Audio Engineering Society

Student Design Competition

Presented at the 150th Convention
2021 June 1–4, Online

# CircuitComparedre : A desktop application to aid audio circuit design

Christopher Johann Clarke (advisor: Jer-Ming Chen)

*Singapore University of Technology and Design*

Correspondence should be addressed to Christopher Clarke (*christopher_clarke@mymail.sutd.edu.sg*)

**ABSTRACT**

Technical ear training towards audio production and critical listening, is a skill that takes a large amount of time to develop. This poses a problem for audio circuit design, when the designer is faced with having to decide between two or more similar performing (sounding) audio circuits. We present an application developed to aid the design process by focusing on objective empirical values, through audio feature extraction, and utilising machine learning to reconcile similarities and differences. The application features the ability to compare up to 6 circuits at once, provides a default and extended feature extraction process, and allows the user to easily create inferences to the comparison made.

## 1 Introduction

The recent passing (12 Feb 2021) of an inspiring figure in audio design, Rupert Neve, has motivated my idea to put forth a design tool for audio electronic design. Anecdotally, in my own audio design process, there is always a struggle to decide between two circuits or more circuit designs in one part of the design.

When pitting two circuits (usually nominally equivalent in performance) circuit designs against each other, it is difficult to make an objective decision, as the electrical characteristics of the circuits are very similar. Characteristics are audio features [1][2][3] such as, Total Harmonic Distortion (THD), Spurious Free Dynamic Range (SFDR), SIgnal-to-Noise And Distortion ratio (SINAD), and Signal-to-Noise Ratio (SNR), which contribute to the sonic signature [4] of a circuit.

In Figure 1, the squares A and B represent two make-up gain amplification circuits that are being considered for this section of a Voltage Controlled Amplifier (VCA) based compressor design. The dilemma represented in this case: How does one make an objective decision between the two circuits? Even with trained ears, the differences might be too minute to distinct between.

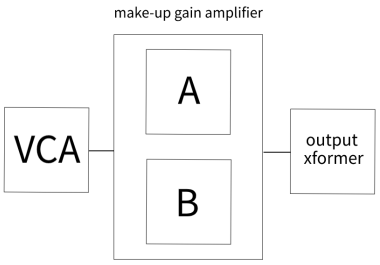


**Figure 1:** Block Diagram of an imaginary dilemma, where A and B refer to equally performant "make-up gain amplifier" circuits to choose from in a VCA dynamic range compressor design.

In the scope of this example, perception is not being used as a basis to judge the performance of the amplifier. Instead, we propose to use objective numerical values of the audio features, and distinguish between the differences using machine learning.

## 2 Description of Application Developed

CircuitComparedre (['sɜːkɪt kəm'pɛːdreɪ]) is an application that has been developed using MATLAB to simplify the process of analysing the data that needs to be collected to assess the differences between the circuits.

### 2.1 Collection of Input Data

The data collection mechanism is not built into the application, and therefore needs to be separately prepared. Figure 2 illustrates a procedure to collect data. The function generator could be a signal source that is computer controllable, which will allow different spacing between each datapoint produced by means of an algorithm or comma-separated values (CSV) file. The function generator should also be of low distortion, to minimise errors in the audio feature calculation.

The data (audio) output of the circuit can be collected with a device with an analog-to-digital converter (ADC). For the convenience of the user, the application has been designed to read audio files. This allows the user to record the signal directly from a USB audio interface.

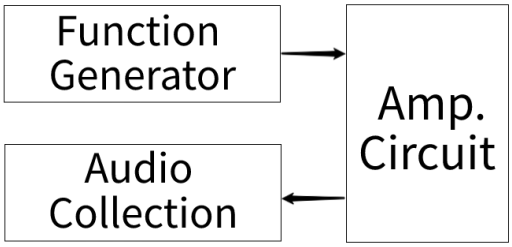


**Figure 2:** Block Diagram of the advised data collection procedure.

The input data should consist of variety in both frequency and amplitude. Specifically, the input data should reflect the range of use-cases that the circuit will be exposed to, in order to map the full extent of device performance. However, due to the constraints of time and storage, a compromise needs to be made with regards to the number of datapoints collected. Collecting datapoints of 2 seconds in length, 0.1 Hz apart, with 10 mV spacing in the input amplitude range from 10 mV to 1 V peak-to-peak, will result in about $2 \times 10^7$ datapoints taking about 1.3 years to record. At a sample rate of 44,100 Hz and a bit depth of 16 bit, this would expend 4 TB.

