Spectogram Prediction with Neural Networks

Mario Alejandro García ✉ and Eduardo Atilio Destéfanis

Universidad Tecnológica Nacional Facultad Regional Córdoba, Argentina
mgarcia@frc.utn.edu.ar

Abstract. A neural network model for spectrogram magnitude prediction is presented. It has one convolutional layer that computes the short-time Fourier transform. By choosing the magnitude of the spectrum as output and discarding the phase, it is possible to avoid complex number operations. The structure of the network and coefficients computation for this alternative are presented in detail. The model coefficients can be directly computed or trained with the gradient descent algorithm. In both cases, the results are satisfactory, but the obtained weights are different. An analysis of the differences is made. The main contribution of this article is to show that the proposed model is trainable. Consequently, the coefficients can be adapted to particular problems.

Keywords: discrete fourier transform, spectrogram, deep learning, convolutional neural network

1 Introduction

The spectrogram is a representation of the variation of the frequency spectrum of a signal. This variation can occur in time (audio, earthquake waves, etc.), space (images) and other domains.

In machine learning, it is common to use the spectral information data to find non-obvious features in the source domain. The frequency spectrum of a signal is obtained through the Fourier Transform (FT). For discrete data, the computational basis of spectral analysis is the Discrete Fourier Transform (DFT).

In the following paragraphs of this introduction, some key concepts will be stated.

Discrete Fourier Transform. The DFT converts a finite sequence of N complex numbers (samples) $\{x_n\} := x_0, x_1, ..., x_{N-1}$ into a sequence of $K = N$ complex numbers $\{X_k\} := X_0, X_1, ..., X_{N-1}$.

$$X_k = \sum_{n=0}^{N-1} x_n \; e^{-i2\pi kn/N} \tag{1}$$

According to the Euler's formula,

$$X_k = \sum_{n=0}^{N-1} x_n \left[\cos(-2\pi kn/N) + i\ \sin(-2\pi kn/N)\right] \tag{2}$$

It is important to note that the DFT is a linear operator [1]. The DFT can then be defined as the linear map $\mathcal{F} : \mathbb{C}^N \to \mathbb{C}^N$ such that $X = \mathcal{F}(x)$ with the following matrix representation.

$$\boldsymbol{X} = \boldsymbol{F}\boldsymbol{x}$$

where

$$\boldsymbol{x} = \begin{bmatrix} x_0 & x_1 & x_2 & \dots & x_{N-1} \end{bmatrix}^\intercal,$$
$$\boldsymbol{X} = \begin{bmatrix} X_0 & X_1 & X_2 & \dots & X_{N-1} \end{bmatrix}^\intercal$$

and according to equation 1,

$$\boldsymbol{F} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & e^{-\frac{i2\pi}{N}} & e^{-2\frac{i2\pi}{N}} & \dots & e^{-(N-1)\frac{i2\pi}{N}} \\ 1 & e^{-2\frac{i2\pi}{N}} & e^{-4\frac{i2\pi}{N}} & \dots & e^{-2(N-1)\frac{i2\pi}{N}} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-(N-1)\frac{i2\pi}{N}} & e^{-2(N-1)\frac{i2\pi}{N}} & \dots & e^{-(N-1)^2\frac{i2\pi}{N}} \end{bmatrix}$$

For the case of equation 2,

$$\boldsymbol{F} = \boldsymbol{F}_C + i\boldsymbol{F}_S$$

where

$$\boldsymbol{F}_C = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & cos(-1\frac{i2\pi}{N}) & cos(-2\frac{i2\pi}{N}) & \dots & cos(-(N-1)\frac{i2\pi}{N}) \\ 1 & cos(-2\frac{i2\pi}{N}) & cos(-4\frac{i2\pi}{N}) & \dots & cos(-2(N-1)\frac{i2\pi}{N)} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & cos(-(N-1)\frac{i2\pi}{N}) & cos(-2(N-1)\frac{i2\pi}{N}) & \dots & cos(-(N-1)^2\frac{i2\pi}{N}) \end{bmatrix}$$

$$\boldsymbol{F}_S = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & sin(-1\frac{i2\pi}{N}) & sin(-2\frac{i2\pi}{N}) & \dots & sin(-(N-1)\frac{i2\pi}{N}) \\ 0 & sin(-2\frac{i2\pi}{N}) & sin(-4\frac{i2\pi}{N}) & \dots & sin(-2(N-1)\frac{i2\pi}{N)} \\ 0 & \vdots & \vdots & \ddots & \vdots \\ 0 & sin(-(N-1)\frac{i2\pi}{N}) & sin(-2(N-1)\frac{i2\pi}{N}) & \dots & sin(-(N-1)^2\frac{i2\pi}{N}) \end{bmatrix}$$

