Objectively Choosing Spectrogram Parameters to Classify Environmental Noises
Edward T. Nykaza\textsuperscript{1}; Anton Netchaev\textsuperscript{2}; Steven Bunkley\textsuperscript{3}; Matthew G. Blevins\textsuperscript{4}
\textsuperscript{1,2,3,4} Engineer Research and Development Center, USA

ABSTRACT
Spectrograms are commonly used to visualize, analyze, and classify audio signals in the same way that social media companies (e.g., Google, Facebook, Yahoo) use images to classify or tag people in photos. A problem unique to using spectrograms to classify acoustic signals is that the user must choose the spectrogram input-parameters, which may affect the accuracy of the resulting classifier. While the spectrogram – in its simplest form – only has three input-parameters, each parameter has a large number of possible values it can take, resulting in a nearly infinite number of combinations and unique spectrograms. The three input-parameters include the window-type, window-size, and percent-overlap-between-windows. The process of choosing spectrogram parameters, however, is often glossed over in the literature, and there is typically little guidance on how to make this, often, subjective choice. We hypothesize that the choice of spectrogram input-parameters will affect the spectrogram output or features that in turn will affect the performance of the acoustic classifiers. To test this hypothesis, we use Matlab’s built-in spectrogram function, a support-vector-machine classifier, a labeled (i.e., human classified) environmental noise dataset, and randomly sample the spectrogram input-parameter space to objectively choose the spectrogram input-parameters. We find that the random sampling procedure is a useful way of choosing the spectrogram input-parameters, and finding the spectrogram features that are the most important for classifying environment noises. The environmental noises used in this study include the noise from air conditioners, car horns, children playing, dogs barking, drilling, engine idling, gunshots, jackhammers, sirens, and street music.

Keywords: Spectrogram, classification, environmental noise
I-INCE Classification of Subjects Numbers: 50, 74.8

1. INTRODUCTION
With the advent of reliable and continuously-operating environmental noise-monitoring systems\cite{1, 2}, we are now faced with a superabundant amount of noise-monitor data; an intractable amount of data for humans to process. There is both a need for robust classification algorithms that can automatically classify environmental noise-sources, and the goal of removing humans from the classification process. As a result of this need and goal, many of the newer environmental noise-monitoring systems include on-board noise-source classifiers that classify a few noise-sources (e.g., aircraft, thunder, or blast noise). However, there are no systems currently available that are able to classify all environmental noise-sources. Admittedly, this is quite an intractable problem; yet, one that has been solved, or nearly solved, in the speech and music recognition domains. For example, think of the software programs and apps that are available to translate your voice to text, or that can tell you what song is playing on the radio.

One of the ways speech and music signals are classified is using the spectrogram. That is, the output from a short-time Fourier transform (STFT) or spectrogram is used as features in a supervised classification problem. A spectrogram (Figure 1) shows the acoustic intensity or power spectral density (PSD) of a signal over time and frequency \cite{3}. Spectrograms are essentially images of acoustic signals, and can be used to classify signals in the same way that social media companies (e.g., Google, Facebook, Yahoo, etc.) use images to classify or tag people in a photo.

\cite{1} edward.t.nykaza@erdc.dren.mil
\cite{2} anton.netchaev@erdc.dren.mil
\cite{3} steven.l.bunkley@erdc.dren.mil
\cite{4} matthew.g.blevins@erdc.dren.mil
A problem unique to using spectrograms to classify acoustic signals is that the user must choose the spectrogram input-parameters, which may affect the accuracy of the resulting classifier. While the spectrogram – in its simplest form – only has three input-parameters that the user needs to select, each parameter has a large number of possible values it can take, resulting in a nearly infinite number of combinations and unique spectrograms. The three input-parameters include the window-type, window-size, and percent-overlap-between-windows. The process of choosing spectrogram parameters, however, is often glossed over in the literature, and there is typically little guidance on how to make this, often, subjective choice. In many cases, the default parameters of the spectrogram function are used, the choice is based upon a precedent set in the literature, or the conventional wisdom to chose the spectrogram that looks the best.

We hypothesize that the choice of spectrogram input-parameters will affect the spectrogram output or features that in turn will affect the performance of the acoustic classifiers. At the outset, we are pretty confident that the choice of spectrogram input-parameters will affect the resulting classifier, but how big will the effect be, and which input-parameters are the most important for classifying environmental noise-sources? Some of the other questions we hope to answer include: Is it possible to obtain good classifier results with spectrograms that are not computationally intensive (e.g., spectrograms with low sampling rates and a small number of Fast Fourier Transform (FFT) points)? How does the spectrogram classifier compare to other environmental noise classifiers (e.g., classifiers built with sound level meter (SLM) metrics (4), or Mel-Frequency Cepstral Coefficients (MFCC) metrics (5))?  

