The article has been accepted for publication by IEEE. Full citation:
Kai Zhen, Mi Suk. Lee, Jongmo Sung, Seungkwon Beack and Minje Kim, "Psychoacoustic Calibration of Loss Functions for Efficient End-to-End Neural Audio Coding," in IEEE Signal Processing Letters, vol. 27, pp. 2159-2163, 2020.
DOI: 10.1109/LSP.2020.3039765.

Psychoacoustic Calibration of Loss Functions for Efficient End-to-End Neural Audio Coding

Kai Zhen, *Student Member, IEEE*, Mi Suk Lee, Jongmo Sung, Seungkwon Beack, Minje Kim, *Senior Member, IEEE*

***Abstract*—Conventional audio coding technologies commonly leverage human perception of sound, or psychoacoustics, to reduce the bitrate while preserving the perceptual quality of the decoded audio signals. For neural audio codecs, however, the objective nature of the loss function usually leads to suboptimal sound quality as well as high run-time complexity due to the large model size. In this work, we present a psychoacoustic calibration scheme to re-define the loss functions of neural audio coding systems so that it can decode signals more perceptually similar to the reference, yet with a much lower model complexity. The proposed loss function incorporates the global masking threshold, allowing the reconstruction error that corresponds to inaudible artifacts. Experimental results show that the proposed model outperforms the baseline neural codec twice as large and consuming 23.4% more bits per second. With the proposed method, a lightweight neural codec, with only 0.9 million parameters, performs near-transparent audio coding comparable with the commercial MPEG-1 Audio Layer III codec at 112 kbps.**

***Index Terms*—Audio coding, deep neural networks, psychoacoustics, network compression**

I. Introduction

AUDIO coding, a fundamental set of technologies in data storage and communication, compresses the original signal into a bitstream with a minimal bitrate (encoding) without sacrificing the perceptual quality of the recovered waveform (decoding) [1], [2]. In this paper we focus on the lossy codecs, which typically allow information loss during the process of encoding and decoding only in inaudible audio components. To this end, psychoacoustics is employed to quantify the audibility in both time and frequency domains. For example, MPEG-1 Audio Layer III (also known as MP3), as a successful commercial audio codec, achieves a near-transparent quality at 128 kbps by using a psychoacoustic model (PAM) [2]. Its bit allocation scheme determines the number of bits allocated to each subband by dynamically computing the masking threshold via a PAM and then allowing quantization error once it is under the threshold [3].

Recent efforts on deep neural network-based speech coding systems have made substantial progress on the coding gain [4]–[6]. They formulate coding as a complex learning process that converts an input to a compact hidden representation. This poses concerns for edge applications with the computational resource at a premium: a basic U-Net audio codec contains approximately 10 million parameters [7]; in [8], vector quantized variational autoencoders (VQ-VAE) [9] employs WaveNet [5] as a decoder, yielding a competitive speech quality at 1.6 kbps, but with 20 million parameters. In addition, recent neural speech synthesizers employ traditional DSP techniques, e.g., linear predictive coding (LPC), to reduce its complexity [10]. Although it can serve as a decoder of a speech codec, LPC does not generalize well to non-speech signals.

Perceptually meaningful objective functions have shown an improved trade-off between performance and efficiency. Some recent speech enhancement models successfully employed perceptually inspired objective metrics, e.g. perceptual attractors [11], energy-based weighting [12], perceptual weighting filters from speech coding [13], and global masking thresholds [14] [15], while they have not targeted audio coding and model compression. Other neural speech enhancement systems implement short-time objective intelligibility (STOI) [16] and perceptual evaluation of speech quality (PESQ) [17] as the loss [18], [19]. These metrics may benefit speech codecs, but do not faithfully correlate with subjective audio quality. Meanwhile, PAM serves as a subjectively salient quantifier for the sound quality and is pervasively used in the standard audio codecs. However, integrating the prior knowledge from PAM into optimizing neural audio codecs has not been explored.

