Case Study 3

Hammad Khan **2584665K**
Sarvesh Tendulkar **2582865T**
Sebin Thomas **2604952T**

Introduction

This case study involves classifying a time series audio output with a simple classifier and using visualization methods to find an optimal set of parameters that gives a good separation between the different classes.

The audio data is sourced from the ***Stane*** project[1] which is a device that is controlled by tactile input. The device has different textures around it and a microphone attached to it. Rubbing, scratching, and tapping the device makes distinct sounds that are picked up by the microphone and are processed as separate gestures. This contrasts with the capacitive touch sensors used in common touchscreen devices, where the users are disconnected from the tactile feedback and the actions they have performed.

Most machine learning algorithms tend to use a fixed-length feature vector for training and inference. To adapt time series data (like audio data) to fit this methodology, we have to perform windowing, which is the act of breaking up the continuous data into fixed time intervals (which may or may not overlap) and use those chunks to train the model.

Audio data is also very high dimensional and tends to not contain much information in a single sample. The trends are visible only when one considers data over longer periods of time (like milliseconds to seconds). One method is to drop features that do not contribute to the overall classification or averaging features together to reduce noise, another method to extract useful information from audio data is to transform them in a manner which takes the *time* axis into consideration. Methods like the *fourier transform,* which transforms a time varying signal into the sum of several sinusoidal waves, are helpful in achieving this.

The problem statement is to find a good set of parameters that perform windowing and feature selection/transformation so that we are able to extract maximum performance from our baseline classifier (K-nearest neighbor classifier), by using visualization methods as a yardstick, for classifier performance.

Methods

We are given six parameters to control the pre-processing of the data. The classifier is fixed, and the aim is to use visualization techniques to help us select the best set of parameters. The parameters we can control involve the window size, which is the length of the window we are choosing, (measured in samples), window step, which measures how much samples we should move forward, in choosing the next window, decimate, which refers to how much feature reduction

should be applied, feature range, which says what slice of the feature vector we should take (0 to 1 for no slice, to the whole vector), window function and feature function. The selection of above parameters was primarily guided by the linear and manifold visualization techniques as elaborated below.

Linear Methods

Principal Component Analysis:
Principal Component Analysis or PCA, is a linear dimensionality reduction technique which projects the data onto a lower dimensionality space by calculating eigenvalues and eigenvectors using Singular Value Decomposition of the data[2].
The n_components parameter decides the number of components to keep for projecting the data on, in this case it was set to 2 i.e., the original data was projected onto 2 components that best separated the data.

Manifold Methods

Locally Linear Embedding:
Locally linear embedding (LLE) is another linear projection technique. It works by trying to express each point as a linear combination of its neighbouring points and tries to map it into a lower dimension in such a way that the points are still expressed as a linear combination of the same neighboring points. It can basically be explained as a bunch of local Principal Component Analyses merged smoothly together to give one coherent structure.

ISOMAP:
ISOMAP, which is short for Isometric Mapping, works similar to LLE, in that it tries to find a lower dimensional representation of the data which preserves the geodesic distance between different points. Geodesic distance refers to the number of edges between two vertices in a shortest path between them. ISOMAP uses K-nearest neighbors heuristic and might suffer from issues arising due to a small or large value of k being chosen.

Uniform Manifold Approximation and Projection:
Uniform Manifold Approximation and Projection (UMAP) is a non-linear dimensionality reduction technique. The algorithm relies on 3 assumptions about the data[3]:

1. The data is uniformly distributed on Riemannian manifold;
2. The Riemannian metric is locally constant (or can be approximated as such);
3. The manifold is locally connected.

For the UMAP algorithm, min_dist and n_components parameters were set as 0.9 and 2, respectively. The min_dist parameter provides how far apart points are allowed to be in lower dimensional space whereas the n_components determine the number of dimensions the resulting data would be projected on[4].

t-distributed Stochastic Neighbor Embedding:

t-distributed Stochastic Neighbor Embedding (t-SNE) is another non-linear dimensionality reduction technique used to visualize high-dimensional data. It converts similarities between data points as joint probabilities and tries to minimize the difference between those joint probabilities of the low-dimensional projection and the high-dimensional original data[5].