Spectrogram. The magnitude of the Short-Time Fourier Transform (STFT), yields the spectrogram. In the case of a discrete signal of length L, the STFT is simply the DFT of length N segments, where $N < L$, of the signal. The result is a complex matrix $\boldsymbol{S}$ with the magnitude and phase of the signal for each frequency in each segment (time). Generally, the columns in the matrix represent the time dimension and the rows represent the different frequencies. The choice of the N value depends on the goal of the spectral representation. For small N values, high definition is obtained in the time dimension and low definition is obtained in the frequency dimension, while for high N values, the effect is reversed. The segments can be overlapped on m samples, for m between 0 and $N - 1$.

Convolutional Neural Networks. Each neuron of a convolutional neural network (CNN) performs a linear transformation of the input vector before an activation function. The output of the neuron j is defined as:

$$y_j = g(\sum_{i=0}^{N} w_{ij}\ x_i)$$

where
$g()$ is the activation function
x_i is the input i
w_{ij} is the synaptic weight corresponding to the input i of the neuron j
w_{0j} is the bias. ($x_0 = 1$).

Each layer of a CNN is composed of kernels. All kernels have the same number of neurons. Within each kernel, the neurons share the synaptic weights, but each one has its own receptive field (neurons connect to a limited set of inputs). Neurons of the same location in each kernel share the receptive field, that is, they connect to the same inputs. In this way, the output of a CNN layer is the discrete convolution of the inputs and the weights of each kernel. Once the network is trained, each kernel specializes in recognizing or transforming a certain pattern of the input.

1.1 Objectives

The objective of this work is to design an artificial neural network that predicts the spectrogram and to train the network with audio data. For this, the network must calculate the DFT.

The weights of the network can be set with the elements of the matrix $\boldsymbol{F}$ or be trained.

The main contribution of this work is the calculation of the synaptic weights by training. Demonstrating that the network can learn the coefficients of the DFT, suggests that other representations of the signal, better suited to the classification needs, could also be learned.

2 Motivation and related work

There are many deep learning works that use the magnitude of the frequency spectrum as input [2–4]. This article shows that it is possible to compute the same information by adding layers at the beginning of the deep network. The advantage is that the calculation of the spectrum can be adapted to a particular case.

Moreira et al. proposed the calculation of the DFT with cellular neural networks in [5]. They divide the weights into two groups that represent the real and imaginary parts. The training is not carried out, the weights are directly assigned.

Velik, in [6], predicts the DFT with weights calculated from complex exponential functions, also directly assigned, and reports that the neural network cannot be trained.

On the other hand, Anderson and Mallat [7] propose the replacement of the DFT by the Deep Scattering Spectrum (DSS) technique, based on wavelet transforms, because DSS is able to represent invariant characteristics over time (or space in the case of the images). Sometimes these deviations in time are significant for recognition. Our work is carried out in the field of vocal quality classification, where a deviation of the vibration frequency of the vocal cords is important [8].

3 Methods and Materials

3.1 Data

The neural network was designed to predict the spectrogram magnitudes of two seconds audio signals.

The audios are part of the Voice Disorders Database (VDD) [9], recorded by the *Universidad Politécnica de Madrid* in collaboration with the *Hospital Universitario Príncipe de Asturias*. The audio files contain approximately 2 seconds of a sustained vowel /a/. These were recorded from people with vocal pathologies and healthy people.

Inputs. The audios are in WAV format with a sample rate of 25000 samples per second. All files where duration ≥ 2 seconds were selected. Then the 50000 central samples were taken.

The input data is defined as 430 vectors of length $L = 50000$. 300 vectors were randomly chosen for training and 130 for validation.

Outputs. The outputs are calculated as the absolute value of the STFT of the inputs, with segments of size $N = 1760$ and overlap $m = 1540$ elements. It implies a 220 elements displacement by transformation. The module is calculated in order to obtain the spectrogram magnitude. The spectrogram calculated in this way has 220 columns (time) and 881 rows (frequency). Due to the nature

of the audios (voices in the usual pitch), the energy is concentrated almost completely in the first 200 rows (0-2826 Hz). For this reason, it was decided to train the network to predict only the magnitude of rows 1 to 200 of the spectrogram. For this case $K = 200$.

