IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 10, 2019, accepted May 30, 2019, date of publication June 4, 2019, date of current version June 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2920663

DeepVoCoder: A CNN Model for Compression and Coding of Narrow Band Speech

HACER YALIM KELES[1], JAN ROZHON[2], H. GOKHAN ILK[3], (Member, IEEE), AND MIROSLAV VOZNAK[2], (Senior Member, IEEE)

[1]Computer Engineering Department, Ankara University, 06830 Ankara, Turkey
[2]Department of Telecommunications, Faculty of Electrical Engineering and Computer Science, VSB–Technical University of Ostrava, 70800 Ostrava, Czech Republic
[3]IT4 Innovations, VSB–Technical University of Ostrava, 70800 Ostrava, Czech Republic

Corresponding author: H. Gokhan Ilk (ilk@ieee.org)

This work was supported in part by the Ministry of Education, Youth, and Sports Project under Grant LM2015070 and Grant SGS SP2019/41, conducted at the VSB–Technical University of Ostrava, Czech Republic.

ABSTRACT This paper proposes a convolutional neural network (CNN)-based encoder model to compress and code speech signal directly from raw input speech. Although the model can synthesize wideband speech by implicit bandwidth extension, narrowband is preferred for IP telephony and telecommunications purposes. The model takes time domain speech samples as inputs and encodes them using a cascade of convolutional filters in multiple layers, where pooling is applied after some layers to downsample the encoded speech by half. The final bottleneck layer of the CNN encoder provides an abstract and compact representation of the speech signal. In this paper, it is demonstrated that this compact representation is sufficient to reconstruct the original speech signal in high quality using the CNN decoder. This paper also discusses the theoretical background of why and how CNN may be used for end-to-end speech compression and coding. The complexity, delay, memory requirements, and bit rate versus quality are discussed in the experimental results.

INDEX TERMS Convolutional neural network, deep learning, source coding, speech codecs.

I. INTRODUCTION

Speech is the most natural way of communication among humans. Therefore, it is not surprising to see speech coding applications in communications. The need for removing the redundancy for efficient compression and coding has been a challenge for a long time. Although rate distortion theory, which defines the quality with respect to the bit rate, is at the focal point of this discussion [1], we will also focus on other practical issues such as delay (including latency), complexity and memory requirements, which are as important parameters as rate-distortion.

Speech signal has two major forms of redundancy. The first one is called ''short-term'' or ''sample-to-sample'' redundancy [2]. It is possible to observe high correlation between consecutive samples. Linear autoregressive models are an effective means of exploiting short term correlation by considering previous P sample values. The second redundancy stems from pitch, which is actually the fundamental frequency of the almost periodic speech signal. This redundancy, called ''long-term'' only exists in the voiced segments of speech. The rest of the speech is unvoiced and does not have periodicity resulting from the vibration of the vocal cords and looks more like random noise with low energy. Unfortunately, speech is not composed of voiced and unvoiced segments, transition frames, either from voiced to unvoiced or unvoiced to voiced segments, are extremely difficult to model. It is argued in [3] that ''attributes of speech that identify the speaker and speaking style do not vary rapidly over time and hence do not change this rough estimate (true information rate is less than 100 b/s) significantly'' is not always correct as natural speech not only contains lexical information but also speaker identity, speech speed and style and further emotions. Therefore, although speech is extremely redundant for most of the time, it has a complicated structure from time to time. Nevertheless, we believe that a proposed model should be able to handle both ''short-term'' and ''long-term'' redundancy.

Linear autoregressive (AR) models, which lead to adaptive predictive coding of speech signals, date back to half-century ago [2]. The fundamental idea behind adaptive predictive coding is simple: Speech signal is assumed to be stationary for short time intervals and therefore the $a_k, k = 1, 2, \ldots P$ coefficients of an adaptive linear predictive coding (LPC)

The associate editor coordinating the review of this manuscript and approving it for publication was Chenguang Yang.

filter are calculated by a minimum mean square error algorithm for every 10-32 ms of input speech signal, $x[n]$. This duration (where speech signal is assumed to be stationary) may change but speech synthesis is performed according to (1), where P is the prediction order.

$$x[n] = \sum_{k=1}^{P} a_k x\,[\text{n-k}] + e[n] \tag{1}$$

In adaptive predictive coding, there are two challenges. The first one is how to represent and transmit the LPC coefficients for every 10-32 ms. The second challenge is to model the error signal, $e[n]$, which is actually the prediction error, as no prediction is perfect and leaves a residual. The former problem is solved by the line spectral frequencies (LSF) or line spectral pairs (LSP) representation [4] and quantization. It turns out that LPC coefficients in time domain represents the spectral envelope of the signal in the frequency domain. The second challenge however results in more versatile solutions. The ubiquitous are under the analysis-by-synthesis family, regular-pulse [5] and code excited linear prediction (CELP) [6] systems. It is of course possible to represent speech without adaptive prediction. These methods operate either on sinusoidal representation [7] or in the frequency domain [8]. In order to reduce bit rates further, at the expense of speech quality, phase of the residual signal may also be modelled via combining predictive coding with frequency domain solutions on the LPC residual [9].

