ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9413678

SOURCE-AWARE NEURAL SPEECH CODING FOR NOISY SPEECH COMPRESSION

Haici Yang[1], *Kai Zhen*[1], *Seungkwon Beack*[2], *Minje Kim*[1]

[1]Indiana University, Department of Intelligent Systems Engineering, Bloomington, IN, USA
[2]Electronics and Telecommunications Research Institute, Daejeon, South Korea

ABSTRACT

This paper introduces a novel neural network-based speech coding system that can process noisy speech effectively. The proposed source-aware neural audio coding (SANAC) system harmonizes a deep autoencoder-based source separation model and a neural coding system, so that it can explicitly perform source separation and coding in the latent space. An added benefit of this system is that the codec can allocate a different amount of bits to the underlying sources, so that the more important source sounds better in the decoded signal. We target a new use case where the user on the receiver side cares about the quality of the non-speech components in the speech communication, while the speech source still carries the most important information. Both objective and subjective evaluation tests show that SANAC can recover the original noisy speech better than the baseline neural audio coding system, which is with no source-aware coding mechanism, and two conventional codecs.

Index Terms— Speech enhancement, speech coding, source separation

1. INTRODUCTION

Breakthroughs made in deep learning for the past decade have shown phenomenal performance improvements in various pattern recognition tasks, including media compression and coding. Seminal works are proposed in the lossy image compression domain, where autoencoders are a natural choice. With an autoencoder, the encoder part converts the input signal into a latent feature vector, followed by the decoder that recovers the original input [1, 2]. Compression is achieved when the number of bits used to represent the latent vector (or code) is smaller than that of the raw input signal. With increased computational complexity, the deep autoencoders have shown superior compression performance to traditional technology.

Neural speech coding is an emerging research area, too. Autoregressive models, such as WaveNet [3], have shown a transparent perceptual performance at a very low bitrate [4, 5], surpassing that of traditional coders. Another branch of neural speech coding systems takes a frame-by-frame approach, feeding time-domain waveform signals to an simpler end-to-end autoencoding network. Kankanahalli proposes a model that consists of fully convolutional layers to integrate dimension reduction, quantization, and entropy control tasks [6]. Cross-module residual learning (CMRL) inherits the convolutional pipeline and proposes a cascading structure, where multiple autoencoders are concatenated to work on the residual signal produced by the preceding ones [7]. In [8], CMRL is coupled with a trainable linear predictive coding (LPC) module as a preprocessor. It further improves the performance and lowers the model complexity down to 0.45 million parameters, eventually outperforming AMR-WB [9].

In this work, we widen the scope of speech coding applications by taking into account noisy speech as the input. Additional sound sources often accompany real-world speeches. However, traditional speech codecs are mostly based on the speech production models [10, 9], thus lacking the ability to model the non-speech components mixed in the input signals. Efforts to address the problem are partly reflected in the MPEG unified speech and audio coding (USAC) standard [11, 12]. USAC tackles speech signals in the mixture condition by switching between different tools defined for different kinds of signals, such as speech and music. However, the switching decision does not consider the mixed nature within the frame, which requires explicit source separation. Meanwhile, AMR-WB's discontinuous transmission (DTX) mode also considers the mixed nature of input speech by deactivating the coding process for the non-speech periods [9]. Lombard et al. improved DTX by generating artificial comfort noise that smooths out the discontinuity [13]. Still, for the frames where both speech and non-speech sources co-exist, it is difficult to effectively control the bitrate using DTX. Similar ideas have been used in transform coders for audio compression, where the dynamic bit allocation algorithm based on psychoacoustic models can create a spectral hole in low bitrate cases. Intelligent noise gap filling can alleviate the musical noise generated from this quantization process [14, 15], while it is to reduce the artifact generated from the coding algorithm, rather than to model the non-stationary noise source separately from the main source.

