An analysis of dimensionality reduction and music information retrieval techniques for the visual representation of large audio datasets.

Łukasz Piotrak

A thesis presented for the degree of
Bachelor of Engineering

supervisor: Hung Son Nguyen
Polish-Japanese Academy of Information Technology
Department of Computer Science
Poland, February 2021

Abstract

Navigation and search has long been a bottleneck activity when working with sound. Recent advancements in dimensionality reduction techniques have made it possible to replace the classic directory tree with a spacial map of audio files, resulting in a more intuitive representation. I provide a method of evaluating visualizations using methods from spatial statistics and find that a combination of pca + stft + umap methods gives the best scoring, although unexpected 2d embedding on a dataset of instrument recordings.

Przeszukiwanie zbiorów plików od dawna sprawia trudność w efektywnej pracy z dźwiękiem. Wraz z postępem w dziedzinie redukcji wymiarowości danych, powstała możliwość zastąpienia klasycznej przeglądarki plików na bardziej intuicyjną reprezentację, tj. przestzenną mapę plików audio. W tej pracy przedstawiam metodę na ocenę wygenerowanych przestrzennych reprezentacji poprzez zastosowanie metod ze statystyki przestrzennej. Przeprowadzam także analizę metod ekstrakcji cech oraz redukcji wymiarowości i odkrywam, że połączenie metod pca + stft + umap daję najlepszą reprezentację, choć niespodziewaną reprezentację dla zbioru danych składających się z krótkich nagrań gry na instrumentach.

Contents

1 Introduction

The current paradigm for the storage and organization of audio files is that of the classic directory tree. This results in an attribute ontology; files are grouped together into classes (usually by directory name) i.e. “Drums”, “Vocals” and assigned a range of attributes by means of file name or tags. This approach is limited, e.g. the file might be labeled incorrectly or the labels might not addequately describe the sound. Moreover, many properties inherrent to the files are hard to represent e.g. two audio samples might be in separate branches of the taxonomy but be perceptually similar. However, as shown by projects such as The Infinite Drum Machine by Google Creative Lab [6], collections of sounds can be explored more naturally with the help of dimensionality reduction techniques.

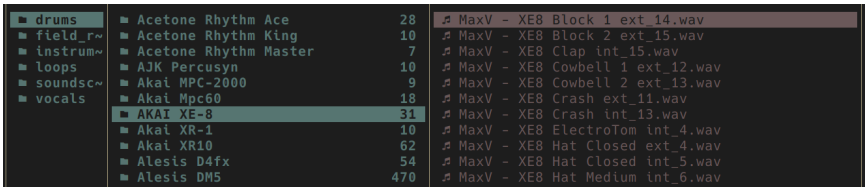

Figure 1: Browsing sound samples using a directory structure.

Advances in this area have enabled the intuitive representation of high-dimensional data. The dimension count of a Dataset can be reduced arbitrarily while still preserving information about its intrinsic properties. Commonly, this approach is used to plot Datapoints as clouds in 2d or 3d space, allowing for a depiction of the data which can be naturally grasped by the human mind.

As with many kinds of information, audio data in its raw form is unfit for such processing. An intermediate representation must to be constructed if any insights are to be gleaned from the data. In order to obtain a representations of audio signals useful to a human observer, a selection of features have to be extracted, which might correspond to certain aspects of the human perception of sound. These can then be used as inputs to produce a visualization by means of dimensionality reduction.

1.1 Related Work

There have been a number of papers and software projects with a focus on dimensionality reduction as applied to audio datasets. This work is based on previous efforts in this domain, such as those by Hantrakul & Sarwate [12] and McDonald et. al [6] which I will describe in more detail below. I expand upon their findings by using a dataset of samples containing harmonic data, widening the pool of feature extraction methods and introducing a novel method of assessing the visualizations using Ripley's H function to measure the homogeneity of density distribution of the plots.

1.1.1 The infinite drum machine

The idea for this work originally came from the excellent web app by McDonald et. al [6]. The app organized thousands of foley sounds on a 2d plane using the t-SNE technique introduced by van Der Maaten & Hinton [14]. Similar sounding samples were placed close together. Sounds could be navigated and played back by hovering over individual points with the mouse. A sequencer was implemented to enable the arrangement of those sounds into audio loops. As far as my research showed this was one of the first applications introducing the excellent idea of using dimensionality reduction techniques to make intuitive "maps" of large audio collections. Since the app was meant more as a proof of concept I immediately saw areas in which it might be extended. Firstly, it only worked on a static dataset, meaning there was no way to make visualizations for other datasets or even add new samples to the existing visualization. Secondly, the functionality was extremely limited by the intended use case. A user is able to select several sounds and use them as samples to program a musical loop and not much besides.

1.1.2 Klustr

Similarly to this work, Klustr, by Hantrakul & Sarwate [12] is a review of feature extraction methods and dimensionality reduction algorithms as applied to a large audio dataset. They used a dataset of 10,000 drum samples, shortened to length 0.25 seconds. The feature set used was: Short-term-Fourier-Transform, Mel-Frequency Cepstral Coefficients, various MIR features (RMSE, spectral centroid, spectral crest, spectral flux, spectral rolloff, zero-crossing rate) and finally features extracted by a Wavenet autoencoder. These features were then applied to a dimensionality reduction step using 3 different dimenionality reduction techniques: PCA, t-Sne and UMAP. The

scoring function is a combination of the silhouette coefficient [1], roundness of plots measured by the Polsby-Popper test [10] and the ration of overlap of convex hulls of the different drum classes. I have used a similar scoring system, though changing the calculation of the ratio of overlap of the convex hulls and adding another scoring method. Their results showed that, for the dataset they used, a combination of STFT, PCA and UMAP yielded the best 2D visualization.

2 Background

2.1 Music information retreival

The objective of Music information retrieval (MIR) is the extraction of information from music. It is a broad field which lies on the intersection of many different research domains. It uses knowledge from musicology and music theory, (music) psychology, psycho-acoustics, audio engineering, computer science and machine learning. TODO With the recent rise of music streaming platforms, which serve millions of people each day, extracting information from music to classify, categorize and ultimately build effective recommender systems seems more relevant than ever. It can be further subdivided into two distinct subdomains. The first includes the analysis of non-audio music formats such as musical notation, song lyrics and even user reviews and bibliographical information (publication date, title etc). The second, named Audio Content Analysis (ACA), uses various techniques from digital signal processing, machine learning, statistics and psychoacoustics, among others, to extract meaningful information from audio signals.

2.1.1 Short-term Fourier Transform (stft)

The Short-term Fourier Transform is a representation of a signal obtained by taking the Fourier transform of short segments of time. The method used to obtain the stft is relatively straightforward. First, the signal is divided into shorter, equal-length segments using a window function. These segments usually have some overalp in order to prevent artifacts of the original signal (so-called spectral leakage) Multiple window functions could be used for this step, however a Hann window is usually used. Usually a window length of 10 to 300ms is chosen. Smith [13] points out three reasons for doing so:

- The human ear analyzes short fragments of signals at a time (10-20ms).

- Signals change over time. It is best to analyze over a time frame where the spectral content stays relatively constant.
- Computation of the fourier transform is costly and computing a whole signal at once may inefficient and undesirable.

Next, the Discrete Fourier Transform (DFT) is computed for each segment. We can thus observe how the frequency content of the signal changes as we progress through time.