For t-SNE algorithm, n_components parameter was set to default value of 2, meaning data would be projected onto a 2-dimensional space.

Results

The main aim of this case study was to explore different data visualization techniques to help guide the decision of selecting a feature vector that provides optimal model performance. The optimal performance could only be achieved by trying out different combinations of the six parameters provided since no other data engineering techniques could be applied like feature selection or changing the model itself, which was KNN in this case. We based our experiments on the five visualization techniques: PCA, ISOMAP, LLE, UMAP and tSNE; starting off with a baseline score and working our way through to achieving best performance with the help of aforementioned methods. Since PCA is only good with linear data, we relied more on UMAP and tSNE which are better at finding non-linear relationships in the data as we applied different feature_fn in experiments later on. For brevity, we aren't including the results of ISOMAP and LLE.

Experiment 1

We started off with the default values for the parameters to set a baseline model performance score. The total score obtained was 62.49% on the default KNN classifier. For all three of the methods, the data points appeared to be clustered together as can be observed in below plots, not providing much of an information in terms of variation in the data.

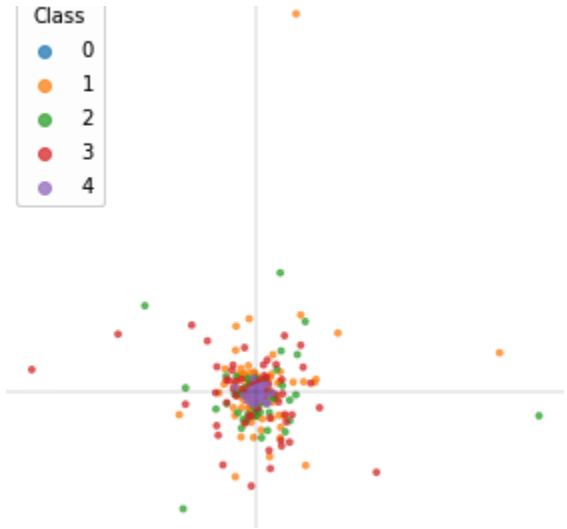

Fig 1(a): PCA

Fig 1 (b): UMAP

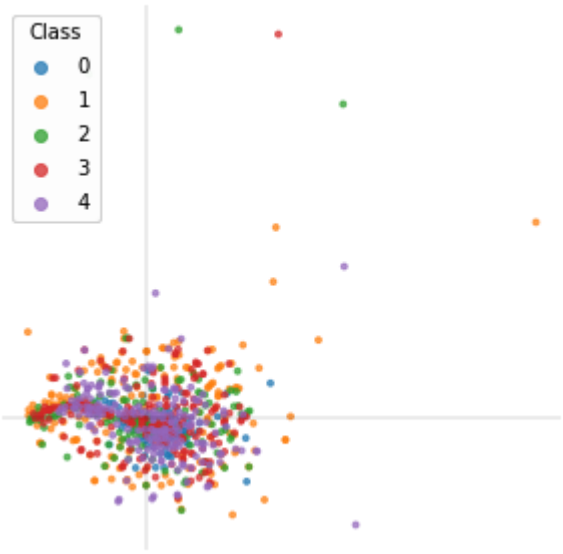

Fig 1(c): tSNE

Experiment 2

For the next experiment, we increased the window size to 512 and changed the 'feature_fn' to 'fft'. This resulted in a 9% increase in overall model performance at 71.28% and increased variability in data for all the three visualization methods.