In recent years, a generative model, i.e. WaveNET, for waveform synthesis has been also proposed [10]. In this approach, each waveform sample is conditioned on the samples at all previous time steps, where the model produces a probability distribution with a set of discrete values for the next sample. Although the WaveNET is composed of causal convolutions [10], the model operates in a probabilistic viewpoint. WaveNET model was used as a speaker-dependent speech synthesis vocoder in [11], where acoustic features are fed into the model as auxiliary features and speech is generated. This unfortunately results in a speaker dependent speech synthesis. This idea is further extended in [3], where WaveNET model is used for both parametric and waveform speech coders. In case of parametric vocoder, a standard parametric open source vocoder [12] has been used to obtain the bit stream that is necessary to drive the WaveNET model that is used as a decoder, operating at 2.4 kb/s. In other words, the encoding of speech is done by a standard speech coder, where this bit stream is passed to a WaveNET model for decoding purposes. In case of a waveform speech coder, the authors mention average bit rates, variable bit rate coders are not suitable for telecommunications purposes due to bit rate allocation purposes with respect to fix bit rate schemes. Nevertheless, the bit rate mentioned in [3] is 42 kb/s.

The major contribution of this paper is the use of two convolutional neural networks (CNN) for compression and coding of voice in an end-to-end solution for encoding and decoding purposes. To the best of our knowledge, deep neural networks (DNNs) are not used for end-to-end speech coding in the literature so far. The end-to-end solution means that input speech is in time domain and represented as raw speech waveform, which does not require any explicit extraction of parameters, i.e. acoustic features; the output is the reconstructed speech waveform. The process also does not require any conversions (such as LSF) or quantization under some circumstances for the CNN encoder. Speech is synthesized by the CNN decoder.

In the remainder of this paper, Section 2 describes the novel DeepVoCoder, which is a DNN based speech compression and coding method with emphasis on CNN and inception modules. In Section 3, we discuss the important parameters of delay, complexity and bit rate versus quality. In section 4, the results of our experiments are presented for a limited number of selected parameters. There are countless number of parameters and combinations that can be used for our DeepVoCoder approach and each choice aims optimization of a different objective. Finally, Section 5 draws conclusions.

II. THE METHOD

Motivated by the recent successes in the deep convolutional models in different domains ([13-18]), a CNN based encoder-decoder type neural network architecture is designed for speech coding. The Inception model in [13] is considered as a reference in the design of our encoding and decoding networks, where the raw speech waveform is analyzed through cascades of multiple inception modules (Fig. 1). Each module encodes the relevant acoustic features of the waveform in multiple-scales. The encoded features at the output layer of the encoder network is utilized as the "source" by the decoder to reconstruct the speech waveform. The details of the encoder and decoder network configurations are provided below.

A. ENCODER NETWORK CONFIGURATION

The encoder network is designed to take 32 ms raw speech samples as the inputs, i.e. a vector of 256 samples (from 8kHz). It is composed of 6 blocks and two additional convolutional layers at the end. We refer to the set of layers between two max-pooling layers, for downsampling, as a block. The block diagram in Fig.1 depicts the encoder architecture.

The network starts with three 1 dimensional convolutional layers of filter sizes 9, 1 and 9, respectively. All these convolutional layers are configured to extract 128 feature vectors. After the first layer, a non-linear activation function, i.e. leaky ReLU [19], is applied after batch normalization [20]. The following two convolutional layers' responses are kept linear. The convolution layer of size 1, in the middle, is used to align the feature planes with a more useful direction after the non-linearity, which makes the second convolution more effective. Then a max-pooling layer is applied to the feature vectors of the third layer with a stride 2 to downsize the sample count from 256 to 128. The extracted features are then fed to an inception block that is very similar to the architecture that is utilized for a vision network developed for image classification [13]; we adapted this network for

one dimensional speech processing, with minor changes in the activation functions.

In each block, only one inception module is included and following an inception module, resultant concatenated multi-scale features are normalized using batch normalization and passed further to leaky ReLU activation before max-pooling. The feature vectors that are computed by the last convolutional layer of the first block are passed through 5 inception modules. After the fifth inception module, two cascaded convolutions are applied to the resultant features. At this level, i.e. before the cascaded convolutions, the number of samples in each feature vector is reduced to 4 and a total of 512 feature vectors are generated to represent the original speech samples. In order to obtain a condensed representation, first convolutional layer projects the feature planes, using size 1 convolutions, to 20 vectors. Following that, without applying any non-linearities, a second convolution layer, using 20 filters with a filter size of 3, is applied to generate the final activations for another 20 feature vectors. The resultant activations are passed through *sigmoid* function to get the encoded speech. As a result, the encoded speech is represented using 80 coefficients in the range (0-1) in order to quantize them easily and more effectively.