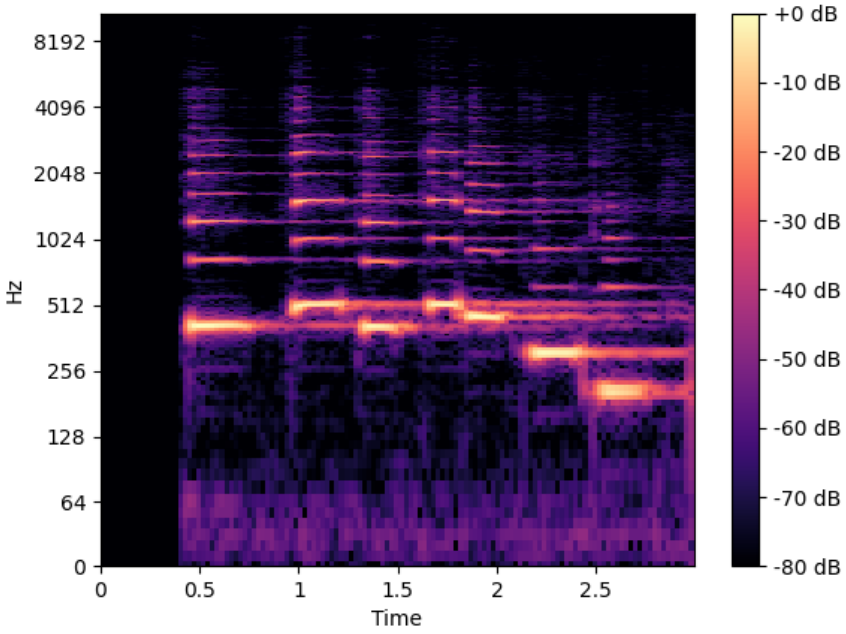

Figure 2: An stft spectrogram

2.1.2 MFCC

The motivating idea of mel frequency cepstral coefficients is to compress information about the vocal tract (smoothed spectrum) into a small number of coefficients based on an understanding of the cochlea. An example mfcc spectrogram can be seen in 7 There is no one standard way of calculating the coefficients, however the main steps may be summarized as follows:

1. Take a Fourier tranform of a short, windowed fragment of the signal (as in the stft)
2. Apply the mel filterbank (shown in 3) to convert the frequencies to the mel scale.
3. At each of the mel frequencies, take the logarithm of the power. $log(sum(x^2))$
4. Take the discrete cosine transform of the resulting values (express the value of x as a sum of cosine functions of different frequencies).
5. The amplitudes of the resulting spectrum are the Coefficients.

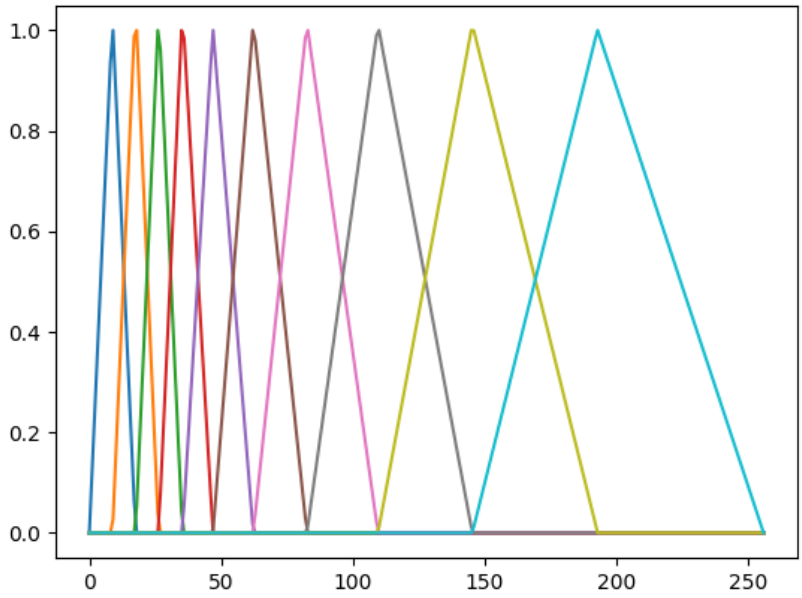

Figure 3: The mel filterbank

2.1.3 MIR metrics

- **spectral centroid**

The mean of the normalized distribution over frequency per frame of a spectrogram.

$$centroid(t) = \frac{\sum_{k=b_1}^{b_2} f_k s_k}{\sum_{k=b_1}^{b_2} s_k}$$

where:

- f_k is the frequency value at bin k.
- b_1 and b_2 are the bin boundaries between which to calculate the centroid.
- s_k is the spectral value at k

- **spectral rollof**

  The frequency for each window such that at least k percent of the energy of the spectrum is contained in this frequency bin and the bins bellow. The rolloff point is i such that:

$$\sum_{k=b_1}^{i} s_k = k \sum_{k=b_1}^{b_2} s_k$$

  where:

  - s_k is the spectral value at bin k.
  - b_1 and b_2 are the bin boundaries between which to calculate the spread.
  - k is the percentage of energy contained between b_1 and i.

- **spectral bandwidth**

  Given by:

$$(\sum_{k} s(k)(f(k) - f_c)^p)^{\frac{1}{p}}$$

  where:

  - $s(k)$ is the spectral magnitude at bin k.
  - $f(k)$ is the bin at k.
  - f_c is the spectral centriod at k.

- **spectral crest**

  The crest of the power spectrum over time. Given by:

  $$crest = \frac{max(s_k \in [b_1, b_2])}{\frac{1}{b_2 - b_1} \sum_{k=b_1}^{b_2} s_k}$$

  where:

  - s_k is the spectral value at bin k.
  - b_1 and b_2 are the bin boundaries between which to calculate the crest.

- **specral flux**

  Can intuitively be thought of as how much the signal changes over time. Given by:

  $$flux(t) = (\sum_{k=b_1}^{b_2} |s_k(t) - s_k(t-1)|^p)^{\frac{1}{p}}$$

  where:

  - s_k is the spectral value at bin k.
  - b_1 and b_2 are the bin boundaries between which to calculate the flux.
  - p is the normalization type.

- **spectral flatness**

  Measures how much noise-like a sound is. Given by:

  $$flatness = \frac{(\prod_{k=b_1}^{b_2} s_k)^{\frac{1}{b_2 - b_1}}}{\frac{1}{b_2 - b_1} \sum_{k=b_1}^{b_2} s_k}$$

  where:

  - s_k is the spectral value at bin k.
  - b_1 and b_2 are the bin boundaries between which to calculate the flux.

- **RMS**

  The root-mean-square. Given by:

$$RMS = \sqrt{\frac{1}{N}\sum_{n=1}^{N}|x_n|^2}$$

- **zero-crossing rate**

  How many times a signal crosses zero a given time frame.

2.2 Dimensionality reduction

2.2.1 PCA

2.2.2 t-sne

2.2.3 umap

2.3 Evaluation metrics

2.3.1 Silhoutte score

$$s(i) = \frac{a(i) - b(i)}{\max a(i), b(i)}$$

Where a is the mean distance between a sample $i \in C_i$ and all other points in the same cluster C_i. This gives us how close a point is to the corresponding cluster. It is given by the equation:

$$a(i) = \frac{1}{|C_i| - 1}\sum_{j \in C_i, i \neq j} d(i, j)$$

where $d(i, j)$ is the distance between points i and j.

b is the mean distance between a sample $i \in C_i$ and all other points in the next nearest cluster C_k. Given by:

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|}\sum_{j \in C_k} d(i, j)$$

The final Silhouette score for a clustering is the mean Silhouette Coefficient over all the datapoints:

$$\frac{1}{|I|}\sum_{i \in I} s(i)$$

Where I is the set of all datapoints. The final value is in the range $[-1, 1]$ with values closer to -1 indicating incorrect clustering and values closer to $+1$ indicating highly dense clustering. Scores around zero indicate overlapping clusters. The score is generally higher for convex well-sperated and dense clusters.