To that end, we propose source-aware neural audio coding (SANAC) to control the bit allocation to multiple sources differently. SANAC does not seek a speech-only reconstruction, e.g., denoising the mixture input and coding it simultaneously [16]. Instead, we target the use case where the user still wants the code to convey the non-speech components to better understand the transmitter's acoustic environment. We empirically show that the sources in the mixture can be assigned with unbalanced bitrates depending on their perceptual or applicational importance and entropy in the latent space, leading to a better objective and subjective quality.

2. MODEL DESCRIPTION

The proposed SANAC system harmonizes a source separation module into the neural coding system. Our model performs explicit source separation in the feature space to produce source-specific codes, which are subsequently quantized and decoded to recover the respective sources. The source-specific code vectors are learned using a masking-based approach as in TasNets [17, 18], while we utilize the orthogonality assumption between the source-specific code vectors to withdraw the separator module in the TasNet architec-

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

ture and reduce the encoder complexity. Soft-to-hard quantization [2] quantizes the real-valued source-specific codes. Bitrate control works on each sources as well.

2.1. Orthogonal code vectors for separation

As a codec system, the model consists of an encoder that converts time-domain mixture frame $\boldsymbol{x} \in \mathbb{R}^N$ into code vector $\boldsymbol{z} \in \mathbb{R}^D$: $\boldsymbol{z} \leftarrow \mathcal{F}_{\text{enc}}(\boldsymbol{x})$. To marry the source separation concept, we assume K mask vectors $\boldsymbol{m}^{(k)} \in \mathbb{R}^D$ that can decompose the code vector into K components: $\sum_{k=1}^K m_d^{(k)} = 1$. Note that $m_d^{(k)}$ is the probability of d-th code value belonging to k-th source. In addition, we further assume that this probabilistic code assignments to the sources are actually determined by one-hot vectors, so that the masking process assigns each code value to only one source, i.e.,

$$m_d^{(k)} = \begin{cases} 1 & \text{if } \arg\max_j m_d^{(j)} = k \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The TasNet models estimate similar masking vectors via a separate neural network module, which led to the state-of-the-art separation performance. In there, the estimated mask values are drastically distributed to either near zero or one, making the masked code vectors nearly orthogonal from each other. However, the sigmoid-activated masks in the TasNet architecture do not specifically assume a hard assignment.

From now on, we assume orthogonal code vectors per source as a result of hard masking, i.e., $\boldsymbol{z}^{(1)} \perp \boldsymbol{z}^{(2)} \perp \cdots \perp \boldsymbol{z}^{(K)}$, where $\boldsymbol{z}^{(k)} = \boldsymbol{m}^{(k)} \odot \boldsymbol{z}$, k-th source's code vector defined by the Hadamard product $\odot$ between the mask and the mixture code.

The proposed orthogonality leads us to a meaningful structural innovation. Instead of estimating the mask vector for every input frame, we can use structured masking vectors that force the code values to be grouped into K exclusive and consecutive subsets. For example, for a two-source case with $D = 8$, $\boldsymbol{m}^{(1)} = [1, 1, 1, 1, 0, 0, 0, 0]^\top$. Hence, we can safely discard the masked-out elements (the latter four elements), by defining its truncated version as $\mathbf{z}^{(k)} \in \mathbb{R}^{D/K}$. The concatenation of the truncated code vectors determines the final code vector: $\boldsymbol{z} = [\mathbf{z}^{(1)\top}, \mathbf{z}^{(2)\top}, \cdots, \mathbf{z}^{(K)\top}]^\top$.

In practice, we implement the encoder as a 1-d convolutional neural network (CNN) of which output is a $2L \times P$ matrix, where $2L$ is the number of output channels (see Figure 1, where $L = 6$, $P = 256$). We collect the first L channels of this feature map (dark red bars) as our codes for speech, i.e., $\mathbf{z}^{(1)}$ corresponds to the vectorized version of the upper half of the feature map of size $L \times P$, or $LP = D/K$, where D should be an integer multiple of K. The other half is for the noise source. Since the decoders are learned to predict individual sources, this implicit masking process can still work for source separation.