In this paper, we present a psychoacoustic calibration scheme to improve the neural network optimization process, as an attempt towards efficient and high-fidelity neural audio coding (NAC). With the global masking threshold calculated from a well-known PAM [20], the scheme firstly conducts priority weighting making the optimization process focus more on audible coding artifacts in frequency subbands with the relatively weaker masking effect, while going easy otherwise. The scheme additionally modulates the coding artifact to ensure that it is below the global masking threshold, which is analogous to the bit allocation algorithm in MP3 [2]. This is, to our best knowledge, the first method to directly incorporate psychoacoustics to neural audio coding.

II. End-to-end neural audio coding

A. Lightweight NAC Module

Given that neural codecs can suffer from a large inference cost due to their high model complexity, one of our goals is

This work was supported by the Institute for Information and Communications Technology Promotion (IITP) funded by the Korea government (MSIT) under Grant 2017-0-00072 (Development of Audio/Video Coding and Light Field Media Fundamental Technologies for Ultra Realistic Tera-Media).

Kai Zhen is with the Department of Computer Science and Cognitive Science Program at Indiana University, Bloomington, IN 47408 USA. Mi Suk Lee, Jongmo Sung, and Seungkwon Beack are with Electronics and Telecommunications Research Institute, Daejeon, Korea 34129. Minje Kim is with the Dept. of Intelligent Systems Engineering at Indiana University (e-mails: zhenk@iu.edu, lms@etri.re.kr, jmseong@etri.re.kr, skbeack@etri.re.kr, minje@indiana.edu).

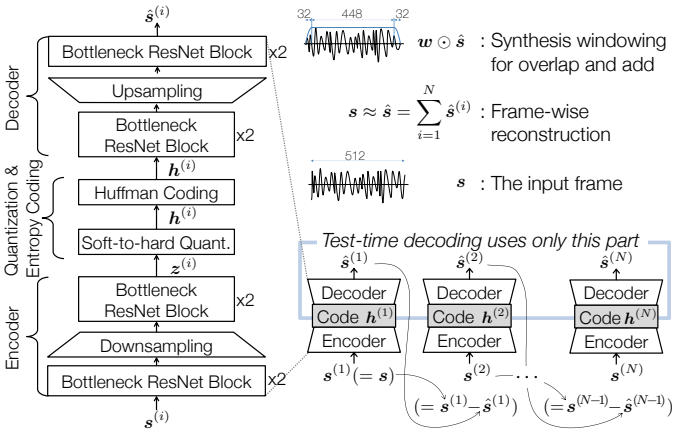

Fig. 1: Schematic diagrams for NAC. The residual coding pipeline for CMRL consists of multiple NAC autoencoding modules. Training and test-time encoding uses all blocks while the test-time decoding uses only the decoder portion.

TABLE I: The 1D-CNN NAC module architecture (Fig. 1). The shape of feature maps is (frame length, channel); the kernel shape is (kernel size, in channel, out channel).

System	Layer	Input shape	Kernel shape	Output shape
Encoder	Change channel	(512, 1)	(9, 1, 100)	(512, 100)
	1st bottleneck	(512, 100)	[(9, 100, 20) (9, 20, 20) (9, 20, 100)] ×2	(512, 100)
	Downsampling	(512, 100)	(9, 100, 100)	(256, 100)
	2nd bottleneck	(256, 100)	[(9, 100, 20) (9, 20, 20) (9, 20, 100)] ×2	(256, 100)
	Change channel	(256, 100)	(9, 100, 1)	(256, 1)
Soft-to-hard quantization & Huffman coding				
Decoder	Change channel	(256, 1)	(9, 1, 100)	(256, 100)
	1st bottleneck	(256, 100)	[(9, 100, 20) (9, 20, 20) (9, 20, 100)] ×2	(256, 100)
	Upsampling	(256, 100)	(9, 100, 100)	(512, 50)
	2nd bottleneck	(512, 50)	[(9, 50, 20) (9, 20, 20) (9, 20, 50)] ×2	(512, 50)
	Change channel	(512, 50)	(9, 50, 1)	(512, 1)

to demonstrate the advantage of the proposed pychoacoustic loss function on model compression. To that end, we choose a compact neural audio coding (NAC) module as the building block. The NAC module is a simplified version of a convolutional neural network (CNN)-based autoencoder [21] with only 450K parameters. As shown in Fig. 1, it consists of a stack of bottleneck blocks as in [22], each of which performs a ResNet-style residual coding [23]. The code vector produced by its encoder part is discretized into a bitstring via the soft-to-hard quantization process originally proposed in [24] for image compression. We detail the description as follows.