2.3.2 Roundness

The overall roundness of the plot is calculated by using the method proposed by Polsby & Popper [10]:

$$PP(D) = \frac{4\pi A(D)}{P(D)^2}$$

where D is the convex hull of all points of the plot, $P(D)$ is the circumference and $A(D)$ the area.

2.3.3 Overlap of cluster convex hulls

The measure of overlap of the convex hulls of each class. I calculate this by taking the ratio of the area of the unary union of convex hulls of each class to the sum of areas of the convex hulls of each class:

$$O = \frac{A(U)}{\sum_{c \in C} A(H_c)}$$

Where $A(U)$ is the unary union of all convex hulls, $A(H_c)$ is the area of the Hull for class c and C is the set of all classes.

2.3.4 Ripleys K function

Ripley's K function is sum of the number of points N within a distance r of a selected point p, per area λ surrounding p. This value is normalized by the total points:

$$K(r) = \frac{\sum_{i=1}^{n} N_{p_i}(r)}{n\lambda}$$

It may be interpreted as a meassure of deviation of a given distribution from the random Poisson distribution. In essence, this let's us measure the homogeneity of the spatial density of the data points. The expected value of $K(r)$ for a random distribution is πr^2. If the output value deviates from this value, this indicates clustering or dispersion in the data. The K-function may be normalized so that the expected value is r:

$$L(r) = \sqrt{K(r)/\pi}$$

Further normalization gives an expected value of 0, called the H-function:

$$H(r) = L(r) - r$$

Now, a positive value of $H(r)$ indicates that the data is clustered at the scale of r. If the value is negative, the data is dispersed.

3 Experiment design and overview

3.1 Dataset

I used the Medley-Solos-db dataset assembled by Lostanlen et al. [8]. Downloaded through the mirdata python library [2]. The dataset consists of 21572 mono WAV files sampled at 44.1 kHz at a bit depth of 32. Every audio clip has a duration of 2972 milliseconds. The data is split into 3 subsets: training, validation and test. Each sample belongs to one instrument category among a taxonomy of 8. Each instrument class was given a distinct color for easier recognition on the plots as seen in 4. I use the training subset of this dataset. The distribution of sounds is summarized in table

clarinet	elec. guitar	f. singer	flute	piano	tenor sax	trumpet	violin
732	955	1142	3167	2609	325	406	2899

Table 1: The distribution of instrument classes

3.2 The processing pipeline

To produce a 2d scatter plot of the dataset, the original audio files, each an array of 65,536 floating-point numbers, has to be reduced to 2 values. The process can be thought of as having 4 distinct steps:

1. **Preprocessing.**

   The audio files are ingested in a format which is easy to run calculations on. In this case a numpy ndarray.

2. **Feature Extraction.**

   Some characteristics of the sound are extracted using a selection of algorithms and mathematical tools. These are then used as an intermediate representation of the sound for further processing.

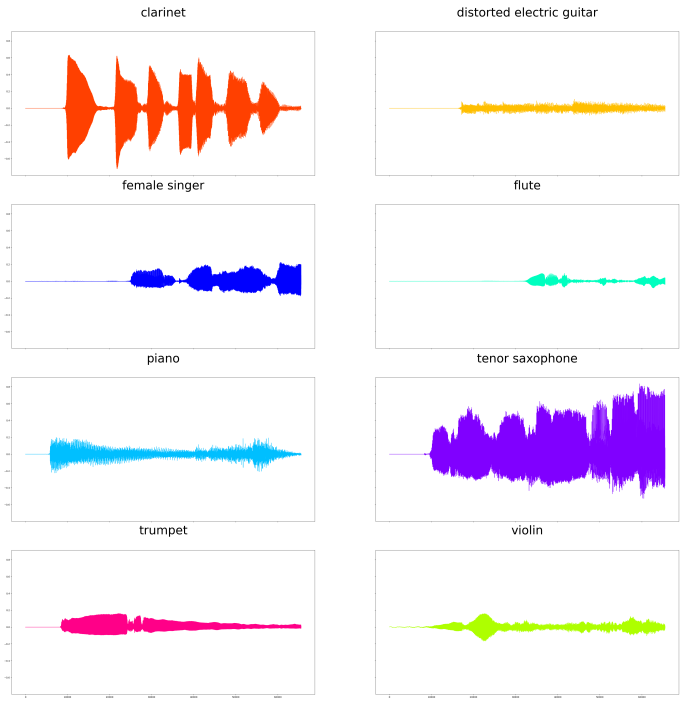

Figure 4: Selected sample belonging to each of the instrument classes. Each consists of 65,536 32-bit floating point numbers.

3. **Feature manipulation**

   Some of the features had to be further modified after extraction. The operations included reshaping feature matrices and aggregation.

4. **Dimensionality Reduction.**

   After selecting a set of features to serve as a representation of the original files an algorithm is applied to reduce them to two dimensions.

Once the plots have been generated, the one which most closely fits the defined criteria must be chosen. As such, an extra, fourth step in which plots are evaluated must be added. Each of these steps will be described in greater detail in the next section.

3.3 Preprocessing

The data is ingested using the Librosa python library [9] used for music and audio analysis. The "librosa.load" method was used to convert the WAV files to a float32 numpy ndarray.

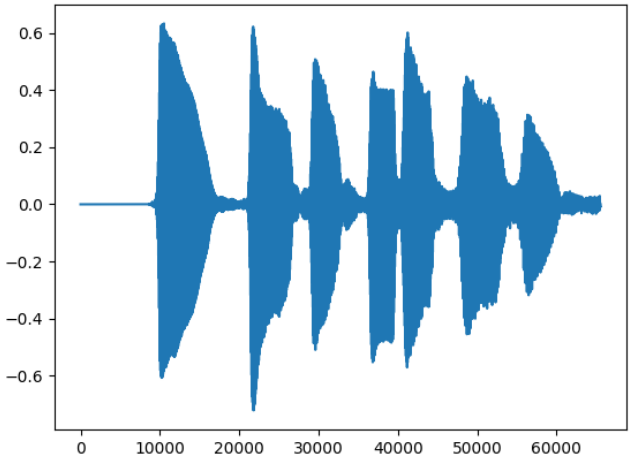

Figure 5: A sample from the clarinet class

After loading the samples, the amplitudes were normalized to be in the range $[-1, 1]$ by dividing by the max amplitude value for the sample. The data loaded in such a way was stored in a hdf5 file.

3.4 Feature Extraction

Finding a compact representation of phenomena is crucial for machine learning processess, including dimensionality reduction. To produce a meaningful representation of the raw data, useful to machines as well as humans, the step of extracting features is required. It can be even thought of as a preliminary dimensionality reduction technique as a raw signal consisting of many thousands of values to just a handful, which, with luck, provide an adequate representation of useful characteristics, innate to the signal. Most of the feature extraction steps were calculated using the implementations found in the librosa library.

