Audio Signals Quality Diagnostics with Image Analysis

Vasily Tolkachev
Zurich University of Applied Sciences (ZHAW)
Institute for Data Analysis and Process Design (IDP)
vasily.tolkachev@gmail.com
www.idp.zhaw.ch

Dr. Oliver Dürr

Dr. Beate Sick

Zurich Machine Learning meetup
22.09.2016

About me

- Sep. 2014 - Now: Research Assistant in Machine Learning & Statistics at

Zürcher Hochschule
für Angewandte Wissenschaften

zh aw School of Engineering

IDP Institut für Datenanalyse
und Prozessdesign

- Sep. 2011 - Aug. 2014: MSc Statistics & Research Assistant at

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

- Sep. 2008 - Aug. 2011: BSc Mathematical Economics at

UNIVERSITY OF LONDON

- https://ch.linkedin.com/in/vasily-tolkachev-20130b35

Project Outline

Sound emitters	
broken	working
52	1991

- Goal: build a discriminative model to distinguish between the broken and the working devices.
- We transformed audio analysis task into image analysis by creating a spectrogram of every sound file.
- Since Convolutional Neural Networks have achieved remarkable results in image analysis, initial attempts included their application on spectrograms (e.g. OxfordNet & modifications) in Lasagne library.
- First classification results (even on the training set) were not very promising, hence deeper investigation on data quality & artefacts began.

Data Artefacts

- Unrepresentative number of broken devices (2.5%), making it hard to predict broken class.

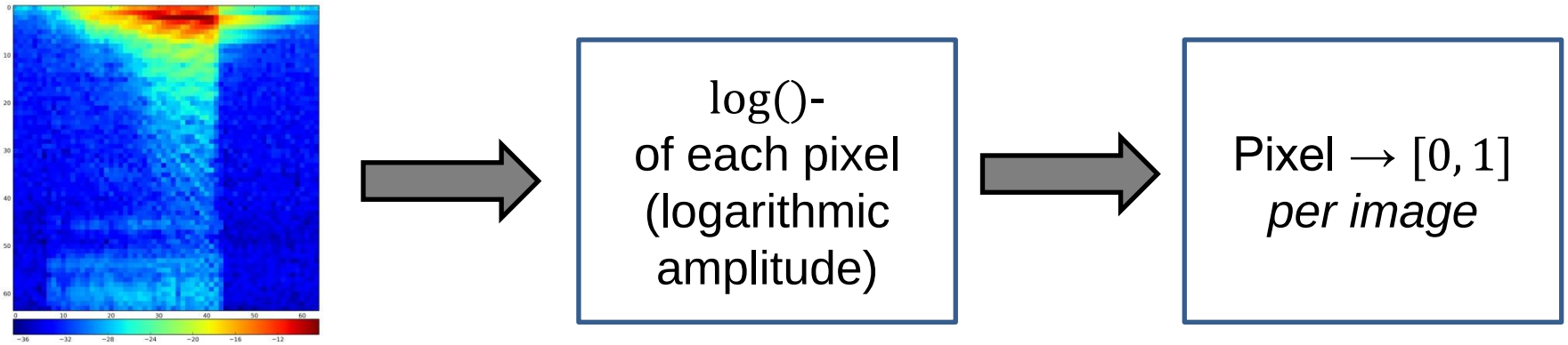
- This is a problem with an *unbalanced sample*, where some classifiers could predict only one class (“working”) for all observations and achieve low loss, which would give meaningless prediction for new samples.

- In general, it was hard to tell the difference between the spectrograms of broken and working devices even by eye inspection.

- This led us to suspect that the response variable also had noise in it (More on this later).

Data transformations

- After reading-in each audio file, a spectrogram* was produced from it and converted to a square image. Various image sizes (28x28 and 64x64 pixels) were investigated for performance. Finally, 64x64 size was found to be “optimal” for the currently used methods.