2. METHODS

In order to answer our main question (i.e., how do the spectrogram input-parameters affect classifier accuracy?) and test our hypothesis, we use an environmental noise dataset (5), randomly sample the spectrogram input-parameter space, input those parameters into Matlab’s built-in spectrogram function (6), input the resulting spectrogram into a support-vector-machine (SVM) classifier (7), and assess the accuracy of the classifier using 5-fold cross validation. We repeat this process for many random samples, and visualize the results using Decision Tree (8) and Random Forest (9) analysis methods.
2.1 Environmental Noise Dataset

The environmental noise dataset analyzed in this paper contains 8732 acoustic signals with approximately 1000 signals for each of the 10 noise-sources (Table 1). These acoustic sources have been cleaned (5) and curated/labeled (10), which allows us to perform a supervised classification analysis, and to address how accurately the each spectrogram can classify the environmental noise sources. A spectrogram for each class of noise is given in Figure 2.

Table 1 – Environmental noise source class list, with a link to an audio example of each source.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air Conditioner</td>
<td>1000</td>
</tr>
<tr>
<td>Car Horn</td>
<td>429</td>
</tr>
<tr>
<td>Children Playing</td>
<td>1000</td>
</tr>
<tr>
<td>Dog Bark</td>
<td>1000</td>
</tr>
<tr>
<td>Drilling</td>
<td>1000</td>
</tr>
<tr>
<td>Engine Idling</td>
<td>1000</td>
</tr>
<tr>
<td>Gunshot</td>
<td>374</td>
</tr>
<tr>
<td>Jackhammer</td>
<td>1000</td>
</tr>
<tr>
<td>Siren</td>
<td>929</td>
</tr>
<tr>
<td>Street Music</td>
<td>1000</td>
</tr>
</tbody>
</table>

Figure 2 – Spectrogram examples of each noise source.
2.2 Spectrogram

Matlab’s spectrogram function (6), in its simplest and most common form, has 3 variable input-parameters: window-type, window-size, and the overlap-between-windows. For this paper, we varied 6 input-parameters. In terms of the most common form, we varied the window-type (windowType), percent-overlap-between-windows (%Overlap), and chose to vary the number of time bins (nTimeBins) instead of the window-size, though these two parameters are directly related. In addition, and in order to cut down on computational processing time, we varied the number of points in the FFT (nfft), the signal length (signalLength), and the sampling rate (sampleRate).

More specifically, we considered 16 windowTypes from Matlab’s signal processing toolbox and used the default parameters for each: bartlett, barthannwin, blackman, blackmanharris, bohmanwin, chebwin, flattopwin, gaussian, hamming, hann, kaiser, nuttallwin, parzenwin, rectwin, taylorwin, triang, tukeywin. We varied the %Overlap between 1 to 99% in steps of 1, the nfft between 2^6 to 2^10 in power of 2 steps, the nTimeBins in steps of 1 between 1 and the minimum of 128 and half the sampling rate, the signalLength between 0.5 and 3 seconds in steps of 0.5, and considered 20 sampleRates between 30Hz and 12 kHz. In total, we sampled an input-parameter space that included over 284 million unique combinations (6 signalLength x 20 sampleRate x 99 %Overlap x 136 nTimeBins x 16 windowType x 11 nfft).

2.3 Analysis Methods

In order to objectively choose input-parameters without considering all 284 million combinations, each of the spectrogram input-parameters was randomly sampled over the entire range of possible values. After randomly sampling the input-parameter space, we input those parameters into Matlab’s spectrogram function and input the resulting spectrogram into a support-vector-machine (SVM) classifier using a Gaussian or radial basis function (RBF) kernel following the precedent of Cvengros et al. (4). It should be noted that the spectrogram 2D matrix fed into the SVM, is first reshaped into 1D array, which is common practice.

The accuracy of each spectrogram and classifier was then assessed using 5-fold cross validation and an overall accuracy measure:

\[ \text{OverallAccuracy} = \frac{(TP + TN)}{(Total)} \]  

where TP is the number of true positives classifications, TN is the number of true negative classifications, and Total is the total number of signals used to test the classifier. In the case of multi-class classification, the total number of TP and TN can be found by summing across the diagonals of the confusion matrix. In terms of calculating the OverallAccuracy, we used a 5-fold cross-validation procedure where 80% of the data is used to train the classifier, and the remaining 20% is used as the test set to assess the accuracy of the classifier. This process is repeated for each fold, or 5 times (hence the 5-fold cross-validation), and the OverallAccuracy of the classifier is obtained from an overall confusion matrix that results from summing the confusion matrices output from each of the 5 iterations. See Sokolova and Lapalme (11), for more information on the different ways to quantify the performance of binary and multi-class classifiers.