3.4.1 STFT

The Short-time Fourer Transform is a basic representation in signal processing, which captures the change in frequency content over time. To extract the stft, I used the librosa implentation. I decided to take the STFT over 32 windows in both the time and frequency domains, finally giving a 32x32 matrix:

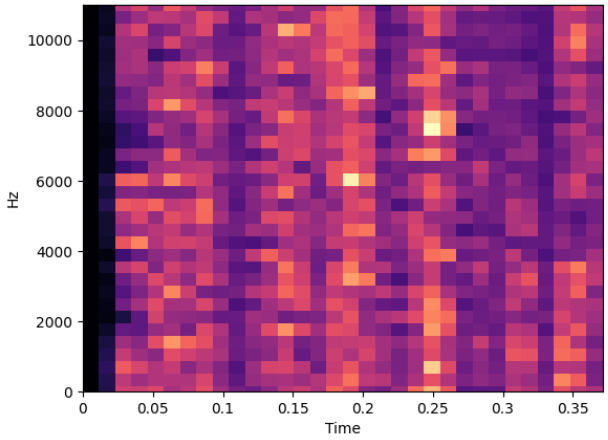

Figure 6: STFT of the sample in figure 5

3.4.2 MFCC

The Mel-Frequency Cepstral Coefficients are a heavily used in both speech recognition and MIR. [5, 12, 11, 3] I used the librosa implementation to calculate the mfcc's. I decided to go with a Cepstral Coefficient count of 20 and hop length of 256, resulting in a feature size of 20x257:

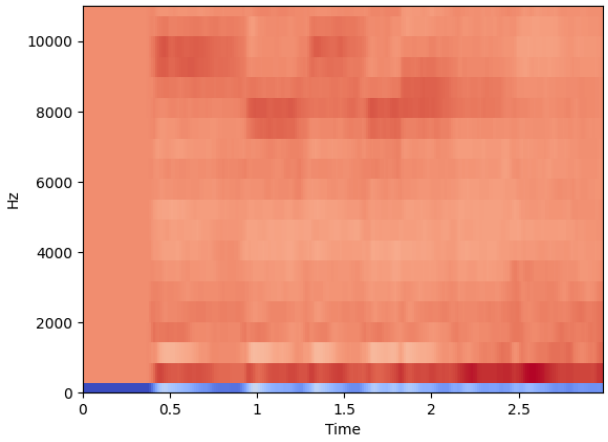

Figure 7: MFCC of the sample in figure 5

3.4.3 MIR Metrics

A number of metrics from the field of audio analysis has also found to be useful when extracting timbre information from audio signals [3, 12]. In my case these will include:

- Root mean square
- Spectral Centroid
- Spectral Crest
- Spectral Flux
- Spectral Roll
- Zero crossing Rate

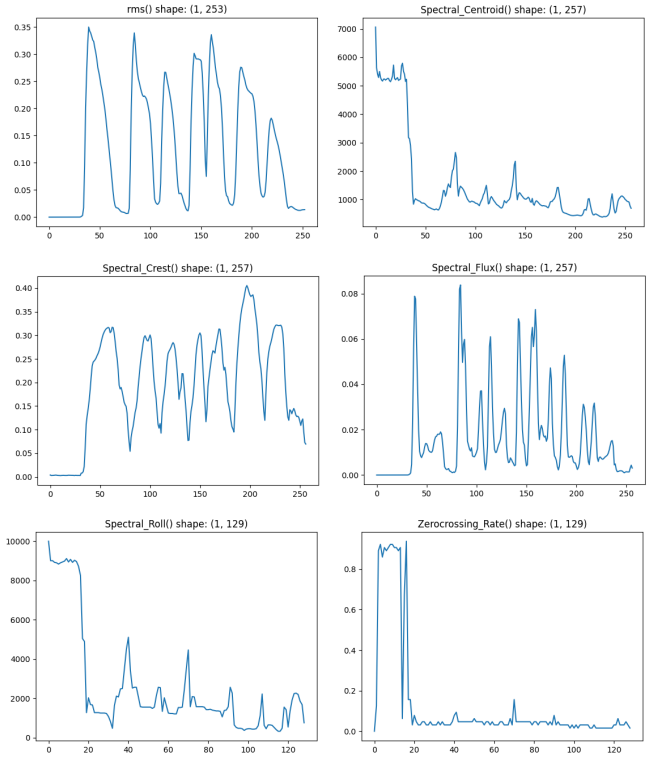

Figure 8: Graphs of MIR features for the sample in figure 5

3.5 Feature Manipulation

Many of the features are of different shapes and sizes. Since the matrix passed to the dimensionality reduction algorithm must be rectangular (i.e. a vector for each sample) a way must be found to force the features into a 1-d vector of values. I have used two approaches.

- **Flattening**

  If the feature matrix is 2 dimensional, we can simply concatenate consecutive rows into one feature vector transforming a (k, n) shape matrix to an array of length $k \times n$. The raw STFT and MFCC feature matrices must be flattened in this way to be used in the further steps of dimensionality reduction.

- **Aggregation over selected axis**

Another approach to reducing feature dimensions is that of aggregation. We can calculate an aggregate value over a particular axis, in essence reducing the dimensionality of the matrix by this axis. We can then treat the reduced matrix as any other feature vector. As suggested by Dupont et al. [4] and Fedden [5], I chose three aggregation functions: the average, standard deviation, and the mean of the difference between consecutive values in the vector.

In my case this procedure was applied to obtain the following four features:

- raw stft, reducing size from 32x32 -> 32x3. I call this "statistically shortened stft"
- raw mfcc, reducing size from 20x257 -> 20x3. I call this "statistically shortened mfcc"
- a second time to the statistically shortened stft, reducing size from 32x3 -> 3. I call this "vertically statistically shortened stft"
- a second time to the statistically shortened mfcc, reducing size from 20x3 -> 3. I call this "vertically statistically shortened mfcc"

Inspired by the successes of Klustr [12], with using a preliminary dimensionality reduction step on raw features using PCA before feeding applying dimensionality reduction proper, I also tried this approach on the raw stft and mfcc features. I decided to make features out of the top n principal components, where n took the values: [3, 5, 8, 12, 20].

3.6 Dimensionality Reduction

The last step in the pipeline is the final dimension reduction of the final collection of features to just 2 values. 4 different algorithms were used in this step:

- Principal Component Analysis

  The basic, tried-and-tested dimensionality reduction method. This method doesn't accept any additional parameters except the number of Principal Componenets to output. As such, for each collection of features we obtain 1 plot.

- T-stochastic neighbour embedding

  What has come to be a widespead technique in the field of machine learning for it's ability to create useful 2d maps of data. Used by

McDonal et al. and Hantrakul et al. to create visualizations of audio data. T-sne's uses extend far beyond just audio data, however. It is commonly used in the field of single-cell genomics to visualize human genetic data [7] and is able to seperate samples from different continents and even reflects some local, sub-continental patterns.

3 parameters influence the visualization in a significant way:

- Perplexity, which can be interpreted as how much attention the algorithm should give to local or global structure. In the original article van der Maaten & Hinton suggested that perplexity values should generally fall in the range 5-50. In my case, smaller values tend to result in plots with less dense clusters with higher values giving more well-seperated clusters. I chose the values [5, 10, 20, 40, 60] as parameters for the t-sne plots.
- The learning rate. Usually falls in the range [10-10000]. I however determined that values higher than 1000 seemed to lose global structure. I chose the values [20, 50, 100, 200, 300]
- Iterations. I decided to go with a constant value of 3000. After heuristic tests I determined that the plots seemed to be stable for this dataset.

