. This can also be formulated in terms of a *target average bitrate*, which is commonly used in software that renders MP3s or videos.

By learning a variable latent autoencoder along with a scoring function, we are able to produce a variable number of codes per window of input data. The objective is to minimize the reconstruction error of the autoencoder over the entire input.

4 Methodology

We propose a simple, end-to-end framework for learning to encode a window of continuous data into a variable number of discrete latent codes. In contrast to existing fixed bitrate quantization schemes, our framework allows users to incrementally remove codes (corresponding to increasing compression rates and lower bitrate), returning data with gradually increasing degradation [25].

4.1 Perceiver Quantizer

The core of our architecture design builds upon the success of the Perceiver IO [10] architecture. This is different from most previous work that relies on some form of convolution-based autoencoder [5, 36]. The Perceiver is an attention-based model that enables an $M \times C$ input array to be coalesced into an $N \times D$ latent array, then decoded back into the input space. These input and latent spaces sizes are decoupled, and model learns to *reshape* the input data into latent space via a cross attention mechanism. The self-attention mechanism within the latent space naturally enables us to change the number of latent codes during inference time (e.g., by deleting rows of the $N \times D$ latent array).

Jaegle et al. [10] showed promising results for multimodal auto-encoding, demonstrating Perceiver's ability to handle images, audio and classification labels. However, in their experiments, the Perceiver used a pre-determined number of latents N and there was no quantization bottleneck. We propose a **Perceiver Quantizer** architecture to solve our problem statement as follows (see Figure 1):

We add a quantization bottleneck to the middle of the Perceiver. Specifically, we introduce a Gumbel Softmax [11] layer to quantize the latent space, similar to the approach used in DALL-E [23]. The encoder must thus learn how to compress the data into discrete codes.

As depicted in Figure 1, we take as input an audio sequence "chunked" into an array of size $M \times C$ where $M = \frac{T}{C}, C = 16$. We attend over the input features via a learned, fixed-size latent array with maximum N codes and D dimensionality. The encoder performs L rounds of causal self-attention over the fixed-size encoding, and a Gumbel Softmax layer is used to perform (differentiable) quantization into a distribution over latent codes.

To decompress the quantized codes, the codes are embedded via a learned $N \times D$ lookup table. This goes through L additional rounds of causal self-attention. The self-attention outputs are attended over via (fixed) Fourier positional features for M positions with H dimensionality. The cross-attention outputs are then unchunked to produce a decompressed T-length audio sequence.

Inroducing Order via Causal Self-Attention Although the Perceiver architecture enables us to remove latent codes during inference time, it is unclear which latents we should remove. A possible end-to-end approach is to learn a bitmask over the latents that determines which latents are more important [15], and prune the unimportant latents according to some code allocation strategy (Sec. 4.2). However, since such a pruning strategy leads to non-contiguous codes, it requires storing both the bitmask and the codes in order to reconstruct the input.

In practice, we want to produce a contiguous set of latent codes that we can compactly store, which can be achieved by imposing an ordering to the latent variables. To this end, we propose changing the self-attention from full attention to causal attention [22] (e.g., any given latent can only attend to its left latents). This is similar to Nested Dropout in that it imposes an ordering of the latents, but it does not introduce stochastic noise that hinders training.

We hypothesize that this setup should enable us to learn an ordered discrete representation of data. By the definition of causal attention, the codes on the left are unaffected by the right. This ordered set of latent codes enables us to construct $q(x_w, n)$ for an arbitrary n by simply discarding latent codes on the right during inference. We expect that removing the latent codes from right to left results in a gradual degradation of reconstruction quality compared to a non-causal self-attention.

4.2 Code Allocation Strategies

Now that we have proposed a method to learn $q(x_w, n)$, we propose a few strategies to construct $s(x_w)$. We plan to compare these strategies to see which one is the simplest and most effective.

Attention-Based Scoring A parameter-free approach to allocate codes within a given window of input $s(x_w)$ is to introspect the attention weights of the decoder $d(z)$ during inference. To reconstruct the input data, the decoder must attend, for each output position o, to the latent space, using its cross attention mechanism by assigning an attention weight $a_1^{(o)}, ..., a_N^{(o)}$ for all N latent codes. We consider the average attention weight over all O output positions assigned to each latent position as $a_i = \sum_{o=1}^{o=O} a_i^{(o)}/O$, Note that this averaging can be extended to the case where *multi-headed attention* is used, where we would take the average over the different heads.