Inception Modules: The Inception modules are initially designed for a visual recognition challenge (ILSVRC14) with a vision network called GoogLeNet [13]. The modules are used in a cascaded way to generate deeper and the wider networks; hence called as inception modules to imply a networks-in-networks architecture. The essential components of the design of inception modules are parallel convolutional layers that analyze the same features in multiple scales, using different filter sizes, simultaneously. However, this is costly when the depth of a network is increased. The design of the module efficiently avoids parameter increase by using 1×1 convolutions to reduce the depth before applying convolutions of 3×3 and 5×5 (in images). These 1×1 convolutions project the incoming features to a lower dimensional space that allows increasing the depth without creating computational bottlenecks.

In speech domain, we want to encode both short-term and long-term correlations in the speech samples, hence inception modules are very suitable for our purpose. Pooling of sparse features at the end of inception modules, provide features in varying scales to the upcoming inception modules for further analysis using multiple aperture sizes. Especially in the higher layers, the ones that are close to the output after multiple pooling operations, the receptive fields of the convolutions, i.e. of sizes 3 and 5, become wide enough to learn long-term correlations in a frame effectively using only a few parameters.

To the best of our knowledge, deep neural networks (DNNs) are not used for end-to-end speech coding in the literature so far. Moreover, the inception modules are included in the speech coding research for the first time. Different combinations of inception modules can be designed and evaluated for speech coding purposes; as a baseline, we selected to use exactly the same configurations for the convolutional layers, i.e. the number of filters and filter sizes, with the original image recognition network, yet one-dimensional versions are utilized. Inception modules and the activation functions are also configured as they are in [13], i.e. ReLU. Then we optimized that baseline performance experimentally by using different linear and non-linear activations and improved the quality of the decoded speech. Fig.1 depicts the final design of our encoder network architecture that we obtained our ''best'' results after many experiments. The number of filters in each inception block are given in Table 1 and they are exactly the same as the decoder.

TABLE 1. Decoder network configuration parameters. MP: Max pooling, BN: Batch normalization, LR: Leaky ReLU.

Layer/Block	Number of Filters				Size of Filters			
ConvBlock1	20				1			
	20				3			
	LR							
Upsampling (↑2)								
InceptionBlock1	64	96	16	MP	1	1	1	MP
		LR	32	32		LR	5	1
		128				3		
	Concatenate							
	BN/LR							
Upsampling (↑2)								
InceptionBlock2	128	128	32	MP	1	1	1	MP
		LR	96	64		LR	5	1
		192				3		
	Concatenate							
	BN/LR							
Upsampling (↑2)								
InceptionBlock3	192	96	16	MP	1	1	1	MP
		LR	48	64		LR	5	1
		208				3		
	Concatenate							
	BN/LR							
Upsampling (↑2)								
InceptionBlock4	160	112	24	MP	1	1	1	MP
		LR	64	64		LR	5	1
		224				3		
	Concatenate							
	BN/LR							
Upsampling (↑2)								
InceptionBlock5	128	128	24	MP	1	1	1	MP
		LR	64	64		LR	5	1
		256				3		
	Concatenate							
	BN/LR							
Upsampling (↑2)								
ConvBlock2	128				1			
	128				5			
	BN/LR							
	1				1			

B. DECODER NETWORK CONFIGURATION

The decoder network architecture design is very similar to the encoder architecture. We keep the inception modules exactly the same with respect to the filter sizes and number of filters applied in each block, as in our encoder network; only the

