Technology
Arts Sciences
TH Köln

On spoken digit classification utilizing a basic SVD approach.

Fabian Rosenthal   Philipp Schwarz

Module: Angewandte Mathematik

Lecturer: Prof. Dr. Stefan Michael Grünvogel

September 15, 2022

Abstract

In classification of handwritten digits a simple algorithm based on Singular Value Decomposition (SVD) is known and has been applied by offices like United States Postal Service for decades. This paper adapts the SVD algorithm to classification of spoken digits from audio recordings based on their spectrogram representation. To test this digit classification we present an evaluation study on the large AudioMNIST data set. Achieving an accuracy of only 65 % the SVD classification can not keep up with benchmark networks on this data set. *(Abstract author: Fabian Rosenthal)* ***Index Terms***— Digit classification, SVD, AudioMNIST, spoken digit, applied mathematics

Contents

List of Figures

List of Tables

1 Introduction

Philipp Schwarz

In the past, significant gains have been made in the field of speech recognition: Knowledge of speech characteristics and human auditory perception in combination with data science and machine learning techniques have worked together to achieve greater accuracy [1, 2, 3, 4]. Nonetheless, it is not uncommon to use ideas from image processing in audio classification e.g. in tasks of speech recognition [5].
Some use cases for such classification algorithms can include extracting information from spoken words in an audio file or detecting (i.e. recognizing) the speaker. For example, a voice assistant has to be able to interpret a spoken phone number whilst interacting with the user. In our example, an audio recording of a spoken digit shall be classified as one of ten feature classes.

Often, it is undesirable to outsource the task and rely on third party machine learning systems. Either for privacy or data security concerns, or simply because such systems are often 'black boxes' but insights into the inner processes of the classification are required [6].

Recent works in this field concentrate mostly on the usage of either Deep, Recurrent or Convolutional Neural Networks [7, 8, 4]. Especially the field of speech recognition makes use of this kind of machine learning [8].
Singular Value Decomposition instead has not been used frequently for that matter. As an example, Robles-Guerrero et.al. could be mentioned, who used this approach to classify sound patterns in bee colonies [9].
Apart from auditive classification problems, SVD has been mainly used in image classification. Most prominent by Turk & Pentland, who presented the method of 'Eigenfaces' [10]. An overview over this field of research can be found at [11]. Often SVD is used in a side process of a larger classification or process algorithm. Another example is Kim et.al., who used SVD for noise reduction on input signals [12].
This work presents a simple approach of spoken digit classification, not based on deep learning or neural networks, but closely related to image processing algorithms and image recognition methods. In our case, the algorithm will be trained on the spectogram data of audio samples. Therefore, we will examine the research question if a basic SVD classifier trained on spectrogram data will perform sufficiently on unseen test data samples. As evaluation we present a study on the AudioMNIST data set. In the course of this report the algorithm will be analyzed and evaluated in depth. Lastly, the viability of the procedure to classify a rather complex data set will be discussed.

2 Foundations

2.1 Spectrogram

Fabian Rosenthal

In digital domain an audio recording is a discrete time series of digital values, where the number of values per second is determined by the sample rate (f_s) and the range of possible values of a single sample is determined by the number of quantization levels (bit depth) in analog-to-digital conversion [13]. Information about the signal's frequency content can be computed with Fast Fourier Transformation (FFT) [14]. This method is not only commonly used in audio measurement but is crucial to numerous computations in audio feature engineering.
The FFT furthermore can be utilized to represent the change of signal's frequency content over time: Therefore, FFT is consecutively computed on shifting and overlapping chunks (windowed segments) of the signal. Each FFT result for the chunks corresponds to one vertical line in the latter image and are visualized side-by-side. Usually, the levels of the FFT output are then represented by a heat map [15]. In order to reflect human auditory perception, the values (levels) are usually logarithmized with the decadic logarithm. This allows to visualize a huge range of digital values. It is also common to display spectrograms - or any other representation of audio frequencies over time - with logarithmized frequency axis to match human perception of frequencies in intervals, i.e. octaves in particular. But even if they are not used in plots, the 2-dimensional array representation of spectrograms can be used for multiple applications and further calculations.
Two examples of spectrogram plots of the spoken digits 0 and 8 can be examined in Figure 1. Notice the difference in energy distribution over the length of the spoken words as well as how the center frequency (bright yellow/orange) differs for both digits. The tone center of digit 8 seems to be an octave lower compared to digit 0.
Additionally, there is another technique — chirplet transformation [16] — to generate spectrograms and therefore change their representation of non-stationary signals [17]. In audio classification, it has recently been applied to data sets containing bird calls [18, 19].

2.2 Singular Value Decomposition

Philipp Schwarz

In the past, Singular Value Decomposition (SVD) has emerged as an efficient way to process images in different ways. It helps reducing the information contained in an image — in multiple linear independent components — reducing the needed dimensions for processing.
A singular value decomposition of a $m \times n$ matrix M of rank r is the matrix factorization

$$M = U \Sigma V^* \tag{1}$$

where U is an $m \times m$ complex unitary matrix, Σ is an $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, V is an $n \times n$ complex unitary matrix, and V^* is the conjugate transpose of V [21, 22]. The Matrix

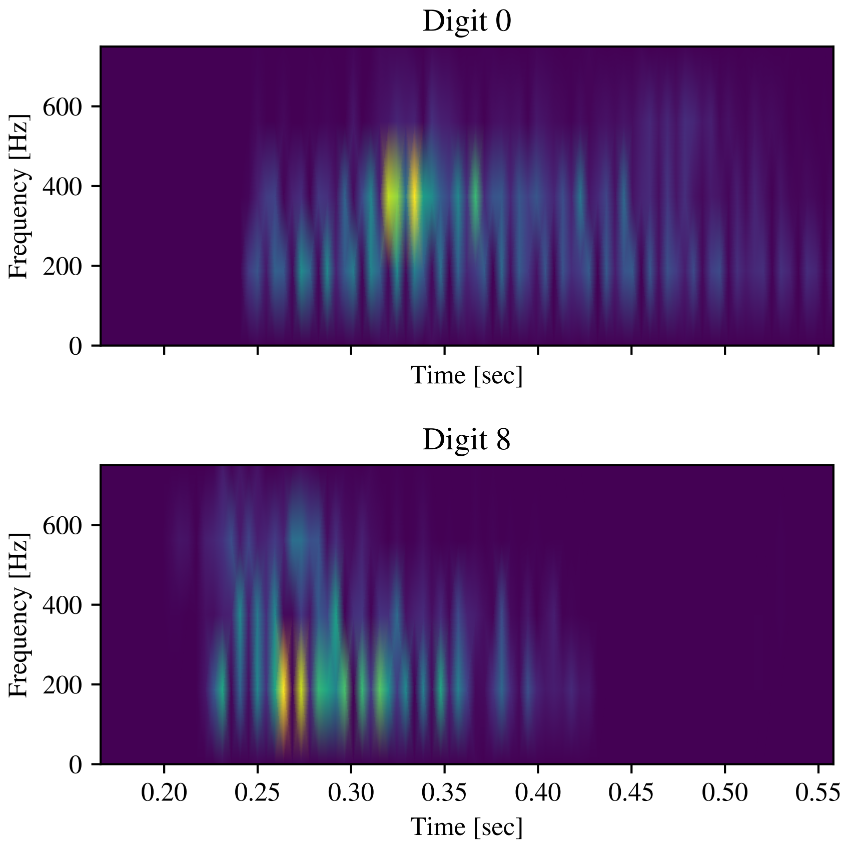


Figure 1: Plots of spectrograms of single recordings of the spoken digits 0 and 9 (Speaker 0 from AudioMNIST data set). Y-axis is set to an upper limit of 1 kHz. All axes are shared in both plots. Levels are represented by color. The colormap ranges from dark blue (low levels) to bright yellow (high levels). The plots have been generated by creating a pseudocolor plot of a 2D array using quadrilaterals with Matplotlib's pcolormesh in Python [20].

