- ORIGINAL ARTICLE -

Power Cepstrum Calculation with Convolutional Neural Networks

Cálculo del Power Cepstrum con Redes Neuronales Convolucionales

Mario Alejandro García[1] and Eduardo Atilio Destéfanis[1]

[1] *Universidad Tecnológica Nacional Facultad Regional Córdoba, Argentina*
mgarcia@frc.utn.edu.ar

Abstract

A model of neural network with convolutional layers that calculates the power cepstrum of the input signal is proposed. To achieve it, the network calculates the discrete-time short-term Fourier transform internally, obtaining the spectrogram of the signal as an intermediate step. Although the proposed neural networks weights can be calculated in a direct way, it is necessary to determine if they can be obtained through training with the gradient descent method. In order to analyse the training behaviour, tests are made on the proposed model, as well as on two variants (power spectrum and autocovariance). Results show that the calculation model of power cepstrum cannot be trained, but the analysed variants in fact can.

Keywords: Cepstrum, Discrete Fourier transform, Spectrogram, Deep learning, Convolutional neural network

Resumen

Se propone un modelo de red neuronal con capas de convolución que calcula el power cepstrum de una señal de audio. Para logarlo, la red calcula internamente la Transformada Discreta de Fourier de Tiempo Reducido, obteniendo el espectrograma de la señal como paso intermedio. Si bien los pesos de la red neuronal propuesta se pueden calcular de forma directa, uno de los objetivos de este trabajo es determinar si esta puede ser entrenada con el método del gradiente descendiente. Para analizar el comportamiento del entrenamiento se realizan pruebas sobre el modelo propuesto y también sobre dos variantes (power spectrum y autocovarianza). Los resultados indican que el modelo de cálculo del power cepstrum no se puede entrenar, pero las variantes analizadas sí.

Palabras claves: Cepstrum, Transformada discreta de Fourier, Espectrograma, Aprendizaje profundo, Red neuronal convolucional

1 Introduction

In machine learning, it is common to use the spectral information of data in order to find non-evident features in the origin domain. The frequency spectrum of a signal is obtained through the Fourier transform (FT). For discrete data, as this work, the discrete Fourier transform (DFT) is used. The spectrogram is a representation of the spectrum variation. This variation can take place in time (for example, audio), in space (images) and in other domains.

Considering the frequency spectrum as if it was a signal, its spectral information can be analysed in order to find non-evident features in the frequency domain. Consequently, a new level of spectral analysis is created, the *cepstral analysis*, whose representation, again in the time domain, is called *cepstrum*.

During the pattern recognition process, the spectrum, the spectrogram and the cepstrum are usually calculated in the feature extraction stage. These features will be later used in a classifier.

On the other hand, in deep learning models, the first layers of the neural network are the ones in charge of extracting features. It is supposed that it is in this integration of both stages in one same model where the advantage of this method lies, since it is possible to find both the optimal parameters of feature extraction and classification together [1].

The architecture of a neural network that calculates the spectrum (power spectrum), the spectrogram and the cepstrum (power cepstrum) is defined in this work. Then, training tests with audio data are done, and the adaptation capability of the proposed model is analysed with the aim of establishing if it can be used as layers of feature extraction in a bigger deep learning model.

As the DFT coefficients are known, the optimal weights of the network can be calculated in a direct way ("theoretical weights" hereafter). However, it is important to know the training capacity of the network. The ability to adapt to particular needs of classification depends on its capability.

In section 2, some key concepts for the rest of the article are enunciated. In section 3 related works are presented. In section 4 a neural network in order to calculate the power cepstrum is proposed. In section 5 details of the experiments made are provided. Finally, results and conclusions are shown in sections 6 and 7 respectively.

2 Background

2.1 Discrete Fourier Transform

The DFT converts a finite sequence of N complex numbers (samples) $\{x_n\} := x_0, x_1, ..., x_{N-1}$ in another sequence of $K = N$ complex numbers $\{X_k\} := X_0, X_1, ..., X_{N-1}$, where x_n is in the time domain and X_k is in the frequency domain.

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

According to Euler's formula, $e^{i\xi} = \cos\xi + i\sin\xi$. Then, equation 1 can be written as:

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

It is important to highlight that the DFT is a linear operator [2]. Then, the DFT can be defined as the linear map $\mathscr{F} : \mathbb{C}^N \rightarrow \mathbb{C}^N$ such that $X = \mathscr{F}(x)$ with the matrix representation below.

$$X = Fx$$

where

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

and from equation 1,

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

For the case of equation 2,

$$F = F_C + iF_S$$

where

$$F_C = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \theta_1 & \theta_2 & \dots & \theta_{N-1} \\ 1 & \theta_2 & \theta_4 & \dots & \theta_{2(N-1)} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & \theta_{N-1} & \theta_{2(N-1)} & \dots & \theta_{(N-1)^2} \end{bmatrix}$$

$$F_S = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & \rho_1 & \rho_2 & \dots & \rho_{N-1} \\ 0 & \rho_2 & \rho_4 & \dots & \rho_{2(N-1)} \\ 0 & \vdots & \vdots & \ddots & \vdots \\ 0 & \rho_{N-1} & \rho_{2(N-1)} & \dots & \rho_{(N-1)^2} \end{bmatrix}$$

for

$$\theta_j = cos(-j\tfrac{i2\pi}{N})$$

$$\rho_j = sin(-j\tfrac{i2\pi}{N})$$

2.2 Spectrogram

The spectrogram is the outcome of the application of the short-term Fourier transform (STFT). For the case of a discrete signal of length L, the STFT is simply the DFT of segments of length N, for $N < L$, of the signal. The result is a matrix S of complex values with the signal magnitude and phase for each frequency in each segment (time). Generally, the time dimension is represented in the matrix columns and the frequency, in the matrix rows. The choice of the value of N depends on the objective of the spectral representation of the data. For small values of N , high definition in the time dimension and low definition in the frequency dimension are obtained; whereas for high values of N, the effect is reversed. The segments can be overlapped in a number of samples m between 0 and $N-1$. Frequently, instead of using the X_k elements of the spectrum, the spectrogram is made with the spectrum magnitude, $|X_k|$, or with the square of the magnitude (power spectrum), $|X_k|^2$.