- Uniform Manifold Approximation and Projection

  A relatively new dimensionality method. It is very effective at preserving both the local and global structure of the original data in it's projections. Similarly to t-sne it is also based on manifold learning. The important hyperparameters are:

  - **number of neighbours**

    This parameter controls the number of approximate nearest neighbors to use to construct the initial high dimensional graph. Low values tend to focus on the fine, local structure. Higher values put an emphasis on the wider structure since they take into account a larger number of neighbour points. I chose values [5, 10, 15, 30, 50, 100, 200]
  - **minimum distance**

    is the value used by the algorithm to determine what the minimum distance points on the embedding can be from each other. I chose to sweep throught the whole range: [0.0, 0.001, 0.01, 0.1, 0.5, 0.75, 0.99]

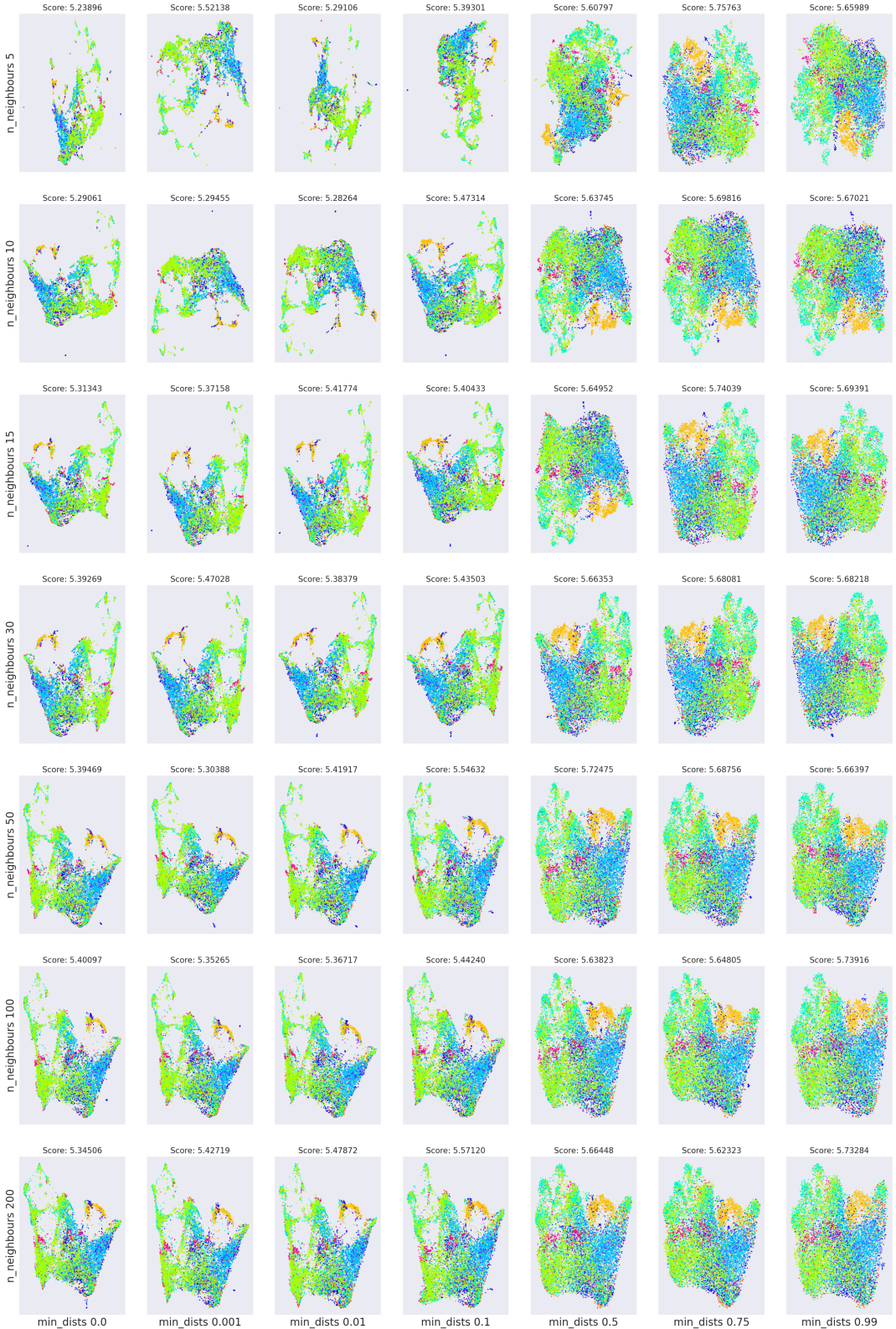

Figure 9: A Umap grid search over stft.
It is clear to see that lower values of hyperparameters (top left) give more densely clustered plots with a focus on the local structure, while higher values (bottom right) are more distributed and focus more on global relationships.

Since the objective is to make a round and homogenously dispersed plot, while still keeping the clusters seperate, I suspect that values, which strike a balance between being globally spread out and having enough detailed local structure to seperate the clusters will be evaluated as teh best.

3.7 Scoring the plots

The experiment was designed with a particular goal in mind - generating plots from the original audio data which would enable an intuitive grasp of the dataset. In order to achieve this, I define several metrics for evaluating the plots:

- How well the embedding reflects the inherrent relationships between datapoints. This is measured by the Silhouette score with the class labels corresponding to cluster labels. Also, the overlap of the convex hulls of classes is an indicator of this quality.

- Readability and ease-of-navigation of the plot. In order for the plot to be readable, points should be as evenly distributed as possible, avoiding clumps, which might be hard to navigate. Ripley's function is an indicator of homogeneity of the density distribution of points on the plot. I figured, that a regular, uniform shape of the plot would increase readability, as such metric of readability is given by the Polsby-Popper method for measuring the roundness of the convex hull of the plot.

Each plot produced is scored using these metrics. Since each of these metrics had a different range of values, to compute a final score for the plot, each metric was normalized to the range $[0, 1]$. The final score for plot $p \in P$ is a weighted sum of all the normalized individual metrics given by:

$$T(p) = 2silhouette(p) + 2ripley(p) + overlap(p) + roundness(p)$$

Where:

- $silhouette(p)$ is the normalized Silhouette score. Because the silhouette metric is in the range $[-1, 1]$ the value must be shifted to be in range $[0, 1]$: $s(p) + 1$. Where $s(p)$ is the Silhouette score for the plot. The shifted silhouette score is then normalized relative to the max silhouette score, finally giving:

$$silhouette(p) = \frac{s(p) + 1}{\max\{s(p)|p \in P\}}$$

- *ripley*(p) is a metric based on the Ripley H-function. First, the average Ripley H-function is taken for plot p for radii in the set: $R = (0.05, 0.1, 0.25, 0.5)$ $\sum_{r \in R} H_p(r)$ Since the H-function can assume values both negative and positive, I decided to take the absolute value. This causes a loss of information, since negative values indicate dispersion and positive ones indicate clustering. However, this distinction is not important for the purposes of the experiment. The only information important to us is how much the plot deviates from being uniformly dense. Since we want values close to 1 to indicate a better score I take the inverse, giving:

  $$H(p) = abs(\frac{\sum_{r \in R} H_p(r)}{|R|})^{-1}$$

  With $H_p(r)$ being Ripley's H-function for plot p taken for radius r. Finally, the value is normalized relative to the max value for all plots.

