1 Feature Selections

Even best minds in the world will perform poorly on test which they have studied the wrong material- neural networks are no different. In order to properly train a neural network, the model must be presented with an appropriate set of training input x and a complementary set of training labels y [1, 3, 4]. It becomes quickly apparent that the nature of information contained in the input object x, called *features*, is *extremely important* to the performance of the classifier. Consider if you were tasked to identify cats and dogs from images, but instead were presented with only the top-most row of pixels- The task would be nearly impossible because of an inappropriate or in complete set of information.

Tuomas Virtanen, machine learning and audio engineer writes in his book, "Computational Analysis of Sound Scene and Events" [16]:

> For recognition algorithms, the necessary property of the acoustic features is low variability among features extracted from examples assigned to the same class, and at the same time high variability allowing distinction between features extracted from examples assigned to different classes.

In constructing a neural network classifier, the development of appropriate features is of the utmost importance. To ensure the construction of a suitable model, we derive features based from three sources (i) a spectrogram matrix of the waveform, (ii) the time-space representation of the waveform, and (iii) the frequency space representation of the waveform. It is important to note that although this algorithm will classify sound waves to instruments, the model will never actually be presented with a waveform directly, instead it will rely on these features.

Once we produce an sufficient set of features, we concatenate them into a single object, $\hat{x}$, called the *feature-vector* [3]. In the training process, this object, along with the appropriate classification label, y is presented to the neural network for processing. This process of constructing a feature vector from any data set is vital and is used to represent the data set in a far more compact and non-redundant format [16, 7]

To ensure suitable performance of this sound wave classification neural network, a great deal of time has been devoted to the construction of the elements of the feature vector. These features are derived are principles of music, digital signal processing, previous work success, and most importantly physics. In the following sections, we outline the set of 24 features used in the classification process

1.0.1 Audio Preprocessing

Preprocessing a data set is a necessary step to execute prior to feature extraction [2, 4]. In the case of audio files, preprocessing usually consists of ensuring that the data set contains the following:

1. A suitably sized number of files, of reasonable audio quality, with normalized ampltudes
2. Audio encoded in a standard, and consistent format
3. A consistent sample rate between audio files
4. A consistent number of channels

Note that different projects may require a different set of requirement from preprocessing[16]. For this project, we have chosen to use the following parameters:

1. Roughtly 4000 audio files Professionally or semi-professionally recorded in a studio. Citation needed!. All ampltiudes have been normalized to ± 1 unit.
2. All audio has be converted into *.WAV* files from other formats, such as *.AIF* or *.MP3* using a MATLAB program
3. All audio is sampled at $44,100$ Hz
4. All audio has been down-mixed into mono-channel waveforms.

1.1 Spectrogram Feature

The field of neural classification is well studied in the application of image-processing. Many large-scale, and introductory neural network projects find themselves under the umbrella of image classification [1, 3, 8, 10]. As a result, model architectures for image related tasks are well-explored and have experimentally shown successful behavior. Following in those footsteps, it make senses to provide an image-like representation of a sound wave as a feature. We do this in the form of a spectrogram.

A spectrogram is a representation of the energy distribution of a sound wave as a function of both space and time. In a conventional spectrogram, the passing of time is shown along the x-axis, and the frequency spectrum is shown on the y-axis. Thus each point in the 2-Dimensional space is an energy at a given time and frequency. Examples spectrograms from the wave form data set are shown in Fig. (1.1).

Insert spectrograms here

Figure 1: Spectrogram representations of various waveforms

A spectrogram is produced by the method of *frame-blocking*, which is very prevalent in audio signal classification. Frame-blocking creates a set of analysis frames, $a^{(i)}$ each of which is N samples in length, and has a fixed overlap with the next adjacent frame. Each of the k frames then allows for a section of the signal in somewhat stationary state [7, **?**, 5, 16]. For this project, we have chosen to use frames of size $N = 4096$ with a 75% or 3072 sample overlap. At a sample rate of $f_s = 44100$ samples/second, each frame represents a slice of time that is about 0.1 seconds long.