We repeat the random sampling process and build an output table that contains the 6 variable input-parameters (windowType, %Overlap, nTimeBins, nfft, signalLength, sampleRate) and the resulting OverallAccuracy. From this output table we identify the best combination using the statistical software R (12). This is done by feeding the output table into a Decision Tree (8) to learn the parameter values that give the best classification results. Then we use a Random Forest model (9) to learn which features are the most important for predicting the OverallAccuracy.

3. RESULTS & DISCUSSION

3.1 Results

The analysis outlined in Section 2.3 resulted in a total of 1800 random samples, and took over 256 hours of computational time spread across multiple computers. In some cases, we had to make use of a high performance computer (HPC) given that some of the combinations used over 112GB of RAM. In total, we sampled 0.0006 % (1800 of 284 million) of the possible combinations (Figure 3), and found that the best-tested combination resulted in an overall accuracy of 39.5% (Table 2). In addition, we found that the worst combinations (not shown here) were as low as 14%. Together, this range of classification accuracy (i.e., 25%) confirms that the choice of spectrogram input-parameters can
drastically affect the resulting classifier performance.

![Figure 3 – Distribution of the input-parameter space examined in this paper.](image)

<table>
<thead>
<tr>
<th>sampleRate</th>
<th>signalLength</th>
<th>windowType</th>
<th>%Overlap</th>
<th>nTimeBins</th>
<th>nfft</th>
<th>Overall Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000</td>
<td>3</td>
<td>kaiser</td>
<td>51%</td>
<td>49</td>
<td>1024</td>
<td>39.5%</td>
</tr>
<tr>
<td>8000</td>
<td>3</td>
<td>kaiser</td>
<td>57%</td>
<td>41</td>
<td>512</td>
<td>39.2%</td>
</tr>
<tr>
<td>8000</td>
<td>2</td>
<td>blackmanharris</td>
<td>93%</td>
<td>55</td>
<td>1024</td>
<td>38.9%</td>
</tr>
<tr>
<td>1000</td>
<td>3</td>
<td>barthann</td>
<td>34%</td>
<td>3</td>
<td>512</td>
<td>38.3%</td>
</tr>
<tr>
<td>1000</td>
<td>2</td>
<td>taylor</td>
<td>77%</td>
<td>12</td>
<td>256</td>
<td>37.7%</td>
</tr>
</tbody>
</table>

Table 2 – Top 5 random sampling results.

We also used a Decision Tree to see which parameters, on average, gave the best results and a Random Forest tree analysis to see which parameters were the most important. From the Decision Tree (Figure 4), we found that having a sampling rate (sampleRate) greater than 350 Hz and nfft greater than 192, on average, resulted in an accuracy of 35%.

The variable importance for each of the input-parameters was found by using a Random Forest Decision Tree method which gives the total decrease of node impurity by splitting on the variable and averaging over all of the random trees (13). The node impurity was determined using the Gini Impurity Index, which is a measure of misclassification (14). A higher decrease in node impurity indicates more variable importance. Based upon these definitions, we found that the number of
points in the FFT (nfft) and the sampling rate (sampleRate) are the two most important variables (Table 3).

Table 3 – Variable Importance as determined by Random Forest Decision Tree.

<table>
<thead>
<tr>
<th>Input-parameter</th>
<th>Feature Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>nfft</td>
<td>1.83</td>
</tr>
<tr>
<td>sampleRate</td>
<td>1.63</td>
</tr>
<tr>
<td>%Overlap</td>
<td>0.48</td>
</tr>
<tr>
<td>nTimeBins</td>
<td>0.33</td>
</tr>
<tr>
<td>windowType</td>
<td>0.27</td>
</tr>
<tr>
<td>signalLength</td>
<td>0.19</td>
</tr>
</tbody>
</table>

3.2 Discussion

While we were confident at the outset that the choice of spectrogram input-parameters would affect the accuracy of the resulting classifier, we didn’t know that the accuracy would vary by 25% (i.e., 14-39%). Perhaps the more useful result, however, is the identification of the most important input-parameters (i.e., sampleRate and nfft, Table 3). Intuitively this makes sense, as the Nyquist–Shannon Sampling Theorem tells us that a higher sampling rate is required to capture the higher frequency components of signals. In regard to the importance of the number of points in the FFT (nfft), this too makes sense given that a higher bin count or finer frequency resolution of the spectrogram is useful for capturing small frequency differences between the environmental noise-sources.

Identifying the input-parameters that are not as important is also useful and interesting. For example, we found that the percent-overlap-between-windows (%Overlap), and time resolution (i.e., nTimeBins) did not have as profound effect on the classification accuracy. In fact, though not shown in the figures or tables of this paper, only a single time bin is used in the spectrogram in some of the
For example, the 50th best combination had \(n_{\text{TimeBins}} = 1\) and an accuracy of 36%. This may suggest that using the spectrogram or STFT is unnecessary, and that the FFT could be a more efficient way to solve this problem. We also found that the signal length \((\text{signalLength})\) and window type \((\text{windowType})\) were the least important in comparison the other input-parameters. In regards to \(\text{signalLength}\), it is likely for this dataset that most of the useful information was adequately captured in the first half second of the signal.