Fig 2 (a): PCA

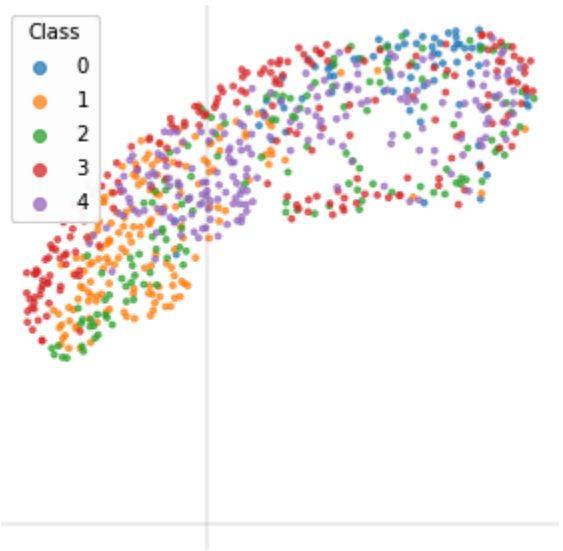

Fig 2 (b): UMAP

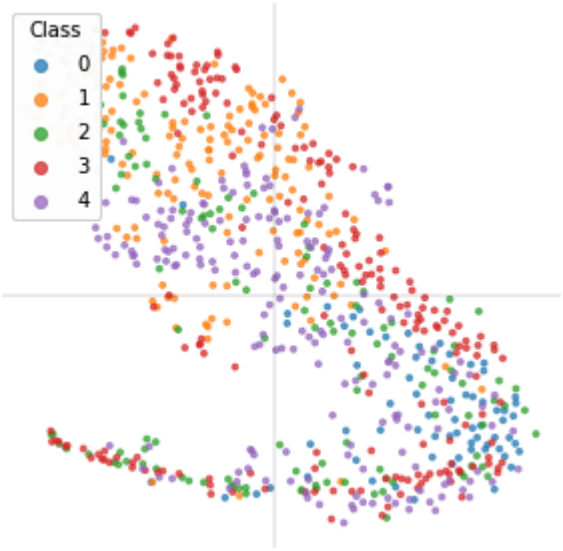

Fig 2 (c): tSNE

Experiment 3

In the next experiment, we changed the 'window_fn' to 'blackmanharris' and reduced the step size to 64. As a result, there was a slight increase in the classifier's accuracy at 73.70% and the data appeared better separated for UMAP and tSNE however, the PCA failed to capture the underlying relationship in the data due to its limitations and this continued to be the case for PCA for remainder of the experiments.