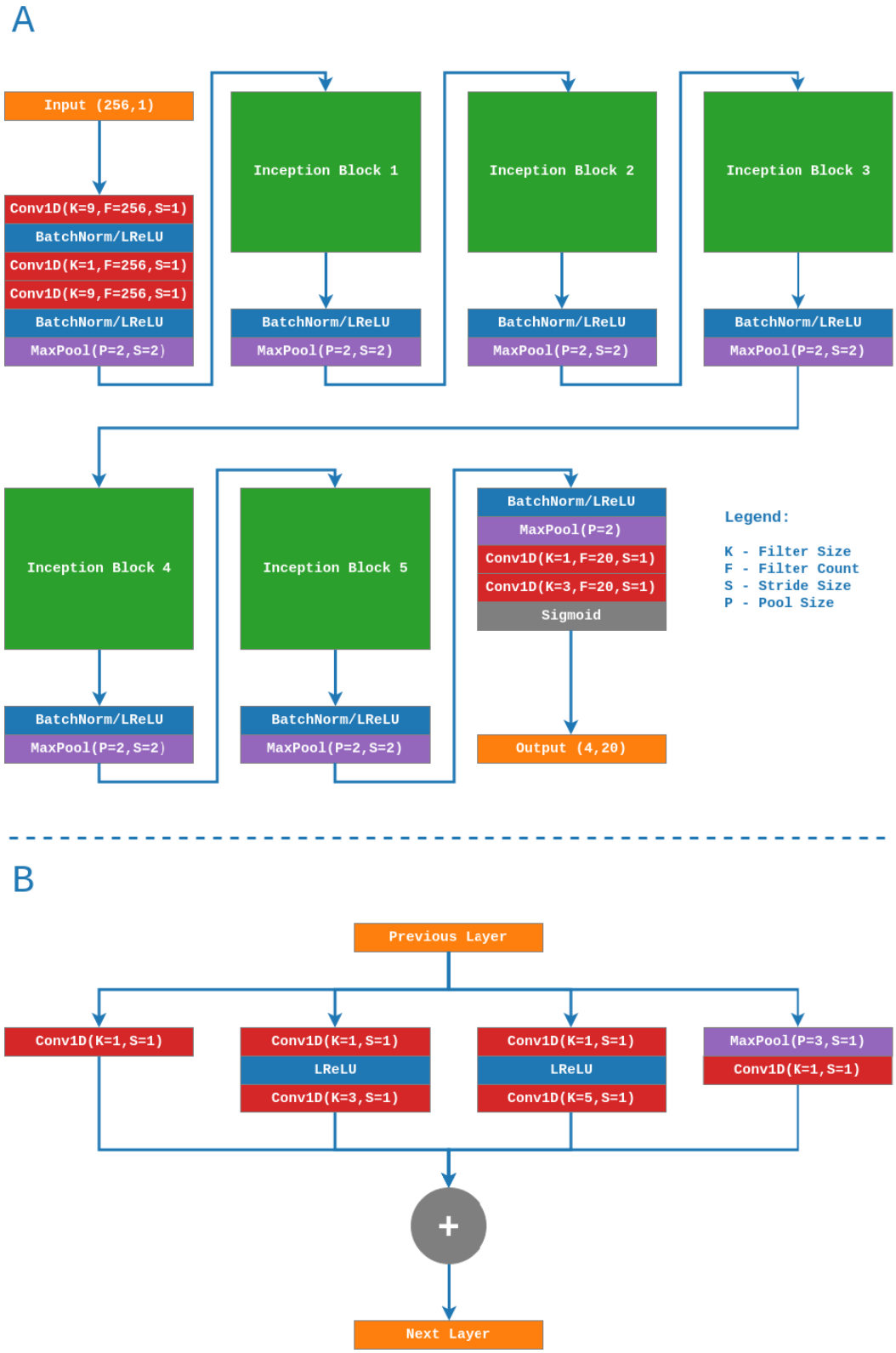

FIGURE 1. A) Encoder CNN architecture. CONV1D: One dimensional convolutional layer, BacthNorm/LRelu: Batch normalization layer and then Leaky ReLU layer applied in this order, LRelu: Leaky ReLU activation layer, MaxPool: Max pooling layer. B) Details of the inception block.

pooling layers are replaced with upsampling layers. Upsampling layers are implemented using bilinear interpolation to double the size of the input features and then convolution is applied using a transposed filter. The initial and final convolutional layers are slightly different. The parameters of the decoder network are provided in Table 1.

The input to the decoder network is a two-dimensional tensor of size 4×20, i.e. 4 samples for 20 feature vectors. This is the output tensor size of the encoder network. The input is processed by two cascaded convolutions with filter sizes 1 and 3, using 20 filters in each layer. After the second convolution, the resultant features are passed through leaky ReLU activation function. The features are then upsampled to their double sizes, i.e. each feature vector then contains 8 samples. Following upsampling, the resultant features are processed with a sequence of 5 inception blocks, the details of which are summarized in Table 1. The configuration of each inception block is kept similar to the corresponding block in the encoder network. For instance, InceptionBlock1 contains 4 parallel convolutional blocks; the first block has only one convolutional layer with 64 filters of size 1, the second block has a convolutional layer with 96 filters of size 1, followed by a leaky ReLU activation function, and another convolutional layer with 128 filters of size 3. In Table 1, the individual layers in each block are provided in different rows; the number of filters in parallel branches of the inception blocks are provided on the left half of the table as separate columns, and corresponding cells on the right half provides the filter sizes corresponding to the parallel layers on the left half.

At the end of the inception blocks, after the last upsampling, the number of features in each feature vectors becomes 256, i.e. the original sample window size of our speech signal. At this layer, two cascaded convolutions are applied to the features using 128 filters in each layer with filter sizes of 1 and 5, respectively. After batch normalization, the activations are passed through leaky ReLU function. The resultant feature vectors are then convolved with a filter of size 1 to generate a projection to a window of 256 samples. The resultant samples are passed through hyperbolic tangent function, i.e. tanh, to generate samples in $(-1, 1)$ range.

C. MODEL TRAINING

We trained the proposed encoder-decoder network model end-to-end from scratch using a publicly available, multilingual, multi-speaker database [21], in an unsupervised manner. As the objective of the CNN is to obtain an identity function, the input of the CNN encoder and the output of the CNN decoder are exactly the same for model training. We used 8 hours of speech data, in total, which are sampled at 8 kHz. The data is split into multiple non-overlapping 32 ms raw speech frames, i.e. a total of 900000 frames. The samples in the frames are first normalized into $(-1, +1)$, then quantized into logarithmic scale using 256 level μ-law companded [22] with 8 bits. The μ-law coded frames are then shuffled to generate random batches for training, the batch size is configured as 256. Before training, we set aside a random 15% of the training frames for model validations.