  $$ripley(p) = \frac{H(p)}{\max\{H(p)|p \in P\}}$$

- *overlap*(p) is the ratio of overlap of convex hulls for the clusters to the area of the whole plot. Normalized relative to the max value for all plots:

  $$overlap(p) = \frac{O(p)}{\max\{O(p)|p \in P\}}$$

- *roundness*(p) is calculated using the Polsby-Popper method. Also normalized relative to the max value for all plots:

  $$roundness(p) = \frac{PP(p)}{\max\{PP(p)|p \in P\}}$$

4 Experiment Evaluation

4.1 Results

The final, best scoring plots for different feature sets are shown in 10. Looking at the plots in this figure, the resulting plots seem to be satisfactory. That is, most of the plots seperate the classes relatively well as well as retaining a round and fairly uniform structure as well as well-seperated clusters. The exceptions are plots which have gone through a preliminary pca reduction step, leaving 3 prinicipal components, labeled *pca 3 stft* and *pca 3 mfcc*. The

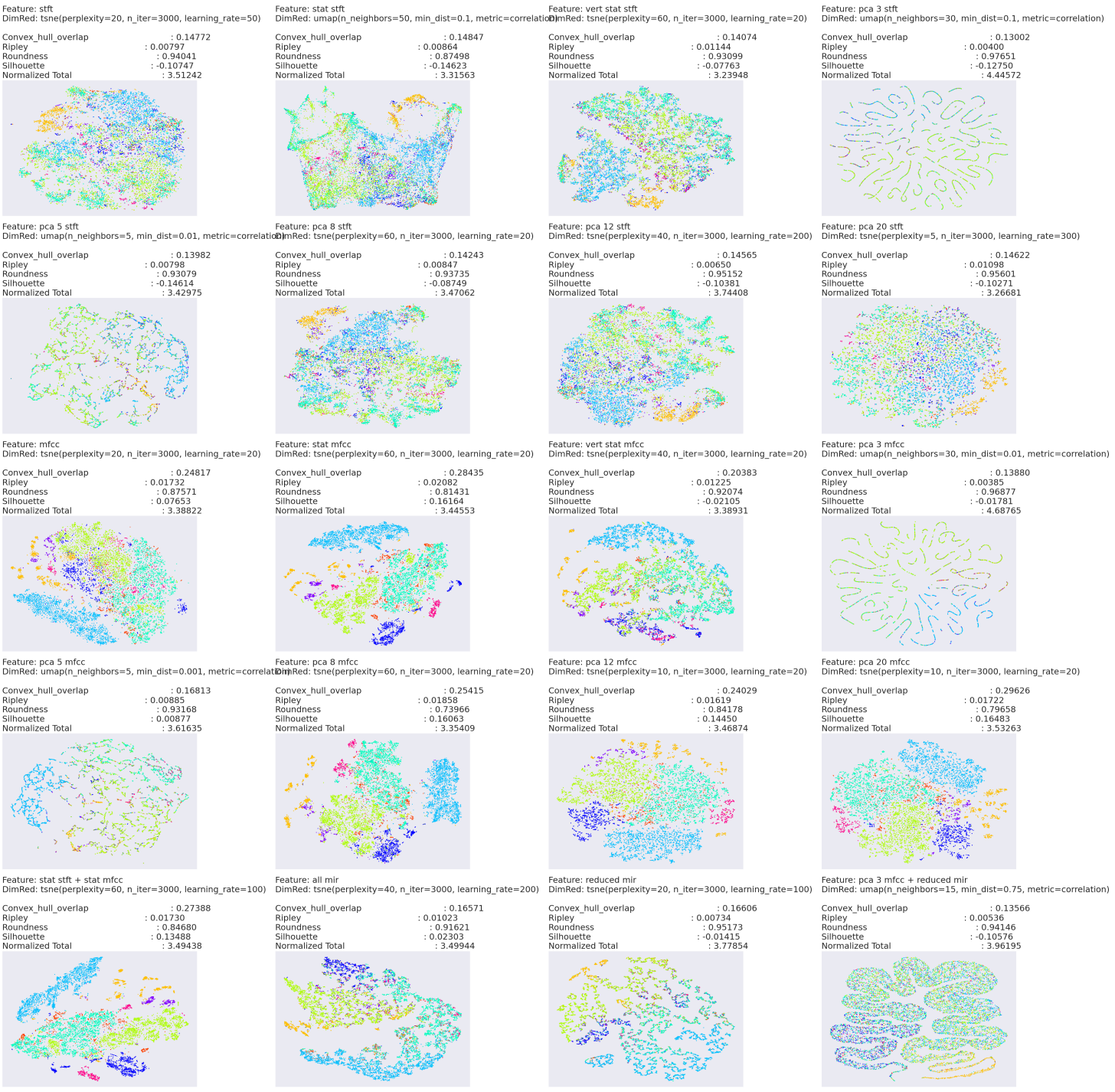

Figure 10: The best scoring plots for each feature set.

interesting, string shaped plots only appear with the combination of pca 3 + umap. This is most probably a side-effect of the way in which umap builds it's high dimensional representation of the data using the fuzzy simplical complex. Somehow, umap decides that the global structure contained in the 3d pca plot shown in 11 is best represented in this way. A closer look at the umap grid search (12) reveals that this is indeed the case. The more nearest neighbours the umap algorithm uses to determine connectedness in the high-dimensional representation, the more pronounced becomes the pattern.

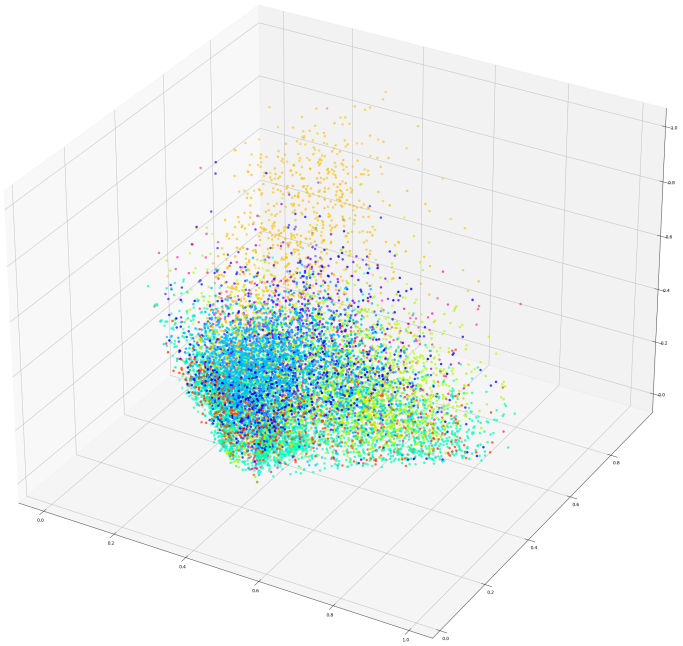

Figure 11: A plot of the pca 3 mfcc feature

This phenomenon is all the more interesting when we take a look at the top scoring plots overall, shown in 13. All the best scoring plots are from

