

# Noise clustering

## **THE PROBLEM:**

- Voice interfaces become more common for day-to-day communication. Users tend to use these interfaces on the go in very noisy environments.
- There is no software-level audio noise cancellation technique that was proven to work in real-time applications.
- Common intersections between DSP tasks and machine learning algorithms go through deep learning and require long and costly learning time.

The future belongs to  
algorithms that just know.

## **GOALS:**

- To find a way to reduce noise from speech (voice) recordings
- To cluster an audio signal into two groups: 'voice' and 'noise'  
(to find a model, not to predict)
- To apply machine learning techniques and algorithms for signal processing tasks
- To find a method that could be applied in real-time applications  
('production-ready')

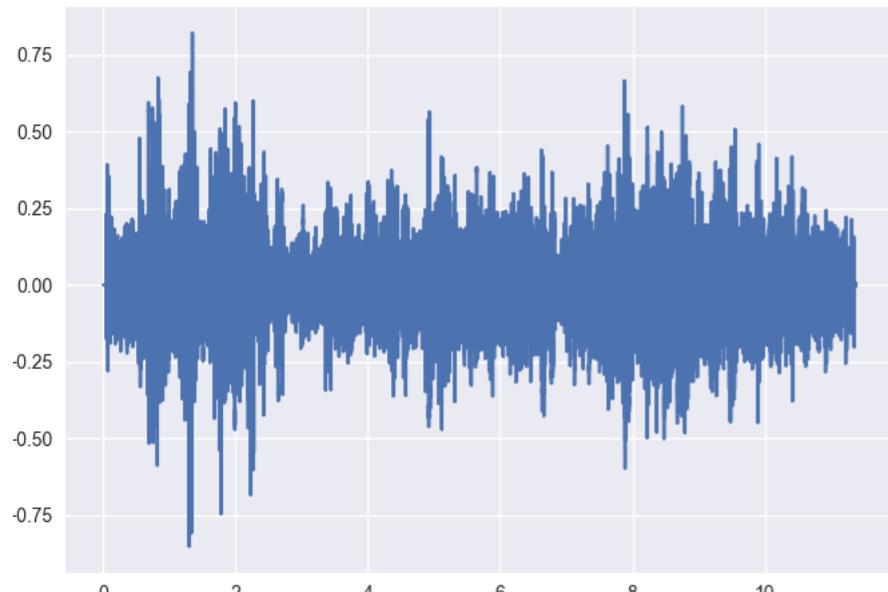
## **POTENTIAL APPLICATIONS:**

- Fast noise reduction on mobile voice messages
- Better speech-to-text capabilities, that wouldn't require deep learning
- Audio signal manipulations (voice filters, fun)

# Methods

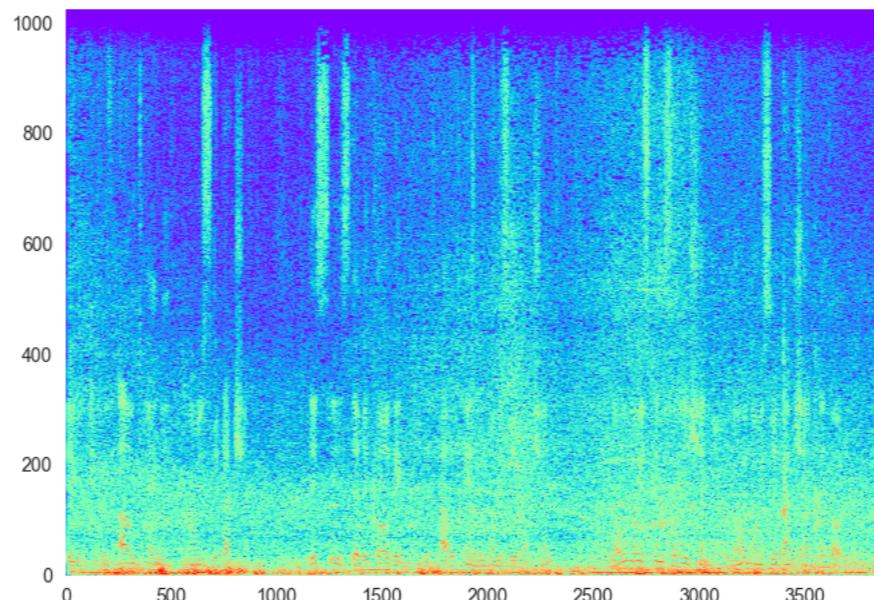
# DATASET: AUDIO SIGNAL AS A DATASET

Audio time series



```
array([ 0.0000000e+00, 0.0000000e+00,  
0.0000000e+00, ..., -9.15527344e-05, -6.10351562e-05,  
-9.15527344e-05], dtype=float32)
```

Spectrogram



```
array([[ 2.03640684e-05 -0.00000000e+00j,  
5.99053465e-02 -0.00000000e+00j,  
4.25433785e-01 -0.00000000e+00j, ... ,  
3.46554875e-01 -0.00000000e+00j,  
1.12801301e+00 -0.00000000e+00j,  
-2.69750923e-01 -0.00000000e+00j],  
...,  
[ -1.78232608e-06 -0.00000000e+00j,  
-7.45655780e-05 -0.00000000e+00j,  
-4.86985336e-05 -0.00000000e+00j, ... ,  
1.32114466e-04 -0.00000000e+00j,  
-9.93174908e-05 -0.00000000e+00j,  
5.87694813e-04 -0.00000000e+00j]],  
dtype=complex64)
```