We use Keras framework [23] with Tensorflow backend for training the model and testing. The loss function for error backpropagation is computed using the mean square error (MSE) between the resultant frame samples and the input speech samples. We use Adam optimizer [24] with a learning rate 0.001, beta_1 0.9 and beta_2 0.999 with early stopping, i.e. the training is terminated when MSE stops improving for the validation frames for at least 5 epochs. We configured training for 256 epochs, yet, the model usually stops improving the MSE score, i.e. converges, between 180-220 epochs.

D. QUANTIZATION

The parameters obtained as the output of the encoder need to be quantized for telecommunication purposes, in order to be transmitted to the decoder input. In general, the output of the encoder has 20 filters and feature vector is either reduced to dimension of 4 (for 32 ms frames) or 2 (for 16 ms frames). Fig. 2 gives the probability density function of these coefficients. The final activation layer of the encoder was chosen to be sigmoid in order to get the source as narrow as possible, for an efficient quantization. Although these vectors (output coefficients of each filter in vector format of either 2 or 4) can be easily and effectively quantized with a matrix quantizer (for the filter bank of 20), we preferred an optimum vector quantizer of LBG implementation [25] for almost transparent quantization. In this approach, we did not exploit the possible intracorrelation in between the vectors of different filters. Our fundamental motivation is to keep the quantization distortion at the minimum, in order not to accumulate additional distortion on top of CNN modeling.

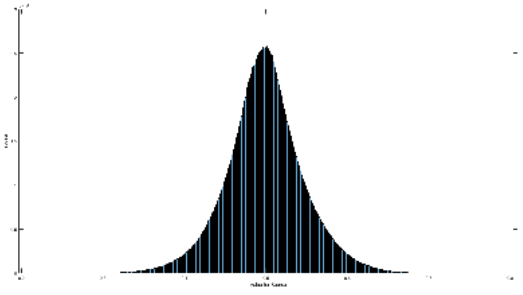

FIGURE 2. PDF of source, output of CNN encoder.

III. DISCUSSION

The parameters that are important for the design of DeepVoCoder can be described from both CNN topology and speech compression and coding perspective. The most important parameters for a voice coder are delay, complexity, memory requirement, bit rate and quality. The parameters of a CNN are total number of layers, number of convolutions in each layer and size of filters in each convolution. The other important parameter is of course the activation functions. We will consider each parameter from speech coder perspective and explore its effect on the CNN topology.

A. DELAY

Delay is extremely important for a speech coder as unacceptable delay durations cause echo which needs to be canceled [26]. Delay introduced by DeepVoCoder is merely the

frame size, which is actually the input dimension. Processing time required for encoding and decoding should also be added in order to calculate the overall latency. Therefore, although the input dimension should be as small as possible (for minimum delay) it should be considered as large as possible because of bit rate. If the frame size is too small, source needs to be updated more often, leading to an increase in bit rate. We decided on 32 ms, leading to 256 samples for narrowband (sampled at 8 kHz) speech. The decision comes from the fact that speech is assumed to be stationary in between 20-32 ms. This parameter not only defines delay and bit rate but also it defines better quality if analysis is done for longer frames and synthesis quality degrades for larger frames, for the same source. Although delay is not the only parameter during design of a DNN topology, our experiments showed that 32 ms is more favorable than 10, 16 and 20 ms. 16 ms is also important as it is equivalent to 128 samples, which is a power of 2 and therefore pooling can be done until 1 sample is reached at the end of the encoder, i.e. bottleneck.

B. COMPLEXITY

It is well known that CNN training takes significantly long time and requires computational resources. However, training complexity is not an issue as it is done offline only once. CNN encoder and CNN decoder complexity are however important as they determine the applicability of real-time implementation. DeepVoCoder also differs from conventional speech coders in this sense because in conventional speech coders, encoder complexity is always much higher than decoder complexity. In DeepVoCoder topology however, the complexity of the encoder and decoder are similar. The complexity is determined by i) total number of layers, each grouped into blocks and separated by pooling (for the encoder), or by up sampling (for the decoder), ii) number of convolutions per layer, iii) number of filters and filter sizes for each convolution. DeepVoCoder has around 1.2M learnt parameters in the encoder and 1.3M parameters in the decoder networks.

C. BIT RATE

Overall bit rate is determined by quantization of source only. In case where the final dimension of the bottleneck is 1, there is only a single parameter to be quantized by an optimum scalar quantizer. As a consequence of the activation function, source is always within a predetermined interval and easy to quantize. Moreover, we round the source to two-digit decimal points and observed that output speech quality is not significantly affected. Therefore, a uniform quantizer performs as good as an optimum scalar quantizer. Nevertheless, dimension of one for the source is too small even with many filters at the bottleneck and therefore we preferred dimensions of 2 or 4, where source is quantized with an optimum vector quantizer [25]. Overall bit rate in b/s is calculated by multiplication of ''total number of filters at the bottleneck'' times, ''dimension of source'' times, ''number of bits per sample for the source'' times ''update rate necessary per second''. In order to reduce the bit rate necessary to transmit speech over DeepVoCoder, one can reduce the number of filters at the bottleneck, increase the number of poolings in order to reduce the dimension of the source or spent less bits during quantization per sample for the source. The product is updated every 32 ms, which means it is multiplied by 31.25 (8000/256), in order to get the overall bit rate in bits/s. Reducing any of these parameters will also result with a reduced speech quality.