With the average attention score computed a_i, we treat this value as the importance of the latent code. We define the scoring function as the total of these cumulative sums: $s(x_w) = \sum_{i=1}^{N} \sum_{j=i}^{N} a_j$.

Intuitively, we can understand this heuristic as follows. When a latent is pruned (or, equivalently, not allocated), it is equivalent to having zero attention weight applied to it from the decoder's perspective. The cumulative sum is used to represent the cost of removing all codes to the right $z_{\geq j}$ of the j-th code. The drawback of this approach is that attention-based scoring requires running both the encoder and decoder during inference, and may lead to slower compression encoding times.

Entropy-Based Scoring An alternative parameter-free approach is based on the ideas proposed in Variational Bayesian Quantization [35]. Because we defined our quantization procedure using the Gumbel Softmax approach, our latent codes are probabilistic and our autoencoder is a Discrete VAE [11]. Thus, we can infer the entropy of each latent code to see how *uncertain* each code is.

We can employ the heuristic that highly uncertain codes should be pruned because their information is deemed inaccurate anyway. We propose computing the entropy of each latent code: $H(z_i) = -\sum_c^{\mathbb{C}} P(z_i^{(c)}) \log P(z_i^{(c)})$, where $P(z_i^{(c)})$ is the probability of producing code c from alphabet $\mathbb{C}$. This can be easily computed during inference without running the decoder.

By considering all entropies in a given window of data, we define our score as $s(x_w) = \sum_{i=1}^{N} \sum_{j=i}^{N} 1/H(z_i)$. The intuition for this scoring function is similar to that of Attention-Based Scoring.

5 Experimental Protocol

5.1 Datasets

We evaluate our model alongside baselines in three increasingly challenging speech settings: Librispeech [20] is a benchmark dataset for speech recognition, comprising 1000 hours of single speakers (1,166) reading from books. TED-LIUM [8] contains 452 hours of audio from online single-speaker presentations, containing more varied cadence and unique speakers (1,970) compared to Librispeech. This American Life (TAL) [17] poses a more challenging setting, comprising 663 podcasts, each containing multiple speakers, music, and other noise. TAL has a total of 6,608 unique speakers across 637 hours of audio. All raw audio is re-sampled to a 16 kHz sampling rate and segmented into 3-second snippets, corresponding to audio experiment settings from van den Oord et al. [31]

5.2 Evaluation

We measure the **compression rate** of each algorithm via average bit rate in bits per second (bps). A sequence of n latent codes $z \in \mathbb{C}^n$ with vocabulary size $|\mathbb{C}|$ comprises $\log_2(|\mathbb{C}|^n)$ bits.

We will also assess **audio reconstruction quality** of each approach. Following Arik et al. [1], we compute the following losses between the reference audio waveform x and reconstructed waveform $\hat{x}$: To capture how the reconstructed waveform fits large-amplitude spectral components, we compute *Spectral convergence (SC)*:

$$\text{SC} = \parallel |\text{STFT}(x)| - |\text{STFT}(\hat{x})| \parallel_F \; / \; \parallel |\text{STFT}(x)| \parallel_F$$

where STFT is the Short-Time Fourier Transform [7] and $\| \cdot \|_F$ represents the Frobenius norm over time and frequency. We additionally compute *Mean Squared Error (MSE)* as MSE $=\| x - \hat{x} \|_2^2$.

For **subjective evaluation** of reconstruction quality, we will conduct Mean Opinion Score (MOS) tests using Amazon Mechanical Turk for rating audio quality [30, 5]. Each subject will be instructed to listen to a sample of reconstructed audio and rate its *naturalness* and *clarity* on a five-point Likert scale (1: Bad, 2: Poor, 3: Fair, 4: Good, 5: Excellent).

Modeling efficiency For each approach, we will measure the complexity of training a model (or models) to perform VBR quantization, including the number of independent models to be trained, the total number of parameters across all models, and the total wall-clock training time. Each of these metrics will be reported as their raw values for each target compression rate investigated.

5.3 Experiments

RQ1: Does our framework allow us to learn variable bitrate models? As our aim is to learn a single model capable of encoding continuous data at various bitrates, we assess the reconstruction quality of our models at several fixed average bitrates ($B \coloneqq \{b_0, \ldots, b_n\}$ such that $b_i < b_{i+1}$). For each target bitrate b_i, we compare the following five approaches:

Our first multi-model baseline follows Dhariwal et al. [5]: we train a different constant bitrate model for each target bitrate b_i. Each of the $|B|$ models is trained independently (**MM-Ind**), and for a target bitrate b_i we use only the corresponding model to encode our data.

