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 21 features used in the classification process

1.0.1 The Design Matrix

The 21 features are from each file are organized into one of two array structures. each called a *design matrix*. A design matrix is an N-Dimensional array of real floating-point numbers that stores the features in a convenient way for the neural network to digest. For this hybrid neural-network, we use two matrices, X_1 and X_2, each containing data from b audio file samples. each has shape given by:

$$X_1 \in \mathbb{R}^{(b \times N' \times k' \times 1)} \tag{1}$$

$$X_2 \in \mathbb{R}^{(b \times p)} \tag{2}$$

Where $N' \times k' \times 1$ gives the shape of a single spectrogram matrix S, described in sect. (1.1), The first axis, with length b indicates that there are b samples stored in that design matrix. Similarly, p is the number of features derived from the time and frequency-representation of features outlined in sec. (1.2) and sec.(1.3). A row of X_2 has shape $1 \times p$ where there are p features within the *feature-vector* for each sample. Matrix X_1 is presented to the convolutional branch of the network to be process as an a collection of images, and matrix X_2 is presented to the perceptron branch to be processed as a collection of vectors.

1.0.2 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 amplitudes
2. Audio encoded in a standard, and consistent format
3. A consistent sample rate and bit depth 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. Roughly 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 and Bit depth needed - 16? 24?
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 even introductory neural network projects find themselves under the umbrella of image classification [1, 3, 8, 10]. As a result, model architectures for image processing related tasks are well-explored and have shown experimentally successful behavior. Following this success, 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 matrix.

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 takes a raw waveform or signal, s and decomposes it into a set of analysis frames, a_i, with each being 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, 18, 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

$$A = \{a_0, a_1, a_2, ..., a_{k-1}\} = \begin{bmatrix} a_0[0] & a_0[1] & a_0[2] & \dots & a_0[N-1] \\ a_1[0] & a_1[1] & a_1[2] & \dots & a_1[N-1] \\ a_2[0] & a_2[1] & a_2[2] & \dots & a_2[N-1] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{k-1}[0] & a_{k-1}[1] & a_{k-1}[2] & \dots & a_{k-1}[N-1] \end{bmatrix} \tag{3}$$

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

$$A_{i,j} = A_i[j] = A[i][j] = A[i, j] \tag{4}$$

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:

$$H[n] = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi n}{N-1}\right)\right] \tag{5}$$

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

$$\widetilde{A_i} = A_i \odot H \tag{6}$$

Finally, we perform a *Discrete Fourier Transform* (DFT) to bring each analysis frame 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

$$\mathbb{W} = \frac{1}{\sqrt{N}}\begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \dots & \omega^{(N-1)^2} \end{bmatrix} \tag{7}$$

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 models require all activations, weights, and biases to be real floating-point numbers. Since the the DFT matrix introduces complex values, we compute the square of the element-wise $L2$-norm of the resultant array.

$$S_{i,j} = \left\|(\mathbb{W}\widetilde{A}^T)_{i,j}\right\|_2^2 \tag{8}$$

Where $\mathbb{W}$ is the DFT matrix from eqn. (7) and $\widetilde{A}^T$ is the transpose of the analysis frames matrix from eqn. (6). The matrix S, is the spectrogram representation of the initial waveform, and has shape $N \times k$ of real floating-point numbers. We can index matrix S similarly to that of matrix A in eqn. (4):

$$S_{i,j} = S[i][j] = S[i,j] \tag{9}$$

Each column of the S 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 time-series analysis frames. Given the discretized nature of digital audio, the frequency-space 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 20 Hz to 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 matrix will contain mostly zero, or zero-like entries. As a result of this, we can simply crop each matrix to only represent frequencies that are only between 0 Hz and 8000 Hz. This makes the input array smaller, and eliminates redundant and non-useful information. The number rows in the S matrix is reduced from N down to N'. Since each raw audio file may contain a different number of samples to begin with, a different number of analysis frames are created. To ensure all samples 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 pixel channel, as opposed to 3 channels 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 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:

$$X_1 = \left\{ S^{(0)}, S^{(1)}, S^{(2)}, ..., S^{(b-1)} \right\} \in \mathbb{R}^{(b \times N' \times k' \times 1)} \tag{10}$$