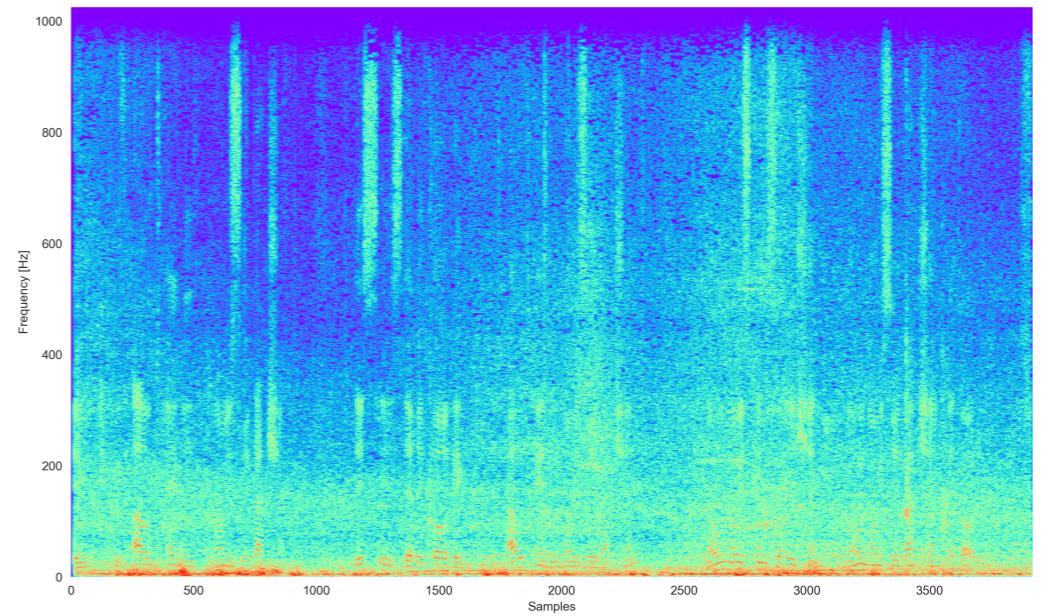
3D table

3.000000000000000e+00	0.000000000000000e+00	0.000000000000000e+00
4.000000000000000e+00	0.000000000000000e+00	1.000000000000000e+00
5.000000000000000e+00	0.000000000000000e+00	-1.000000000000000e+00
6.000000000000000e+00	0.000000000000000e+00	-2.000000000000000e+00
7.000000000000000e+00	0.000000000000000e+00	0.000000000000000e+00
8.000000000000000e+00	0.000000000000000e+00	0.000000000000000e+00
9.000000000000000e+00	0.000000000000000e+00	0.000000000000000e+00
1.000000000000000e+01	0.000000000000000e+00	0.000000000000000e+00
1.100000000000000e+01	0.000000000000000e+00	0.000000000000000e+00
1.200000000000000e+01	0.000000000000000e+00	0.000000000000000e+00
1.300000000000000e+01	0.000000000000000e+00	0.000000000000000e+00
1.400000000000000e+01	0.000000000000000e+00	0.000000000000000e+00
1.500000000000000e+01	0.000000000000000e+00	0.000000000000000e+00
1.600000000000000e+01	0.000000000000000e+00	0.000000000000000e+00

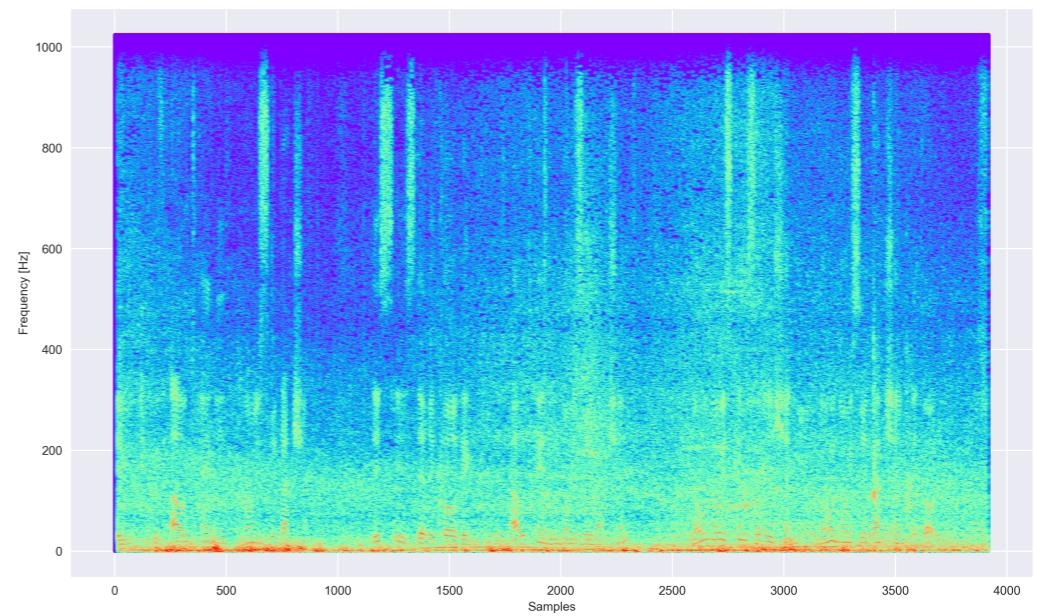
```
array([[ 0, 0, 0],  
[ 1, 0, 0],  
[ 2, 0, 0],  
...,  
[ 487, 1024, 0],  
[ 488, 1024, 0],  
[ 489, 1024, 0]])
```