U consists of so called left singular vectors as column vectors u_i, whereas the matrix V contains the right singular column vectors v_i and Σ the singular values σ_i. Therefore, the SVD factorization of M in Equation 1 is equal to

$$M = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{2}$$

Used extensively in research, the principal component analysis (PCA) utilizes this behavior by computing the dominant vectors present in a given data set since principal components, are described by the orthogonal matrix U [23, 24]. In image recognition the method of using *Eigenfaces* has been prominently established by Turk and Pentland [10] for the use in face recognition. Hence, an image can be described by a linear combination of a number of Eigenfaces, a set of features characterizing the variation between images. This procedure creates a training matrix, containing the variance of the training images to the average of all training images in a column vector. Defining the average image by

$$\bar{x} = \frac{1}{n} \sum_i x_i \tag{3}$$

where x_i is the i-th image and n is the number of images. The image variance then is calculated by $x_i - \bar{x}$. Applying the aforementioned principal component analysis the Eigenfaces correspond to the Eigenvectors sorted by their ascending Eigenvalues.
A data point can then be classified (or assigned to a principal component for visualization) by computing the residuals for the test point and Eigenfaces of classes of interest in a least squares algorithm.

3 Method

3.1 Study Design

Fabian Rosenthal

This study will answer the research question whether a simple SVD algorithm can also work as a classifier of spectrogram data as a representation of audio material. The algorithm we are planning to adapt originally was introduced to classify handwritten digits [21, 25, 26]. For this reason we chose classification of spoken digits as a evaluation problem for our algorithm. So an unseen spectrogram test sample should be classified as the correct digit class of the spoken digit it was generated on. Therefore, we make the assumption, that variation between digits (digit classes) is measurable higher than variation between speakers and recording takes for the same digit. In regards of the spectrogram representation we assume that the spectrograms of multiple samples for the same digit class are similar and contain recognizable patterns. A high classification accuracy on spectrograms would mean that this similarity is more important or dominant than variation due to speaker characteristics like gender, accent etc.
The basic underlying idea of this study is to generate spectrogram data from an evaluation data set, shuffle them and split them into portions for training and testing. The training data was stacked class-wise and each digit class was subjected to a Singular Value Decomposition separately to model the variation in each class using orthogonal basis vectors. This basically results in a least squares algorithm on a reduced ranks model [21]. The first left singular vector u should represent the most dominant spectrogram representation of the class, the following vectors u_i then represent the most important variation of the data of this class. A test sample can be predicted by computing residuals for each basis of the trained classes — in our case the digit classes 0–9 — and consecutively by choosing the lowest residual value of all tested bases.
The beauty of this algorithm definitely lies in it's simplicity: In it's core it is a very common algebraic operation[1] which can even be interpreted geometrically. Also the classification criterion is quite clear and comprehensible. Furthermore, there is a interesting possibility of tuning the model by selecting the number of

[1]Which means that the actual classification can completely be done with standard functions from Scipy's linear algebra package (scipy.linalg) in Python and does not require installing a huge machine learning library.

Eigenvectors to use in residual calculation.
The overall accuracy of the classifier can be evaluated by calculating the rate of misclassifications which will be called error-rate. Also a plot of a confusion matrix will give great insight how the algorithm performs on different classes.

3.2 Data Set

Fabian Rosenthal

The study was conducted on the AudioMNIST[2] data set. This is a large collection of spoken digits (0-9) in English consisting of 30,000 audio recordings. The total duration of the data is ca. 9.5 hours. 60 different speakers recorded 50 repetitions per digit.
The mean speaker age is 28 years (SD = 6 years). In regards of gender 48 (80 %) speakers have been male, 12 (20 %) female. Furthermore, it is interesting to have a look at the spoken accents: 95 % of the speakers are not native English speakers. The data set contains a huge variety of accents but a great majority (67 %) of the recorded speakers had a German accent. A list of spoken accents can be found in Appendix.
The authors of the data set choosed quiet rooms for the recordings (e.g. VR-rooms, cinemas) and every recording was made with the same recording equipment.
The algorithm was trained on 22,500 (75 %) of the audio recordings. A quarter (7,500 recordings) had been reserved as an unseen test data set.

3.3 SVD Algorithm

Philipp Schwarz

For classification, this work uses the *Eigenfaces* approach proposed by Turk and Pentland [10]. Their method was specialized to recognize faces in images, but it can be used for various applications in the area of image recognition. A handwritten digit recognition algorithm utilizing this approach is reported by Elden [21]. Since the program presented in this work uses image recognition on audio spectograms, the use of this approach promises good results.
A set of training images, flattened to a 1D vector representation, are stacked to a matrix. This matrix only describes one feature class, so for multiple classes, i.e. various digits, multiple of these matrices with their own associated training sets are needed.
To reduce the data needed in the classification and to boost efficiency, the Eigenvectors of these matrices are used, which are obtained by a Singular Value Decomposition. These Eigenvectors span a multidimensional space, where every spectogram image can be represented. Turk and Pentland, who used this method prominently in their face recognition system, gave these vectors the fitting name 'Eigenfaces' and for the vector space 'face space' [10]. In our case we could call them 'Eigenspectograms' and 'spectral space'.

The classification itself can therefore be described as a least squares problem minimizing the residuals derived from the test sample and the Eigenfaces of the feature classes.

[2]Published at: https://github.com/soerenab/AudioMNIST

Elden [21] breaks Turk and Pentland's approach [10] down as a least squares problem on a reduced rank model:

$$\min_{\alpha_1} \|z - \sum_{i=1}^{k} \alpha_i u_i\| \tag{4}$$

> where z represents an unknown digit and u_i represents the singular images. We can write this problem in the form
>
> $$\min_{\alpha} \|z - U_k \alpha\|_2 \tag{5}$$
>
> where $U_k = (u_1, u_2, \dots u_k)$ since the columns of U_k are orthogonal, the solution of this problem is given by $\alpha = U_k^T z$, and the norm of the residual vector of the least squares problems is
>
> $$\|(I - U_k U_k^T)z\|_2 \tag{6}$$
>
> i.e., the norm of the projection of the unknown digit onto the subspace orthogonal to $span(U_k)$.

Moreover, Elden excludes the use of the average image in this process. This means that the columns in the final matrix consist only of the training images themselves.

The amount of Eigenvectors used in the calculation of the residuals determines how exactly the test samples have to match the numbers from the test set, since every spectogram can be represented as a linear combination of these Eigenvectors.
To be able to use this behavior to calibrate the algorithm, the amount of Eigenvectors can be freely adjusted using the k-parameter. If the value of k is too small, images with too much variance will be recognized. However, if k is set too high, numbers, 'that are not good enough' will be rejected. In a consequence, an optimal value for k has to be determined by trial and error.

3.4 Implementation

Philipp Schwarz

The aforementioned algorithm was implemented in Python. While a simplified overview of the program is given in Figure 2, a more comprehensive and detailed chart is presented in Figure 11.
Firstly, data is read from the file system and the spectogram for every audio sample is calculated using the spectogram function from the *scipy-signal* package. Since the human voice does not show harmonics above 4 kHz while speaking [27], we used the spectrograms to only 6 kHz. This reduces memory demands and calculation times drastically.
For storage purposes, every sample is added to a dictionary object containing the spectogram, shape and meta data (i.e. file name, digit class, speaker id), which are extracted from the file name.
In order to process the spectograms in the following steps, they need to have the same dimensions (lengths). In reality, the duration of the audio recordings varies. Therefore, they are zero padded to the duration of the longest

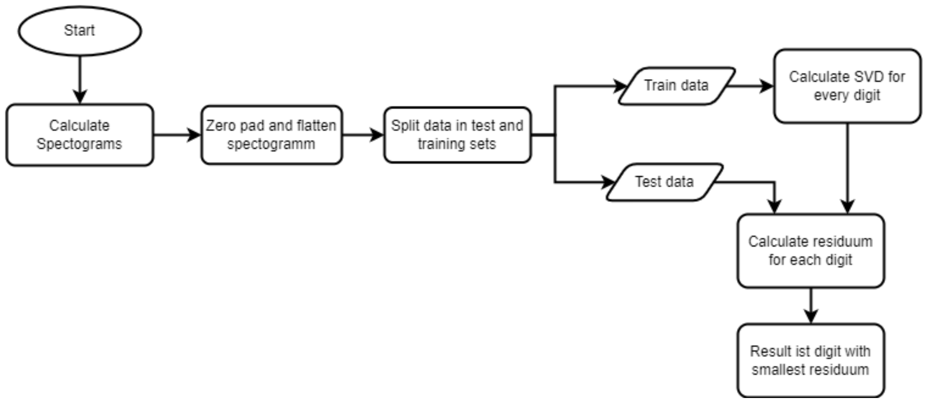

Start
Calculate Spectograms
Zero pad and flatten spectogramm
Split data in test and training sets
Train data
Test data
Calculate SVD for every digit
Calculate residuum for each digit
Result ist digit with smallest residuum

Figure 2: Simplified process chain of the digit classification algorithm.

spectogram. This longest duration is determined during the spectogram calculations of the first step. The resulting spectograms and corresponding meta data are then saved on the disk, to be reused in future calculations to reduce the effort needed.