We concatenate each analysis frame, $a^{(i)}, i \in [0, k-1]$ into a single $k \times N$ matrix, called A. Each row is a frame, each column is an index in that frame

$$A = \left\{a^{(0)}, a^{(1)}, a^{(2)}, ..., a^{(k-1)}\right\} = \begin{bmatrix} a^{(0)}[0] & a^{(0)}[1] & a^{(0)}[2] & \dots & a^{(0)}[N-1] \\ a^{(1)}[0] & a^{(1)}[1] & a^{(1)}[2] & \dots & a^{(1)}[N-1] \\ a^{(2)}[0] & a^{(2)}[1] & a^{(2)}[2] & \dots & a^{(2)}[N-1] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a^{(k-1)}[0] & a^{(k-1)}[1] & a^{(k-1)}[2] & \dots & a^{(k-1)}[N-1] \end{bmatrix} \tag{1}$$

We use bracket notation, $[j]$ to indicate that each object is array-like. Many programming languages could also use the following indexing conventions for matrix A:

$$A_{i,j} = a^{(i)}[j] = A[i][j] = A[i, j] \tag{2}$$

Note that these all represent equivalent entries.

After frame-blocking, we apply a *windowing function* to each frame. A standard *Hann Window* of N samples is generated and reshaped into a $N \times 1$ column-array,H. The n-th index in a Hann window with N samples is defined:

$$H[n] = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi n}{N-1}\right)\right] \tag{3}$$

The window function is applied to each analysis frame by computing the dot-product of the Hann window array, H and each row of the analysis frames matrix, $A_{i,:}$. Result is an $k \times N$ array, $\widetilde{A}$:

$$\widetilde{A} = A \cdot H^T \tag{4}$$

Finally, we perform a *Discrete Fourier Transform* (DFT) to bring the signal from a time domain into a frequency domain. [12, 13]. The Discrete Fourier Transform is applied by producing an $N \times N$ *transform matrix*, often noted at $\mathbb{W}$. Let $\omega^k = e^{\frac{-2\pi i}{N}k}$, then the DFT matrix for a time-space containing N samples

$$\mathbb{W} = \frac{1}{\sqrt{N}}\begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \dots & \omega^{(N-1)^2} \end{bmatrix} \tag{5}$$

Each column of the matrix is a complex sinusoidal oscillating with an integer number of periods within the N-sample length window [15, 13].The DFT is applied by a taking the matrix -product of $\mathbb{W}$ and $\widetilde{A}^T$. The transpose of $\widetilde{A}$ then makes each analysis frame into a column vector, which gives the appropriate dimension for multiplication.

Most standard implementations of neural network require all activations to be real floating-point numbers. Since the the DFT matrix introduce complex values, we compute the square of the element-wise $L2$-norm of the resultant array.

$$S_{xx} = \left\|\mathbb{W}\widetilde{A}^T\right\|_2^2 \tag{6}$$

Where $\mathbb{W}$ is the DFT matrix from eqn. (5) and $\widetilde{A}^T$ is the transpose of the analysis frames matrix from eqn. (4). The matrix S_{xx}, is the spectrogram representation of the initial waveform, and now contains $N \times k$ real floating-point numbers.

Each Column of the S_{xx} matrix is now a single-frame that has been moved into a frequency-space representation, thus there are k columns, just as there were k analysis frames. Given the discretized nature of digital audio, the fequency-sapce representation is not a continuous function, but rather a column vector, where the frequency has been assigned to one of N bins. Since human hearing extends from about 20 Hz to around 20 kHz, audio files, such as music and voice recordings are typically sampled around 44.1 kHz, to ensure that the full range of audio is held by the recording [12, 16].

However, standard musical instruments seldom extend above 8 kHz [12, 16, 17]. This means that when constructing the spectrogram, we will rarely ever see energy present above this frequency at any time, and the S_{xx} matrix will contain mostly zero, or zero-like entries. As a result of this, we can simply crop each matrix, to display frequencies that are only between 0 Hz and 8000 Hz. This makes the input array smaller, and eliminates redundant and non-useful information. This brings the number rows in the S_{xx} matrix from N down to N'. lastly, each raw audio file may contain a different number of samples to begin with, thus a different number of analysis frames are created. To ensure all sample are a homogeneous size, we assert that each matrix must have exactly k' columns. If $k' < k$, columns of zeros are added to pad the input, and if $k' > k$, columns are removed.