D. QUALITY

Quality is rather the result than a parameter. Input frame size, CNN encoder and decoder topology determines the quality of the synthesized speech signal. Major issue is at the bottleneck, i.e. the last convolutional layer of the encoder network. Total number of filters at the bottleneck and the number of layers in order to reach to the bottleneck are extremely important. It is also important to measure the quality in objective terms. Perception of voice by humans is still an ongoing discussion and active research field [27]. CNN decoder output, i.e. synthesized speech, is configured to be exactly the same with the CNN encoder input, i.e. original waveform. Hence, DeepVoCoder in its proposed form acts as a waveform coder and therefore PESQ [28] (perceptual evaluation of speech quality) to mimic MOS (mean opinion score) is an acceptable metric in order to present the quality.

IV. RESULTS OF THE EXPERIMENTS

In order to evaluate the performance of the proposed CNN encoder, we conducted some experiments mainly with two different settings. In all experiments we use the same set of test speech frames, which is selected randomly from Vox Forge dataset that are not included in the training process, to assess the quality of the encoding.

In our experiments, we fixed the inception model, i.e. the topology of the CNNs. Two different settings were determined by the input frame sizes of 256 and 128 samples, i.e. 32 and 16 ms, respectively. In the initial phase, parameters are not quantized in order to measure the figure of merit of the model itself. Since the topology and therefore the number of layers is exactly the same, the output of the CNN encoder are 4 and 2 samples for 32 and 16 ms, respectively. Of course, in the latter case they need to be updated twice more frequently, that is 62.5 times/s, instead of 31.25 times/s.

Table 2 provides some common parameters for the proposed DeepVocoder versus well knowns codecs for

TABLE 2. Codec parameters.

Codec/ Parameter	DeepVoCoder _16ms	DeepVoCoder _32ms	Speex	G.729
Frame(ms)	16	32	20	10
Delay(ms)	16	32	30	15
PESQ RawMOS	2.568	2.814	2.838	3.663
Bit_Rate (kb/s)	Model	Model	4.8	8.0

TABLE 3. Run time comparison.

	Execution time	
Codec/Parameter	DeepVoCoder_32ms	Speex
Single CPU (60 seconds)	56.817±0.106 s	161±0.784 ms
Single CPU (Single frame)	32±0.168 ms / 32 ms frame	< 1ms / 20 ms frame

performance comparison. It is possible to observe that DeepVocoder model, before quantization, has the potential to provide comparable speech quality with Speex at 4.8 kb/s and can run at comparable delays as low as G.729. Regarding the complexity, it is difficult to give execution times as DeepVocoder runs on GPU. Traditional speech coding algorithms however, run on CPU. Therefore, we limit Keras library to use single CPU for a fair comparison in a non-optimized manner. In addition, we include other possible setups in our results to provide a broader picture of the performance. Table 3 gives an overview of run time for best performing DeepVocoder_32 ms and freely available Speex vocoder. These values are obtained with a mean and standard deviation, after 1,000 runs, for full duplex operation, i.e. the execution times involve both encoding and decoding. It should be also noted that measurement overhead becomes unreliable for values below 1 ms. It is however, evident from Table 3 that the proposed DeepVocoder runs in real-time, even under constraints, if encoding and decoding are allowed on different CPUs (or GPU). Moreover, if DeepVocoder runs on 1,875 frames, that consists of a series of bulk 60 seconds of speech, slight advantages are observed in Table 3.

If the environment however is suitable, DeepVocoder can be massively parallelized, i.e tensor operations on convolutional neural networks can run much faster on GPU. This leads to vast gains in computing performance measured as 707±7.15 ms (more than 80 times faster than single CPU, 60 seconds). These values are obtained from the computer equipped with:

- CPU AMD Ryzen 7 2700X (8C/16T) @ 3.7 GHz,
- GPU nVidia Geforce RTX 2080 Ti @ 1.755 GHz.

Table 4 provides the bit rates of the model after quantization. Quantization is done with an optimum vector quantizer [25]. It is possible to observe that 32 ms frames provide better quality at the same bit rate due to two reasons. The first reason stems from the model that it is possible to remove redundancy more efficiently over longer frames. The second reason stems from quantization efficiency as larger dimensions of a vector with correlated values are easier to quantize. We have employed MIT's former Media Lab's Music, Mind and Machine group's SQAM (Sound Quality Assessment Material) database [29] as our test files. These files include native male and female speakers of English, French and German. There are four sets of files provided as supplementary material. The first set of files are original, clean speech for all samples of male and female speakers for three languages. SQAM database is stereo and CD quality. Therefore, all files

TABLE 4. Bit allocation table for the DeepVocoder parameters.