It is possible to filter the sample set to test only certain digits or speakers. To provide this functionality the data frame structure from the *pandas* package is used and filtered accordingly. This filter function returns the indices of the selected data entries; without filtering all indices are used in the following steps. Only the indices of the dictionary object are used in the following steps to avoid memory overhead in function calls and data handling. Using this set of indices the samples are then split into training and test subsets using the functionality from the *sklearn* package. Due to the long computation durations we did not have the resources to run multiple full tests with the whole data set. Therefore, we set a random seed to shuffle the data set before splitting. Resulting error rates are given without tolerances for that reason.

To prepare the training data for SVD, all single training spectrogram arrays are flattened (from 2D to 1D shape) in order to be sorted by their digit class and column-stacked in associated *numpy* arrays for each class.
Consecutively, each array containing all training samples for each digit class is passed to SVD, using the *linalg.svd* function from *numpy.* The resulting list of SVD arrays is saved to file for future testing/classification.

After the training data is either generated or read from disk, classification will be performed on either all of the assigned samples or just a certain number of test samples (reduced by user). For every single spectrogram under test, the classification algorithm iterates over every digit class (i.e. 0–9) and uses a user specified number of Eigenvectors of the U matrix from the SVD corresponding to the digit class. Using these Eigenvectors, the residual is calculated by vector norm computation as described in subsection 3.3. The digit class which resulted in the lowest residuum of the ten iterations is then returned as the estimated (i.e. predicted) digit class for the sample under test.
To validate the classification, the estimated digits are compared to their actual

digit classes, which had been stored in the meta data dictionary. Key measures for classification are error rate and accuracy: That is the number of incorrect classified samples over total number of test samples as well as correct classified samples over total number of test samples, respectively. Finally, the result and all calculated residuals are written to disk.
Performance and efficiency of the implementation are measured and validated by timing a single classification function call (i.e. classification of a single test sample). In addition to that, multiple data evaluations are made from the test results after test is completed.

```
├── max_shape: int
├── cutoff_value: int
├── cutoff_index: int
└── specs: dict
    ├── 0
    │   ├── spec: numpy array
    │   ├── digit: int
    │   ├── speaker: int
    │   ├── shape: int (only
    │   │   time dimension)
    │   └── name: string
    └── ...
```

Figure 3: Schematic structure of the Python dictionary used for storing spectograms and meta data.

```
├── samples : int
├── error_rate : float
├── used_EV : int
└── results : dict
    ├── 0
    │   ├── estimated: int
    │   ├── actual: int
    │   ├── correct: bool
    │   ├── error rate: float
    │   ├── duration: float
    │   └── residuals: array
    │       of floats
    └── ...
```

Figure 4: Schematic structure of the Python dictionary, used for storing the test results after classification.

3.5 Data Visualization

Fabian Rosenthal

We took Elden's idea [21] of residual line plots over tested bases for every actual digit class but transformed the data to visualize the results of the least squares problems with boxplots. In general boxplots are a powerful tool in visualization of statistical data and will show multiple statistical information at once. They include the median and quartile values, interquartile ranges and the overall data range. This makes boxplots far superior in regards of information density to Elden's proposal. In addition, it has to be pointed out that they are not as confusing or misleading as Elden's line plots: The residual line plots seem to suggest that neighboring digit bases are somehow semantically connected or similar since the data points (the residual values in iterations over bases) are connected with lines. But this is clearly neither the case in handwritten digits nor in spoken digit spectrograms since both sets are not sorted by any similarity measure.

4 Results

Fabian Rosenthal

On 7,500 unseen test samples of the AudioMNIST data set the best general classification result is an error rate of 0.35 (Accuracy 65 %). It is achieved by using the first 1,500 Eigenvectors after tuning this parameter iteratively and comparing error rates.
Of course this result is far from ideal: In machine learning accuracies deviating from the high 90s (%) may be counted as failures. But this result also clearly shows that the algorithm utilizing the proposed SVD approach is indeed successfully working. Apparently it performs better than guess probability. The following subsection will consecutively inspect this result in depth and give inside to convergence measures.

4.1 Confusion Matrix Analysis

Fabian Rosenthal

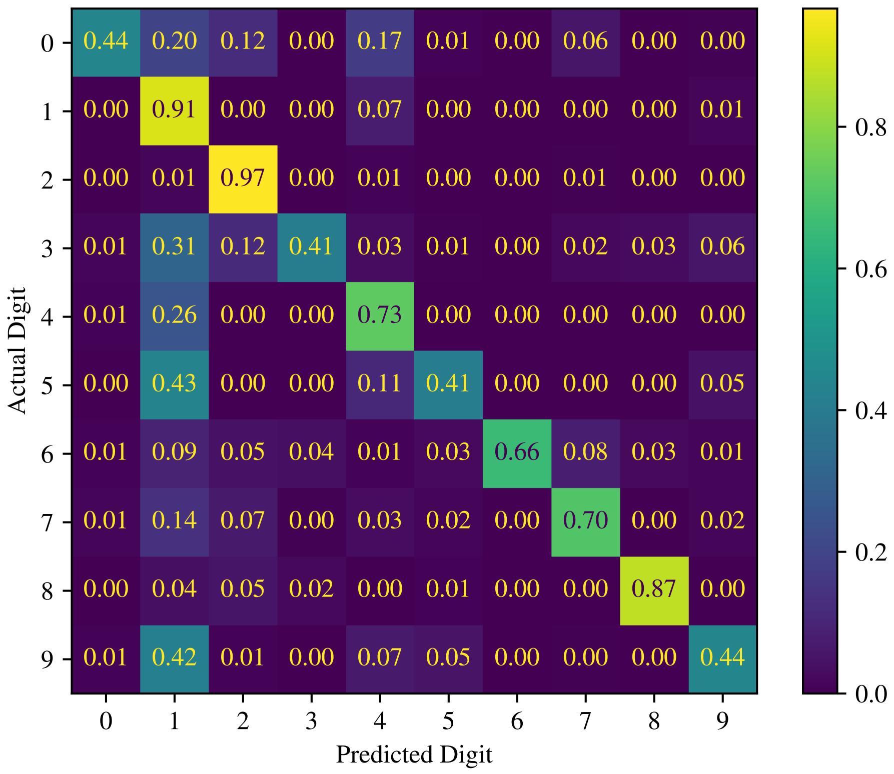
Besides the overall error rate, a confusion matrix is a great tool to inspect classification results and understand the classifier's performance on each class, as well as underlying misclassification problems.
For the test classification of 7,500 samples the best accuracy was determined for classes of digit two (0.97) and digit one (0.91). The worst classification accuracies are seen in digit classes three and five (both 0.41) as well as zero and nine (both 0.44). A deeper inspection of the confusion matrix shows, that the classifier seems to confuse digits of any class with class one too often. For the actual class five test samples have been misclassified as class one (0.43) more often than being classified correctly. This is similar in actual class 9 which is misclassified with class one almost as often (0.41).