Each spectrogram is now $N' \times k' \times 1$ (1 representing a single gray-scale channel, as oppose to 3 for RGB inputs) and effectively encodes the energy distribution of the waveform a a function of both time and frequency. The spectrogram is the first feature used in this model. For this classifier, we have chosen $N' = 560$ and $k' = 256$. For training, a mini-batch of b samples are concatenated into a single array object. See Neural Network section for more details. For a batch of b samples of $N' \times k' \times 1$ spectrograms, we shape X_1 such that:

$$X_1 = \left\{ S_{xx}^{(0)}, S_{xx}^{(1)}, S_{xx}^{(2)}, ..., S_{xx}^{(b-1)} \right\} \in \mathbb{R}^{(b \times N' \times k' \times 1)} \tag{7}$$

This matrix is presented to the *Convolution* branch of the neural network from processing. See Neural Network section for more details

1.2 Time-Space Features

The features described in this section are derived from time-domain representations of each audio sample. In some cases, this can be from the waveform directly, or my manipulation of rows and columns in the analysis frames matrix A from eqn. (1). For each feature, we detail the physical significance and provide a visualization in feature-space.

1.2.1 Time Domain Envelope

The time domain envelope (TDE) is a method of determining which parts of the sound wave contains most of the energy of the signal. Generally, we start by computing the RMS-Energy of each frame. The larger the RMS energy of the frame, the larger the amplitude of the waveform in that frame. For frames with smaller amplitude, we conclude that the signal has either not begun, or is decaying. The RMS-Energy of a single analysis-frame $a^{(i)}$ is given by [12, 16]

$$RMS\big[a^{(i)}\big] = \sqrt{\frac{1}{N}\sum_{j=0}^{N-1} a^{(i)}[j]} \tag{8}$$

We adapt this feature to **compute the RMS-Energy of the full waveform**. By doing this, we acquire a rough estimate for the energy of a waveform in the entirety of the audio file [7]. For instruments with long sustain or release times, such as strings or undampened mallet percussion, hold a comparably large waveform RMS when compared to those instruments without sustain such as brass or percussion.

PLOT OF TDE FEATURES HERE

Figure 2: Time domain envelope visualized in feature-space

1.2.2 Zero Crossing Rate

The zero crossing rate (ZXR) of a signal or frame is use to measure how many times that a signal crosses it's equilibrium point. This can be done per total sound wave, or per unit time, such an per analysis-frame. In differentiating speech and music, it is commonly to use the difference between the frame with the highest ZXR and the lowest ZXR. In some cases, the ZXR can be correlated to frequency as well [5, 18]. The ZXR for a frame $a^{(i)}$ is given by [16, 7]

$$ZXR\big[a^{(i)}\big] = \frac{1}{2}\sum_{j=1}^{N-1} \Big|\operatorname{sign}(a^{(i)}[j]) - \operatorname{sign}(a^{(i)}[j-1])\Big| \tag{9}$$

Where $\operatorname{sign}(x)$ returns $+1$ if $x > 0$, -1 if $x < 0$ and 0 if $x = 0$.

We adapt this feature to **compute the zero crossing rate for the full waveform**. This provides a rough estimate for the average frequency in the full waveform, and can help discern clean periodic signals (low ZXR) from those that may have more volatile behavior (high ZXR)[16].

PLOT OF ZXR FEATURES HERE

Figure 3: Zero crossing rate visualized in feature-space

1.2.3 Center of Mass

The temporal center of mass (TCM) of a signal is used to compute where in time the amplitude sort of *bunches up*. We treat the full waveform array, a, with M samples as a 1-Dimensional discrete mass distribution. The TCM of that waveform is then given:

$$TCM\big[a\big] = \frac{\sum_{j=0}^{M-1} ja[j]}{\sum_{j=0}^{M-1} a[j]} \tag{10}$$

This includes negative amplitude values acting as "negative masses".

The TCM turns out to be a very unique feature, and very powerful in classification due to it's variance. For instruments with heavier attacks, we expect the TCM to be a very low value as most of the energy and amplitude is concentrated close to the start of the audio file. For instruments with long sustain times, we expect a very centrally located center of mass, and for instruments with long release times, we expect a very late TCM value.