1) Encoder: The CNN encoder maps an input frame of T time-domain samples, $\boldsymbol{s} \in \mathbb{R}^T$ to the code vector, i.e., $\boldsymbol{z} \leftarrow \mathcal{F}_{\text{enc}}(\boldsymbol{s})$. Striding during the 1D convolution operation can downsample the feature map. For example, $\boldsymbol{z} \in \mathbb{R}^{T/2}$ when the stride is set to be 2 and applied once during encoding. The detailed architecture is summarized in TABLE I.

2) Soft-to-hard quantization: Quantization replaces each real-valued element of the code vector $\boldsymbol{z}$ with a kernel value chosen from a set of K representatives. We use soft-to-hard quantizer [24], a clustering algorithm compatible with neural networks, where the representatives are also trainable. During training, in each feedforward routine, the c-th code value z_c is assigned to the nearest kernel out of K, $\boldsymbol{\beta} \in \mathbb{R}^K$, which have been trained so far. The discrepancy between z_c and the chosen kernel $h_c \in \{\beta_1, \beta_2, \cdots, \beta_K\}$ (namely the quantization error) is accumulated in the final loss, and then reduced during training via backpropagation (i.e., by updating the means and assignments). Specifically, the cluster assignment is conducted by calculating the distance, $\boldsymbol{d} \in \mathbb{R}^K$, between the code value and all kernels, and then applying the softmax function to the negatively scaled distance to produce a probabilistic membership assignment: $\boldsymbol{a} \leftarrow \text{softmax}(-\alpha \boldsymbol{d})$. Although we eventually need a hard assignment vector $\mathbf{a}$, i.e., a one-hot vector that indicates the closest kernel, during training the quantized code $\boldsymbol{h}$ is acquired by a soft assignment, $\boldsymbol{a}^\top \boldsymbol{\beta}$, for differentiability. Hence, at the test time, $\mathbf{a}$ replaces $\boldsymbol{a}$ by turning on only the maximum element. Note that a larger scaling factor α makes $\boldsymbol{a}$ harder, making it more similar to $\mathbf{a}$. Huffman coding follows to generate the final bitstream.

3) Decoder: The decoder recovers the original signal from the quantized code vector: $\hat{\boldsymbol{s}} = \mathcal{F}_{\text{dec}}(\boldsymbol{h})$, by using an architecture mirroring that of the encoder (TABLE I). For upsampling, we use a sub-pixel convolutional layer proposed in [25] to recover the original frame length T.

4) Bitrate Analysis and Control: The lower bound of the bitrate is defined as $|\boldsymbol{h}|\mathcal{H}(\boldsymbol{h})$, where $|\boldsymbol{h}|$ is the number of down-sampled and quantized features per second. The entropy $\mathcal{H}(\boldsymbol{h})$ forms the lower bound of the average amount of bits per feature. While $|\boldsymbol{h}|$ is a constant given a fixed sampling rate and network topology, $\mathcal{H}(\boldsymbol{h})$ is adaptable during training. As detailed in [24], basic information theory calculates $\mathcal{H}(\boldsymbol{h})$ as $-\sum_k p(\beta_k) \log_2 p(\beta_k)$, where $p(\beta_k)$ denotes the occurrence probability of the k-th cluster defined in the soft-to-hard quantization. Therefore, during model training, $\mathcal{H}(\boldsymbol{h})$ is added to the loss function as a regularizer navigating the model towards the target bitrate. Initiated as 0.0, the blending weight increases by 0.015 if the actual bitrate overshoots the target and decreases by that amount otherwise. Because this regularizer is well defined in the literature [24] [21] [26], we omit it in following sections for simplicity purposes.