# USING 3D TABLES FOR CLUSTERING THE VISUALIZATION

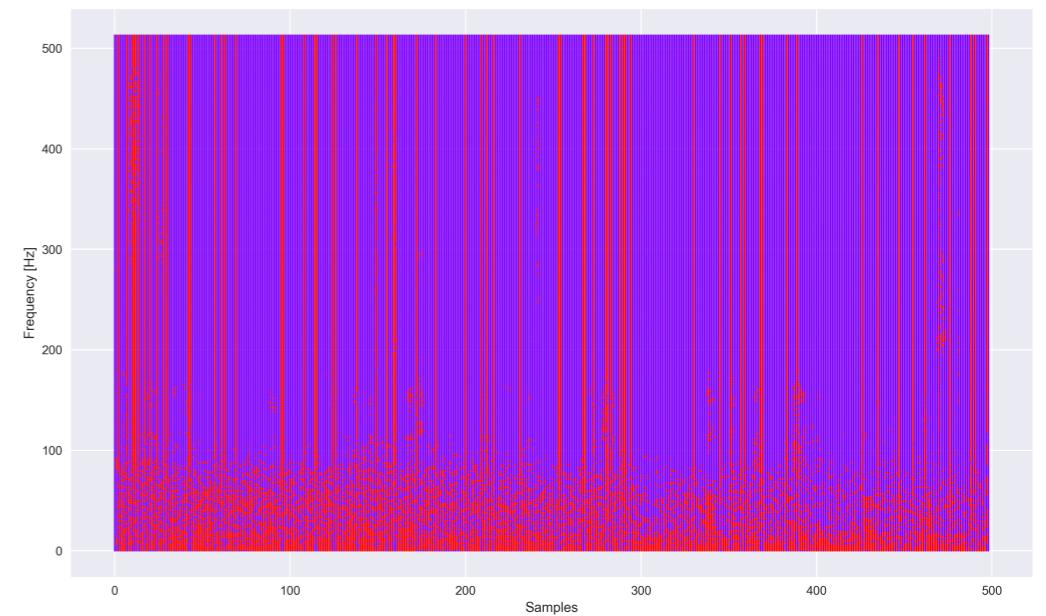
```
plt.pcolormesh(spectrogram)
```



```
plt.scatter(X[:,0], X[:,1], c=X[:,2])
```



```
plt.scatter(X[:,0], X[:,1], c=spectral_prediction)
```



## **ALGORITHMS:**

- Spectral Clustering
- 'rbf' VS. 'nearest\_neighbors' to create the affinity matrix
- 'discretize' VS. 'kmeans' to assign the labels
- ICA (Independent Component Analysis)

# PROCESS:

1. Load audio file

=> Get time series (array)

2. Run Short-time Fourier transform

=> Get spectrogram of amplitudes (matrix, bins \* frames)

3. [optional] Change the structure of the data to improve clustering results

=> Get a new matrix with row per sample

4. Cluster the data using a clustering algorithm

=> Get a matrix with cluster label per sample

5. Remove (or reduce) the samples that were clustered as noise

=> Get noise reduced spectrogram (matrix)

6. Run Inverse short-time Fourier transform

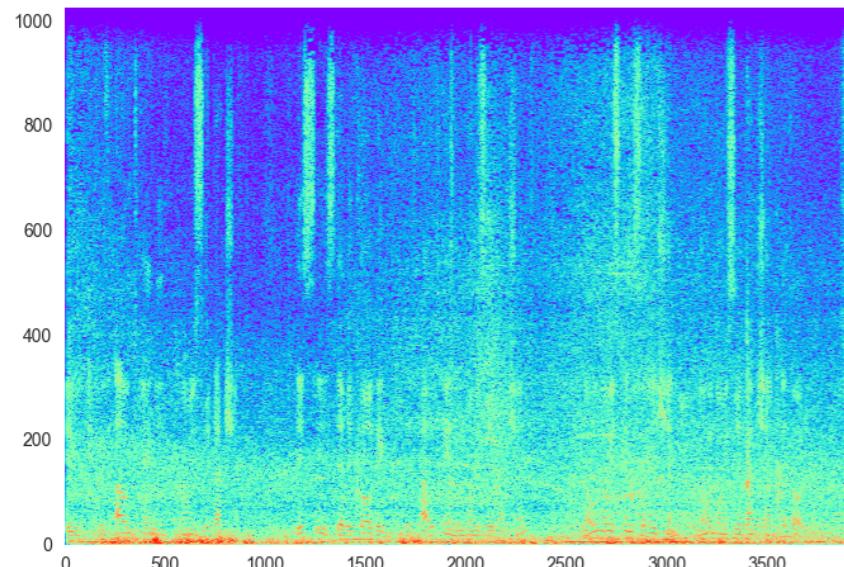
=> Get time series (array)