Then, the output is a vector of size 200×220. Note that the output values are in $\mathbb{R}$ because the module of the spectrum is taken.

3.2 Neural Network

To obtain the output defined in the previous section, the neural network must consist of two parts, one that calculates the STFT and another that obtains the absolute value. If the operation of calculating the absolute value is not included in the network, it is not possible to perform the training with the data as defined.

The STFT computation is done with a convolution layer, where the synaptic weights are the elements of the matrix $\boldsymbol{F}$ and the activation function is linear. This is possible because both, the DFT and the operation performed by each neuron, are linear transformations. It is important to note that the values of the matrix $\boldsymbol{F}$ are constant.

There are two ways to implement the calculation, depending on whether the DFT equation 1 or 2 is chosen. If the matrix $\boldsymbol{F}$, corresponding to the equation 1 is used, the weight matrix $\boldsymbol{W}$ consisting in the complex coefficients w_{ij}, will have size $N \times K$. In the case of the equation 2, $\boldsymbol{W}$, of size $N \times 2K$ will consist in the (real) values of the matrices $\boldsymbol{F}_C$ and $\boldsymbol{F}_S$.

In terms of efficiency, there is no difference between the two alternatives. For this work the second one is chosen because, since the output only conserves information of the spectrum magnitude, it is possible to avoid complex numbers operations. This could be a practical advantage because, among the neural networks software libraries, there is still no (in general) support for the complex numbers [10]. For the rest, the two approaches are equivalent.

Convolution layer. In order to calculate the STFT with a convolution layer, it is convenient to write the equation 2 in the following way.

$$X_k = \sum_{n=0}^{N-1} x_n \ \cos(-2\pi kn/N) + \sum_{n=0}^{N-1} i \ x_n \ \sin(-2\pi kn/N) \tag{3}$$

In matrix form,

$$\boldsymbol{X} = \boldsymbol{F}_C\boldsymbol{x} + i\boldsymbol{F}_S\boldsymbol{x}$$

Fig. 1 shows the proposed neuronal model. It can be seen that the output of the convolutional layer contains $\boldsymbol{F}_C\boldsymbol{x}$ and $\boldsymbol{F}_S\boldsymbol{x}$ values. These are the convolution of the input and each one of the $2K(400)$ kernels.

In the case of the direct assignment of the synaptic weights (without training), the first K kernels take the values of the K columns of matrix $\boldsymbol{F}_C$, and

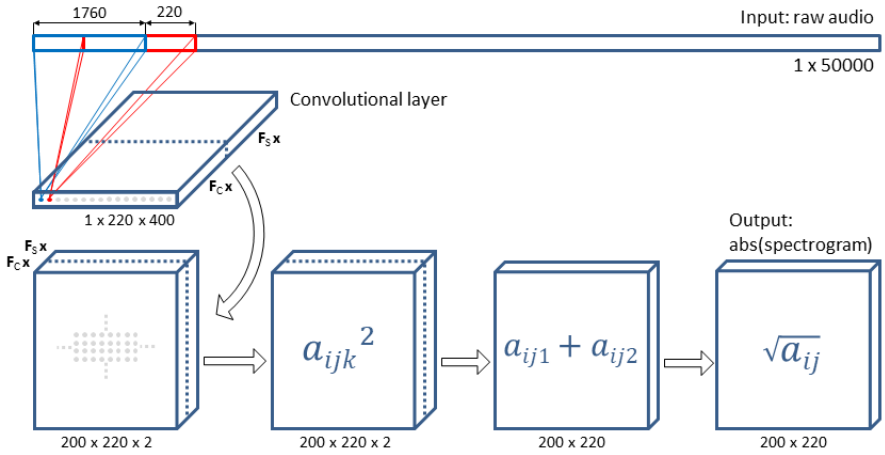

Fig. 1. Artificial neural network model that receives audio as input and predicts the magnitude of the spectrogram.

remaining kernels take the values of $\boldsymbol{F}_S$ columns. In this model, the neurons do not have the w_{0j} weight because the DFT has no term independent. Formally, the assignment of the weights is done as follows:

$$w_{ijk} = f_{Cik} \tag{4}$$

$$w_{ij(k+K)} = f_{Sik} \tag{5}$$

where

w_{ijk} is the synaptic weight of the input i of the neuron j of the kernel k.

f_{Cik} is the element in the row i and column k of $\boldsymbol{F}_C$.

f_{Sik} is the element in the row i and column k of $\boldsymbol{F}_S$.