B. Cross-Module Residual Learning

To scale up for high bitrates, cross-module residual learning (CMRL) [26] implants the multistage quantization scheme [27] by cascading residual coding blocks (Fig. 1). CMRL decentralizes the neural autoencoding effort to a chain of serialized low complexity coding modules, with the input of i-th module being $\boldsymbol{s}^{(i)} = \boldsymbol{s} - \sum_{j=1}^{i-1} \hat{\boldsymbol{s}}^{(j)}$. That said, each module only encodes what is not reconstructed from preceding modules, making the system scalable. Concretely, for an input signal $\boldsymbol{s}$, the encoding process runs all N autoencoder modules in a sequential order, which yields the bitstring as a concatenation of the quantized code vectors: $\boldsymbol{h} = \left[\boldsymbol{h}^{(1)\top}, \boldsymbol{h}^{(2)\top}, \cdots, \boldsymbol{h}^{(N)\top}\right]^\top$. During decoding, all de-

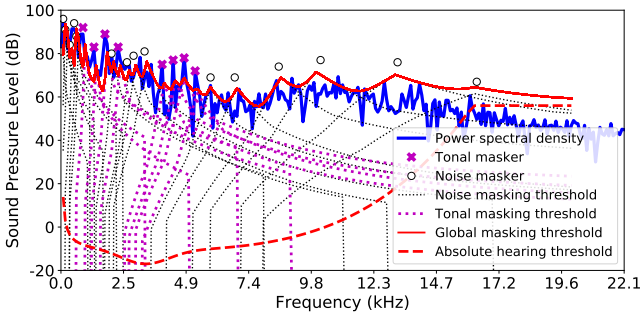

Fig. 2: Visualization of the masker detection, individual and global masking threshold calculation for an audio input.

coders, $\mathcal{F}_{\text{dec}}(\boldsymbol{h}^{(i)})\ \ \forall i$, run to produce the reconstructions that sum up to approximate the initial input signal as $\sum_{i=1}^{N}\hat{\boldsymbol{s}}^{(i)}$.

III. The Proposed Psychoacoustic Calibration

The baseline model uses the sum of squared error (SSE) defined in the time domain: $\mathcal{L}_1(\boldsymbol{s}||\hat{\boldsymbol{s}}) = \sum_{i=1}^{N}\sum_{t=1}^{T}\left(\hat{s}_t^{(i)} - s_t^{(i)}\right)^2$. In addition, another loss is defined in the mel-scaled frequency domain to weigh more on the low frequency area, as the human auditory system does, $\mathcal{L}_2(\boldsymbol{y}||\hat{\boldsymbol{y}}) = \sum_{i=1}^{N}\sum_{l=1}^{L}\left(y_l^{(i)} - \hat{y}_l^{(i)}\right)^2$, where $\boldsymbol{y}$ stands for a mel spectrum with L frequency subbands as proposed in [21].

A. Psychoacoustic Model-1

Without loss of generality, we choose a basic PAM that computes simultatenous masking effects for the input signal as a function of frequency, while the temporal masking effect is not integrated. According to PAM-1 defined in [2], for an input frame, it (a) calculates the logarithmic power spectral density (PSD) $\boldsymbol{p}$; (b) detects tonal and noise maskers, followed by decimation; (c) calculates masking threshold for individual tonal and noise maskers $\boldsymbol{U} \in \mathbb{R}^{F\times R}, \boldsymbol{V} \in \mathbb{R}^{F\times B}$, where R and B are the number of maskers. The global masking threshold at frequency bin f is accumulated from each individual masker in (c) along with the absolute hearing threshold $\boldsymbol{Q}$ [20], as $m_f = 10\log_{10}\left(10^{0.1Q_f} + \sum_r 10^{0.1U_{f,r}} + \sum_b 10^{0.1V_{f,b}}\right)$. Fig. 2 shows an example of $\boldsymbol{p}$ of a signal and its global masking threshold based on the simultaneous masking effect.