Codec	**DeepVoCoder_16ms**			**DeepVoCoder_32ms**	
Bottleneck dimension	2			4	
Total # of filters at the bottleneck	20			20	
# of bits/filter	5	6	7	9	10
Overall bit rate (kb/s)	6.25	7.50	8.75	5.625	6.25
PESQMOS	1.977	2.177	2.282	2.121	2.348

are first converted to mono and then converted to narrow band speech. For the second set of files, samples are modeled with DeepVocoder with 20 filters, where each filter has four dimensional coefficients for the bottleneck. The model output is then quantized at 6.25 kb/s (2.5 bits/coefficient) as shown in Table 4 for the third set of files. The final speech sample set is the output of the Speex vocoder operating at 4.8 kb/s for subjective comparison. Speex is a VBR (variable bit rate) codec where 4.8 kb/s stands for the average bit rate, whereas DeepVocoder is CBR (constant bit rate) and therefore bit rate is always exactly 200 bits/32 ms frame.

V. CONCLUSION & FUTURE WORK

The major contribution of this paper is the use of two convolutional neural networks (CNN) for compression and coding of voice for the first time in literature. We propose an end-to-end solution which means input speech is in time domain and represented as raw speech waveform. The encoding does not require any explicit extraction of parameters, i.e. acoustic features; the output is the reconstructed speech waveform. The process also does not require any conversions such as LSF. In last several decades, many improvements have been made on traditional speech codecs by many research groups. Over the course of time, they become so complex that optimization and fine tuning becomes increasingly more difficult. Bit allocation table given for the DeepVocoder in Table 4, however is extremely simple, straightforward and flexible. It is possible to change the number of filters at the bottleneck and/or assign a different number of bits during quantization in order to define the overall bit rate. Speech is synthesized by the CNN decoder. The decoded speech is always stable and quality increases in a graceful manner by increasing the frame size (delay), complexity (number of parameters for the CNN) and bit rate (number of samples used at the encoder bottleneck and/or number of bits used to quantize these samples) in an extremely well behaving and flexible manner. Although trained with limited number of speech data, without applying any manual preprocessing of the input samples or post processing of the output samples, the proposed networks encode raw speech samples in comparable performances with the existing, well known codecs. This experimental work depicts the potential of promising, flexible deep learning-based architectures in signal coding domain

with different input and/or DNN topologies to improve our novel proposal.

REFERENCES

[1] T. Berger, *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1971.
[2] B. S. Atal and M. R. Schroeder, "Adaptive predictive coding of speech signals," *Bell Syst. Tech. J.*, vol. 49, no. 8, pp. 1973–1986, Oct. 1970.
[3] W. B. Kleijn, F. S. C. Lim, A. Luebs, J. Skoglund, F. Stimberg, Q. Wang, and T. C. Walters, "Wavenet based low rate speech coding," Dec. 2017, *arXiv:1712.01120*. [Online]. Available: https://arxiv.org/abs/1712.01120
[4] F. Itakura, "Line spectrum representation of linear predictor coefficients of speech signals," *J. Acoust. Soc. Amer.*, vol. 57, p. 535, Apr. 1975.
[5] P. Kroon, E. Deprettere, and R. Sluyter, "Regular-pulse excitation—A novel approach to effective and efficient multipulse coding of speech," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-34, no. 5, pp. 1054–1063, Oct. 1986.
[6] M. Schroeder and B. Atal, "Code-excited linear prediction(CELP): High-quality speech at very low bit rates," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Tampa, FL, USA, Apr. 1985, pp. 937–940.
[7] R. McAulay and T. Quatieri, "Speech analysis/Synthesis based on a sinusoidal representation," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-34, no. 4, pp. 744–754, Aug. 1986.
[8] D. W. Griffin and J. S. Lim, "Multiband excitation vocoder," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-36, no. 8, pp. 1223–1235, Aug. 1988.
[9] A. McCree, K. Truong, E. B. George, T. P. Barnwell, and V. Viswanathan, "A 2.4 kbit/s MELP coder candidate for the new U.S. Federal Standard," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. Conf.*, Atlanta, GA, USA, vol. 1, 1996, pp. 200–203.
[10] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," Sep. 2016, *arXiv:1609.03499*. [Online]. Available: https://arxiv.org/abs/1609.03499
[11] A. Tamamori, T. Hayashi, K. Kobayashi, K. Takeda, and T. Toda, "Speaker-dependent WaveNet vocoder," in *Proc. Interspeech*, Aug. 2017, pp. 1118–1122.
[12] D. Rowe. (2011). *Codec 2- Open Source Speech Coding at 2400 Bits/S and Below*. [Online]. Available: http://www.tapr.org/pdf/DCC2011-Codec2-VK5DGR.pdf
[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[14] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3128–3137.
[15] L. A. Lim and H. Y. Keles, "Foreground segmentation using convolutional neural networks for multiscale feature encoding," *Pattern Recognit. Lett., v.*, vol. 112, pp. 256–262, Sep. 2018.
[16] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, 2015, pp. 234–241.
[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, Jun. 2014, pp. 580–587.
[18] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung, "Transferring rich feature hierarchies for robust visual tracking," 2015, *arXiv:1501.04587*. [Online]. Available: https://arxiv.org/abs/1501.04587
[19] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, 2013, pp. 1–6.
[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: https://arxiv.org/abs/1502.03167
[21] *Vox Forge Open Source Speech Corpus*. Accessed: May 10, 2019. [Online]. Available: http://www.voxforge.org
[22] *Pulse Code Modulation of Voice Frequencies*, document ITU-T G.711, ITU, 1988.
[23] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io
[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Jan. 2017, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[25] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COMM-28, no. 1, pp. 84–95, Jan. 1980.
[26] *Transmission Systems and Media, Digital Systems and Networks*, document ITU G 168, ITU, 2015.
[27] P. Pocta and J. Holub, "Predicting the quality of synthesized and natural speech impaired by packet loss and coding using PESQ and P.563 models," *Acta Acustica United Acustica*, vol. 97, no. 5, pp. 852–868, Sep./Oct. 2011.
[28] *Perceptual Evaluation of Speech Quality (PESQ): An Objective Method for End-to-End Speech Quality Assessment of Narrow-Band Telephone Networks and Speech Codecs*, document ITU-T 862, 2001.
[29] *MIT Former Media Lab, Music Mind and Machine Group, SQAM (Sound Quality Assessment Material) Database*. Accessed: May 10, 2019. [Online]. Available: https://sound.media.mit.edu/resources/mpeg4/audio/sqam/