2.2. Soft-to-hard quantization

Quantization is a mapping process that replaces a continuous variable with its closest discrete representative. Since it is not a differentiable process, incorporating it into a neural network requires careful consideration. Soft-to-hard quantization showed successful performance both in image and speech compression models [2, 6, 7]. The idea is to formulate this cluster assignment process as a softmax classification during the feedforward process, which finds the nearest one among M total representatives $\boldsymbol{\mu}_m$ for the given code vector $\boldsymbol{y}$ as follows:

$$d_m = \mathcal{E}(\boldsymbol{y} || \boldsymbol{\mu}_m), \quad \boldsymbol{p} = \text{Softmax}(-\alpha \boldsymbol{d}), \tag{2}$$

$$\text{Testing: } \bar{\boldsymbol{y}} = \boldsymbol{\mu}_{\arg\max_m p_m}, \quad \text{Training: } \bar{\boldsymbol{y}} = \sum_{m=1}^M p_m \boldsymbol{\mu}_m,$$

where the algorithm first computes the Euclidean distance vector $\boldsymbol{d}$ against all the representatives (i.e., the cluster means). The negative distance values work as similarity scores for the softmax function. Using the softmax result, the probability vector $\boldsymbol{p}$ of the cluster membership, we can construct the quantized code vector $\bar{\boldsymbol{y}}$: during test time, simply choosing the closest one will do a proper quantization. Since the $\arg\max$ operation is not differentiable, for training, we do a convex combination of the cluster centroids to represent the quantized code, as a differentiable surrogate of the hard assignment process. The discrepancy between training and testing is reduced by controlling the scaling hyperparameter α, which makes the softmax probabilities more drastic once it is large enough (e.g., a one-hot vector in the extreme case). Note that it also learns the cluster centroids $\boldsymbol{\mu}$ as a part of the learnable network parameters rather than employing a separate clustering process to define them.

In previous works the quantization has been on scalar variables, i.e., $\boldsymbol{y} \in \mathbb{R}^1$ [6, 7, 8]. In this work, the soft-to-hard quantization performs vector quantization (VQ). We denote the CNN encoder output by $\boldsymbol{Z} \in \mathbb{R}^{2L \times P}$, which consists of $K = 2$ code blocks: $\boldsymbol{Z} = [\boldsymbol{Z}^{(1)}; \boldsymbol{Z}^{(2)}]$. Then, each code vector for quantization is defined by the p-th feature spanning over L channels: $\boldsymbol{y} = \boldsymbol{Z}^{(k)}_{1:L,p}$, having $L = 6$ as the VQ dimension in our case.

2.3. Source-wise entropy control

The theoretical lower bound of the bitrate, as a result of Huffman coding, can be defined by the entropy of the quantized codes. The frequency of the cluster means defines the entropy of the source-specific codes: $\mathcal{H}\big(\boldsymbol{\mu}^{(k)}\big) = -\sum_{m=1}^M q_m^{(k)} \log q_m^{(k)}$, where $q_m^{(k)}$ denotes the frequency of m-th mean for the k-th source. Meanwhile, the entropy for the code of the mixture signal is smaller than or equal to the sum of the entropy of all sources: $\mathcal{H}(\ddot{\boldsymbol{\mu}}) \leq \sum_{k=1}^K \mathcal{H}\big(\boldsymbol{\mu}^{(k)}\big)$, where $\ddot{\boldsymbol{\mu}}$ is the set of quantization vector centroids learned in a source-agnostic way, i.e., directly from the mixture signals. Therefore, in theory, SANAC cannot achieve a better coding gain than a codec that works directly on the mixture.