Since the weights of neurons at the same kernel are shared, only one set of weights per kernel is stored. Note that there is not subindex j in the second terms of the equations 4 and 5. Then, the weights matrix W has size $N \times 2K$ (1760×400).

Magnitude of the spectrum. From equation 3, the magnitude of the frequency spectrum is as follows.

$$|X_k| = \left| \sum_{n=0}^{N-1} x_n \ \cos(-2\pi kn/N) + \sum_{n=0}^{N-1} i \ x_n \ \sin(-2\pi kn/N) \right|$$
$$= \sqrt{\left(\sum_{n=0}^{N-1} x_n \ \cos(-2\pi kn/N) \right)^2 + \left(\sum_{n=0}^{N-1} x_n \ \sin(-2\pi kn/N) \right)^2} \quad (6)$$

The results of the summations in the equation 6 are the scalars in the location k of the $\boldsymbol{F}_C\boldsymbol{x}$ and $\boldsymbol{F}_S\boldsymbol{x}$ vectors. Thus,

$$|X_k| = \sqrt{(\boldsymbol{F}_C\boldsymbol{x})_k^2 + (\boldsymbol{F}_S\boldsymbol{x})_k^2}$$

where $(\boldsymbol{F}_C\boldsymbol{x})_k$ and $(\boldsymbol{F}_S\boldsymbol{x})_k$ are the kth elements in $\boldsymbol{F}_C\boldsymbol{x}$ y $\boldsymbol{F}_S\boldsymbol{x}$ vectors respectively.

The output of the convolution layer of the model in Fig. 1 is an array containing the vectors $\boldsymbol{F}_C\boldsymbol{x}$ and $\boldsymbol{F}_S\boldsymbol{x}$ corresponding to all segments in the input signal. After the convolution, four operations arranged in layers are carried out. The first operation is a change in the form of the array in order to simplify the third operation, the second calculates the square of each element, the third adds pairs of values corresponding to the same frequency (k) and time segment, and finally, the calculation of the square root of each element is performed. The result is a matrix of size 200×220 with the magnitudes of the spectrum elements.

Training. Instead of assigning the weights directly, these can be trained through the gradient descent method on the mean squared error (MSE) function. The calculation of the gradient includes the derivatives of the three last layers (operations) of the model. In the next section, the results of this process are presented and compared with those obtained through direct assignment.

4 Results

Below, the result of 30,000 training cycles of the proposed model is shown. The weights were initialized with random values between -10^6 and 10^6 uniformly distributed. The optimization method Adam [11] was used with the parameters provided by the authors. The weights were updated in batches of size 300 (the whole of training data). The calculations were performed on an NVIDIA Titan Xp GPU donated through NVIDIA's GPU Grant Program.

The MSE reached on validation data was 1.41×10^{-6} (9.79×10^{-6}% of the mean expected output), while for the same model with directly assigned weights an MSE $< 10^{-9}$ was achieved. Fig. 2 shows the expected output for one vector in validation dataset and the output obtained by the network after the training. At first glance there are no differences.

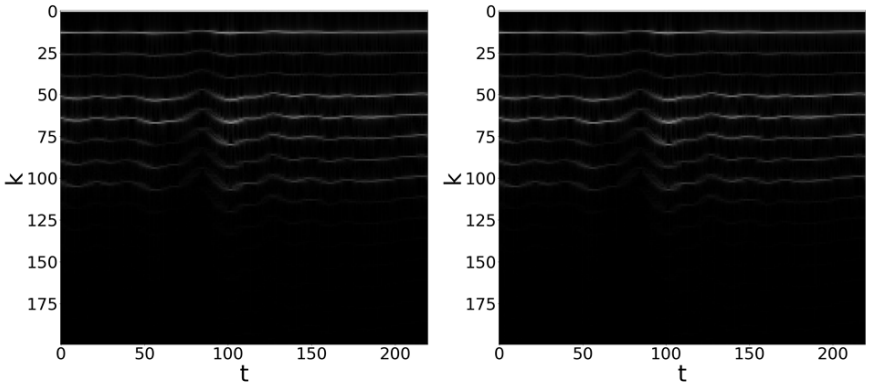

Fig. 2. Output (spectrogram) of the Neural network. Expected output (left) and output obtained with trained weights (right). At first glance there are no differences.