HACER YALIM KELES was born in Ankara, Turkey, in 1979. She received the B.S., M.S., and Ph.D. degrees in computer engineering from Middle East Technical University, Turkey, in 2002, 2005, and 2010, respectively. Her Ph.D. Thesis received the Thesis of the Year Award from Prof. Dr. Mustafa Parlar Education and Research Foundation, Middle East Technical University, in 2010. From 2000 to 2007, she was a Researcher with The Scientific and Technological Research Council of Turkey (TUBITAK). During the years at TUBITAK, she primarily involved in different pattern recognition problems using multimedia data including audio and video. In 2010, she has received the Research and Development Grant from the Ministry of Industry and Trade of Turkey, and she has established a Research and Development Company. Her follow-up Project SOYA was supported by TUBITAK, in 2011, and later received as one of the ten best venture projects and it is selected to be sent to Silicon Valley for investment opportunities. She is the first woman who took this grant in Turkey. She has been an Assistant Professor with the Department of Computer Engineering, Ankara University, since 2013. Her research interests include predominantly in the areas of computer vision and machine learning, particularly in deep learning. She is also interested in the optimization of computational problems using GPUs.

JAN ROZHON was born in Ostrava, Czech Republic, in 1986. He received the B.S., M.S., and Ph.D. degrees in telecommunications from the VSB–Technical University of Ostrava, in 2008, 2010, and 2016, respectively, where he has been an Assistant and an Assistant Professor with the Department of Telecommunications, since 2014. He has authored or coauthored over 50 research papers indexed in Scopus database with all of them focused on the field of telecommunications. His research interests include the performance evaluation of telecommunication infrastructure, speech and video quality measurement and evaluation, artificial intelligence applications in telecommunications, speech coding, and software-defined networking.

H. GOKHAN ILK (M'00) was born in Ankara, Turkey, in 1971. He received the B.Sc. degree from Ankara University, Ankara, in 1993, the M.Sc. degree in instrument design and applications from the Institute of Science and Technology, University of Manchester, in 1994, and the Ph.D. degree from the University of Manchester, Manchester, U.K, in 1997. He is currently a Professor with the Electrical and Electronics Engineering Department, Ankara University, where he is currently on sabbatical leave with IT4I VSB, Technical University of Ostrava (TUO). His publications are in optical communications, digital speech processing and coding, and hyperspectral image signal processing. His current research interests include digital signal processing in networking, speech, image, and video processing. He has a book in Turkish on Applied Signal Processing. He is the Founder of the Ankara University Speech Processing Group and shared Turkcell's (biggest GSM operator in Turkey) Best Academic Study Award, in 2007.

MIROSLAV VOZNAK (M'10–SM'16) received the Ph.D. degree in telecommunications from the Faculty of Electrical Engineering and Computer Science, VSB–Technical University of Ostrava, in 2002, and the Habilitation degree from VSB–Technical University of Ostrava, in 2009. He was an appointed Full Professor in electronics and communication technologies, in 2017. He is a member of numerous IEEE conference committees, and he has served as a Member of the Editorial Board for several journals such as the *Journal of Communications* or a Guest Editor of *Wireless Personal Communications*. His research interests generally include on information and communications technology, particularly on the quality of service and experience, network security, wireless networks, and in the last couple years big data analytics in mobile cellular networks.