7. Write a file using the new time series array

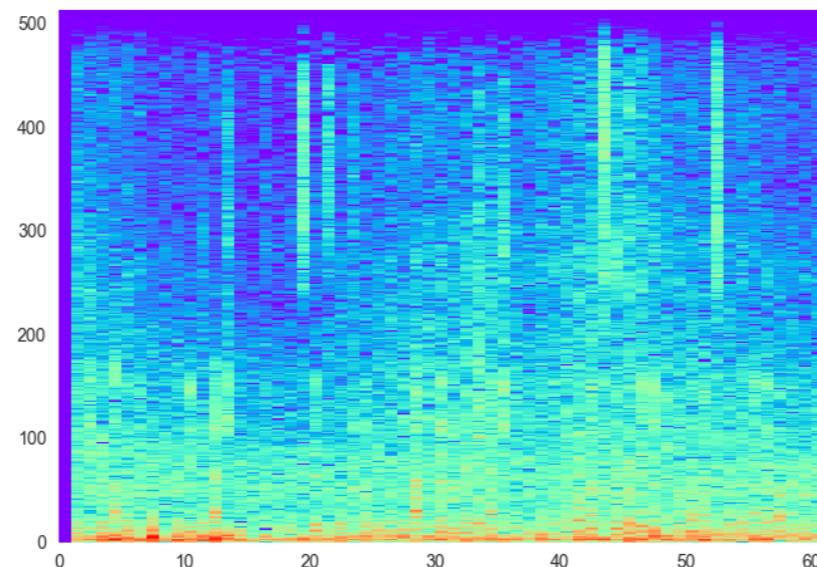
=> Output .wav file

# REDUCING COMPUTATION TIME

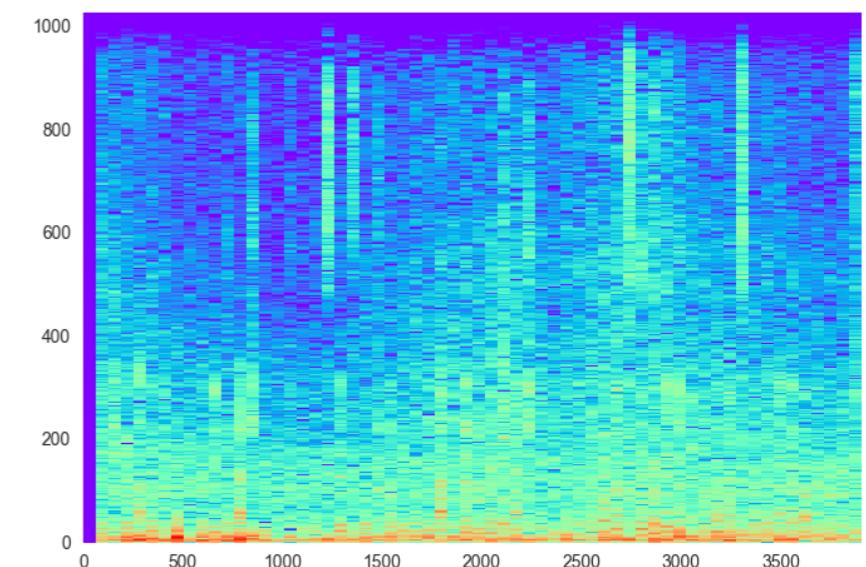
High-res > Low-res > High-res



$2049 \times 1960 =$   
**4,016,040 samples**



$683 \times 70 =$   
**47,810 samples**



$2049 \times 1960 =$   
**4,016,040 samples**

# OPEN-SOURCE PYTHON3 PACKAGES:

- **Scikit-learn**

- Spectral Clustering: <http://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html>
- FastICA: <http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.FastICA.html>

- **LibROSA** (Brian McFee, PhD @ NYU Steinhardt)  
<http://librosa.github.io/librosa/index.html>

- Graphical representation

- **Matplotlib**

- <http://matplotlib.org/>

- **Seaborn**

- <https://seaborn.pydata.org>

- **Pandas**

- <http://pandas.pydata.org/>

- blind\_source\_separation.py GitHub Gist by **abinashpanda**  
<https://gist.github.com/abinashpanda/11113098>

# Results

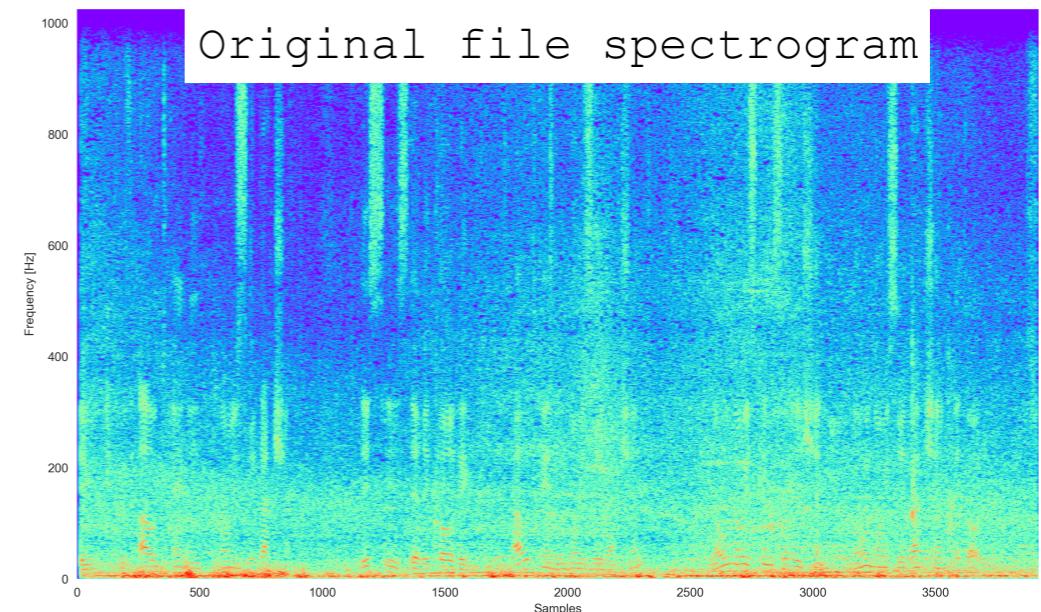
[https://dodiku.github.io/audio\\_noise\\_clustering/results/](https://dodiku.github.io/audio_noise_clustering/results/)

THE ORIGINAL RECORDING



# SPECTRAL CLUSTERING 01

- Full spectrogram (matrix) as dataset

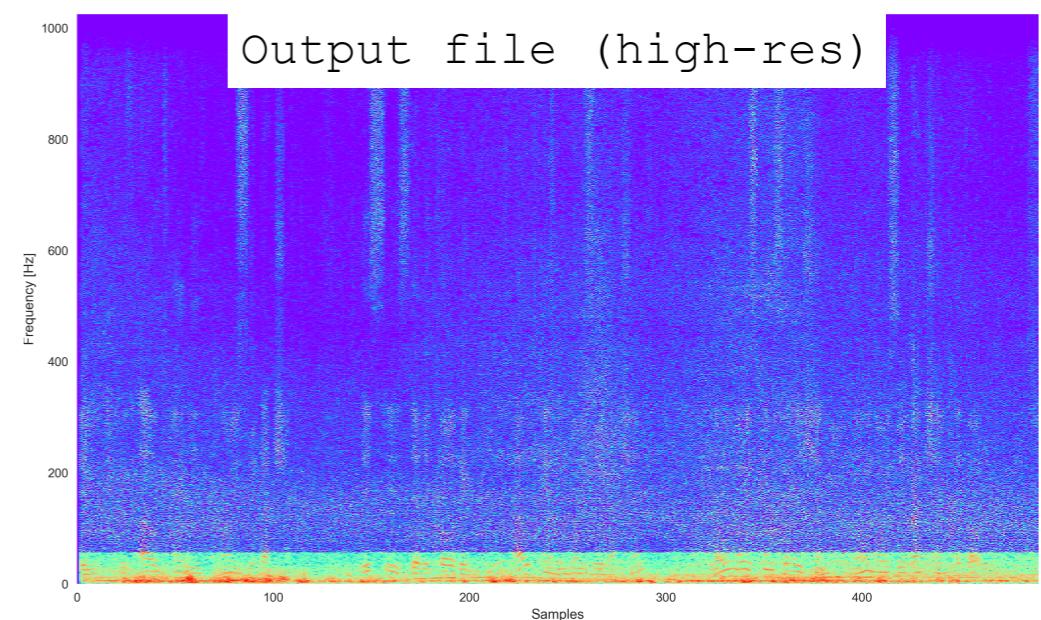


- Using 'nearest\_neighbors' to create the affinity matrix (adjacency matrix)