4.2 Boxplot Analysis

Fabian Rosenthal

The residual line plots in Figure 12 can give quite an intuitive insight into the underlying algorithm in classification and it's problems, when results are somewhat clear. A even better understanding of the classification outcome can be obtained by examining the boxplots of the calculated residuals of tested SVD-bases for every actual class (See example in Figure 6. The set of boxplots for every class can be found in Figure 13).
Those plots can give important insight into whether a classification was made with larger error tolerance or if the least squares problem was solved with a small margin in a statistical sense. Furthermore, there are three types of cases in the residual boxplots we want to point out:

- The median residual for one single tested base deviates strongly negative from the median values of all other tested digit classes whilst corresponding to the correct (actual) digit class (e.g. Figure 6 or plot for actual digit class 2 in Figure 13). Furthermore, in this particular example of Actual Digit 2 the boxplot shows, that the median residual value for the correct digit class was lower than any other class's lower quartile residual value. This



	0	1	2	3	4	5	6	7	8	9
0	0.44	0.20	0.12	0.00	0.17	0.01	0.00	0.06	0.00	0.00
1	0.00	0.91	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.01
2	0.00	0.01	0.97	0.00	0.01	0.00	0.00	0.01	0.00	0.00
3	0.01	0.31	0.12	0.41	0.03	0.01	0.00	0.02	0.03	0.06
4	0.01	0.26	0.00	0.00	0.73	0.00	0.00	0.00	0.00	0.00
5	0.00	0.43	0.00	0.00	0.11	0.41	0.00	0.00	0.00	0.05
6	0.01	0.09	0.05	0.04	0.01	0.03	0.66	0.08	0.03	0.01
7	0.01	0.14	0.07	0.00	0.03	0.02	0.00	0.70	0.00	0.02
8	0.00	0.04	0.05	0.02	0.00	0.01	0.00	0.00	0.87	0.00
9	0.01	0.42	0.01	0.00	0.07	0.05	0.00	0.00	0.00	0.44

Figure 5: Confusion matrix of the AudioMNIST classification results for 7500 test samples and 1500 Eigenvectors used in the least squares calculation. Values have been normalized to 1 for every row (actual digit). This lets the diagonal be read as the error per actual class.

explains statistically why misclassification is rarely seen for this actual digit class.

- The median residuals for some tested bases are significantly higher than most of the classes' median residuals but median residuals of multiple other tested bases do not differ significantly (e.g. actual digit class 0). This explains very rare misclassification for some tested bases but fairly high confusion rates for multiple other tested bases.

- A single base's median residual deviates negatively just as much from the other median values, so that a classification is done correctly. For actual digit class 6 this can be observed for about two thirds of the cases, even, if most of the median residuals do not deviate significantly at all (e.g. actual digit class 6).

As a side note, it might be worth mentioning that Elden's line plots somehow show differing of median values of neighboring bases in reduced line chaos. More focused and parallel lines between two tested bases seem to correspond with differing median values.

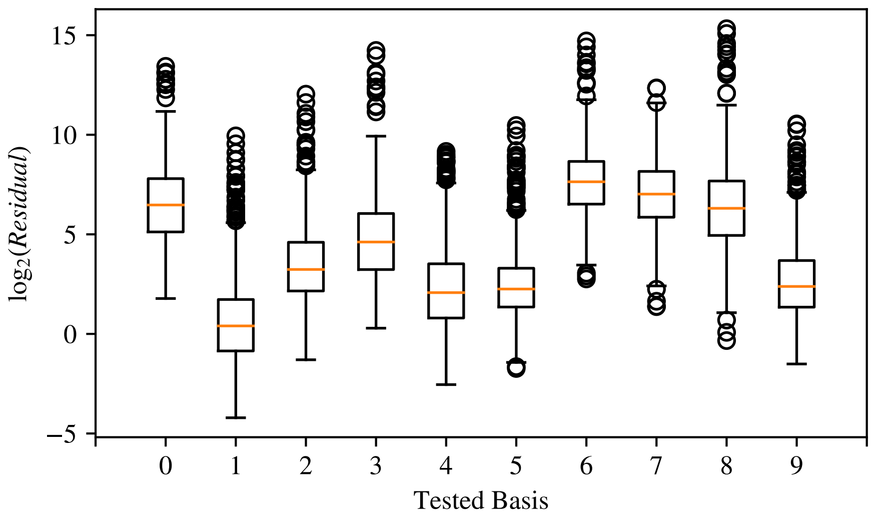


Figure 6: Boxplot of measured residuals over bases tested against actual digital class 1 in digit classification. Y-values are scaled with log_2 to handle the outliers. Orange lines mark the median values. Upper and lower edges of the boxes show the 3rd and 1st quartile, respectively. The whiskers have the length of 1.5 inter-quartile-ranges (IQR) measured from the box. Outliers are visualized with circles and are those data points exceeding the distance of 1.5 IQR from the box.

4.3 Error Analysis

Philipp Schwarz

The following subsection gives insight into behavior of the classifier during the actual classification. An increase of classification accuracy of our proposed algorithm, can be seen by varying the number of used Eigenvectors (k-values). To determine the best value, we iterate over a range of k-values. The results are then plotted as error rates over k-values (see Figure 7). Hereby it becomes clear, that error rates do not deviate significantly from each other, but better classification results can be reported for larger k-values. However, when the number of used Eigenvectors had been set to the maximum possible k-value[3], the error rate jumped to around 0.9.
For this reason, a default value for the amount of Eigenvectors was set to $k = 1500$ as the value resulting in the lowest error rate in our comparison.
Another very important part of testing the algorithm is to check for it's convergence behavior. By plotting error rates over number of tested samples it can be observed, that the overall error rate seems to converge after around 75 samples tested. The accuracy of out digit recognition algorithm converges quite fast after 1 % of the test data. However, but stays however at a high level of around 30%. This may indicate, that the accuracy of the algorithm, is bound by the training methods used or are inherent to the procedure of using spectogram

[3]Maximum possible k-value is 6848, which is the product of maximum spectrogram length and frequency bins.

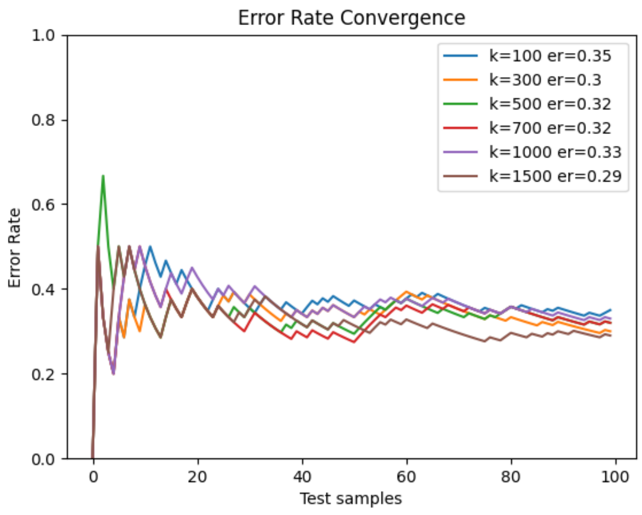


Figure 7: Comparison plot of error rates for multiple number of Eigenvectors used in classification.

images as classification basis.

4.4 Performance

Philipp Schwarz

Since classifications need to be performed fast in a real-time environment and training set can be rather large — in our case 22,500 samples — a good performance is of major importance to the system.
For the training process alone (including the spectogram calculations, spectogram stacking operations and the SVD calculations of 22,500 samples) the run time was around 8–11 Minutes. Considering the amount of training data, this seems to be a sufficient performance of the program. Especially, since the training data only has to be calculated once.
A different image emerges with the classification algorithm: Calling the digit classification function with included residual calculation for all 10 classes takes far too long. Various test runs on multiple systems yield a time per function call of 7–19 seconds (per test sample). For any real-time application this is unbearable. This is also a major disadvantage for large test runs.
Naturally, training and classification run times are highly dependent on the machine they are ran on. Since python uses all the resources it can get, different hardware systems will perform highly divergent (See Figure 10).
In regards of resources, the usage of one data structure to store spectograms need a lot of memory. Comparisons over multiple different test runs show peak memory usages of 3,479–4,066MB.
Overall one could describe the performance of the system as lopsided. With a rather quick training times and a very slow classification algorithm, it represents somehow opposite stats compared to Neural Networks, used often in image classification.

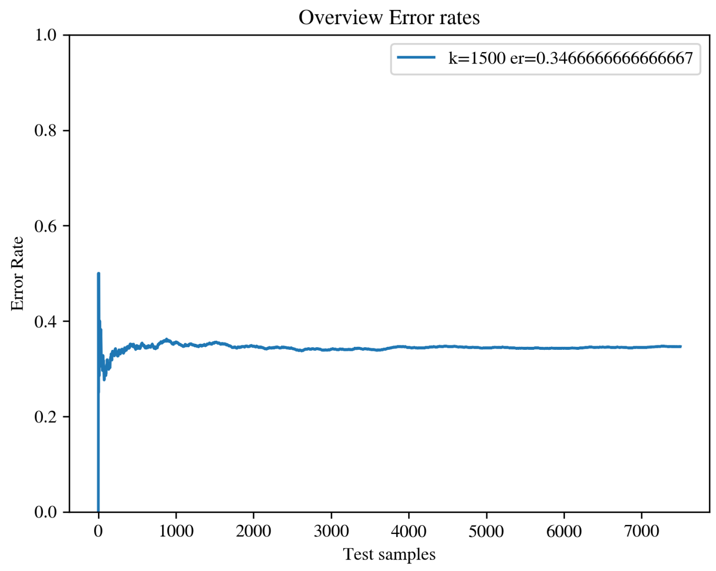


Figure 8: Convergence plot of the overall error rate over number of test samples for 1500 used Eigenvectors.