Which is consistent with the shape of the X_1 matrix outlined in eqn(1). This matrix is presented to the *Convolution* branch of the neural network for 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. Time space can be represented in one of three ways:

- The raw waveform s
  Indexed by $s[i]$ with $i \in [0, 1, 2, 3, ..., M-2, M-1]$
- A single analysis frame a_i
  Indexed by $a_i[j]$ with $i \in [0, 1, 2, 3, ..., k-2, k-1]$ and $j \in [0, 1, 2, 3, ..., N-2, N-1]$
- The full analysis frame matrix A
  Indexed by $A[i][j]$ with $i \in [0, 1, 2, 3, ..., k-2, k-1]$ and $j \in [0, 1, 2, 3, ..., N-2, N-1]$

From time space, we use the following 7 features:

- Time Domain Envelope
- Zero Crossing Rate
- Temporal Center of Mass
- Auto Correlation Coefficients ($\times 4$)

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 Often, we compute the RMS-Energy of each frame - a larger energy of the frame, the larger the average amplitude of the waveform in that frame. For frames with smaller amplitude, we can assume that the signal has either not begun, or is decaying.

We adapt this feature to **compute the RMS-Energy of the full waveform**. The RMS-Energy of the full waveform, s is given by [12, 16]

$$\mathrm{RMS}\big[s\big] = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} s[i]^2} \tag{11}$$

By using the full waveform, we acquire a rough estimate for the energy in the entirety of the audio file [7]. For instruments with long sustain or release times, such as strings or undampened mallet percussion, we expect a comparably large waveform RMS when compared to those instruments without sustain such as plucked stings or dampened 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, per unit time, or per analysis-frame. This feature is most commonly associated with differentiating speech from music, because speech presents a more jagged and often less periodic waveform. In some cases, the ZXR can be correlated to frequency as well [5, 18].

We adapt this feature to **compute the zero crossing rate for the full waveform**. The ZXR for the full waveform s is given by [16, 7]

$$\mathrm{ZXR}\big[s\big] = \frac{1}{2}\sum_{i=1}^{M-1}\Big|\operatorname{sign}(s[i]) - \operatorname{sign}(s[i-1])\Big| \tag{12}$$

Where $\operatorname{sign}(x)$ returns $+1$ if $x > 0$, -1 if $x < 0$ and 0 if $x = 0$. 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 noise or 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 roughly where in time the amplitude of the waveform *bunches up*. We treat the full waveform array, s, with M samples as a 1-Dimensional discrete mass distribution. The TCM of that waveform is then given:

$$\mathrm{TCM}\big[s\big] = \frac{\sum_{i=0}^{M-1} i s[i]}{\sum_{i=0}^{M-1} s[i]} \tag{13}$$

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 near the start of the waveform. 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 later TCM value.

PLOT OF TCM FEATURES HERE

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

1.2.4 Auto Correlation Coefficients

Auto correlation coefficients (ACC) are rough estimates of the signal spectral distribution . 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 [16]. For a full waveform signal s, with M samples, the k-th ACC (indexed from 1 to K) is given by:

$$\mathrm{ACC}_k\big[s\big] = \frac{\sum_{i=0}^{M-k-1} s[i]s[i+k]}{\sqrt{\sum_{i=0}^{M-k-1} s^2[i]}\sqrt{\sum_{i=0}^{M-k-1} s^2[i+k]}} \tag{14}$$

PLOT OF ACC FEATURES HERE

Figure 5: First four auto correlation coefficients visualized in feature-space

1.3 Frequency-Space Features

The features described in this section are derived from the frequency-domain representations of each audio sample. Frequency space is represented by the transposed spectrogram matrix, S^T, described in sec. (1.1). This is done to ensure that each row is now an analysis frame, each column is a frequency bin. This gives a similar structure to the time-space analysis frames matrix, A, in eqn. (3). For each feature, we detail the physical significance and provide a visualization in feature-space.