Global masking threshold as discussed is used in various conventional audio codecs to allocate minimal amount of bits without losing the perceptual audio quality. Typically, the bit allocation algorithm optimizes n_f/m_f (NMR), where n_f denotes the power of the noise (i.e., coding artifacts) in the subband f and m_f is the power of the global masking threshold. In an iterative process, each time the bit is assigned to the subband with the highest NMR until no more bit can be allocated [28]–[30]. The global masking curve acquired via PAM-1 comprises both input-invariant prior knowledge as in the absolute hearing threshold and input-dependent masking effects. We propose two mechanisms to integrate PAM-1 into NAC optimization: priority weighting and noise modulation.

B. Priority Weighting

During training we estimate the logarithmic PSD $\boldsymbol{p}$ out of an input frame $\boldsymbol{s}$, as well the global masking threshold $\boldsymbol{m}$ to define a perceptual weight vector, $\boldsymbol{w} = \log_{10}(\frac{10^{0.1\boldsymbol{p}}}{10^{0.1\boldsymbol{m}}} + 1)$: the log ratio between the signal power and the masked threshold, rescaled from decibel. Accordingly, we define a weighting scheme that pays more attention to the unmasked frequencies:

$$\mathcal{L}_3(\boldsymbol{s}||\hat{\boldsymbol{s}}) = \sum_i \sum_f w_f \left(x_f^{(i)} - \hat{x}_f^{(i)}\right)^2, \tag{1}$$

where $x_f^{(i)}$ and $\hat{x}_f^{(i)}$ are the f-th magnitude of the Fourier spectra of the input and the recovered signals for the i-th CMRL module. The intuition is that, if the signal's power is greater than its masking threshold at the f-th frequency bin, i.e. $p_f > m_f$, the model tries hard to recover this audible tone precisely: a large w_f enforces it. Otherwise, for a masked tone, the model is allowed to generate some reconstruction error. The weights are bounded between 0 and ∞, whose smaller extreme happens if, for example, the masking threshold is too large comparing to the sufficiently soft signal.

C. Noise Modulation

The priority weighting mechanism can accidentally result in audible reconstruction noise, exceeding the mask value m_f, when w_f is small. Our second psychoacoustic loss term is to modulate the reconstruction noise by directly exploiting NMR, n_f/m_f, where $\boldsymbol{n}$ is the power spectrum of the reconstruction error $\boldsymbol{s} - \sum_{i=1}^{N}\hat{\boldsymbol{s}}^{(i)}$ from all N autoencoding modules. We tweak the greedy bit allocation process in the MP3 encoder that minimizes NMR iteratively, such that it is compatible to the stochastic gradient descent algorithm as follows:

$$\mathcal{L}_4 = \max_f \left(\text{ReLU}\left(\frac{n_f}{m_f} - 1\right)\right). \tag{2}$$

The rectified linear units (ReLU) function excludes the contribution of the inaudible noise to the loss when $n_f/m_f - 1 < 0$. Out of those frequency bins where the noise is audible, the max operator selects the one with the largest NMR, which counts towards the total loss. The process as such resembles MP3's bit allocation algorithm, as it tackles the frequency bin with the largest NMR for each training iteration.

IV. Experiments

A. Experimental Setup

1) Data Preparation and Hyperparameters: Our training dataset consists of 1,000 single-channel clips of commercial music, spanning 13 genres. Each clip is about 20 seconds long, amounting to about 5.5 hours of play time. The sampling rate is 44.1 kHz and downsampled to 32 kHz for the lower bitrate setup. Each frame contains $T = 512$ samples with an overlap of 32 samples, where a Hann window is applied to the overlapping region. Note that the choice of frame size is to align the system's hyperparameters to the previous work [21], [26], [31], but it does not necessarily mean that 512 results in an enough frequency resolution for PAM-based lost terms. For training, hyperparameters are found based on validation with another 104 clips: 128 frames for the batch size; $\alpha = 300$ for the initial softmax scaling factor; 2×10^{-4} for the initial learning rate of the Adam optimizer [32], and 2×10^{-5}