5 Discussion

Fabian Rosenthal

The results as reported above shall now be put into context. As mentioned before, we are aware that the presented results are not as high as it would be desired in deployment. To conveniently use a classifier e.g. in a digital speech assistant you would need far better accuracy. Gladly, the creators of the AudioMNIST data set also report results for a digit classification utilizing an AudioNet classifier. Even if the authors had not maximization of classification accuracy as their top priority, this is a useful baseline to compare our results to. Furthermore state-of-the-art results can then be found in [28]. A comparison with those results is given in Table 1. One reason for the rather large gap to those benchmark-results could lie in the simple algebraic approach we chose. There might be too many degrees of freedom in a huge data set of spoken digits compared to images of handwritten digits. Especially the reported diversity in speaker accents (almost no native speaker; wide variety and mixture of accents; see Table 3) could have had a diminishing effect on classification results.
Furthermore, the SVD classifier might not work as good on spectrograms as it does on handwritten digits. Spectrograms as representations of a time signal will always have left to right dominant elements such as fundamental frequencies (compare Figure 1), even if the spoken words differ quite significantly from a linguistic standpoint. Therefore, the geometric characteristics of a single digit class may not always be as high or as unique compared those of a handwritten digit class and make classification problematic. This could also explain why we had to use a relatively high number of singular images to reduce error rates ($k = 1500$).

Authors	Classifier/Network	Avg. accuracy
Becker et al. 2018 [6]	AudioNet	95.82 % ± 1.49 %
Tukuljac et al. 2022 [28]	LF-custom	98.0% ± 0.41%
Schwarz & Rosenthal	SVD Least Squares	65.0 %

Table 1: Comparison of classification results in three different studies on AudioMNIST data set (Task: digit classification). This study's results (Schwarz & Rosenthal) are based on a single calculation. Hence, no tolerance can be given here.

5.1 Comparison to Neural Networks

Philipp Schwarz

In comparison with recent methods for number classification, foremost Neural Networks (NN), as used by Park et.al. [8], our approach can not hold up. While NNs need a lot of effort and time to be trained appropriately, our approach can complete the training process with far less resources. But even with this, not unimportant, advantage, the very slow classification process and high error rates show, that modern Neural Network approaches seem more suited for this kind of classification problem.
A typical statement made about NNs is, that they are 'black boxes' and cannot give information about the underlying decision processes. This is something we wanted to address with the SVD approach. But also with this basic approach, We do not actually know what feature or information in the spectrogram exactly made the algorithm classify a sample. We simply know that it chooses the smallest residual value, which is somehow unintuitive from an interpretational point of view. Compared to modern networks with solutions for interpretability as presented by Becker et al. [6] the SVD approach shows almost no advantage.

5.2 Possible Tuning

Fabian Rosenthal

With regards to improvement of the classification result there are a few steps which could be taken in future work:

- Firstly, there are possibilities to further control a spectrogram's time and frequency-resolution: Choosing a higher chunk length of the shifted FFT windows will result in a better frequency resolution. Shorter window lengths will help the overall temporal resolution but as well result in a worse frequency resolution. This relationship of time and frequency resolution of bandlimited signals is known as Küpfmüller's uncertainty principle [29]. As suggested in [4], tuning window length and overlap might result in better classification accuracy. Research shows that for specific classifiers (normalized cross-correlation) even zero overlap can outperform higher overlaps [30]. However, the effects of FFT window length on classification accuracy might vary across data sets and classification tasks [4]. Best parameters could be determined by performing a cross-validated grid search or even nested cross validation on training data [31, 32].

- Secondly, improvement can be made with a feature engineering approach. Maybe the patterns in our spoken digit spectrograms lack similarity or are just not quite dominating enough to result in a high classification accuracy. An interesting starting point could be transformation to Mel spectrum and representation of the audio in Mel spectrograms respectively. The Mel scale was created to offer a better relation to auditory perception of the human ear. The signal is therefore transformed to cepstrum [33]. This technique might support classification improvement with the SVD algorithm as it can be found in many other machine learning contexts [8, 34, 35].

- A third major factor could be the data preprocessing: It is possible that better temporal alignment of the data set will improve the classification error. With the available resources we have neither been able to implement a check for some kind of misalignment nor a complex control of the temporal alignment. We are relying on the author's work of AudioMNIST at this point. However, badly aligned recordings will result in a SVD matrix containing only 'blurred' information and therefore in low classification accuracy. In our case the implementation of a on-set detection for the spoken digits could be beneficial. This, of course, is not important when applying a machine learning model which can recognize patterns regardless of an absolute starting point in the data. This can partly explain the large gap to classification results in [6].

- Fourthly, chirplet transformed spectrograms could be implemented. Research sees advantages in classification accuracy when passing chirplet spectrograms instead of regular short-time FFT spectrograms to convolutional neural networks (CNN) [18]. It remains to be tested if chirplet transformation is beneficial in our task as well.

- Lastly, there could be a potential to optimize our algorithm by implementing a cross validation to calculate the error rates. The data set therefore would be split into k folds (typically in a range of 5–10) instead of training and testing data portions. Single folds are then iteratively assigned as test data whilst the other folds are exchanged to training data sets. Finally, every combination is tested and the final result is the average error for all folds. This might have a marginal influence. Nonetheless, bigger deviations from our results are not likely: Due to the large number of samples and uniform distribution over classes and speakers it is highly likely that AudioMNIST is robust to errors caused by random assignment.

5.3 Conclusion

Fabian Rosenthal

This work provides a novel adaptation of a Singular Value Decomposition (SVD) based approach to pattern recognition in audio classification. We implemented an algorithm known from handwritten digit classification and evaluated it on spoken digit classification with the large AudioMNIST data set (30,000 recordings). Through iterating over different values for the number of Eigenvectors during residual calculation in the least squares problem we found a value of 1,500 to provide the best accuracy. The classification error is reported as 0.35.

However, the error is not uniformly distributed over all digit classes. As analysis of the confusion matrix and residual boxplots for each class have shown, the algorithm performs sufficient on digit 1, 2 and 8 but misclassifies majorities of actual classes 0, 3, 5, and 9.
Furthermore, the computation time for classification of a single sample (7–19 seconds[4]) is far too high to implement this program in a real-time setting such as digital voice assistants.
For this reasons the algorithm has to be further developed to be recommended. This work gives important starting points for revision and tuning including tweaking of spectrogram window length and overlap, transformation to Mel spectrograms, utilizing chirplet transformation and alignment via on-set detection in preprocessing. In addition to that, the original handwritten digit classifier was not forced to decisions, if the smallest residual did not deviate significantly from the other values [21]. Difficult cases would maybe be directed to manual supervision. For our program it could be beneficial to pass 'difficult' samples to another classifier which then complements with a different approach e.g. a convolutional neural network.

NOTE: Section authorship does not necessarily correlate with code authorship. Our program has been developed in strong collaboration.

[4]Depending on system stats (see Figure 10). Classifying a 12-digit telephone number could take up to almost 4 minutes.

References

[1] L. R. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*, ser. Prentice Hall signal processing series. Delhi: Pearson, 1993.

[2] M. Casey, "General sound classification and similarity in mpeg-7," *Organised Sound*, vol. 6, no. 2, pp. 153–164, 2001.

[3] M. S. Medvedev and Y. V. Okuntsev, "Using google tensorflow machine learning library for speech recognition," *Journal of Physics: Conference Series*, vol. 1399, no. 3, p. 033033, 2019. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1399/3/033033/meta

[4] E. C. Knight, S. Poo Hernandez, E. M. Bayne, V. Bulitko, and B. V. Tucker, "Pre-processing spectrogram parameters improve the accuracy of bioacoustic classification using convolutional neural networks," *Bioacoustics*, vol. 29, no. 3, pp. 337–355, 2020.