Using 'arpack' as the eigen solver

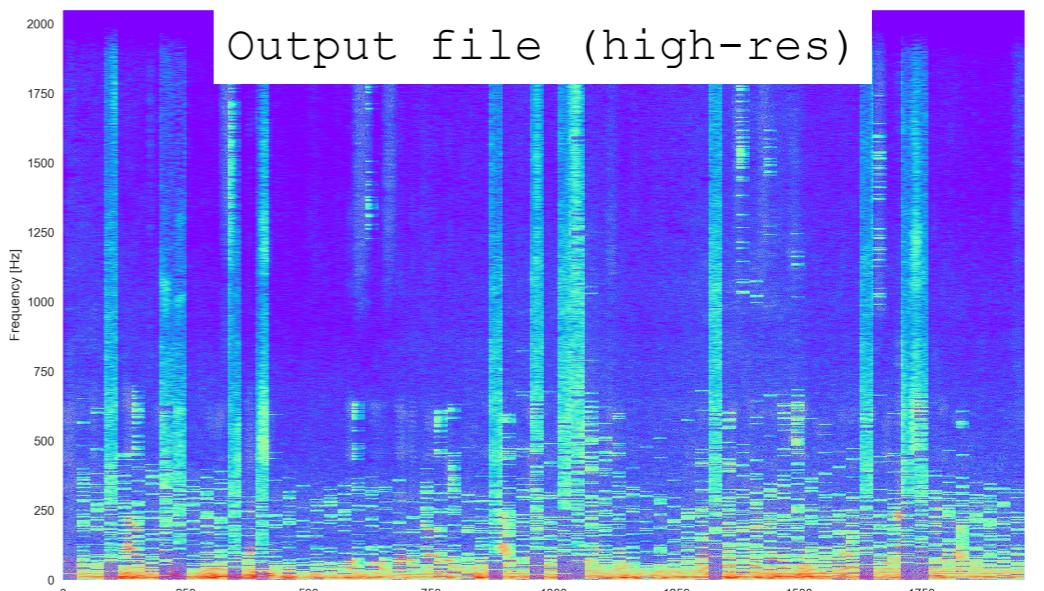
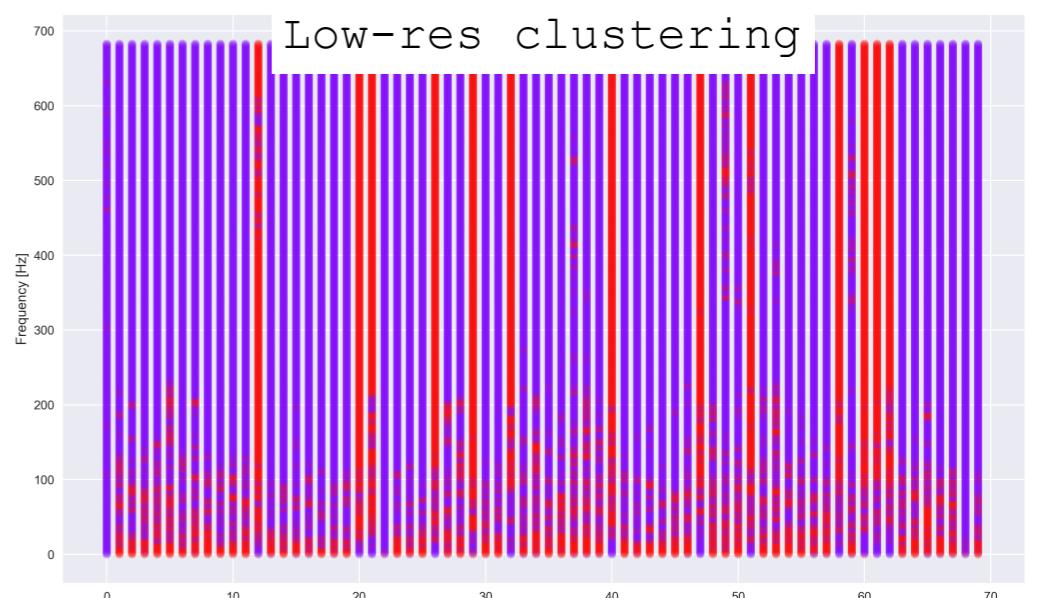
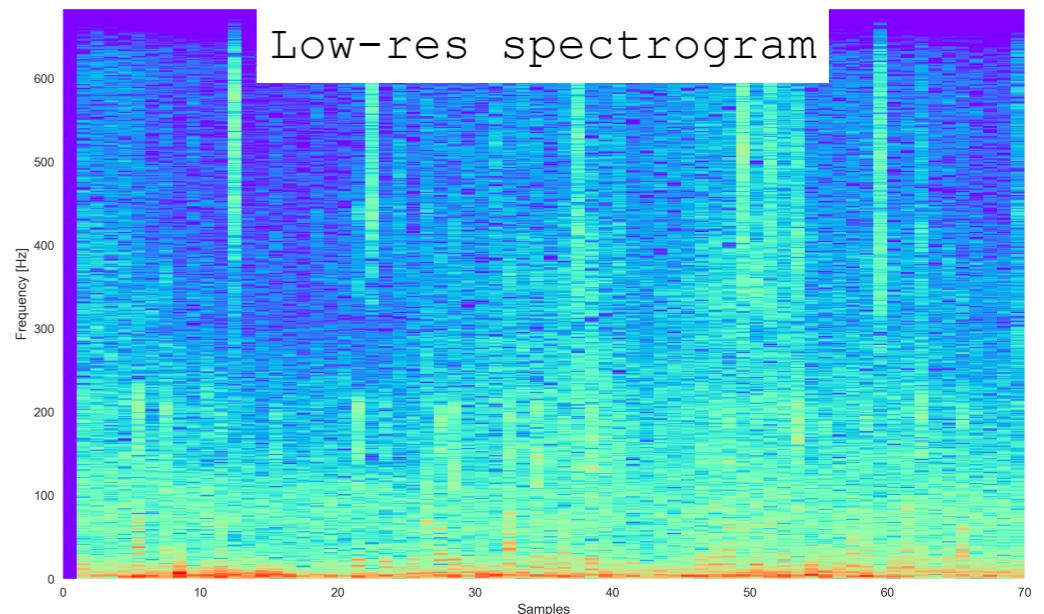