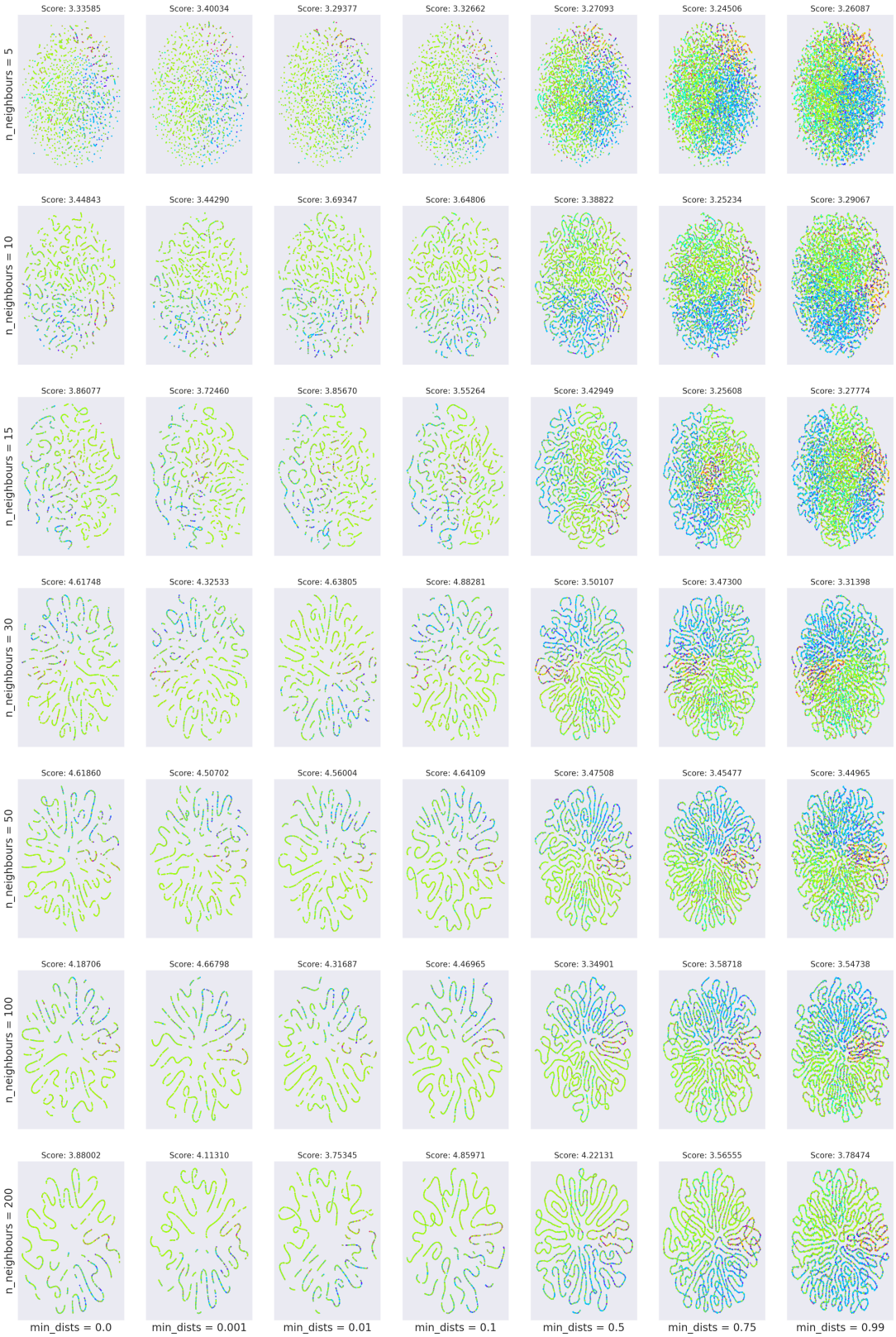

Figure 12: A umap grid search on the pca 3 mfcc feature. As the value of n neighbours increases, the string like pattern gets more pronounced.

this particular feature set.

This result is counterintuitive. It seems that many of the other plots would score better on practically every metric. The plots seem to be less clustered and less uniformly distributed. However, when looking more closely at the values, we can see that the silhouette and and overlap scores are very close to other plots, maybe slightly worse. However, when we look at the Ripley score, we see that these plots have values significantly closer to zero (meaning a higher score after normalization).

Looking at the top scoring plots for the ripley metric presented in 14, it is clear that the pca 3 plots do indeed score very highly in this regard. The first 8 top scoring ripley metrics come from a combination of pca 3 + umap. My hypothesis is that each of the string like shapes is composed of points which are placed at distance apart from each other equal to that of the min dist umap parameter. Also the strings themselves seem to be roughly equally spaced apart from each other. This means that for a circle taken from any point we obtain an approximately equally distributed number of points. Hence, the high score of these plots for the Ripley metric.

embedding pca for each fe best umap for each fe best tsne for each fe

best overall for each feature

best overall for each feature with onlhy x metric

go through the measures and explain why silhouette is insuficcient on it's own. (like klustr paper)

compare stat shortened stuff to non stat shortened stuff

combinatioin of mir with stat shortened to just mir and just stat shortened

go through each dimensionality reduction method and describe the results for each

describe the end results, what scored the best and why

4.2 Limitations

4.3 Future work

4.4 Conclusion

References

[1] sklearn metrics silhouette score.

[2] Rachel M Bittner, Magdalena Fuentes, David Rubinstein, Andreas Jansson, Keunwoo Choi, and Thor Kell. mirdata: Software for reproducible