[5] K. Palanisamy, D. Singhania, and A. Yao, "Rethinking cnn models for audio classification." [Online]. Available: http://arxiv.org/pdf/2007.11154v2

[6] S. Becker, M. Ackermann, S. Lapuschkin, K.-R. Müller, and W. Samek, "Interpreting and explaining deep neural networks for classification of audio signals," *ArXiv*, vol. abs/1807.03418, 2018.

[7] Ehsan Variani, Tom Bagby, Erik McDermott, and Michiel Bacchiani, "End-to-end training of acoustic models for large vocabulary continuous speech recognition with tensorflow," 2017. [Online]. Available: https://research.google/pubs/pub46294/

[8] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Interspeech 2019*. ISCA, sep 2019. [Online]. Available: https://doi.org/10.21437%2Finterspeech.2019-2680

[9] A. Robles-Guerrero, T. Saucedo-Anaya, E. González-Ramírez, and J. I. de La Rosa-Vargas, "Analysis of a multiclass classification problem by lasso logistic regression and singular value decomposition to identify sound patterns in queenless bee colonies," *Computers and Electronics in Agriculture*, vol. 159, pp. 69–74, 2019.

[10] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991. [Online]. Available: https://direct.mit.edu/jocn/article/3/1/71/3025/Eigenfaces-for-Recognition

[11] R. A. Sadek, "Svd based image processing applications: State of the art, contributions and research challenges," *(IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No. 7*, vol. 3, no. 7, 2012. [Online]. Available: https://arxiv.org/pdf/1211.7102

[12] K. S. Kim, J. H. Seo, J. U. Kang, and C. G. Song, "An enhanced algorithm for knee joint sound classification using feature extraction based on time-frequency analysis," *Computer Methods and Programs in Biomedicine*, vol. 94, no. 2, pp. 198–206, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169260709000376

[13] M. J. Pelgrom, *Analog-to-Digital Conversion*, 4th ed., ser. Springer eBook Collection. Cham: Springer International Publishing and Imprint Springer, 2022.

[14] C. Sidney Burrus, *Fast Fourier Transforms.* OpenStax CNX, 2012.

[15] J. O. Smith, *Mathematics of the Discrete Fourier Transform (DFT).* `http://ccrma.stanford.edu/~jos/mdft/`, accessed 09.09.2022, online book, 2007 edition.

[16] S. Mann and S. Haykin, "The chirplet transform: physical considerations," *IEEE Transactions on Signal Processing*, vol. 43, no. 11, pp. 2745–2761, 1995.

[17] B. Palczynska, "Identification of non-stationary magnetic field sources using the matching pursuit method," *Energies*, vol. 10, no. 5, p. 655, 2017. [Online]. Available: https://www.mdpi.com/1996-1073/10/5/655

[18] J.-j. Xie, C.-q. Ding, W.-b. Li, and C.-h. Cai, "Audio-only bird species automated identification method with limited training data based on multi-channel deep convolutional neural networks." [Online]. Available: https://arxiv.org/pdf/1803.01107

[19] F. Zhang, L. Zhang, H. Chen, and J. Xie, "Bird species identification using spectrogram based on multi-channel fusion of dcnns," *Entropy (Basel, Switzerland)*, vol. 23, no. 11, 2021. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/34828205/

[20] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[21] L. Elden, *Matrix Methods in Data Mining and Pattern Recognition: Classification of Handwritten Digits.* Philadelphia: siam, 2007.

[22] W. Dahmen and A. Reusken, *Numerik für Ingenieure und Naturwissenschaftler*, 2nd ed., ser. Springer-Lehrbuch. Berlin and Heidelberg: Springer, 2008.

[23] A. N. Gorban, B. Kégl, D. C. Wunsch, and A. Zinovyev, Eds., *Principal Manifolds for Data Visualization and Dimension Reduction*, ser. Lecture Notes in Computational Science and Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 58. [Online]. Available: http://nbn-resolving.org/urn:nbn:de:bsz:31-epflicht-1623234

[24] M. Ringnér, "What is principal component analysis?" *Nature Biotechnology*, vol. 26, no. 3, pp. 303–304, 2008. [Online]. Available: https://www.nature.com/articles/nbt0308-303

[25] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[27] dpa Microphones, “Facts about speech intelligibility,” 2021. [Online]. Available: https://www.dpamicrophones.com/mic-university/facts-about-speech-intelligibility

[28] H. Peic Tukuljac, B. Ricaud, N. Aspert, and L. Colbois, “Learnable filter-banks for cnn-based audio applications,” *Proceedings of the Northern Lights Deep Learning Workshop*, vol. 3, 2022. [Online]. Available: https://septentrio.uit.no/index.php/nldl/article/view/6279

[29] K. Küpfmüller and G. Kohn, *Theoretische Elektrotechnik und Elektronik: Eine Einführung*, 15th ed., ser. Springer-Lehrbuch. Berlin: Springer, 2000.

[30] J. S. Ulloa, A. Gasc, P. Gaucher, T. Aubin, M. Réjou-Méchain, and J. Sueur, “Screening large audio datasets to determine the time and space distribution of screaming piha birds in a tropical forest,” *Ecological Informatics*, vol. 31, pp. 91–99, 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01346004

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[32] T. Agrawal, *Hyperparameter Optimization Using Scikit-Learn.* Berkeley, CA: Apress, 2021, pp. 31–51. [Online]. Available: https://doi.org/10.1007/978-1-4842-6579-6_2

[33] V. T. Tiwari, “Mfcc and its applications in speaker recognition,” *Int. J. Emerg. Technol.*, vol. 1, 01 2010.

[34] H. Meng, T. Yan, F. Yuan, and H. Wei, “Speech emotion recognition from 3d log-mel spectrograms with deep learning network,” *IEEE Access*, vol. 7, pp. 125 868–125 881, 2019.

[35] C.-H. H. Yang, J. Qi, S. Y.-C. Chen, P.-Y. Chen, S. M. Siniscalchi, X. Ma, and C.-H. Lee, “Decentralizing feature extraction with quantum convolutional neural network for automatic speech recognition,” in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 6523–6527.

A Appendix

A.1 Getting Started

Fabian Rosenthal

For testing purposes we provide a small subset of AudioMNIST. It has been randomly created with dev_dataset.py and represents a minimal working example with 30 audio files in each digit class (300 total). It is important to point out that the classification error on this data set is not representative. Nonetheless it is a useful way to test the main program without downloading the full data set. In addition, the main program will only classify 5 test samples by default to limit computation time. So testing will most likely stop before converging to the lowest possible error rate. This program was created with Python 3.10.4.

1. To get started, unzip the files in *AMA-rosenthal-schwarz.zip* to a local directory (e.g. *number_classifier*).

2. This step only has to be taken, if you would like to perform a complete classification run. Otherwise continue with item 3. To perform a complete classification, download the AudioMNIST data set by running the prompt

   ```
   git clone https://github.com/soerenab/AudioMNIST.git .
   ```

   from command line or download the zip file from *here* manually. Afterwards, copy the *data* folder to your local *number_classifier* directory.

3. Please make sure that your installed Python version and packages match the requirements in *requirements.txt*. We do not use any 'exotic' packages though.

4. Consecutively, you can run the main program (number_classifier.py) for testing or grading purposes from command line. `cd` to the unzipped directory and run a prompt as shown in Figure 9:

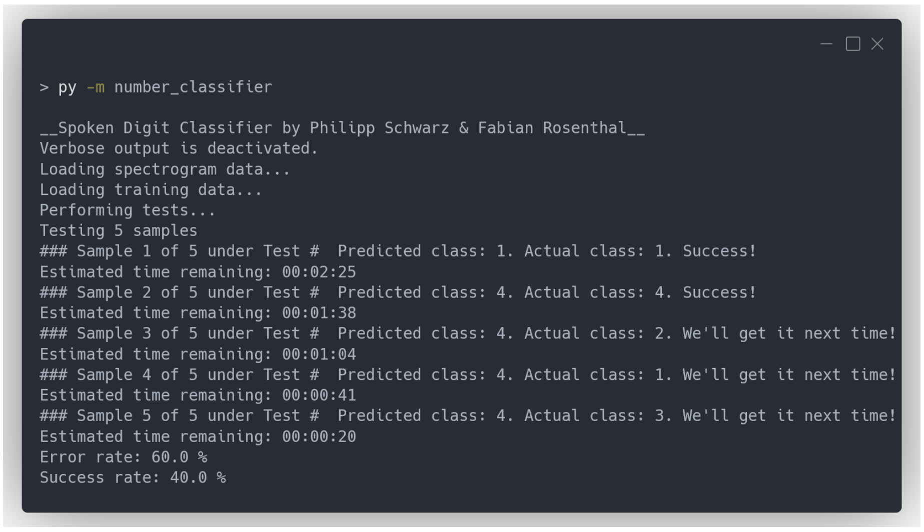
   ```
   py -m number_classifier
   ```

   Of course you can also run the file in any Python IDE of your liking.