However, SANAC can still benefit from the source-wise coding, especially by exploiting the perceptual factors. Our main assumption in this work is that the perceptual importance differs from source-by-source, leading to a coding system that can assign different bitrates to different sources. For noisy speech, for example, we will try to assign more bits to the speech source. Consequently, although the user eventually listens to the recovered mixture of speech and noise (a) the perceptual quality of the speech component is relatively higher (b) the codec can achieve a better coding gain if the noise source' statistical characteristics is robust to low bitrates.

Our argument is based on the codec's ability to control the entropy of the source-specific codes. In SANAC, we adopt the entropy control mechanism proposed in [2], but by setting up a per-source loss between the target $\xi^{(k)}$ and the actual entropy values: $\big(\xi^{(k)} - \mathcal{H}\big(\boldsymbol{\mu}^{(k)}\big)\big)^2$. While this loss does not guarantee the exact bitrate during the test time, in practice, we observe that the actual bitrate is not significantly different from the target.

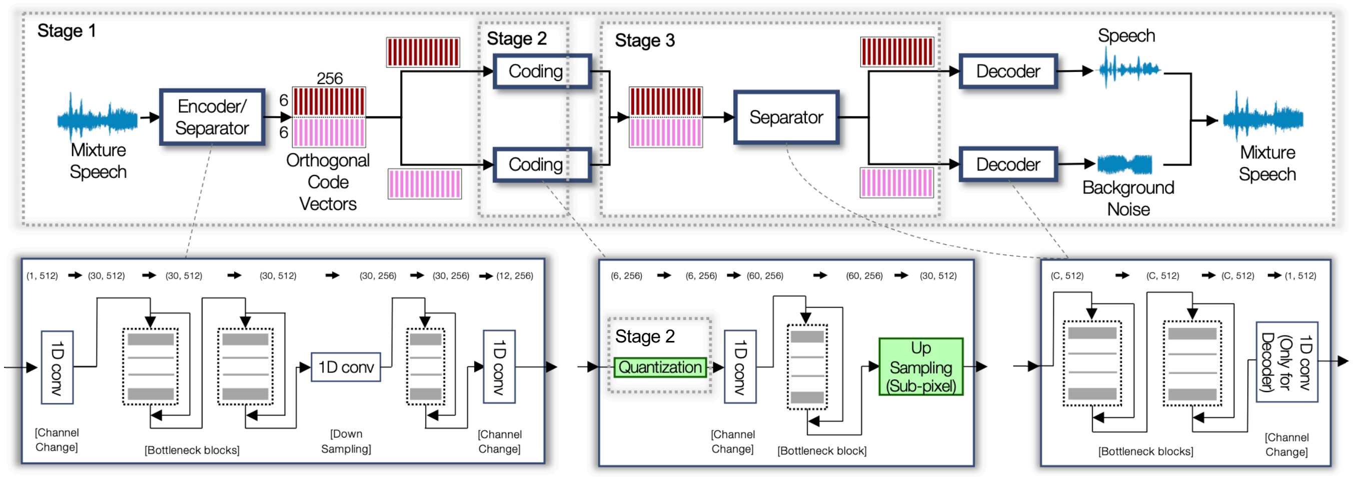

Fig. 1: Schematic diagram of source-aware speech coding. At Stage 1, the Stage 2 and 3 modules are not trained, while Stage 2 updates all parameters except for the Stage 3 modules. At Stage 3, all parameters are optimized.

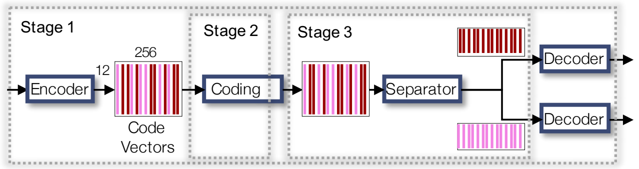

Fig. 2: Schematic diagram of DNN-based baseline model

2.4. Decoding and the final loss

The source-specific truncated codes, after the quantization, $\bar{\boldsymbol{Z}}^{(k)}$, are fed to the decoder part of the network. The decoder function works similarly to Conv-TasNet [18] in that the decoder runs K times to predict K individual source reconstructions from K sourse-specific feature maps as the input. However, SANAC's decoding is different from Conv-TasNet's as the decoder input is the quantized codes. In addition, our model cares about the quality of the recovered mixture, not only the separation quality.