- Runtime = 4.4s



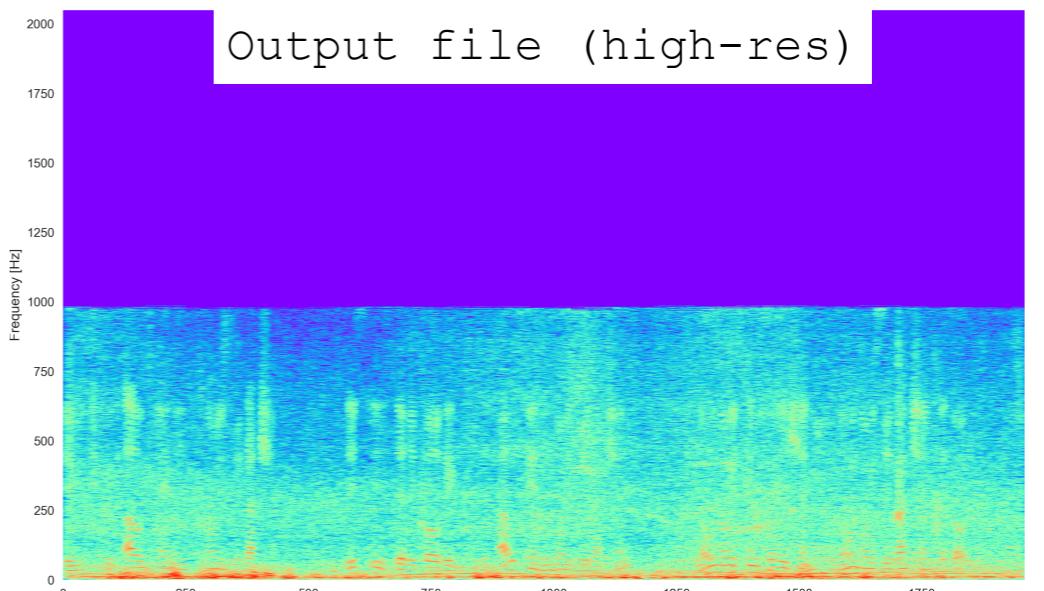
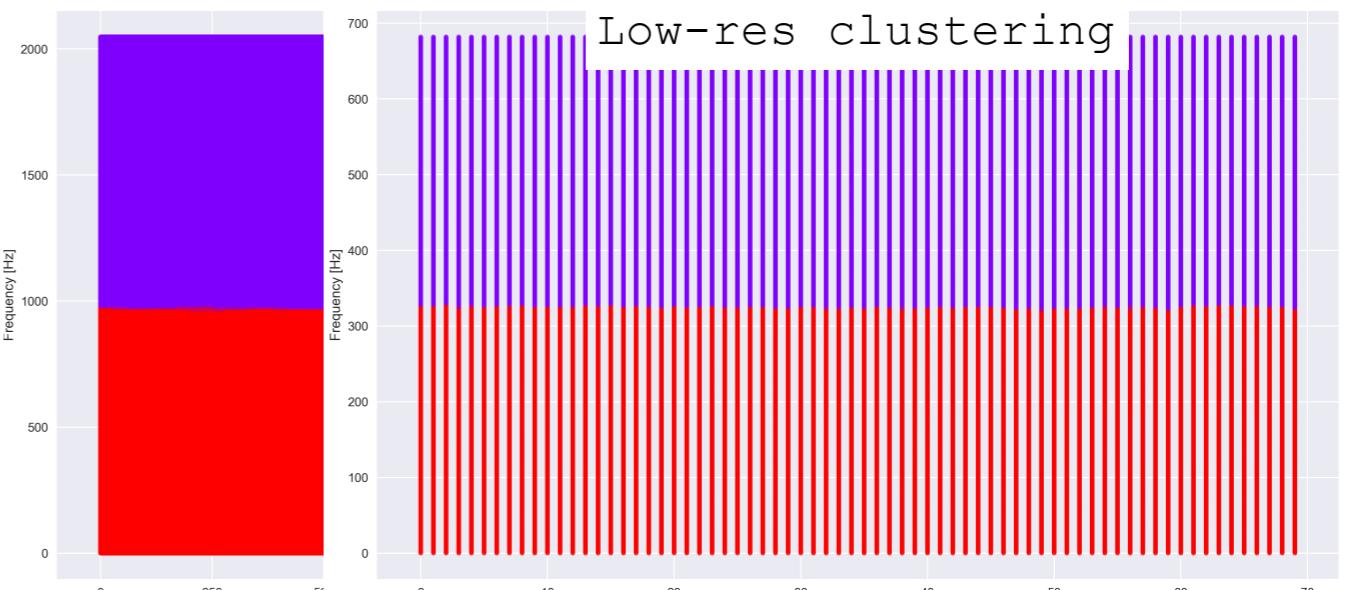
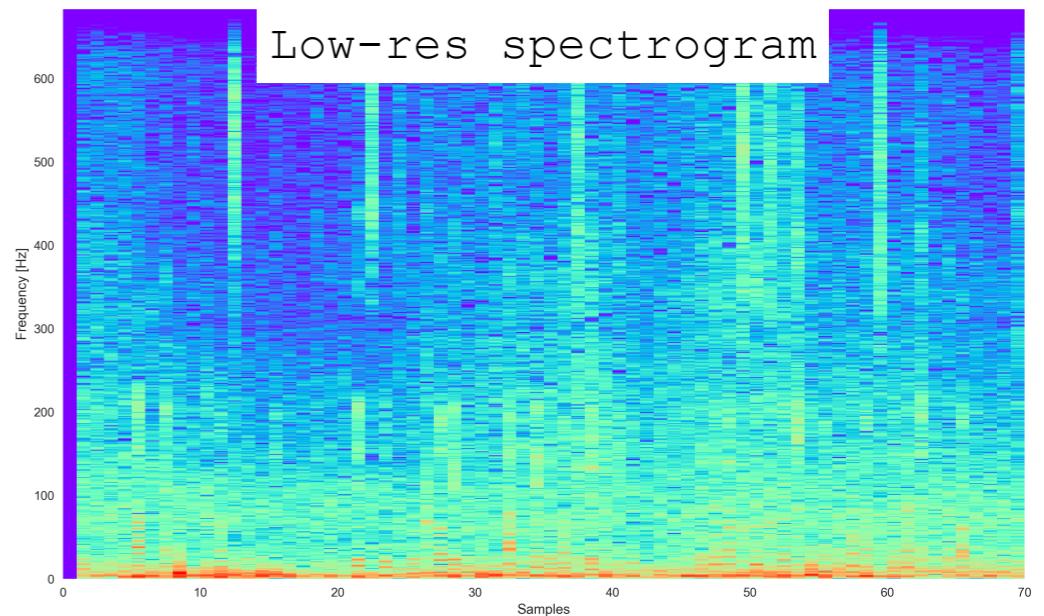
# SPECTRAL CLUSTERING 02