2.3 Cepstrum

The power cepstrum, or just cepstrum, is the power spectrum of the logarithm of the power spectrum of the signal [3]. It is mainly used in signal analysis to find the "frequency" of the occurrence of harmonics in the spectrum. The independent variable of the cepstrum represents a new frequency in the frequency domain, which results in a variable in temporal domain. In order to avoid confusions, but emphasising on the connection with familiar concepts, Bogert *et al.* called this variable *quefrency* (frequency changing the order of the first syllables), using the same criterion that was chosen for the name cepstrum, spectrum with the first letters reversed [4].

The cepstrum was defined in [5] as

$$C = \left|\mathscr{F}^{-1}\left\{\log\left(|\mathscr{F}\{f(x)\}|^2\right)\right\}\right|^2 \tag{3}$$

Note that, as $log\left(|\mathscr{F}\{f(x)\}|^2\right)$ is an even function of the frequency, the sine (imaginary part) of $\mathscr{F}^{-1}$ is voided and, therefore, the cepstrum is equal to the square of the cosine transform of $log\left(|\mathscr{F}\{f(x)\}|^2\right)$

multiplied by $1/N$. For the same reason, it is also frequent to calculate the cepstrum as

$$C = \frac{1}{N}\left|\mathscr{F}\left\{\log\left(|\mathscr{F}\{f(x)\}|^2\right)\right\}\right|^2$$

Factor $1/N$ comes from the inverse DFT (IDFT). For the most common applications of the cepstrum, which imply the detection of its highest peak, or the calculation of the relationship between the energy of the peak and the total energy, this factor can be eliminated.

Although the power cepstrum is calculated in this article, other versions of cepstrum there exist, such as real cepstrum, complex cepstrum, phase cepstrum and autocovariance, The basis of the calculation of abovementioned versions is the same; therefore, they could also be calculated with neural networks. The calculation expression of autocovariance is the same as that of the power cepstrum, but without the logarithm [3].

The use of the cepstrum is common in the automatic analysis of voice. The location of its highest peak (in a certain range) is used to detect the fundamental frequency of voice, in other words, the frequency at which vocal cords vibrate [6]. The ratio between the peak magnitude and the rest of the cepstrum is used to determine certain features of vocal quality. This ratio also allows the classification of voices according to their level of noise, breathiness and nasality [7, 8]. On the other hand, the mel frequency cepstral coefficients (MFCC) are very useful for speech recognition [1].

2.4 Convolutional Neural Networks

Every neuron in a convolutional network (CNN) makes a linear transformation of its inputs before applying the activation function. The output of 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 input i, w_{ij} is the synaptic weight corresponding to input i of neuron j and w_{0j} is the bias ($x_0 = 1$).

Every layer of a CNN is made up of kernels. Each kernel has the same amount of neurons. Inside each kernel, the neurons share the synaptic weights, but each neuron has its own receptive field (they are not connected to all the inputs but to a subset of them). The neurons in the same position as every kernel share the receptive field, i.e., they are connected to the same inputs. Therefore, the output of a CNN layer is the discrete convolution between the inputs and the weights of each of the kernels. Once the network is trained, each kernel specialises in recognising or transforming a certain pattern of the input data.

3 Related Work

In the year 2018, this team presented a neural model able to calculate the spectrogram of an audio signal [9]. In this current work, a change is made in such model, since the power spectrum instead of the spectrum magnitude is calculated, and it is extended to calculate the cepstrum.

There are many works on deep learning that use spectral information as input [10, 11, 12, 13]. In this article, it is shown that it is possible to calculate the same data by adding layers at the beginning of the network, with the advantage that the calculation of the spectrum can be adapted to the particular case.

The calculation of the spectral representation through neural networks was proposed by Moreira *et al.* [14]. In their study, they propose the calculation of the DFT with Cellular Neural Networks by separating the weights into groups that represent the real and imaginary parts. Training is not done, weights are assigned in a direct way. Velik, in [15], predicts the DFT with weights calculated from complex exponential functions also directly assigned, but she reports that the network is not able to be trained. Hoshen *et al.* show in [16] that a convolutional layer is able to approximate (through training) a filterbank that contains values comparable to the DFT coefficients. In [17], Sainath *et al.* are able to improve the state of the art in speech recognition by using filters learnt with a convolutional layer. Although the number of studies that use the original signal as input of the deep learning models continues to grow, neither the calculation of the cepstrum with neural networks nor the training of a network that predicts the power spectrum has been studied.

On the other hand, Anderson and Mallat [18] propose to replace the DFT by the Deep Scattering Spectrum (DSS) technique based on wavelet transforms, because DSS is able to represent invariant features in time (or space in the case of images). On some occasions, these deviations in time are significant for recognition. The current research is done in the vocal quality classification domain, where deviations in the frequency of vocal cords vibration are important [19].