From frequency space, we use the following 13 features:

- Mel Frequency Cepstral Coefficients ($\times 12$)
- Frequency Center of Mass

1.3.1 Mel Frequency Cepstral Coefficients

Mel Frequency Ceptstral Coefficients (MFCC's) are a very prolific feature used widly in audio classification tasks [7, 16, 18]. MFCC's are produced by converting a linear-frequency axis with units such as Hertz into units of Mels, which are design to account for the non-linear perception of frequency in humans. The Hertz to Mel and Mel to Hertz transforms are given [16]:

$$M_f[h] = 2595 \log_{10}\left(1 + \frac{h}{700}\right) \tag{15}$$

$$H_f[m] = 700\left(10^{\left(\frac{m}{2595}\right)} - 1\right) \tag{16}$$

Where M_f is the frequency in units of Mels, given $[h]$, a frequency in Hertz, and H_f is the frequency in Hertz given $[m]$ a frequency in Mels.

Mel filters are produced by grouping frequency-space into R overlapping bins called *Mel Filter-banks*. In units of Mels, each filter bank, R_i remains a constant width, but when shown in Hertz, the bins increase logarithmically.

Figure of a few Mel Filter Banks goes here

Figure 6: Mel Filter Banks shown in frequency space with units of Hertz

Each of the R filters is created to be N' samples long, and transformed back into units of Hertz. When applied to an analysis frame in the frequency spectrum, the dot-product between the filter and the spectrum gives an approximation of the energy in that filter bank. Thus, each filter is concatenated into a matrix M of shape $R \times N'$, where each row is a filter. We apply the Mel Filter banks to the spectrogram to create matrix B:

$$B = S^T M^T \tag{17}$$

Matrix B has shape $k' \times R$.

Using S^T ensure that each row of the matrix is an analysis frame in frequency space. Similarly, each column of the matrix M is a appropriately scaled Mel filter-bank. This way, the matrix product in eqn. (17) allows that $B_{i,j}$ is the dot product between the i-th analysis frame and the j-th filter-bank. Finally, we compute the average energy across all k' frames. The resulting R floating-point numbers are approximation of the signal's energy in those particular frequency bands. For this project, we have chosen to use $R = 12$ filter-banks, which are all used as features.

PLOT OF MFCC FEATURES HERE

Figure 7: Mel Frequency Ceptral Coefficients in feature-space

1.3.2 Center of Mass

The frequency center-of-mass (FCM) for an audio file provides a strong representation of how overtones and energy is distributed in the signal's frequency domain. As with the temporal center-of-mass, we treat each row of the S^T matrix as it's own 1-Dimensional mass distribution, and compute the center of mass of that row. This encodes the FCM for a single analysis frame (in frequency-space) in the waveform. For an frequency-analysis frame, $s^{(i)}$, the FCM is given by:

$$\mathrm{TCM}_i\left[s^{(i)}\right] = \frac{\sum_{j=0}^{N'-1} j s^{(i)}[j]}{\sum_{j=0}^{N'-1} s^{(i)}[j]} \tag{18}$$

We compute the FCM for each of the k' frames, and then average the results. We use the average FCM across k' frames to compute the FCM feature:

$$TCM = \sum_{i=0}^{k'} \mathrm{TCM}_i\left[s^{(i)}\right] \tag{19}$$

The average FCM gives a strong approximation of the instrument or signal source's range. For example, a flute or violin will have a considerably high FCM value, even in their lower registers. Similarly, basses or tubas will have considerably low FCM values. Given that the standard frequency range of some musical instruments is fixed, it is guaranteed that for any particular instrument, the FCM will consistently remain within certain bounds [12, 17].

PLOT OF FCM FEATURES HERE

Figure 8: Frequency center-of-mass visualized in feature-space

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.