Our training loss considers all these goals, consisting of the main mean squared error (MSE)-based reconstruction term and the entropy control terms. More specifically, for the noisy speech case $\boldsymbol{x} = \boldsymbol{s} + \boldsymbol{n}$ ($k = 1$ for speech and $k = 2$ for noise), the MSE loss is for the speech source reconstruction $\hat{\boldsymbol{s}}$ and the mixture reconstruction $\hat{\boldsymbol{x}}$, while the noise source reconstruction $\hat{\boldsymbol{n}}$ is implied in there. We regularize the total entropy as well as the ratio between the two source-wise entropy values:

$$\begin{aligned}\mathcal{L} = &\lambda_{\text{MSE}}\big(\mathcal{E}_{\text{MSE}}(\boldsymbol{s}||\hat{\boldsymbol{s}}) + \mathcal{E}_{\text{MSE}}(\boldsymbol{x}||\hat{\boldsymbol{x}})\big)\\ &+ \lambda_{\text{EntTot}}\Big(\xi - \mathcal{H}\big(\boldsymbol{\mu}^{(1)}\big) - \mathcal{H}\big(\boldsymbol{\mu}^{(2)}\big)\Big)^2\\ &+ \lambda_{\text{Ratio}}\left(\psi - \frac{\mathcal{H}\big(\boldsymbol{\mu}^{(1)}\big)}{\mathcal{H}\big(\boldsymbol{\mu}^{(2)}\big)}\right)^2,\end{aligned} \tag{3}$$

where ξ and ψ are the target total entropy and the target ratio, respectively.

3. EXPERIMENT

3.1. Dataset

500 and 50 utterances are randomly selected from training and test set of TIMIT corpus [19]. We generate 10 contaminated samples out of each clean utterance by adding 10 different non-stationary background sources, {*bird singing, casino, cicadas, typing, chip eating, frogs, jungle, machine gun, motorcycle, ocean*} used in [20]. Every contaminated speech waveform was segmented into frames of 512 samples (32ms), with overlap of 64 samples. We apply a Hann window of size 128 samples only to the overlapping regions. Since there are $16000/448$ frames per second and each frame produces P code vectors for VQ, for the entropy of a source-specific codebook ξ, the bitrate is $16000P\xi/488$, e.g., 9.14kbps when $P = 256$ and $\xi = 1$.

3.2. Training process

Adam optimizer with an initial learning rate 0.0001 trains the models [21]. Both SANAC and the baseline are trained in three stages. Every jump to next stage is triggered when the validation loss stops improving in 3 consecutive epochs. We stop the training updates after validation loss does not improve for 20 epochs.

- *Stage 1*: For the first three epochs, the model trains the encoder to separate the input into the speech and background sources, that are represented by the two orthogonal code vectors. No quantization is involved in yet, but this stage better initializes the parameters for the quantization process. The encoder consists of a few bottleneck blocks that are commonly used in ResNet [22]. With the bottleneck structure, the input and output feature maps can be connected via an identity shortcut with less filters to learn in between. The encoder module also employs a 1-d convolution layer to downsample the feature map from 512 to 256, followed by another bottleneck block and a channel changing layer to yield two sets of code maps of 6×256 each. The learned source-specific feature maps are fed directly to the channel changer with no quantization, followed by an upsampler and the final decoder. In terms of upsampling, we interlace two adjacent channels into one, doubling the number of features up to 512 while halving the number of channels as introduced in [23], and then adopted for neural speech coding [6, 7].