4 Proposed Neural Network

In this section, the proposed neural network in order to calculate the cepstrum is presented. Further training tests will be performed later both on this model and on its modifications. The neural network predicts the cepstrum of two-second-long audio signals. The input is a vector of size $L = 50000$. The expected output for every audio file is a matrix, where each column contains the power cepstrum of a time segment. Note that the output values are elements of $\mathbb{R}$.

In order to obtain the defined output, the neural network must be made up of two parts: one part that calculates the power spectrum and another one that calculates the power cepstrum. In turn, each part has to fulfil two functions, the calculation of the DFT and that of the other operations (module, square and logarithm). To train the network, it is necessary to calculate the error gradient. As a consequence, the abovementioned operations are included in the network as layers. Back error propagation goes through the layers considering the derivatives of represented functions.

The calculation of the STFT of the power spectrum is made with a convolutional layer, where synaptic weights are the elements of matrix F and the activation function is linear. This is possible since both the DFT and the operation made by each neuron are linear transformations. It is important to note that the values of matrix F are constant.

There are two ways of implementing the calculation, depending on whether equation 1 or equation 2 of the DFT is chosen. If matrix F from equation 1 is used, weight matrix W made by complex coefficients w_{ij}, will have a size $N \times K$. In the case of equation 2, W, sized $N \times 2K$, will be made by the (real) values of matrices F_C and F_S. In terms of efficiency, there is no difference between the two alternatives. For this work, the second one is chosen because, as the output only retains information about the spectrum magnitude, it is possible to avoid the use of operations with complex numbers. This could be a practical advantage since, among the software libraries for neural networks, the support to complex numbers is still not general [20].

4.1 First Convolutional Layer

In order to calculate the STFT with a convolutional layer, it is convenient to write equation 2 in the following way:

$$X_k = \sum_{n=0}^{N-1} x_n \, \phi(k,n) + \sum_{n=0}^{N-1} i \, x_n \, \psi(k,n) \qquad (4)$$

were

$$\phi(k,n) = cos(-2\pi kn/N)$$
$$\psi(k,n) = sin(-2\pi kn/N)$$

If this is the case, complex numbers operations can be avoided in section 4.2. Ecuation 4 can be written in matrix form as:

$$X = F_C x + i F_S x$$

Figure 1 shows the proposed neural model. It can be noted that the output of the first convolutional layer contains the values of $F_C x$ and $F_S x$. These values are obtained by making the convolution of the input by each of the $2K$ (1760) kernels.

In the case of direct assignment of synaptic weights (without training), the first K kernels are assigned the elements of the K rows of matrix F_C and the rest, those of matrix F_S, in order. In this model, neurons do not

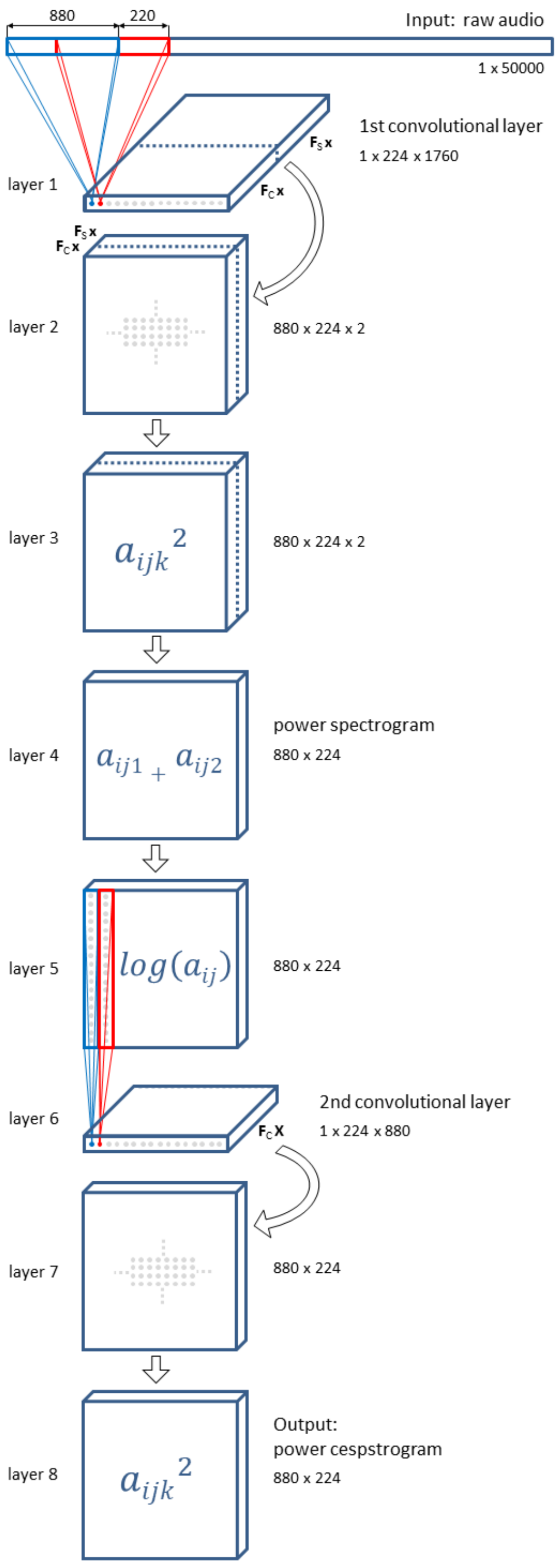

Figure 1: Model of artificial neural network that receives audio as input and predicts the power cepstrum.

use the w_{0j} weight because the DFT does not have an independent term. Formally, weight assignment is done in the following way:

$$w_{ijk} = f_{Cik} \tag{5}$$

$$w_{ij(k+K)} = f_{Sik} \tag{6}$$

where w_{ijk} is the synaptic weight of input i of neuron j of kernel k. f_{Cik} is the element of row i and column k of F_C. f_{Sik} is the element of row i and column k of F_S.