- Full spectrogram (matrix) as dataset
- Column-by-column clustering
- Using 'rbf' to create the affinity matrix
- Reducing res > clustering > applying on high res
- Runtime = 24.2s - 26.9s  
(in full res = ~1:30h)



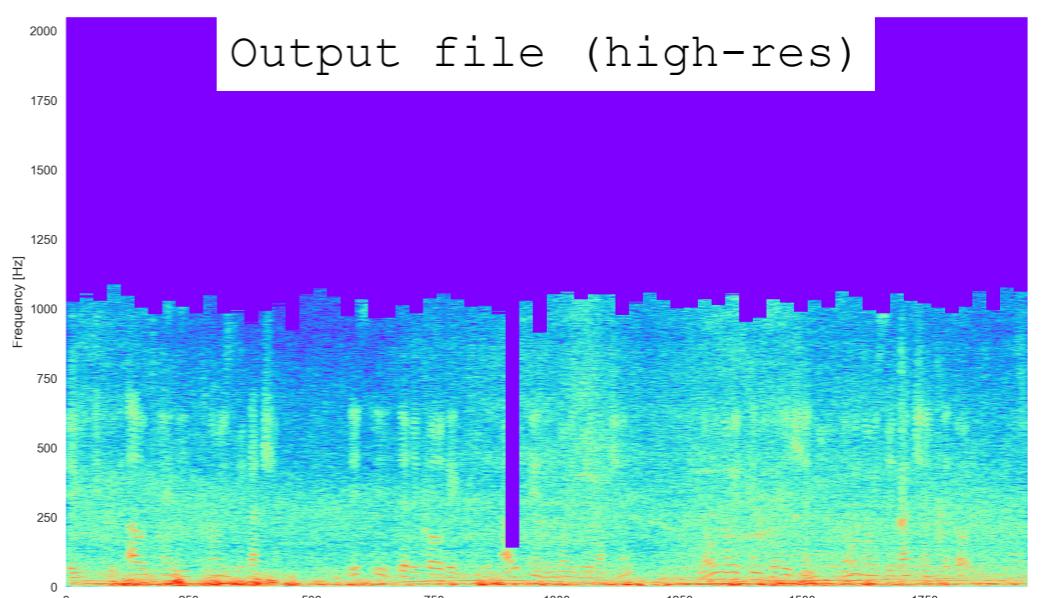
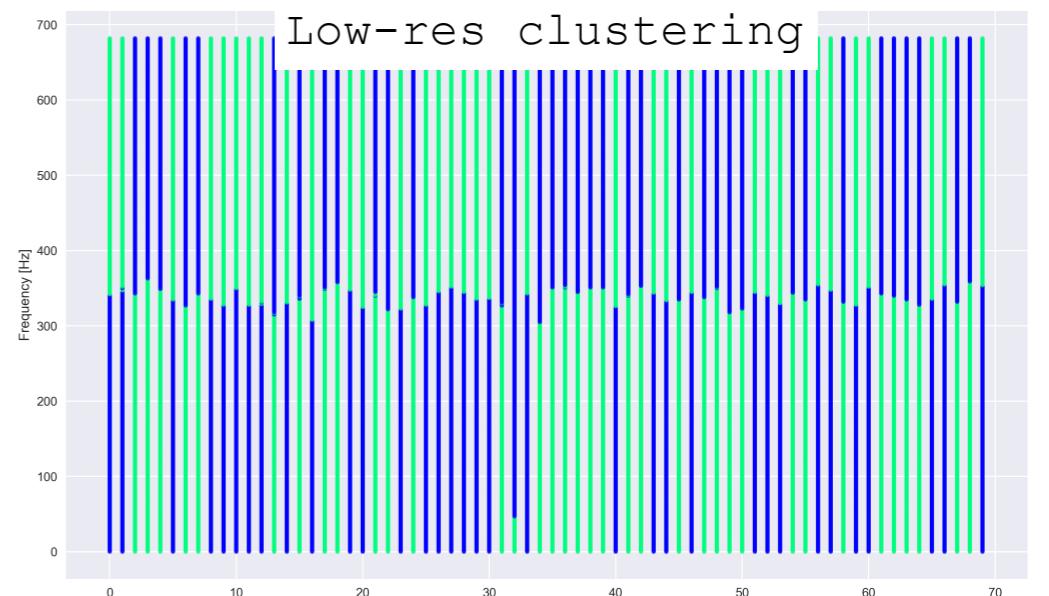
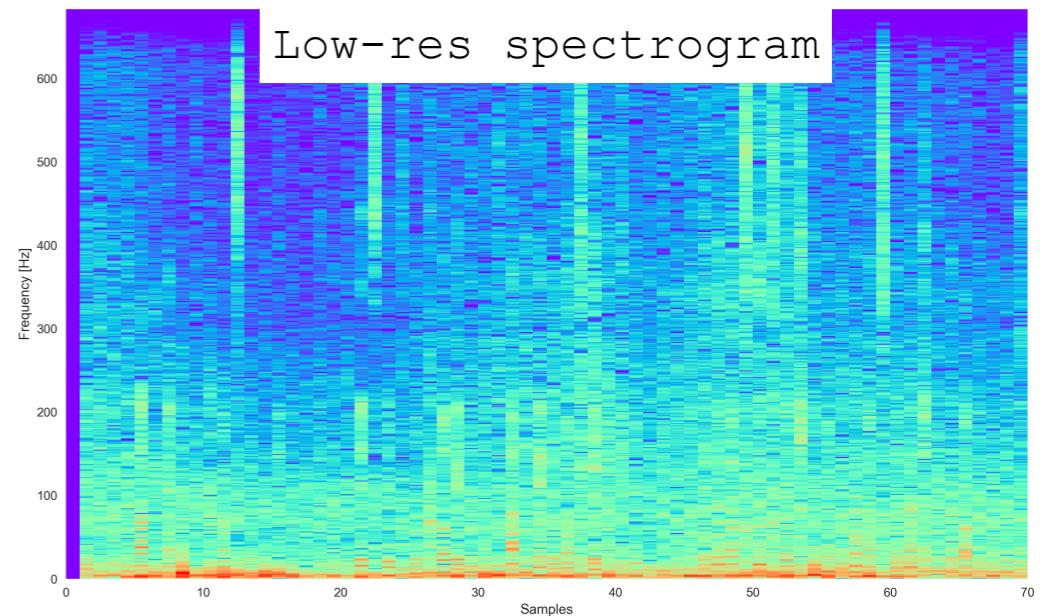
# SPECTRAL CLUSTERING 03