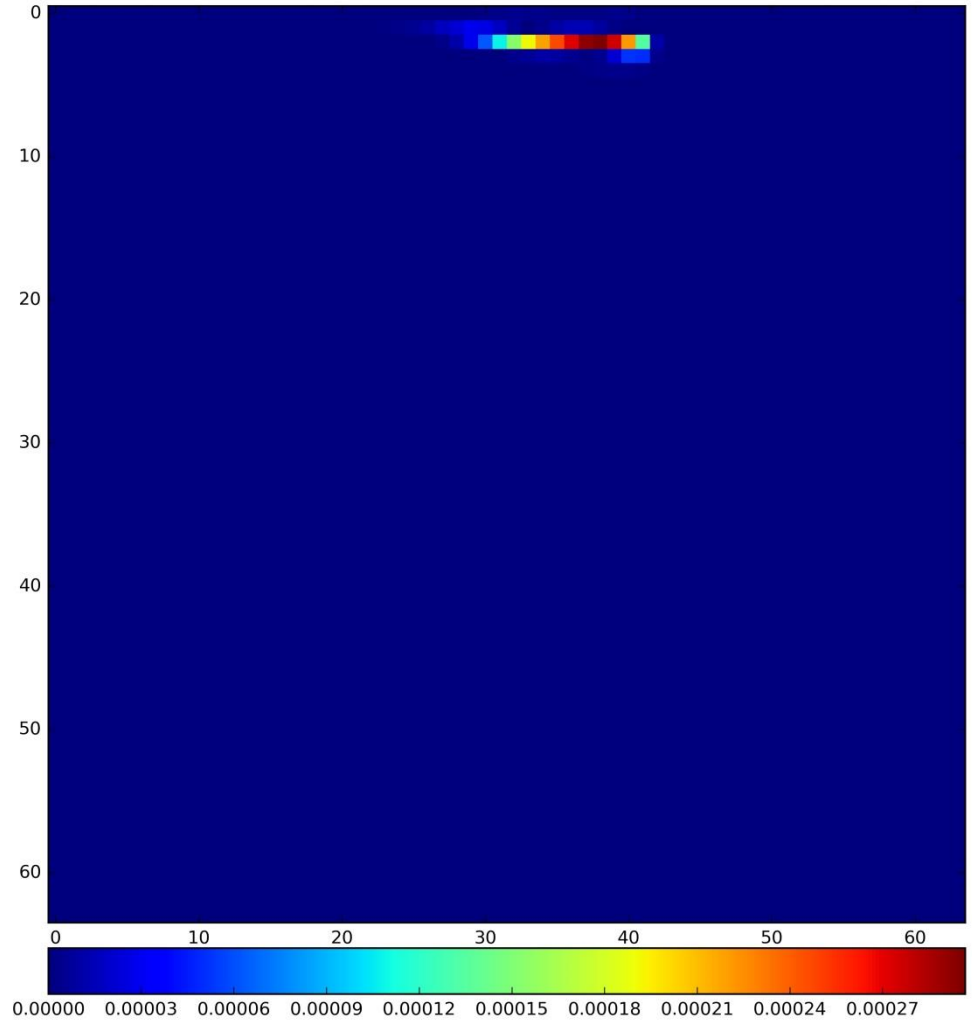
- Apart from physical reasons (transforming the input to dB) these transformations enabled to input the most information into the classifiers.

* (using default parameters of `signal.spectrogram` from `scipy`, with sampling rate 48000 Hz, Tukey window with shape parameter of 0.25, without filters. We used the same number of components for frequency and time domain)