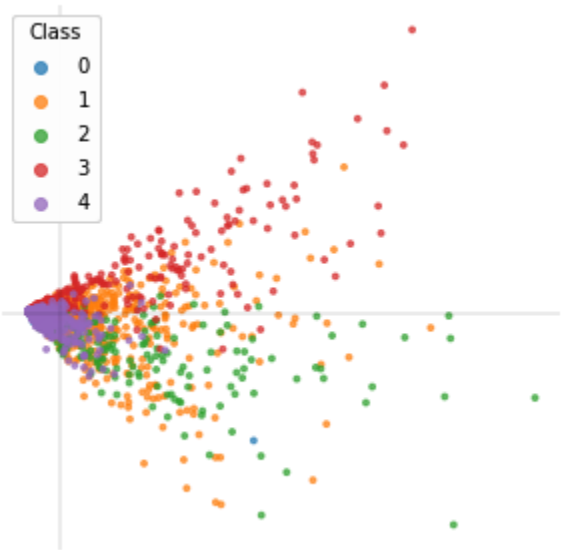

Fig 3 (a): PCA

Fig 3 (b): UMAP

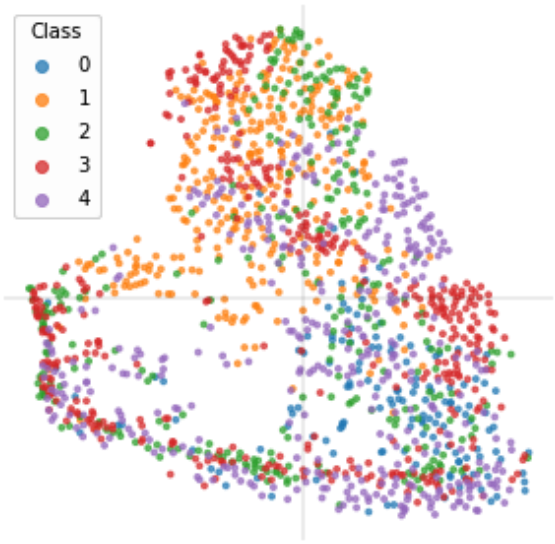

Fig 3 (c): tSNE

Experiment 4

For the next experiment, we increased the decimate parameter to 4, this essentially removes every 4th sample from each window. In this case, we achieved an accuracy of 75.84%. Although there was a slight increase in the prediction accuracy, the visualization plots did not vary much from the previous experiment.

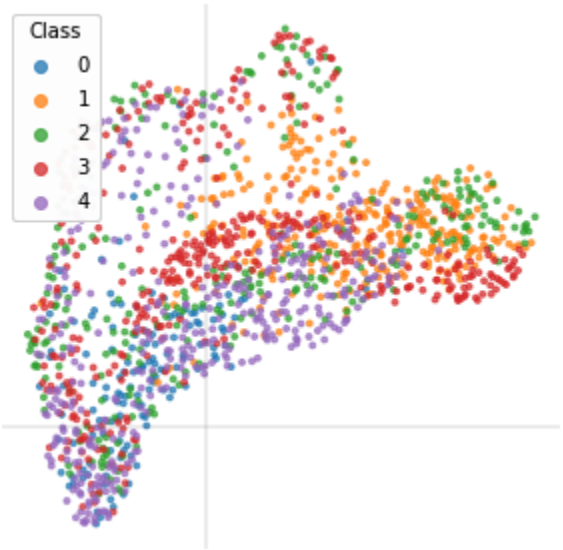

Fig 4 (a): UMAP

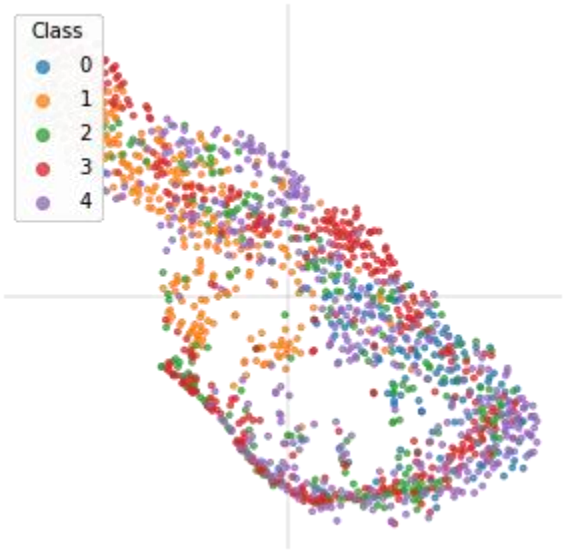

Fig 4 (b): tSNE

Experiment 5

In this experiment, we increased the window size, nearly by a factor of 12, to 6339 and decreased the step size to 50. We also changed the ‘window_fn’ and the ‘feature_fn’ to ‘boxcar’ and ‘fft’ respectively and set decimate to 20. This resulted in a significant increase over previous experiments with an overall accuracy of 85.13%. The UMAP and tSNE plots showed clear separation of the data and the plot shows distinct clusters for classes 3, 4, 0 and the rest. Classes 1&2 appear mixed together with no clear boundary with class 0 showing one clear cluster and spread out between classes 1 and 2.