- Full 3D table as dataset
- All methods showed similar results
- Reducing res > clustering > applying on high res
- Runtime = ~4:10m  
(in full res =  $\infty$ )



# SPECTRAL CLUSTERING 04

- Full 3D table as dataset
- Column-by-column clustering
- All methods showed similar results
- Reducing res > clustering > applying on high res
- Runtime = ~34.4s  
(in full res = ~2.5h)

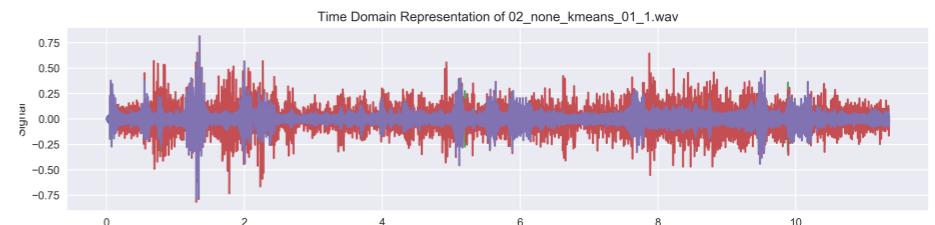


# ICA: INDEPENDENT COMPONENT ANALYSIS

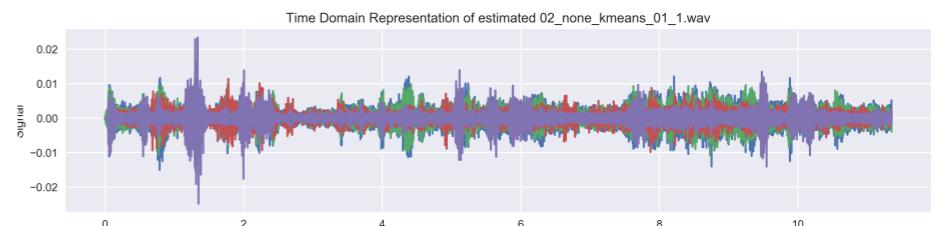
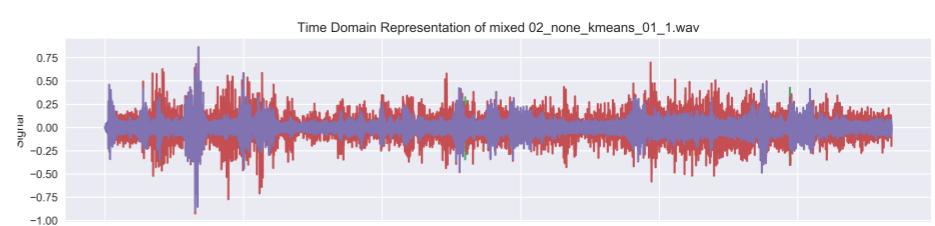
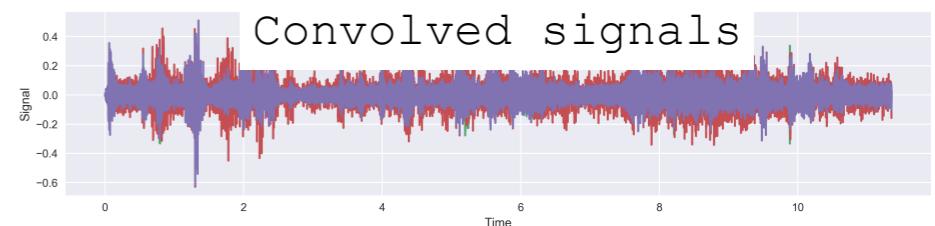
- Convolving 2 spectral clustering results.



- Generating two audio files:
  - one with noise
  - one with (supposedly) clearer speech



- Runtime = 0.2s-0.8s



# Conclusions

# CONCLUSIONS:

- The complex nature of audio signals makes it very hard (maybe impossible?) to cluster away noise.
- The tested clustering technique could not produce better results than existing production techniques.
- Computation time Computation time and the nature of the data makes the research hard to do on a laptop machine .
- DSP is great! The impact of machine learning on audio applications (besides speech-to-text) has yet to come. I'll keep on searching for it.

## **FURTHER RESEARCH:**

- Implementing a solution using other clustering techniques, such as kmeans and hierarchical clustering.
- Expanding the data by adding a column that describes the distance between a sample (a bin) and the centroid of the recording.
- Generating a solid ICA pipeline (results as an input, noise as an input)

Thanks