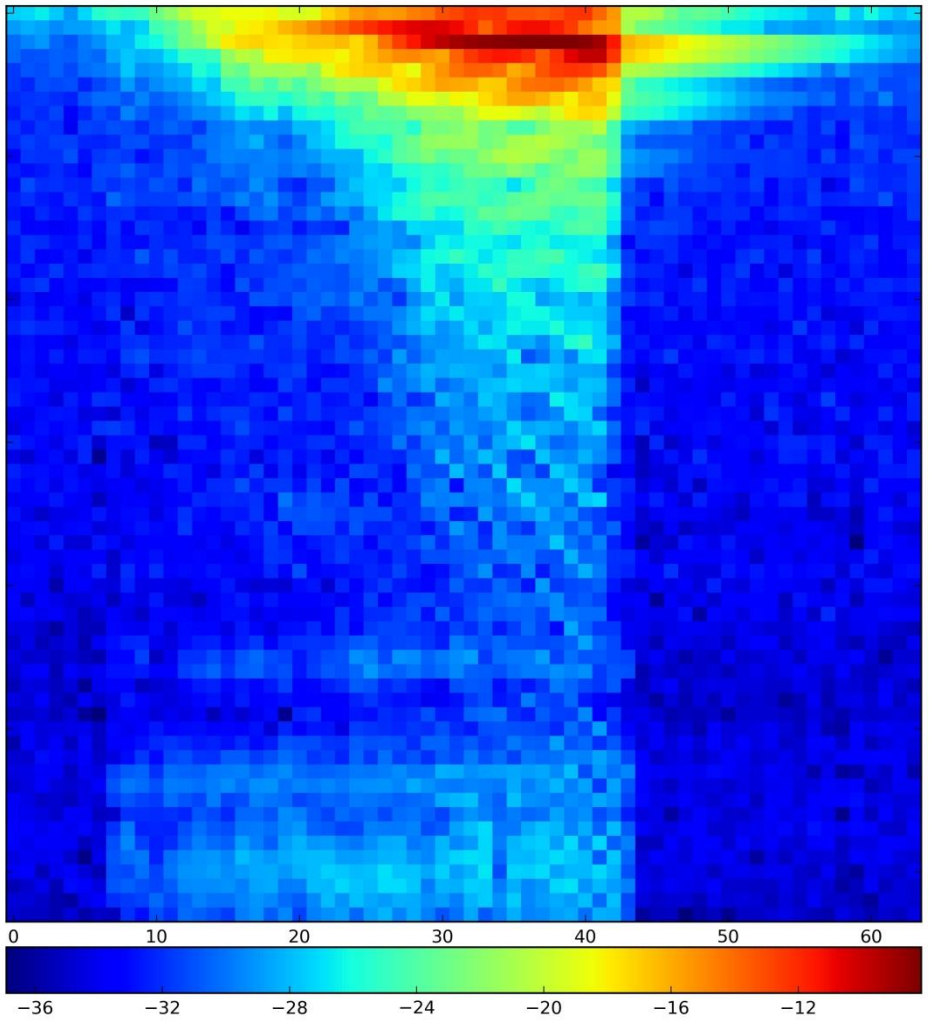
Audacity Demo

Data transformations

Raw Spectrogram of a sample audio file without transformations

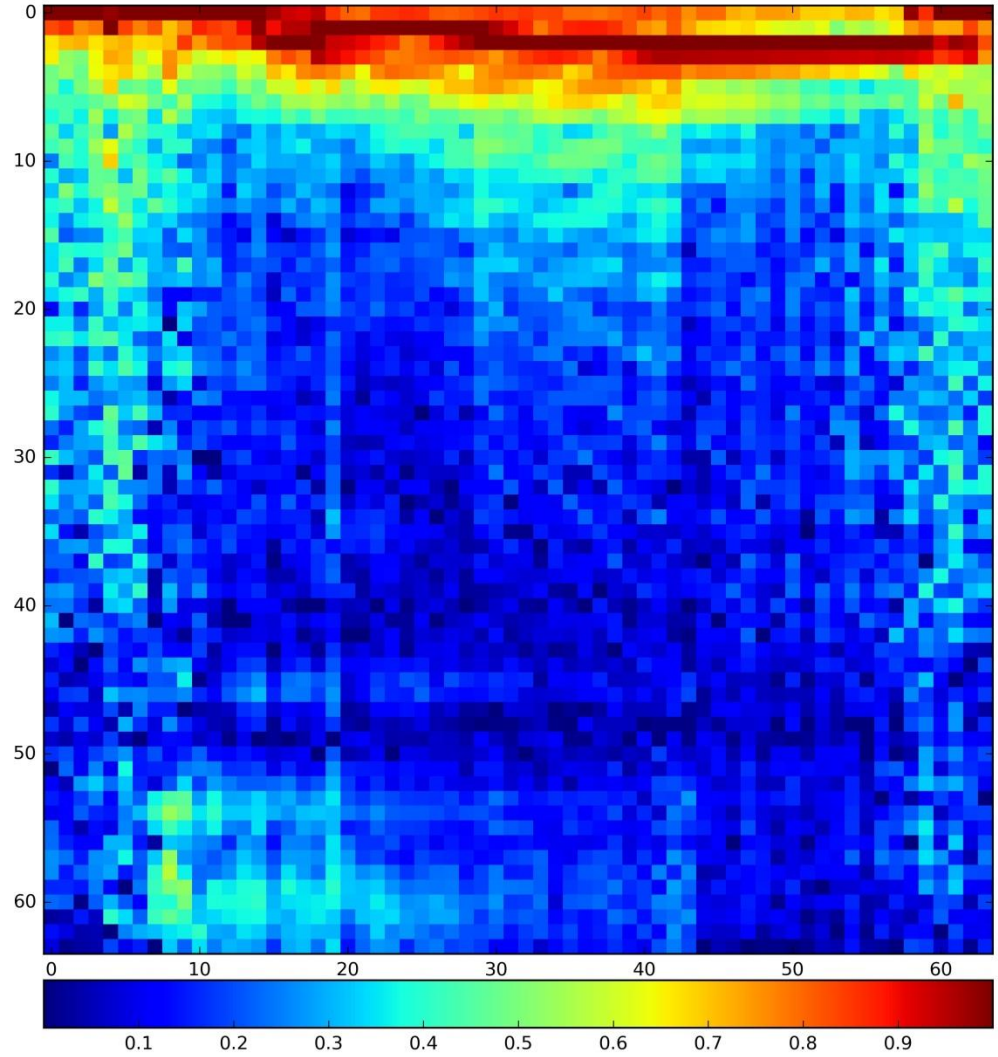


Spectrogram of the same audio file after log()-transformation



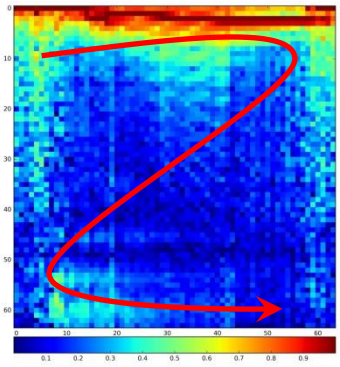
Data transformations

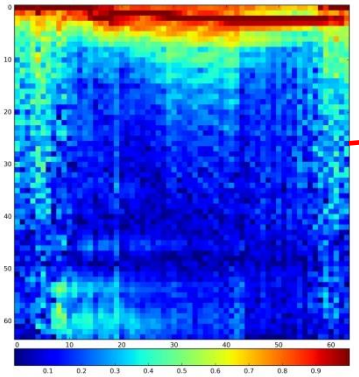
Spectrogram of the same audio
after log()-transformation
and mapping every pixel to [0, 1]

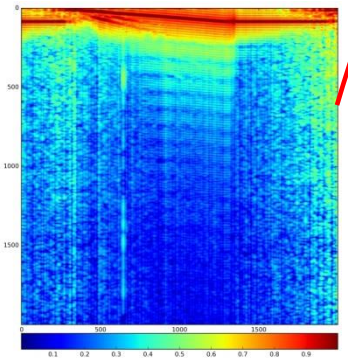