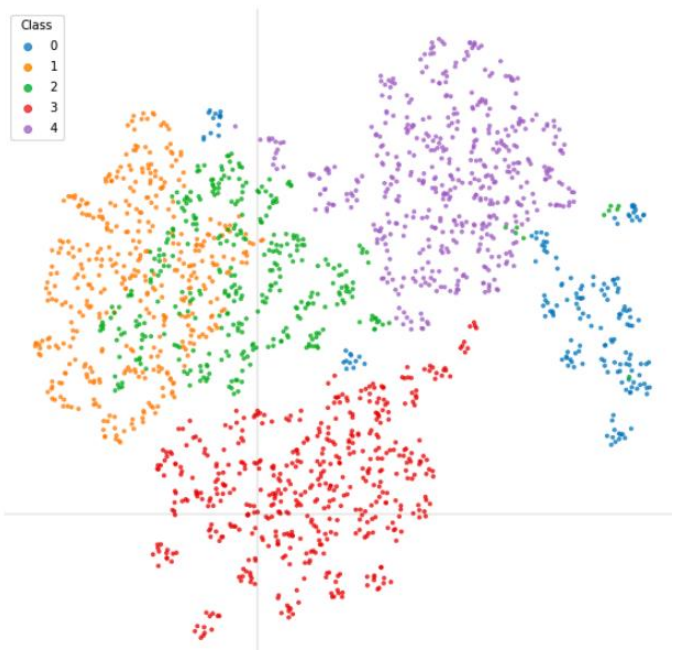

Fig 5 (a): UMAP

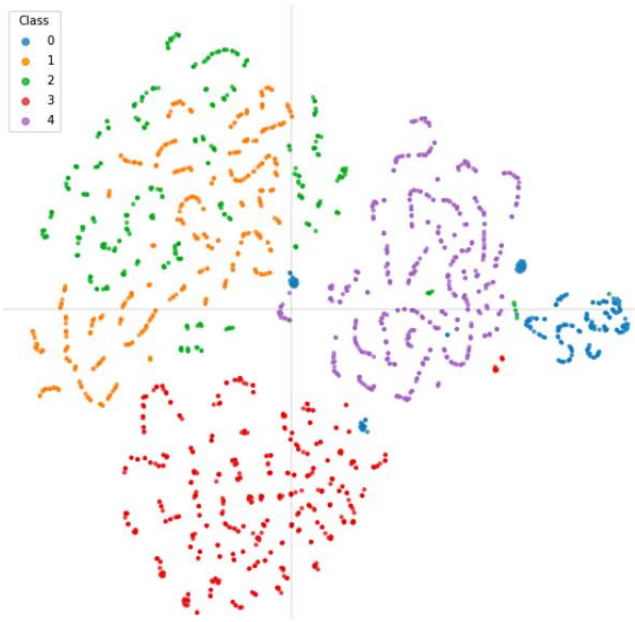

Fig 5 (b): tSNE

Experiment 6

For the final experiment, we decreased the window size to 4096, reduced the step size to 32 and set the deimate to 2. After loading the data with these parameters, we achieved the highest model accuracy of 87.25% and as can be observed from below plots, the data points from the same class were further closer together and there was lesser overlap with other class data points.