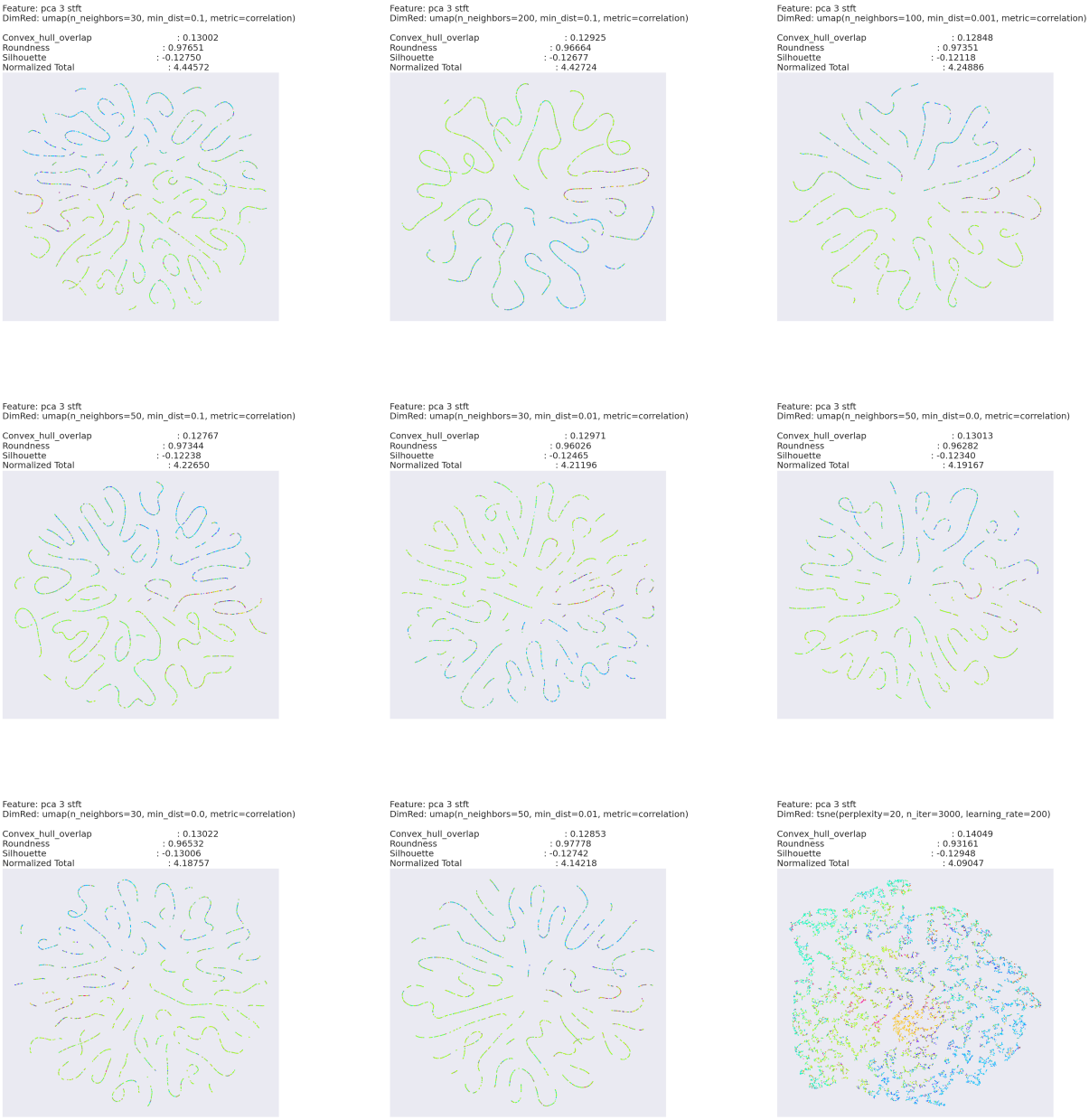

Figure 13: The top 9 scoring plots overall

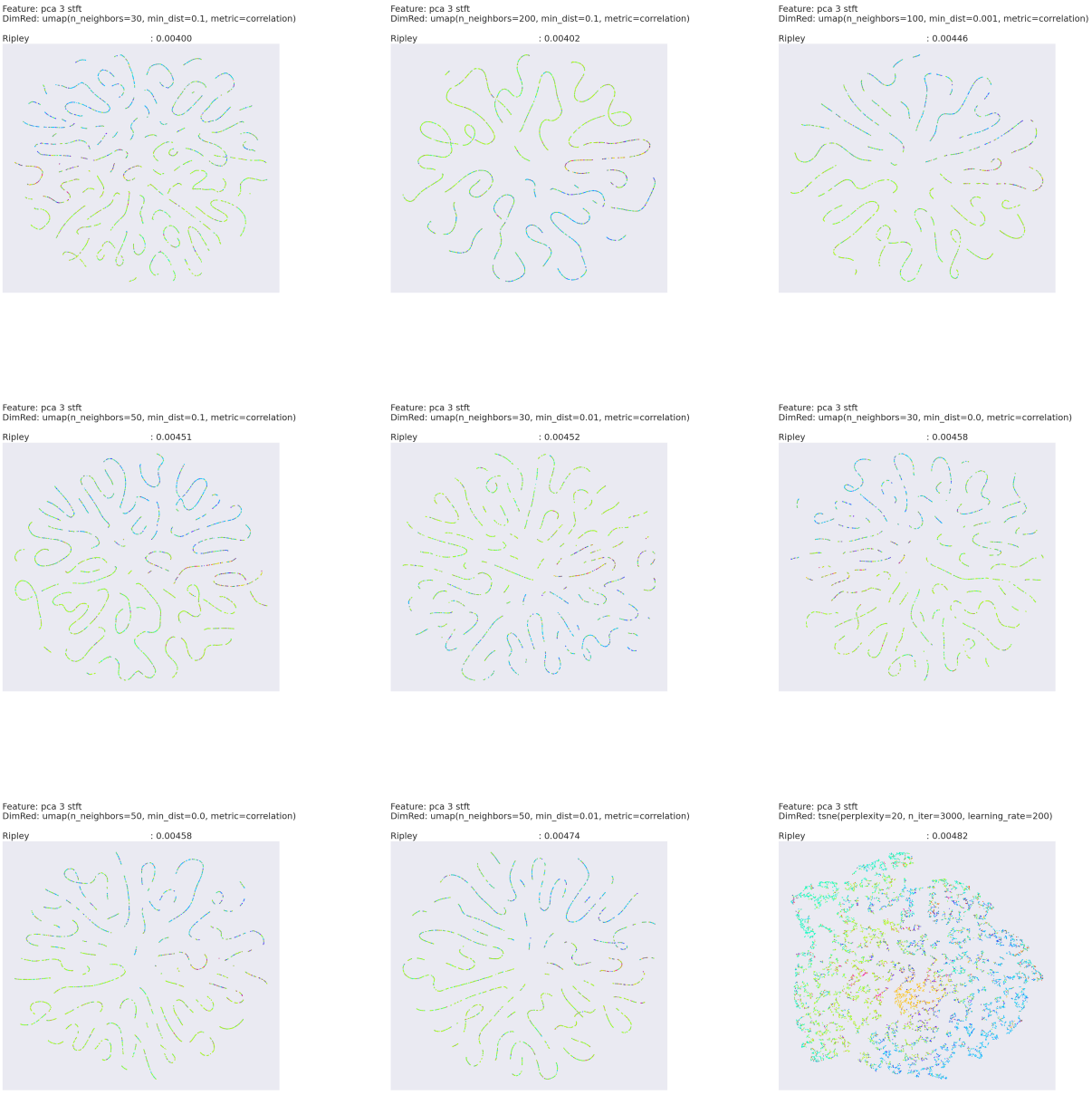

Figure 14: Top 9 scoring plots according to the Ripley metric

usage of datasets. In *International Society for Music Information Retrieval (ISMIR) Conference*, 2019.

[3] Jeremiah Deng, Christian Simmermacher, and Stephen Cranefield. A study on feature analysis for musical instrument classification. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 38:429–38, 05 2008.

[4] Stephane Dupont, Thierry Ravet, Cecile Picard-Limpens, and Christian Frisson. Nonlinear dimensionality reduction approaches applied to music and textural sounds. *2013 IEEE International Conference on Multimedia and Expo (ICME)*, Jul 2013.

[5] Leon Fedden. Comparative audio analysis with wavenet, mfccs, umap, t-sne and pca, 2017.

[6] Yotam Mann Kyle McDonald, Manny Tan. The infinite drum machine. 2016.

[7] Wentian Li, Jane E. Cerise, Yaning Yang, and Henry Han. Application of t-sne to human genetic data. *Journal of Bioinformatics and Computational Biology*, 15(04):1750017, 2017. PMID: 28718343.

[8] Vincent Lostanlen, Carmine-Emanuele Cella, Rachel Bittner, and Slim Essid. Medley-solos-DB: a cross-collection dataset for musical instrument recognition. September 2019.

[9] Brian McFee, Vincent Lostanlen, Alexandros Metsai, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Frank Zalkow, Ayoub Malek, Dana, Kyungyun Lee, Oriol Nieto, Jack Mason, Dan Ellis, Eric Battenberg, Scott Seyfarth, Ryuichi Yamamoto, Keunwoo Choi, viktorandreevichmorozov, Josh Moore, Rachel Bittner, Shunsuke Hidaka, Ziyao Wei, nullmightybofo, Darío Hereñú, Fabian-Robert Stöter, Pius Friesch, Adam Weiss, Matt Vollrath, and Taewoon Kim. librosa/librosa: 0.8.0. July 2020.

[10] Popper Polsby, Daniel. The third criterion: Compactness as a procedural safeguard against partisan gerrymandering. *Yale Law & policy Review 9.2 (1991): 301-353.*, 9:2579–2605, 2008.

[11] Karthikeya Racharla, Vineet Kumar, Chaudhari Bhushan Jayant, Ankit Khairkar, and Paturu Harish. Predominant musical instrument classi-

fication based on spectral features. *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, Feb 2020.

[12] Lamtharn (Hanoi) Hantrakul & Avneesh Sarwate. Klustr: A tool for dimensionality reduction and visualization of large audio datasets, 2017.

[13] Julius Smith. *Spectral Audio Signal Processing*. 01 2008.

[14] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.