Following Williams et al. [34], we next train a different constant bitrate model for each target bitrate b_i. These models are jointly trained in a hierarchical cascade (**MM-Hier**), such that the b_i model receives as input audio encoded at b_{i+1} (output of the b_{i+1} model) and outputs latent codes at b_i. For target bitrate b_i we pass the input data through the stack of i hierarchical encoders: $b_j; j \geq i$. This may serve to be computationally infeasible for large bitrate ranges, and we only compare this baseline in situations where it can be trained.

We then train our Perceiver Quantizer with full self-attention (**Full-Attn**), pruning codes from the right-most rows in the $N \times D$ latent array until a b_i bitrate representation remains. We thus observe whether self-attention in the latent space is inherently robust to pruning and allows VBR compression.

We compare two methods of imposing order in the latent codes: training our Perceiver Quantizer with Nested Dropout [25] (**Nest**), and with causal attention masking (**Causal**). We again prune codes from the right-most, until a b_i bitrate representation remains.

Note that our architecture differs from convolutional approaches used in previous work, so direct comparison is not possible with Dhariwal et al. [5], Williams et al. [34]. For fair comparison, we compare results using the same Perceiver architecture, varying the pruning technique used to produce variable latent codes. Specifically, **MM-Ind** and **MM-Hier** will use the self-attention mechanism in the Perceiver, but the latent space will not adapt (e.g., the $N \times D$ latent array stays constant).

We thus collect $5 \cdot |B|$ data points, and for each method we can compare the quantitative reconstruction quality at a target bitrate b_i. We expect that our causal training approach will result in better reconstruction quality compared to stochastic training with Nested Dropout across all target bitrates, but slightly degraded quality compared to multi-model approaches trained specifically for each bitrate. We aim to establish whether reconstructed audio from our single VBR model is judged to be of comparable quality to multi-model baselines across several target bitrates, when multi-model baselines require significantly greater compute to train.

RQ2: How do we allocate latent codes? In the previous set of experiments, we investigate whether our approaches to VBR quantization can accurately reconstruct audio given varying target compression rates for a given window of input data. We next explore whether adaptive code allocation strategies (Section 4.2) can provide better reconstruction quality at a target bitrate over all windows of a data sample. Here, we split a single audio recording into 3-second windows, encode each window via a VBR model (**Nest** and **Causal**), perform pruning over latent codes for all windows, and then decode each (pruned) window to reconstruct the recording.

For each target bitrate $b_i \in B$, we compare using a fixed pruning strategy (**Fixed**)—where each window of audio is pruned to the same number of codes—against attention-based (**Attn**) and entropy-based (**Entropy**) adaptive pruning. As a baseline, we compare against **MP3**—an industry-standard for lossy audio compression that supports full-file encoding at different bitrates. b_i outputs are encoded using `FFmpeg` [29] set to b_i.

We expect adaptive pruning strategies to allow us to more heavily prune latent codes in audio segments with high noise and/or little substantive speech, while allocating more codes to high-information segments. This should in theory provide better overall reconstruction quality at a target bitrate compared to the baseline. We also seek to measure whether computationally expensive pruning strategies (i.e., Attention-based) provide a perceptible improvement in subjective reconstruction quality compared to faster strategies (i.e., Fixed and Entropy).

Hyperparameters We will investigate whether observed trends depend on model capacity, conducting experiments from Section 5.3 at different maximum bitrates (64, 128, 256 latent codes), model depths (8, 16, 32 layers) and hidden dimensionality (128, 256, 512).

Additionally, in preliminary experiments we have found that both the Gumbel Softmax and Perceiver IO are sensitive to hyperparameters, We will sweep the τ temperature parameter between 2.0 and 0.25 (matching the annealing range used in [2]). We will also experiment with annealing schedules for τ [11, 2].

We will optimize each model using LAMB [9], performing hyperparameter search over training hyperparameters (learning rate, scheduling, warmup, batch size, epochs, and regularization) via Tune [13] and selecting based on best reconstruction quality on a held-out dev set.

References

[1] Sercan Ömer Arik, Heewoo Jun, and Gregory Frederick Diamos. Fast spectrogram inversion using multi-head convolutional neural networks. *IEEE Signal Process. Lett.*, 26(1):94–98, 2019.