We also found that the signal length \((\text{signalLength})\) and window type \((\text{windowType})\) were the least important in comparison the other input-parameters. In regards to \(\text{signalLength}\), it is likely for this dataset that most of the useful information was adequately captured in the first half second of the signal.

We also calculated the memory used and computation time for each input-parameter sample, and found that these two variables had a nearly linear relationship \((R^2 = 0.88)\). Using this information, we found that the input-parameters not only affect the classification accuracy but also dictate the complexity of the classification task. While the computation time and memory usage are directly proportional to size of the \(\text{nfft}\) and \(\text{sampleRate}\), we do find input-parameter combinations that have high overall accuracy and low memory usage (Figure 4). For example, we found some combinations that had an overall accuracy > 35%, with \(\text{sampleRate}\) as low as 500, \(\text{nfft}\) as low as 64, and \(\text{memoryUsage}\) as low as 2.3 GB. This result is encouraging given that in may not be practical for continuously running noise-monitoring systems to have access to the high performance computers needed to calculate spectrograms and SVM classifiers with high \(\text{sampleRate}\) and \(\text{nfft}\).

Figure 4: Overall accuracy as a function of memory usage in gigabytes (GB).

Despite the insights gained from this study, the spectrogram classifiers built in this analysis performed rather poorly. For example, the best spectrogram classifier had an overall accuracy of (40%) whereas the Mel-Frequency Cepstral Coefficients (MFCC) classifier built by Salamon et al. (5) had an overall accuracy of 70%, and a sound level meter (SLM) classifier that used the same metrics as Cvengros et al. (4), had an overall accuracy of 65%. These are pretty fair comparisons, as all analyses used the same environmental noise dataset and the same classification algorithm (i.e., a multi-class SVM with radial basis function kernel). One of the possible reasons that the spectrogram classifier may have performed so poorly could be due to the low sampling of the input-parameter sample space. For example, we only sampled 0.0006% (1800 of 240 million) of the possible input-parameter combinations; albeit, the analyses presented in this paper took over 256 hours to run.

There are many directions future work could and should take. Future work could consider including SLM and MFCC metrics in addition to the spectrogram-based features to improve the classifier accuracy. Additionally, reducing the total number of features may improve the accuracy by eliminating the least important features. Future work should also consider sampling more of input-parameter space, and should consider other sampling designs beyond random sampling. For example, response surface methods may be able to provide a way to more systematic way to explore the large sample space.
4. CONCLUSIONS

In summary, we found that the choice of spectrogram input-parameters can drastically affect the accuracy of a spectrogram-based classifier. In this case, the accuracies varied by 25% (14.5-39.5%). More importantly, we identified the two most important spectrogram input-parameters: the sampling rate (\texttt{sampleRate}) and the number of points in the FFT (\texttt{nfft}), and found that having a \texttt{sampleRate} > 350 Hz and \texttt{nfft} > 192 gave the best accuracies for the environmental dataset considered in this effort. The performance of the spectrogram-based classifiers built in this paper, however, did not perform as well as other environmental metric based approaches (e.g., the MFCC-based and SLM-based classifiers). We’ve speculated on reasons why this may be (e.g., under-sampling or input-parameter space or inferior sampling method), and have presented several different directions that future research can or should take.

This paper has not solved the problem of finding a robust classification algorithm that can automatically classify all environmental noise-sources; yet, some progress has been made. This paper provided a methodology for objectively choosing and analyzing spectrogram input-parameters using random sampling. Some practical insights were gained on relationship between memory usage, computation time, and classifier accuracy. In addition, methods for identifying feature importance were presented (i.e., the visualization and analysis of feature importance through Decision Trees and Random Forests). These methods can be applied to future work to help identify the features that should be included in the classification algorithm that can automatically classify all environmental noise-sources.

ACKNOWLEDGEMENTS

We would like to thank Drew Hulva for writing and compiling the sound level meter metric toolbox that we used in this paper, and would like to thank J. Salamon, C. Jacoby and J. P. Bello from New York University for compiling the dataset and making it freely available for non-commercial use. We would also like to thank FreeSound.org and its users for recording and hosting of wide variety of sound samples. For a complete attribution list of the dataset used in this study please visit \url{https://serv.cusp.nyu.edu/projects/urbansounddataset/index.html}. The US Army ERDC Environmental Quality/Installations business area supported this research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors, do not necessarily reflect the views of the funding agency, and are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

REFERENCES

10. Fabra MTGaUP. Freesound 2016 (\url{http://www.freesound.org/}).