The result obtained by training is widely satisfactory. A comparison between the theoretical weights, calculated from the equations of the DFT, and the trained ones is presented below.

In Fig. 3 It can be seen the values of $\boldsymbol{F}_C$ and $\boldsymbol{F}_S$ for weights assigned directly (theoretical) and for trained weights. In all cases it is clearly observed that the upper rows of the transposed matrices represent the low frequencies and lower rows represent high frequencies.

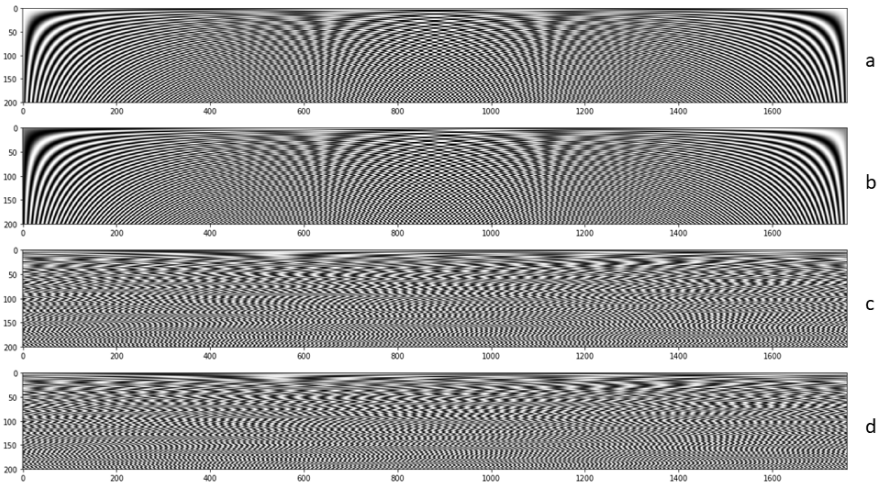

Fig. 3. Transposed matrices of coefficients. $\boldsymbol{F}_C$ with theoretical weights (a), $\boldsymbol{F}_S$ with theoretical weights (b), $\boldsymbol{F}_C$ with trained weights (c) and $\boldsymbol{F}_S$ with trained weights (d).

It is also evident that the two groups, theoretical and trained weights, present different image patterns. The trained weights have a "messy" appearance. The

origin of this phenomenon is that TDF performs a decomposition of the input into a weighted sum of the sinusoidal signals found in the matrices $\boldsymbol{F}_C$ and $\boldsymbol{F}_S$. The training method, for each value of k, finds a pair $\boldsymbol{F}_C$ and $\boldsymbol{F}_S$ formed by sine waves with a 90° phase offset allowing the desired decomposition, but this solution is not necessarily the same as the equation 3.

Fig. 4 shows three examples of trained vs. theoretical weights for particular values of k. Note that, for both theoretical and trained weights there is always a 90° phase shift between $\boldsymbol{F}_C$ and $\boldsymbol{F}_S$. This can be checked by calculating $mod = \sqrt{\boldsymbol{F}_C^2 + \boldsymbol{F}_S^2}$ for any value of k. For theoretical weights, obviously $mod = 1$, while for trained weights a value very close to 1 is always obtained. In this way, an orthogonal base is found to perform the decomposition.