[2] Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. In *ICLR*, 2020.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[4] Mike Cheng, M Taylor, R Hegemann, A Leidinger, T Tominaga, N Shibata, et al. The lame project. *Retrieved November*, 14:2009, 2009.

[5] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *CoRR*, abs/2005.00341, 2020.

[6] Sander Dieleman, Charlie Nash, Jesse H. Engel, and Karen Simonyan. Variable-rate discrete representation learning. *CoRR*, abs/2103.06089, 2021. URL `https://arxiv.org/abs/2103.06089`.

[7] Daniel W. Griffin and Jae S. Lim. Signal estimation from modified short-time fourier transform. In *ICASSP*, pages 804–807, 1983.

[8] François Hernandez, Vincent Nguyen, Sahar Ghannay, Natalia A. Tomashenko, and Yannick Estève. TED-LIUM 3: Twice as much data and corpus repartition for experiments on speaker adaptation. In *SPECOMe*, volume 11096, pages 198–208, 2018.

[9] Zhouyuan Huo, Bin Gu, and Heng Huang. Large batch optimization for deep learning using new complete layer-wise adaptive rate scaling. In *AAAI*, pages 7883–7890, 2021.

[10] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs & outputs. *CoRR*, abs/2107.14795, 2021. URL `https://arxiv.org/abs/2107.14795`.

[11] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2017.

[12] Kushal Lakhotia, Evgeny Kharitonov, Wei-Ning Hsu, Yossi Adi, Adam Polyak, Benjamin Bolte, Tu Anh Nguyen, Jade Copet, Alexei Baevski, Adelrahman Mohamed, and Emmanuel Dupoux. Generative spoken language modeling from raw audio. *CoRR*, abs/2102.01192, 2021. URL `https://arxiv.org/abs/2102.01192`.

[13] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *CoRR*, abs/1807.05118, 2018. URL `http://arxiv.org/abs/1807.05118`.

[14] Aiting Liu, Chao Xing, Yang Feng, and Dong Wang. Learning ordered word representations with γ-decay dropout. In *APSIPA*, pages 1–5, 2016.

[15] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization, 2018.

[16] Huanru Henry Mao. A survey on self-supervised pre-training for sequential transfer learning in neural networks. *CoRR*, abs/2007.00800, 2020. URL `https://arxiv.org/abs/2007.00800`.

[17] Huanru Henry Mao, Shuyang Li, Julian J. McAuley, and Garrison W. Cottrell. Speech recognition and multi-speaker diarization of long conversations. In *INTERSPEECH*, pages 691–695, 2020.

[18] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *ICML*, volume 32, pages 1791–1799, 2014.

[19] Davis Yen Pan. Digital audio compression. *Digit. Tech. J.*, 5(2), 1993.

[20] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *ICASSP*, pages 5206–5210, 2015.

[21] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[22] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[23] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, volume 139, pages 8821–8831, 2021.

[24] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *NeurIPS*, pages 14837–14847, 2019.

[25] Oren Rippel, Michael A. Gelbart, and Ryan P. Adams. Learning ordered representations with nested dropout. In *ICML*, volume 32, pages 1746–1754, 2014.

[26] Samik Sadhu, Di He, Che-Wei Huang, Sri Harish Mallidi, Minhua Wu, Ariya Rastrow, Andreas Stolcke, Jasha Droppo, and Roland Maas. Wav2vec-c: A self-supervised model for speech representation learning. *CoRR*, abs/2103.08393, 2021. URL `https://arxiv.org/abs/2103.08393`.

[27] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. In *INTERSPEECH*, pages 3465–3469, 2019.

[28] Jonathan Sterne. *MP3: The meaning of a format*. Duke University Press, 2012.

[29] Suramya Tomar. Converting video formats with ffmpeg. *Linux Journal*, 2006(146):10, 2006.

[30] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *ISCA*, page 125, 2016.

[31] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, pages 6306–6315, 2017.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[33] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[34] Will Williams, Sam Ringer, Tom Ash, David MacLeod, Jamie Dougherty, and John Hughes. Hierarchical quantized autoencoders. In *NeurIPS*, 2020.

[35] Yibo Yang, Robert Bamler, and Stephan Mandt. Variational bayesian quantization, 2020.

[36] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *CoRR*, abs/2107.03312, 2021. URL `https://arxiv.org/abs/2107.03312`.