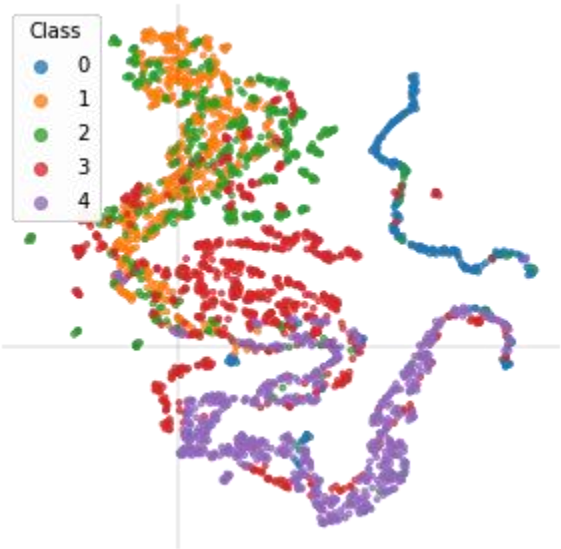

Fig 6 (a): UMAP

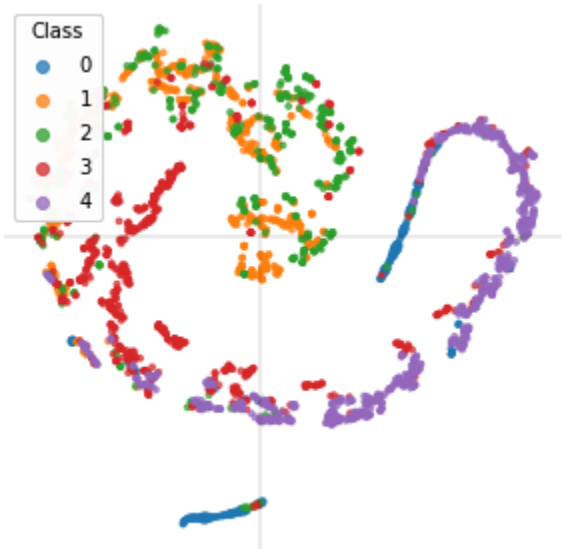

Fig 6 (b): tSNE

Discussion

Visualization techniques certainly help in making informed decisions about the data and should be used before devising any solid feature engineering strategy and predictive model pipeline, as part of exploratory data analysis. Moreover, whenever possible, more than one visualization technique should be used especially when there is high-dimensional data involved as one method might capture the convoluted relationship in the data that the other method may fail to do.

As can be observed from the plots in Results, in this case study, since the data was generally of high dimensions (dependent on the window size and decimate parameters) and the underlying relationship appears to be non-linear, the manifold learning and dimensionality reduction techniques: UMAP and tSNE did better as compared to PCA, ISOMAP and LLE, in fact, after first couple of experiments PCA being a linear dimensionality reduction method, failed to provide variation in the data and essentially resulted in similar plots.

Although, the aim of this case study was not to get the best performance, perhaps implementing a feature engineering strategy with a different classifier might give higher performance but starting with a baseline score of 62.49% on default parameters we tried different combinations of parameters guided by the visualization plots and classifier performance. In general, we realized that higher window size helps in better separating the data, and that along with ‘cepstrum’ and ‘boxcar’ as ‘feature_fn’ and ‘window_fn’ respectively, the data points are grouped together with their own class and are fairly separated from other classes with lesser overlapping. However, for data points of classes 1 and 2 with ‘orange’ and ‘green’ colour coding respectively, we observed that the data points were distinctly inseparable leading to poor classifier performance on them, this can be particularly observed for test-set ‘challenge_test_4’ where the original class was ‘green’ but classifier struggled to make good predictions.

It is interesting to note that the parameter set that gave us good separation between different classes isn’t the same as the one which performed better. Experiment 5 with the following set of parameters gave us a good visual separation of different classes, with a performance of 85.13%

```
'size':6339

'step':50

'decimate':20

'feature_range' :(0.0, 1.0)

'window_fn':"boxcar"

'feature_fn':"fft"
```

After experimenting with different parameters using visualization techniques, we achieved a classifier performance of 87.25% with the following parameters:

```
'size':4096

'step':32

'decimate':2

'feature_range' :(0.0, 1.0)

'window_fn':"boxcar"

'feature_fn':"cepstrum"
```

References

1. Murray-Smith R, Williamson J, Hughes S, Quaade T. Stane: Synthesized Surfaces for Tactile Input. Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08. 2008. https://doi.org/10.1145/1357054.1357257

2. Scikit-learn: Machine Learning in Python [Internet]. [place unknown]: scikit-learn; c2020. sklearn.decomposition.PCA; 2020 [cited 2021 Mar 29]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

3. UMAP [Internet]. [place unknown]: Leland McInnes; c2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction; 2018 [cited 2021 Mar 29]. Available from: https://umap-learn.readthedocs.io/en/latest/index.html

4. UMAP [Internet]. [place unknown]: Leland McInnes; c2018. Basic UMAP Parameters; 2018 [cited 2021 Mar 29]. Available from: https://umap-learn.readthedocs.io/en/latest/parameters.html

5. Scikit-learn: Machine Learning in Python [Internet]. [place unknown]: scikit-learn; c2020. sklearn.manifold.TSNE; 2020 [cited 2021 Mar 29]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html