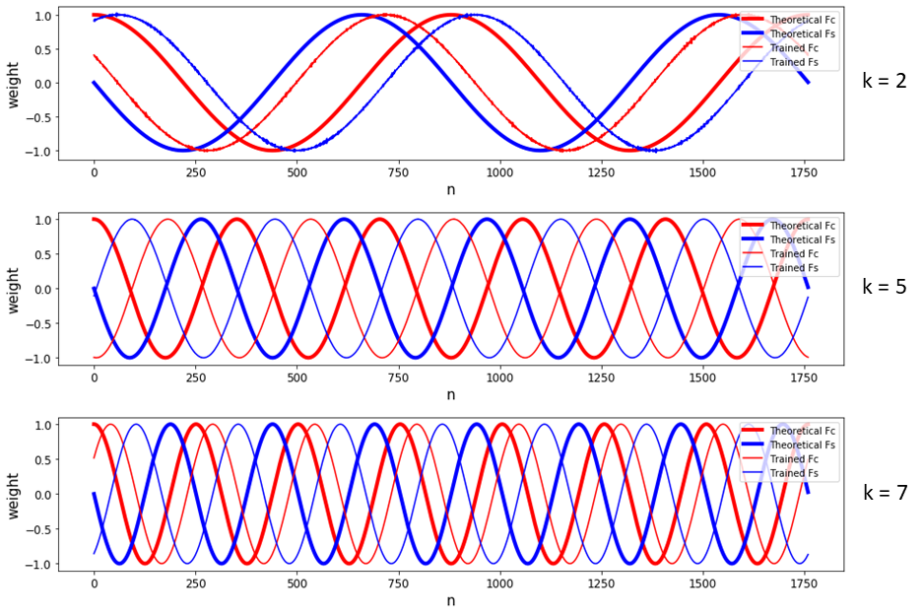

Fig. 4. Trained synaptic weights (thin lines) and theoretical weights (thick lines) for $k = 2$, 5 and 7. The values of $\boldsymbol{F}_C$ are shown in red and the values of $\boldsymbol{F}_S$ in blue.

5 Conclusion

It is concluded that a neuronal model is able to calculate the DFT, both for theoretical and trained weights, and that the trained weights do not necessarily tend towards the theoretical ones, although they share frequency and orthogonality condition. In addition, a convolutional network with the presented characteristics can be trained to calculate the spectrogram of the input signal. If the expected output is the magnitude of the spectrum, it is possible to avoid the operations

of complex numbers by adding operations in layers that calculate the Euclidean distance between the components of the same frequency.

The advantage of training the model is that it can be adapted to particular problems. For example, in the STFT computation it is common to use a window that softens the signal ends. These functions can be achieved with the proposed network by attenuating the ends of the weights. There are several window functions, but for a particular problem a different one may be better.

Then, to solve a particular problem, the weights can be initialized with random values, theoretical weights of the DFT or theoretical weights modified by some function (window function for example) and then, trained to find the optimal combination for the problem.

References

1. McClellan, J., Parks, T.: Eigenvalue and eigenvector decomposition of the discrete fourier transform. IEEE Transactions on Audio and Electroacoustics **20**(1) (1972) 66–74
2. Collobert, R., Puhrsch, C., Synnaeve, G.: Wav2letter: an end-to-end convnet-based speech recognition system. arXiv preprint arXiv:1609.03193 (2016)
3. Xu, Y., Du, J., Dai, L.R., Lee, C.H.: An experimental study on speech enhancement based on deep neural networks. IEEE Signal processing letters **21**(1) (2014) 65–68
4. Putten, M.J., Olbrich, S., Arns, M.: Predicting sex from brain rhythms with deep learning. Scientific reports **8**(1) (2018) 3069
5. Moreira-Tamayo, O., De Gyvez, J.P.: Filtering and spectral processing of 1-d signals using cellular neural networks. In: Circuits and Systems, 1996. ISCAS'96., Connecting the World., 1996 IEEE International Symposium on. Volume 3., IEEE (1996) 76–79
6. Velik, R.: Discrete fourier transform computation using neural networks. In: 2008 International Conference on Computational Intelligence and Security, IEEE (2008) 120–123
7. Andén, J., Mallat, S.: Deep scattering spectrum. IEEE Transactions on Signal Processing **62**(16) (2014) 4114–4128
8. García, M.A., Destéfanis, E.A.: Deep neural networks for shimmer approximation in synthesized audio signal. In: Argentine Congress of Computer Science, Springer (2017) 3–12
9. Arias-Londoño, J.D., Godino-Llorente, J.I., Markaki, M., Stylianou, Y.: On combining information from modulation spectra and mel-frequency cepstral coefficients for automatic detection of pathological voices. Logopedics Phoniatrics Vocology **36**(2) (2011) 60–69
10. Trabelsi, C., Bilaniuk, O., Zhang, Y., Serdyuk, D., Subramanian, S., Santos, J.F., Mehri, S., Rostamzadeh, N., Bengio, Y., Pal, C.J.: Deep complex networks. arXiv preprint arXiv:1705.09792 (2017)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)