With frequency and amplitude data, the granularity required should depend on prior knowledge about the performance of the circuit. This optimises the time taken to collect the data, and the amount of storage needed. As an example, if there is suspicion of a difference in the region of lower frequencies, the spacing of the frequencies can be adjusted with a function such as:

For the $n^{th}$ frequency $F_n$,

$$F_n = exp\left[0.0658n\right] + 20, \tag{1}$$

where $20 \leq F_n \leq 20000$.

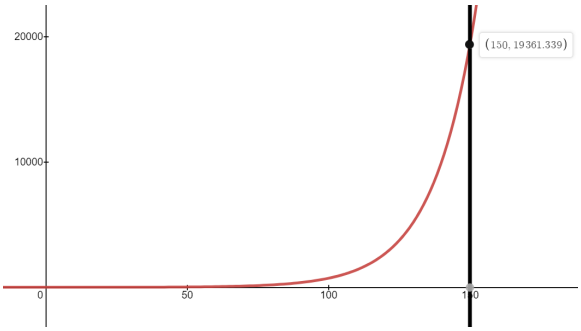


**Figure 3:** Plot of the function in Equation 1

Such a function allows for 150 datapoints along the frequencies between 20 to 20,000 Hz, with 50 of the datapoints below 47 Hz.

Since the performance of the circuit changes with the incoming signal, a range of input amplitudes should also be sampled. In the case of a frequency spacing such as **??**, with amplitudes that are between 50 mV to 600 mV, with 10 mV increments, the dataset will contain 8250 datapoints of the circuit. This gives us enough datapoints for machine learning, while still keeping acquisition time feasible (at about 4.6 hours).

There is a naming convention that the application requires for each of the datapoints collected. To reduce

the error of the measured audio features, it is necessary to label the input frequency and amplitude of the recorded datapoint in this format:

$$(name)\_f(freq.)\_a(amp.).wav, \quad (2)$$

with an example being *distcircuit_f1000_a0.5.wav*.The frequency labelled *freq.*should be the fundamental frequency in Hz of the datapoint collected, and the amplitude labelled *amp.*should be the amplitude in V of the datapoint collected.

## 2.2 Features of the Application

Figure 4 shows the Graphical User Interface (GUI) of the developed application CircuitComparedre.

The application can be divided into 3 separated processes. Loading the audio files for analysis, extracting the features, and training the model. The application supports the analysis of up to 6 different circuits at the same time.

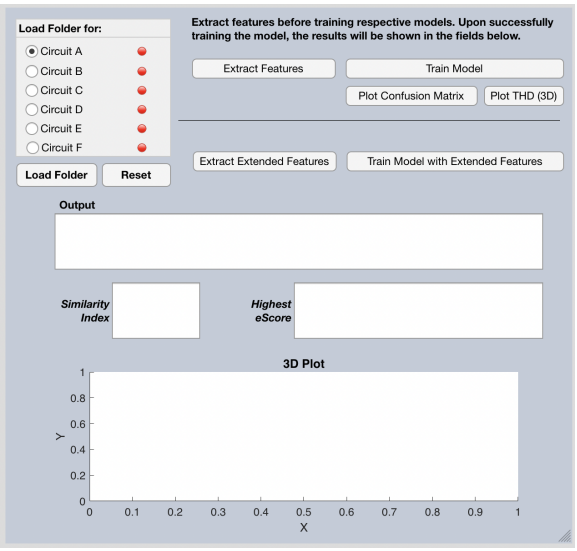


**Figure 4:** GUI of CircuitComparedre

### 2.2.1 Loading the data

The process of loading the data consists of selecting from the radio group a circuit that has not been loaded. If the circuit has not been loaded yet, the lamp on its right side will be red. Clicking on Load Folder, will cause a pop-up to allow you to select a folder. Once a folder has been loaded, the lamp will be green.

### 2.2.2 Feature Extraction

The "Extract Features" and "Extract Extended Features" buttons begin the process of creating the various feature spaces for the combined dataset of Circuit A - F.

**Table 1:** Features extracted for the default and extended feature extraction buttons.

| Button | Features |
|---|---|
| "Extract Features" | RMS (dB) |
| | THD |
| | SFDR |
| | SINAD |
| | SNR |
| "Extract Extended Features" | RMS (dB) |
| | THD |
| | SFDR |
| | SINAD |
| | SNR |
| | Spectral Centroid |
| | Spectral Crest |
| | Spectral Entropy |
| | Spectral Flux |
| | Spectral Flatness |
| | Spectral Kurtosis |
| | Spectral Skewness |
| | Spectral Slope |
| | Spectral Spread |

The set of features from the "Extract Extended Features" button was made using the MATLAB Audio Toolbox: Spectral Descriptors functions [5].