As weights are shared by the neurons of the same kernel, only a set of weights for every kernel is stored. Note that sub index j is not found in the second terms of equations 5 and 6. Weight matrix W is then sized $N \times 2K$ (880×1760).

4.2 Square of the Frequency Spectrum Magnitude

Starting from equation 4, the frequency spectrum magnitude is as follows:

$$\begin{aligned} |X_k| &= \left| \sum_{n=0}^{N-1} x_n\, \phi(k,n) + \sum_{n=0}^{N-1} i\, x_n\, \psi(k,n) \right| \\ &= \sqrt{\left(\sum_{n=0}^{N-1} x_n\, \phi(k,n) \right)^2 + \left(\sum_{n=0}^{N-1} x_n\, \psi(k,n) \right)^2} \end{aligned} \tag{7}$$

Note that complex numbers operations have been avoided in equation 7.

The results of the summations of equation 7 are the scalars in position k of vectors $F_C x$ and $F_S x$. Then,

$$|X_k| = \sqrt{(F_C x)_k^2 + (F_S x)_k^2}$$

where $(F_C x)_k$ and $(F_S x)_k$ are the elements of position k in vectors $F_C x$ and $F_S x$ respectively. Then, the power spectrum is calculated as

$$|X_k|^2 = (F_C x)_k^2 + (F_S x)_k^2$$

The output of the convolutional layer of the model in figure 1 is an array that contains vectors $F_C x$ and $F_S x$ corresponding to all the segments of the input signal. After convolution, three operations located in layers are performed. 1) A change in the shape of the array in order to simplify the third operation (layer 2), 2) the square of each element (layer 3) and 3) the sum of the pairs of values corresponding to the same frequency (k) and to the same time segment (layer 4). The result is a 880×224 matrix with the elements of the spectrogram.

4.3 Second Convolutional Layer

The function of the second convolutional layer is the calculation of the IDFT of the logarithm of the power spectrum. As its input, it receives the output of layer 5, that calculates the logarithm of the elements of the spectrogram (880×224). Different from the first convolution, the input has two dimensions in this layer: frequency $\times$ time. Each kernel has the size of a column of the input matrix, 880 (complete power spectrum) $\times 1$. In this way, for each time segment the calculation is:

$$C_k^* = \frac{1}{N} \sum_{n=0}^{N-1} |X_n|^2 \cos(-2\pi kn/N)$$

and it can be expressed in matrix form as:

$$C^* = \frac{1}{N} F_C |X|^2$$

where C^* is the result of the inverse transform. In section 2.3 it was explained that, for this case, the IDFT can be replaced by the discrete cosine transform (DCT).

In the case of direct assignment of weights, $w_{ijk} = \frac{1}{N} f_{Cik}$.

4.4 Square of the Second Convolution

Finally, the power cepstrum is obtained in the layer 8. It calculates the square of the output of the second convolutional layer. $C_k = C_k^{*2}$.

5 Experiments

5.1 Training

As it was mentioned before, instead of having weights assigned in a direct way, it is possible to train the network. This possibility is important since it implies that the network can be initialized with assigned weights (random or not) and can adapt to new needs.

The proposed neural network will be trained through the gradient descent method of the mean squared error (MSE) function. The calculation of the gradient includes the derivatives of the logarithm, addition and squares.

5.1.1 Variants of the model

Layer 5 calculates the logarithm of the output of layer 4. Discontinuity in the derivative of the logarithm is an issue when back propagating error gradient. Therefore, difficulties can be found when weights of the first convolutional layer are trained. In order to analyse the neural network training capacity considering the situation explained above, three variants of the model are trained, namely:

- Model I. Calculates the power spectrum ($|X_k|^2$). Uses layers 1 through 5 of the proposed model.

- Model II. Calculates the power cepstrum. It is the complete model proposed in section 4.
- Model III. The model proposed in section 4 is modified by having layer 5 removed (logarithm). The model that results from eliminating the logarithm returns the autocovariance, another transformation used in signal analysis which is useful for determining resonance frequency by identifying the highest peak.

5.1.2 Initial Conditions

In [9] we showed that a neural network of similar complexity to that of model I can calculate spectrum magnitude $|X_k|$. In this work, the weights of model I are initialised with random values, as in [9].

Models II and III ave two layers with weights, so training should be more difficult than the one for model I. In order to make a more detailed analysis of the training of models II and III, training is held with three different initial conditions:

- Starting from random weights (R). This condition supposes a greater difficulty than the next conditions, as initial weights are farther from optimal weights than the weights in the other two conditions.
- Starting from theoretical weights with an addition of 10% of noise (N).
- Starting from theoretical weights, but replacing the expected output by the power cepstrum of the audio multiplied by a Tukey window (tapered cosine window) with $r = 0.25$ (W).

Figure 2 shows an example of an audio, of the window function and of the audio multiplied by the window corresponding to conditions W.

Additionally, the following tests are made for every initial condition of models II and III:

- Training the complete model (TCM).
- Just training the first convolutional layer (CL1). The complete model is used. The second convolutional layer is initialised with theoretical weights. Then, the model is trained without any modification in the weights of the second convolutional layer.
- Just training the second convolutional layer (CL2). The complete model is used. The first convolutional layer is initialised with theoretical weights. Then, the model is trained without any modification in the weights of the first convolutional layer.