PLOT OF TCM FEATURES HERE

Figure 4: Temporal center-of-mass visualized in feature-space

1.2.4 Statistical Distribution Data

1.2.5 Auto Correlation Coefficients

Auto correlation coefficients (ACC) are rough estimates of the signal spectral distribution [16]. We can compute any number of ACC's and their value changed depending on the index chosen. It is common to only compute the first k ACC's. For a full waveform signal a, with M samples, the k-th ACC (indexed from 1 to K) is given by:

$$ACC_k\big[a\big] = \frac{\sum_{j=0}^{M-k-1} a[j]a[j+k]}{\sqrt{\sum_{j=0}^{M-k-1} a^2[j]}\sqrt{\sum_{j=0}^{M-k-1} a^2[j+k]}} \tag{11}$$

1.3 Frequency-Space Features

The features described in this section are derived from the frequency-domain representations of each audio sample. Typically, these features are exacted from the spectrogram representation,$S_x x$, outlined in sec. (1.1) and computed in eqn.(6). For each feature, we detail the physical significance and provide a visualization in feature-space.

1.3.1 Mel Frequency Ceptral Coefficients

1.3.2 Center of Mass

The frequency center-of-mass (f_{CoM}) for an audio file provides a strong representation of how overtones and energy is distributed in the signal. For example, a low instrument such as a tuba or bass may have a very low f_{CoM} while flutes and violins may have a very high f_{CoM}.

References

[1] Geron, Aurelien. *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly, 2017.

[2] Geron, Aurelien. *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd ed., O'Reilly, 2019.

[3] Goodfellow, Ian, et al.*Deep Learning*. MIT Press, 2017.

[4] James, Gareth, et al. *An Introduction to Statistical Learning with Applications in R*. Springer, 2017.

[5] Khan, M. Kashif Saeed, and Wasfi G. Al-Khatib. "Machine-Learning Based Classification of Speech and Music." Multimedia Systems, vol. 12, no. 1, 2006, pp. 55–67., doi:10.1007/s00530-006-0034-0.

[6] Levine, Daniel S. *Introduction to Neural and Cognitive Modeling*. 3rd ed., Routledge, 2019.

[7] Liu, Zhu, et al. "Audio Feature Extraction and Analysis for Scene Segmentation and Classification." Journal of VLSI Signal Processing, vol. 20, 1998, pp. 61–79.

[8] Loy, James , *Neural Network Projects with Python*. Packt Publishing, 2019

[9] McCulloch, Warren S., and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity." *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, 1943, pp. 115–133.

[10] Mierswa, Ingo, and Katharina Morik. "Automatic Feature Extraction for Classifying Audio Data." *Machine Learning*, vol. 58, no. 2-3, 2005, pp. 127–149., doi:10.1007/s10994-005-5824-7.

[11] Mitchell, Tom Michael. Machine Learning. 1st ed., McGraw-Hill, 1997.

[12] Olson, Harry E. *Music, Physics and Engineering*. 2nd ed., Dover Publications, 1967.

[13] Peatross, Justin, and Michael Ware. *Physics of Light and Optics*. Brigham Young University, Department of Physics, 2015.

[14] Petrik, Marek. "Introduction to Deep Learning." Machine Learning. 20 April. 2020, Durham, New Hampshire.

[15] Short, K. and Garcia R.A. 2006. "Signal Analysis Using the Complex Spectral Phase Evolution (CSPE) Method." AES: *Audio Engineering Society Convention Paper*.

[16] Virtanen, Tuomas, et al. *Computational Analysis of Sound Scenes and Events*. Springer, 2018.

[17] White, Harvey Elliott, and Donald H. White. *Physics and Music: the Science of Musical Sound*. Dover Publications, Inc., 2019.

[18] Zhang, Tong, and C.-C. Jay Kuo. "Content-Based Classification and Retrieval of Audio." *Advanced Signal Processing Algorithms, Architectures, and Implementations VIII*, 2 Oct. 1998, pp. 432–443., doi:10.1117/12.325703.