### 2.2.3 Training the Model

The "Train Model" button uses the features extracted and begins the process of training a Support Vector Machine (SVM) model. This SVM model [6] [7] makes use of the multi-class Error-Correcting Output Code model in MATLAB, performs 10-fold cross validation [8] and returns the value of the cross-validated classification error as the "Similarity Index". The higher this value, the more likely the circuits are similar.

The "eScore" on the right of the "Similarity Index" returns the value(s) of the predicted classification scores for cross-validated 10-fold observations using the classifier trained on the training observations. A higher

"eScore" relates to a greater confidence level of classification, which indicates differences between the circuits. This meaningfully separates the similar circuits from the different circuits. Read from left-to-right, the numbers will be presented based on the amount of circuits being compared. If there are circuits in A to D, there will be 4 numbers returned, with the first number on the left representing the "eScore" of the cross-validated model's confidence level of circuit A. As an example, the numbers are returned as:

$$0.98, 0.29, 0.43, 0.97 \tag{3}$$

This informs us that the confidence level for Circuits B and C are low, and that there are similarities between them.

#### 2.2.4 Other Features

There are two other features of the application. One of which is plotting a confusion matrix, and the other is a 3D plot of the THD. This allows the user to visually inspect the confusion matrix (a visual representation of the model's predicted labels versus the true labels) to have a detailed view of the misclassification rate. For example, in a 3 circuit comparison, circuit A has a chance to be misclassified as either circuit B or C. This also informs the user about the inter-circuit similarities and differences. Finally, visual inspection can be performed on a familiar (and more intuitive) feature such as the THD, through the use of the "Plot THD (3D)" button. This button returns the plot of frequency (x-axis), amplitude (y-axis), and THD (z-axis), with the colour scheme of red, green, blue, cyan, magenta, and yellow respectively for circuits A - F.

#### 2.2.5 Future Work

Currently, the application is not able to automatically inform the user about the pattern/behaviour of the similarities or differences of the circuits being tested. This would include details about which audio feature played the biggest role in allowing the circuit to be classified (correctly/incorrectly). In this version, this has to be done through visual inspection using the "Plot THD (3D)" button. However, this part of the application is currently being implemented and an update will be released as soon as it is ready.

## 3 Summary

The application proposed aids in the audio circuit design process through audio feature extraction and machine learning. This takes a step towards abstracting the vast amount of information required to analyse circuit comparisons, and presents under 10 numerical values to the user to make sense of the comparison. Although visual inspection is still necessary at this point in time, ground is being broken on the implementation of feature significance and automatic difference detection (see section 2.2.5).

Seasoned (perceptually-capable) audio circuit designers have spent years developing their ears to listen for minute differences between audio circuits. Perhaps this application also aids in closing the gap between new and experienced audio circuit designers, by leveraging on machine learning as an aid.

## References

[1] Analog Devices Inc., E., *Data Conversion Handbook*, Elsevier Science, 2004, ISBN 9780080477015.

[2] Sheingold, D., *Analog-digital Conversion Handbook*, Analog Devices technical handbooks, Prentice-Hall, 1986, asin B000JL2FOO.

[3] Analog Devices Inc., E. and Zumbahlen, H., *Linear Circuit Design Handbook*, Elsevier Science, 2011, ISBN 9780080559155.

[4] Gottinger, B., *Rethinking distortion: Towards a theory of 'sonic signatures'*, Ph.D. thesis, 2007.

[5] MATLAB, "Spectral Descriptors," 2019, retrieved from https://www.mathworks.com/help/releases/R2019a/audio/ug/spectral-descriptors.html . Last accessed 12 May 2021.

[6] Meyer, D., Leisch, F., and Hornik, K., "The support vector machine under test," *Neurocomputing*, 55(1), pp. 169–186, 2003, ISSN 0925-2312, doi:https://doi.org/10.1016/S0925-2312(03)00431-4, support Vector Machines.

[7] Noble, W. S., "What is a support vector machine?" *Nature Biotechnology*, 24(12), pp. 1565–1567, 2006, doi:10.1038/nbt1206-1565.

[8] Moore, A. W. and Lee, M. S., "Efficient Algorithms for Minimizing Cross Validation Error," in W. W. Cohen and H. Hirsh, editors, *Machine Learning Proceedings 1994*, pp. 190–198, Morgan Kaufmann, San Francisco (CA), 1994, ISBN 978-1-55860-335-6, doi:https://doi.org/10.1016/B978-1-55860-335-6.50031-3.