The aim of these three tests is to determine, in case the complete model cannot be trained, whether each of the convolutional layers can reach optimal weights when the other layer is already adapted.

In all the cases, when weights are initialised with random values, they have a uniform distribution between -0.001 and 0.001. Adam (Adaptive Moment Estimation), the optimisation method [21], was used. It is a variant of the gradient descent method, with $\alpha = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Weights were updated in batchs sized 300 (the total training data). The calculations were made on a GPU NVIDIA Titan Xp.

In section 6, the results of this process are presented and compared to those obtained through direct assignment.

5.2 Data

Audios are taken from the Voice Disorders Database (VDD) [22], recorded by the Polytechnic University of Madrid in collaboration with *Príncipe de Asturias* University Hospital. Healthy people and people with vocal pathologies who pronounce a sustained vowel /a/ for approximately two seconds were recorded.

The audios are in WAV format with a rate of 25000 samples per second. For all files with a duration longer than 2 seconds, the 50000 central samples were taken. The input data were then defined by 430 vectors of size $L = 50000$, from which 300 and 130 were randomly chosen for training and for validation respectively.

For outputs, the calculation is made according to equation 3 in the following way:

Firstly, the spectrogram is calculated as the square of the absolute value of the STFT of the inputs with segments sized $N = 880$ and overlap $m = 660$ elements, which implies a displacement of 220 elements in each transformation. The spectrogram calculated in this way has 224 columns (time) by 880 rows (frequency).

The output is calculated as the square of the absolute value of the IDFT of the logarithm of each of the power spectrum (columns) of the spectrogram.

The outputs are defined by vectors of size 880×224.

6 Results

6.1 Model I

The results of 30000 training cycles of the model that calculates the power spectrum are presented hereafter.

The validation MSE reached was 1.63×10^{-6} (9.45×10^{-6}% of the mean value of the expected output), whereas an MSE $< 10^{-9}$ is obtained for the same model with weights assigned in a direct way.

The output obtained through training is very near the expected one. Figure 3 shows the expected output for one element of the validation dataset and the output obtained after training. As it can be seen, there are no visual differences between them.

Despite obtained results are good, it can be observed that trained weights are very different from theoretical

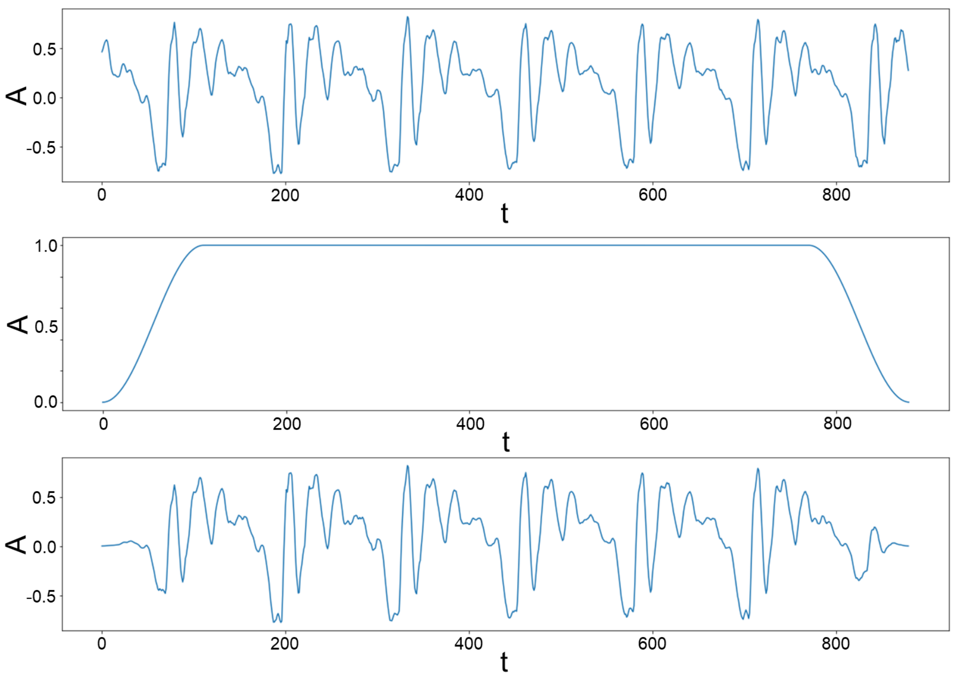

Figure 2: Amplitude (A) in time (t) of 880 samples (0.0352 seconds) of a sustained vowel /a/ (top). Tukey window for $r = 0.25$ (middle). Audio multiplied by Tukey window (bottom).

weights. It is important to analyse these differences in order to gain further understanding about how the neural network calculates the DFT. A comparison between the theoretical weights calculated from the DFT equations and trained weights is presented below.

In figure 4, the values of F_C and F_S for weights assigned in a direct way (theoretical) and for trained weights can be seen. In all the cases, it is clearly observed that the left rows of the transposed matrices represent low frequencies, and the right ones, high frequencies.

It is also evident that both groups, theoretical and trained weights, present different image patterns. Trained weights look more "untidy". The origin of this phenomenon is that the DFT decomposes the input into a weighted sum of sinusoidal signals found in matrices F_C and F_S. The training method for every value of k, finds a pair F_C and F_S made by sinusoidal waves outphased in 90° that allows the desired decomposition. However, this solution is not necessarily the same as in equation 4.

Figure 5 shows some examples of trained weights vs. theoretical weights for particular values of k. Note that the 90° outphase between F_C and F_S is always respected, both among the pairs of theoretical weights and among the pairs of trained weights. This can be proved by calculating $mod = \sqrt{F_C^2 + F_S^2}$ for any value of k. For theoretical weights, obviously $mod = 1$, whereas for trained weights, a value very near to 1 is always obtained. Therefore, an orthogonal base is found to make the decomposition.