5. Additional plots can be generated by running the following prompt. They will be saved as pdf-files to a plots-subdirectory.

   ```
   py -m util
   ```

An overview of all files provided with this report is given in Table 2.

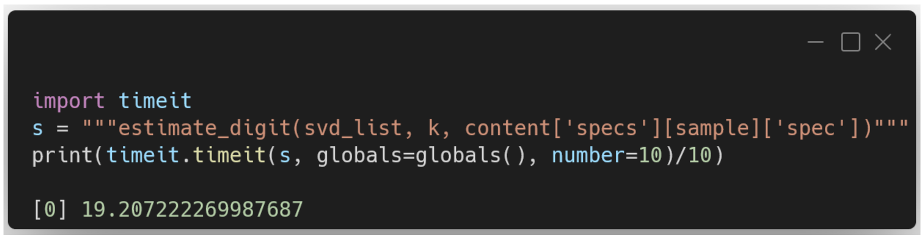


```
> py -m number_classifier

__Spoken Digit Classifier by Philipp Schwarz & Fabian Rosenthal__
Verbose output is deactivated.
Loading spectrogram data...
Loading training data...
Performing tests...
Testing 5 samples
### Sample 1 of 5 under Test #  Predicted class: 1. Actual class: 1. Success!
Estimated time remaining: 00:02:25
### Sample 2 of 5 under Test #  Predicted class: 4. Actual class: 4. Success!
Estimated time remaining: 00:01:38
### Sample 3 of 5 under Test #  Predicted class: 4. Actual class: 2. We'll get it next time!
Estimated time remaining: 00:01:04
### Sample 4 of 5 under Test #  Predicted class: 4. Actual class: 1. We'll get it next time!
Estimated time remaining: 00:00:41
### Sample 5 of 5 under Test #  Predicted class: 4. Actual class: 3. We'll get it next time!
Estimated time remaining: 00:00:20
Error rate: 60.0 %
Success rate: 40.0 %
```

Figure 9: Command line (here: Windows PowerShell) prompt to test-run *number_classifier.py* on reduced data set and output of the classifier. *Fabian Rosenthal*

A.2 Additional tables and plots



```
import timeit
s = """estimate_digit(svd_list, k, content['specs'][sample]['spec'])"""
print(timeit.timeit(s, globals=globals(), number=10)/10)

[0] 19.207222269987687
```

Figure 10: Using the timeit module from Python's Standard Library to time the running time of a single digit prediction. Mean running time per iteration is 19.2 seconds (System stats: Intel(R) Core(TM) i7-4600U CPU @ 2.10GHz (4 CPUs), 2.7GHz; Memory: 8,192MB RAM). On a high performance simulation computer the time per call came down to 7 seconds (System stats: Inten(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz (28 CPUs), 2.60GHz; Memory: 131,072MB RAM). *Fabian Rosenthal*

Table 2: List of files in *AMA-rosenthal-schwarz.zip* their content and authors (Author codes: F. R. := *Fabian Rosenthal*; P. S. := *Philipp Schwarz*).

Name	Content	Author
dev_data	Folder of data subset	-
descriptive_stats.py	Generates data set statistics	F. R.
dev_dataset.py	Creates data subset	F. R.
number_classifier.py	Main program	Both
number_m[...]1500.json	Results data from full run	Both
README.md	Getting started info	F. R.
requirements.txt	Packages required	P. S.
speakerdata.json	Meta data for AudioMNIST	-
spectrograms.py	Generates and stacks spectrograms	F. R.
tester.py	Profiler to test program	P. S.
training.py	Training portion of program	P. S.
util.py	Utility and plot functions	Both

Table 3: Table of different spoken accents recorded in AudioMNIST data set and their percentage of occurence. The values have been rounded to whole numbers (2 % correspond to one single speaker in the AudioMNIST data set with one specific accent). *Fabian Rosenthal*

Accent Description	Percentage
German	68 %
Chinese	5 %
Spanish	3 %
Italian	3 %
German/Spanish	2 %
South Korean	2 %
Madras	2 %
Levant	2 %
English	2 %
Brasilian	2 %
Egyptian/American	2 %
South African	2 %
Arabic	2 %
Danish	2 %
French	2 %
Tamil	2 %

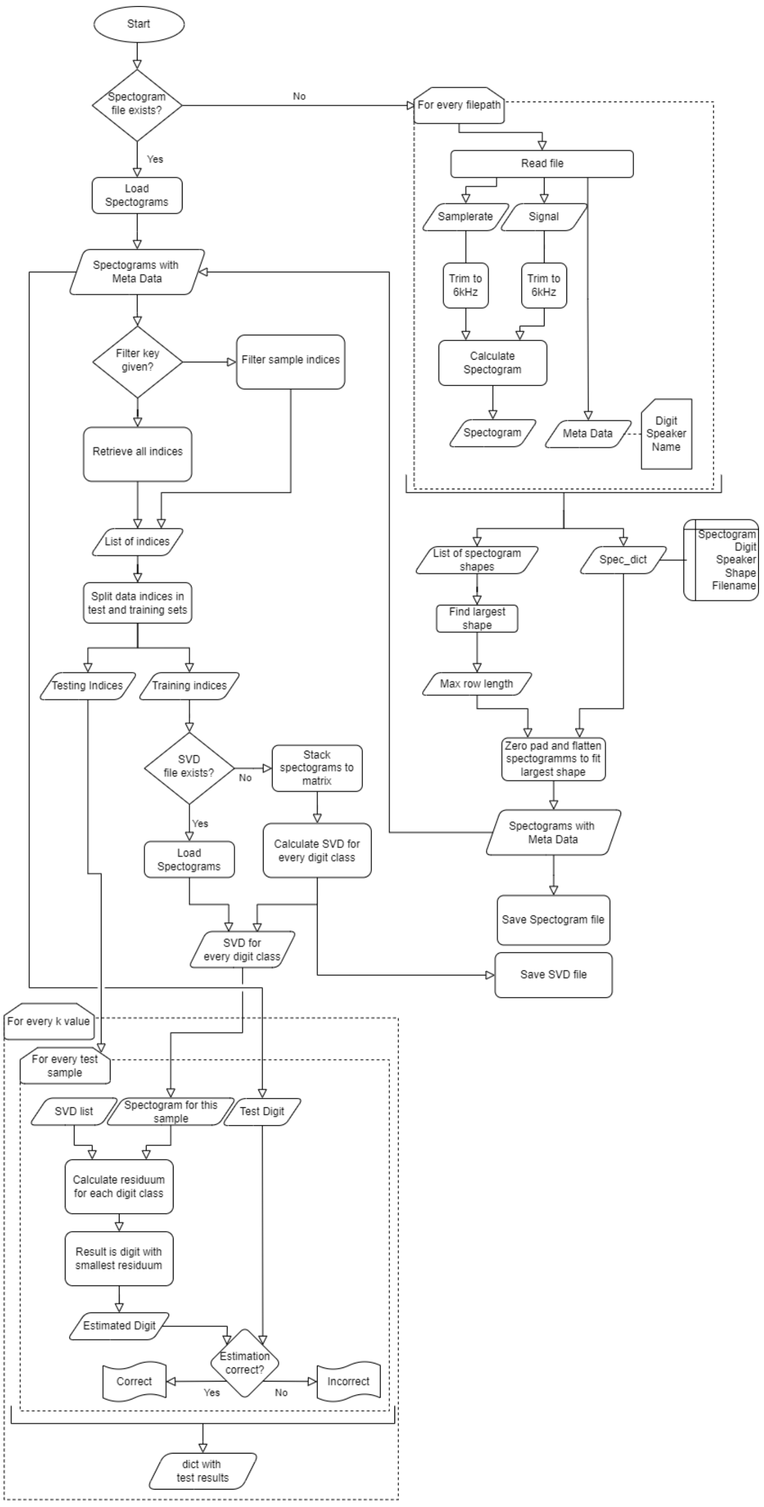


Figure 11: Complete process chain of the implementation of the training and classification algorithm. *Philipp Schwarz*