6.2 Models II and III

Tables 1 and 2 show the errors obtained during the trainings of both calculation models, that of the power cepstrum and that of the variant without the logarithm respectively.

As the output values are not standardised and the outputs of the models compared have different magnitudes, instead of showing the MSE, the ratio between the mean absolute error (MAE) and the output mean value is shown. The empty spaces show that the training was not successful.

Table 1: Ratio between the mean absolute error (MAE) and the output mean value for model II according to the conditions defined in section 5.1.2.

	R	N	W
TCM	-	-	-
CL1	-	-	-
CL2	$1.17\ 10^{-5}$	$5.27\ 10^{-6}$	-

6.2.1 Model II

For the model II, only cases CL2-R and CL2-N could be trained. In the first case, the weights of the first convolutional layer were initialised with theoretical weights and remained fixed, whereas those of the second convolutional layer were initialised with random values and were trained. The second case is similar, but the weights of the second layer are initialised with

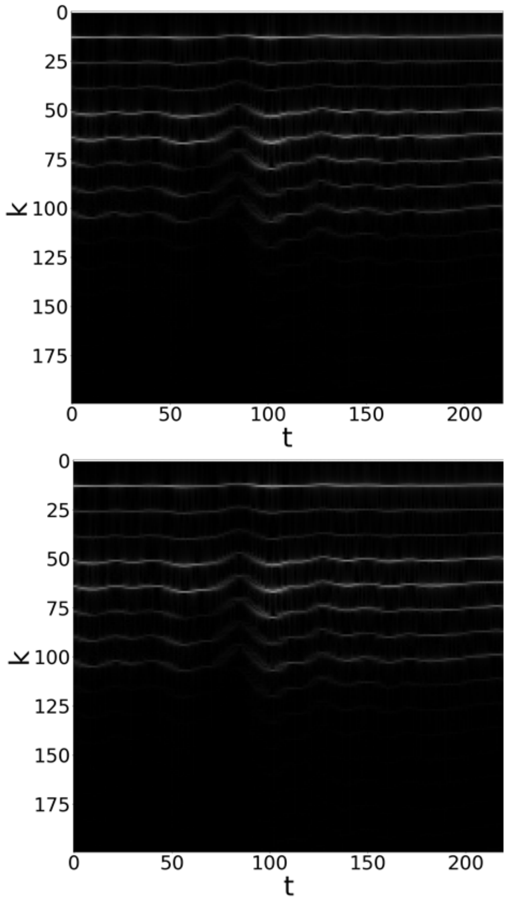

Figure 3: Spectrograms (first 200 rows) of output of the model I, where k is the row number and t is time. Expected output (top) and obtained output with trained weights (bottom).

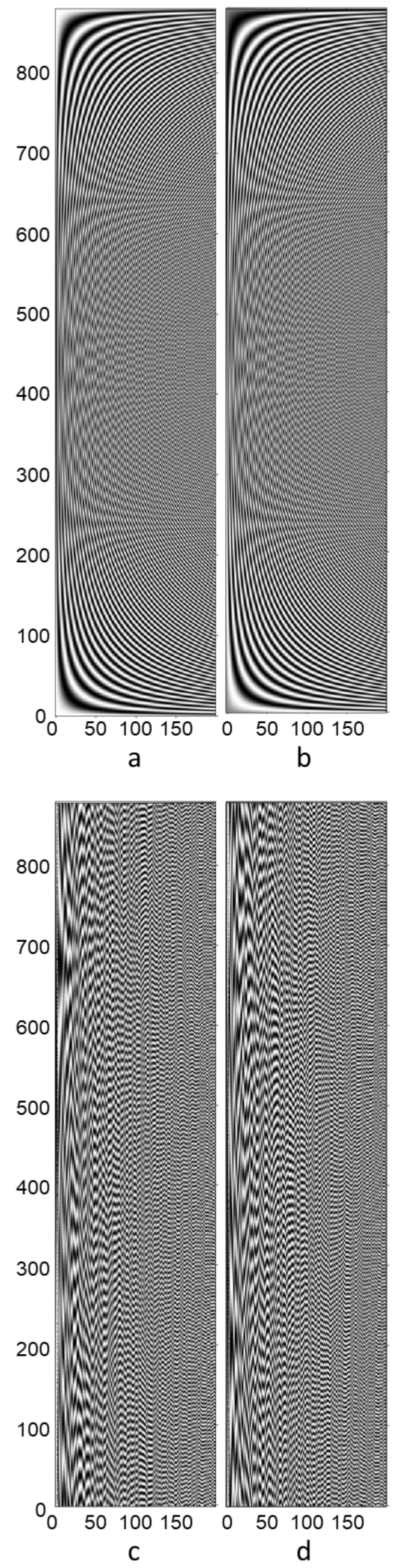

Figure 4: Weights of model I. Kernels 0 to 200 of coefficient matrices F_C with theoretical weights (a), F_S with theoretical weights (b), F_C with trained weights (c) and F_S with trained weights (d).

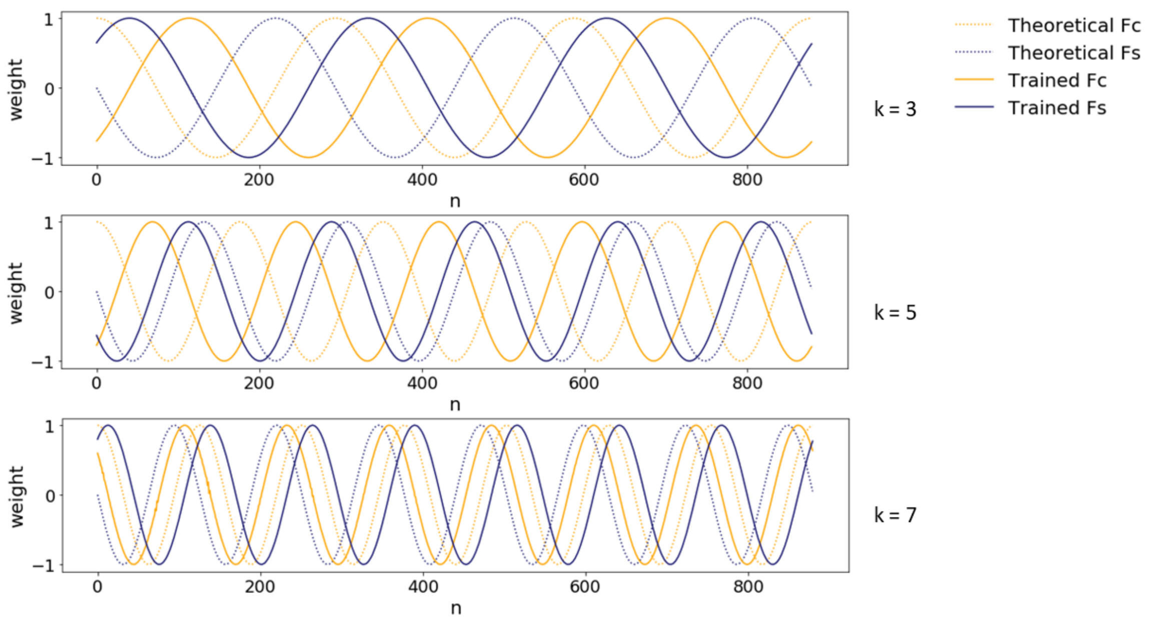

Figure 5: Trained weights vs. theoretical weights for $k = 3$, $k = 5$ and $k = 7$ in model I.

Table 2: Ratio between the mean absolute error (MAE) and the output mean value for model III according to the conditions defined in section 5.1.2.

	R	N	W
TCM	-	$1.33\ 10^{-4}$	$6.46\ 10^{-4}$
CL1	$1.46\ 10^{-4}$	$7.89\ 10^{-5}$	$1.73\ 10^{-5}$
CL2	$1.22\ 10^{-5}$	$5.50\ 10^{-6}$	-

theoretical weights, and noise is added. It is not surprising that these cases are trainable, as the calculation is similar to that of the power spectrum because they make the square of the DFT magnitude.

In none of the TCM and CL1 cases model II could be trained. For these cases, it is necessary to modify the weights of the first convolutional layer. In order to do this, error has to be propagated through layer 5 (logarithm). Due to discontinuity in the derivative of the logarithm, propagated information does not allow a proper search by gradient descent. Hence, TCM and CL1 cases cannot be trained.

Case CL2-W cannot be trained either. In the next section the reason for this is described.

6.2.2 Model III

For model III (without logarithm), case TCM-R could not be trained. This is the complete model in which all the weights are initialised randomly. The rest of the cases (except for CL2-W) could be trained. This means that, in case model III is initialised with weights near the optimal ones (for example, theoretical weights), it can be trained to adapt to particular problems. This also confirms that the problem of model II is the derivative of the logarithm.

Note that case CL2-W cannot be trained because the solution requires modifying the first convolutional layer. Figure 6 shows some of the weights of the first convolutional layer for case CL1-W. It can be clearly seen that weights tend towards the product between theoretical weights and the window used. Therefore, if modifying the weights of the first convolutional layer is not allowed, case CL2-W (both in model III as in model II) cannot be trained.

7 Conclusions

One conclusion is that a neural model is able to calculate the DFT, both for theoretical and for trained weights, and that the latter do not necessarily tend towards the former, although they share frequency and the condition of orthogonality.

Another conclusion drawn is that the complete model presented is able to calculate the cepstrum correctly, but neither can it be trained nor it has the capacity to adapt to other problems through training, since the derivative of the logarithm function is not adequate to backpropagate the error towards the first convolutional layer.

As future work, it is planned to create a neural network to calculate the fundamental frequency (F_0) of voice based on model III and initialised with theoretical weights. This model, equivalent to the autocovariance function, keeps some of the advantages of the cepstrum, such as information to detect the fundamental frequency of the signal. F_0 can be obtained from the location of the autocovariance highest peak.