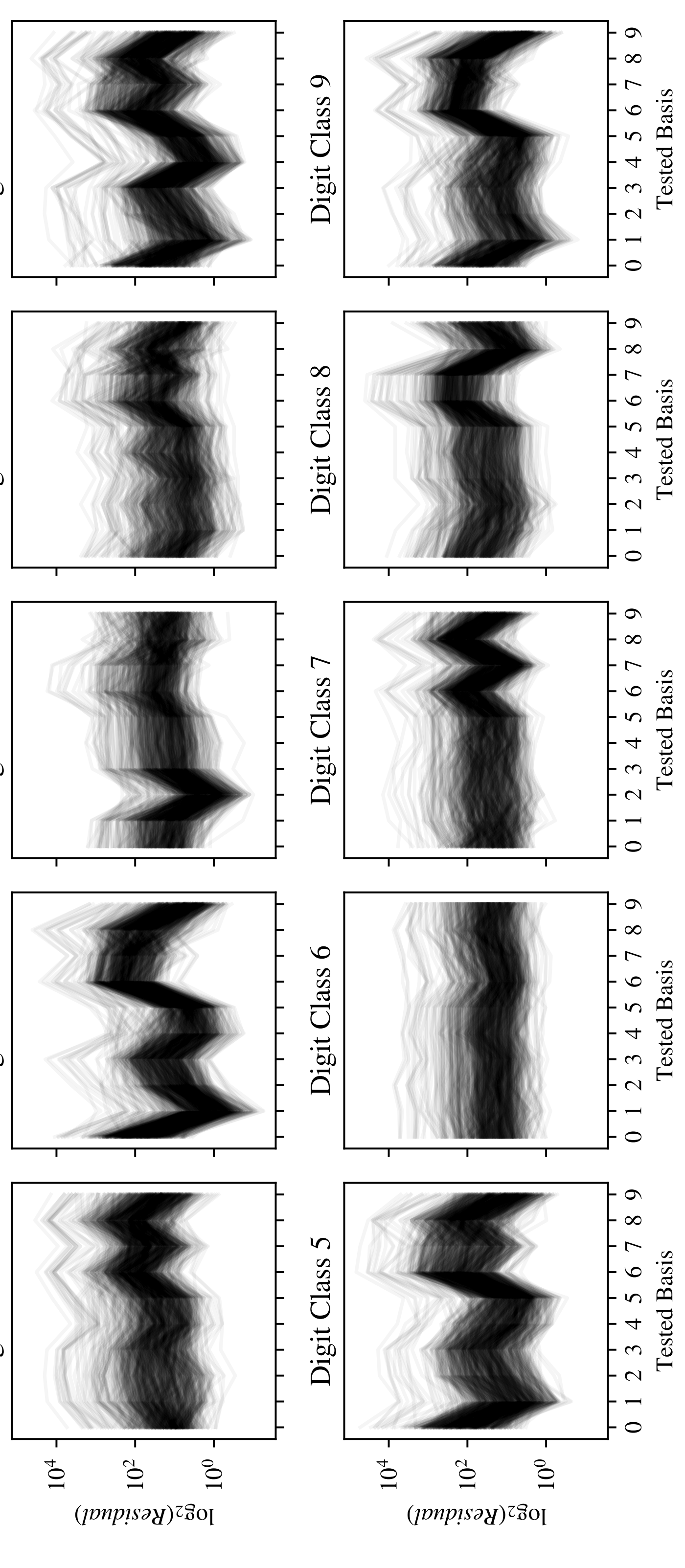


Figure 12: Plots of measured residuals over bases tested against one actual class in digit classification. One graph per actual digit class. Y-values are scaled with log_2 to handle the ranges. Axes are shared. A graphical α was set to 0.04 to support the visualization of line frequencies. *Fabian Rosenthal*

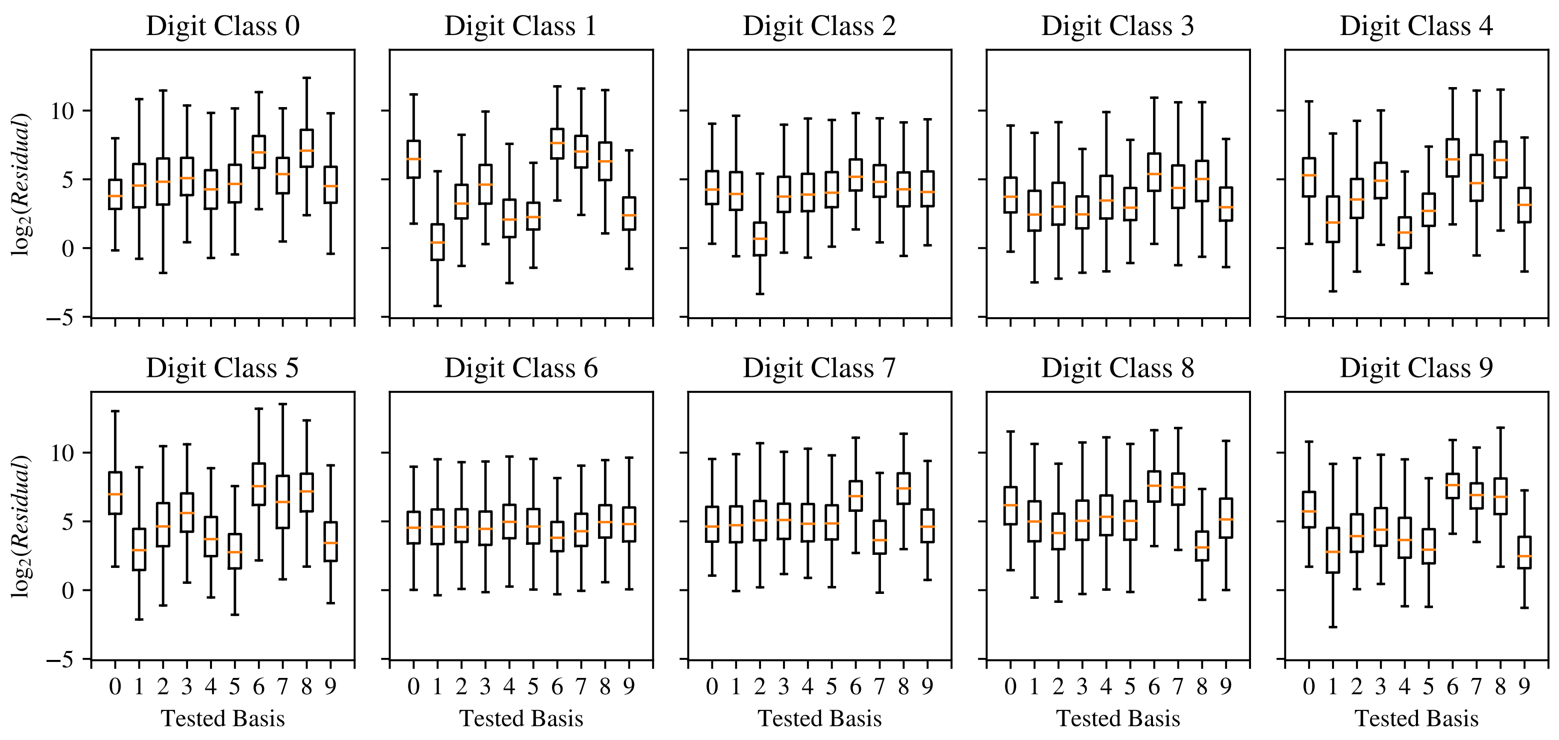


Figure 13: Boxplots of measured residuals over bases tested against one actual class in digit classification. One graph per actual digit class. Y-values are scaled with log_2 to handle the ranges. Axes are shared. Orange lines mark the median values. Upper and lower edges of the boxes show the 3rd and 1st quartile, respectively. The whiskers have the length of 1.5 inter-quartile-ranges (IQR) measured from the box. Outliers are hidden due to readability. *Fabian Rosenthal*