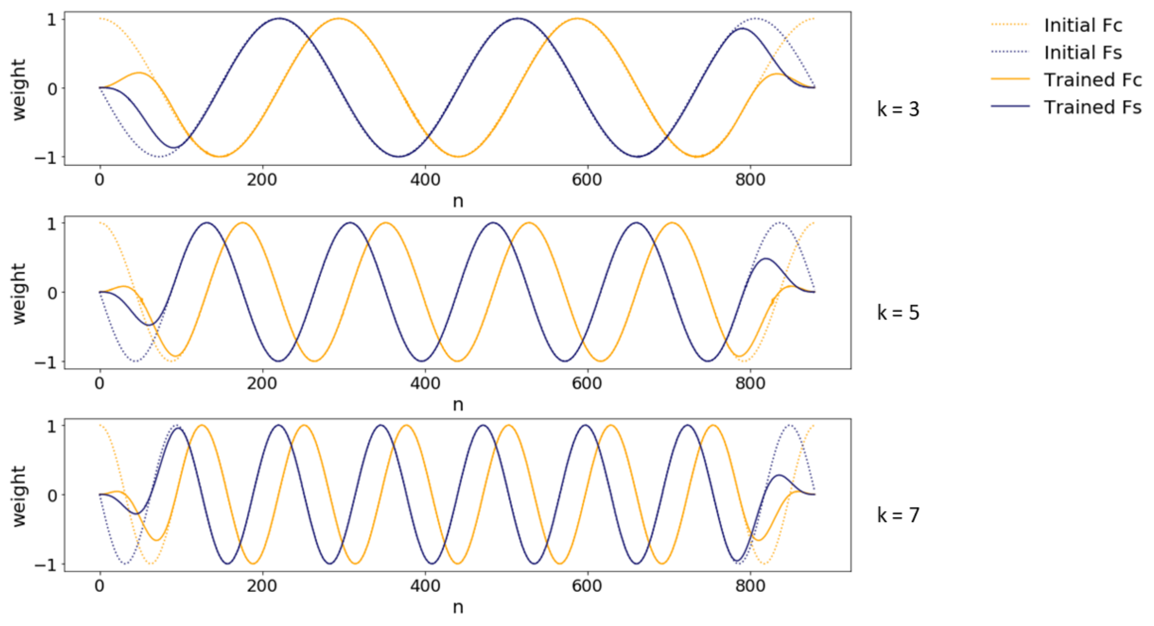

Figure 6: Trained weights vs. initial weights for case CL1-W in model III.

Acknowledgements

We gratefully acknowledge the support of NVIDIA Corporation through the NVIDIA GPU Grant Program.

Competing interests

No competing interests exist.

References

[1] L. D. Dong Yu, *Automatic speech recognition, a deep learning approach*. Springer-Verlag London, 1 ed., 2014.

[2] J. McClellan and T. Parks, "Eigenvalue and eigenvector decomposition of the discrete fourier transform," *IEEE Transactions on Audio and Electroacoustics*, vol. 20, no. 1, pp. 66–74, 1972.

[3] A. M. Noll, "Cepstrum pitch determination," *The journal of the acoustical society of America*, vol. 41, no. 2, pp. 293–309, 1967.

[4] A. V. Oppenheim and R. W. Schafer, "From frequency to quefrency: A history of the cepstrum," *IEEE signal processing Magazine*, vol. 21, no. 5, pp. 95–106, 2004.

[5] B. P. Bogert, "The quefrency alanysis of time series for echoes; cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking," *Time series analysis*, pp. 209–243, 1963.

[6] R. Randall, J. Antoni, and W. Smith, "A survey of the application of the cepstrum to structural modal analysis," *Mechanical Systems and Signal Processing*, vol. 118, pp. 716–741, 2019.

[7] C. Madill, D. D. Nguyen, K. Yick-Ning Cham, D. Novakovic, and P. McCabe, "The impact of nasalance on cepstral peak prominence and harmonics-to-noise ratio," *The Laryngoscope*, 2018.

[8] V. S. McKenna and C. E. Stepp, "The relationship between acoustical and perceptual measures of vocal effort," *The Journal of the Acoustical Society of America*, vol. 144, no. 3, pp. 1643–1658, 2018.

[9] M. A. García and E. A. Destéfanis, "Spectrogram prediction with neural networks," in *XXIV Congreso Argentino de Ciencias de la Computación (Tandil, 2018).*, 2018.

[10] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv preprint arXiv:1609.03193*, 2016.

[11] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "An experimental study on speech enhancement based on deep neural networks," *IEEE Signal processing letters*, vol. 21, no. 1, pp. 65–68, 2014.

[12] M. J. Putten, S. Olbrich, and M. Arns, "Predicting sex from brain rhythms with deep learning," *Scientific reports*, vol. 8, no. 1, p. 3069, 2018.

[13] A. M. Badshah, J. Ahmad, N. Rahim, and S. W. Baik, "Speech emotion recognition from spectrograms with deep convolutional neural network," in *Platform Technology and Service (PlatCon),*

2017 International Conference on, pp. 1–5, IEEE, 2017.

[14] O. Moreira-Tamayo and J. P. De Gyvez, "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*, vol. 3, pp. 76–79, IEEE, 1996.

[15] R. Velik, "Discrete fourier transform computation using neural networks," in *2008 International Conference on Computational Intelligence and Security*, pp. 120–123, IEEE, 2008.

[16] Y. Hoshen, R. J. Weiss, and K. W. Wilson, "Speech acoustic modeling from raw multichannel waveforms," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 4624–4628, IEEE, 2015.

[17] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[18] J. Andén and S. Mallat, "Deep scattering spectrum," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4114–4128, 2014.

[19] M. A. García and E. A. Destéfanis, "Deep neural networks for shimmer approximation in synthesized audio signal," in *Argentine Congress of Computer Science*, pp. 3–12, Springer, 2017.

[20] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," *arXiv preprint arXiv:1705.09792*, 2017.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] J. D. Arias-Londoño, J. I. Godino-Llorente, M. Markaki, and Y. Stylianou, "On combining information from modulation spectra and mel-frequency cepstral coefficients for automatic detection of pathological voices," *Logopedics Phoniatrics Vocology*, vol. 36, no. 2, pp. 60–69, 2011.

Citation: M. García and E. Destéfanis. *Power Cepstrum Calculation with Convolutional Neural Networks*. Journal of Computer Science & Technology, vol. 19, no. 2, pp. 132–142, 2019.
DOI: 10.24215/16666038.19.e13
Received: March 15, 2019 **Accepted:** July 04, 2019.