- After these transformations, each spectrogram image was flattened out into a vector of pixel values.
- Hence, for 2043 spectrograms we constructed a data frame of size 2043×64^2, with 2043 observations(rows) and 4096 variables representing each pixel (columns).

Mapping images to a vector





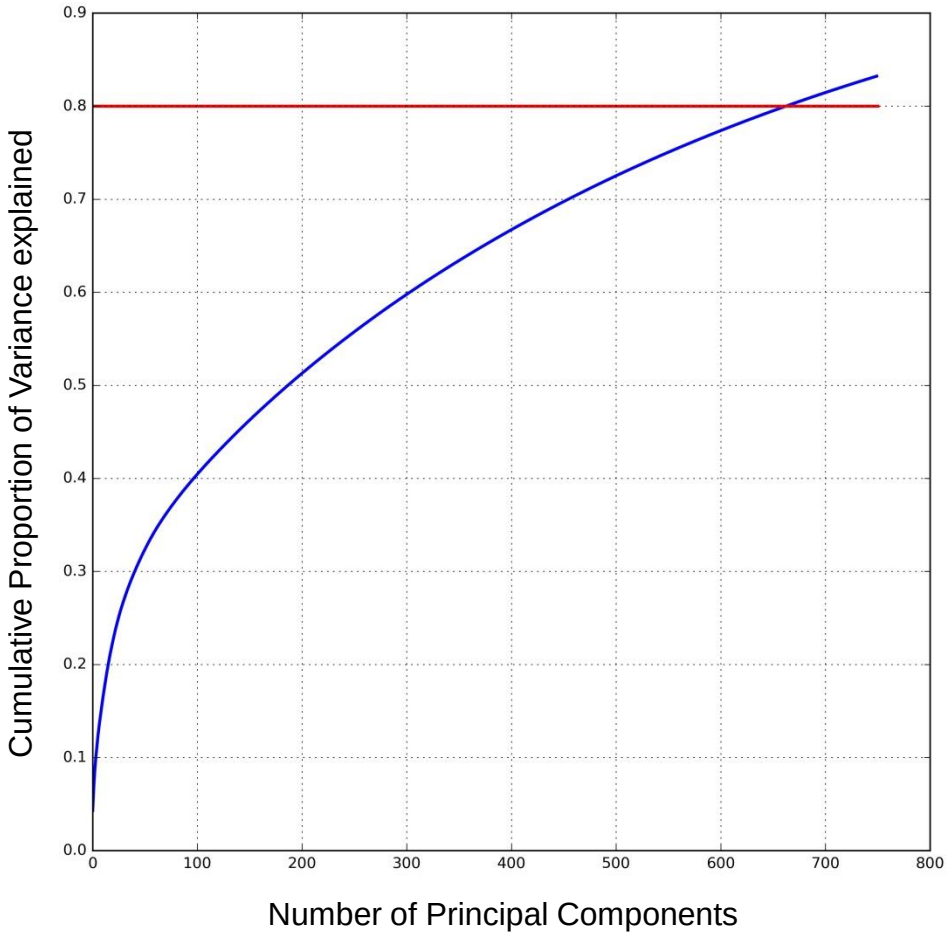


	Pixel 1	Pixel 2	…	Pixel 64^2
Image 1	1	0.9296		0.0048
Image 2	0.761	1		0.0586
…			…	
Image 2043	1	0.9066		0.0296

Principal Components Analysis (PCA)

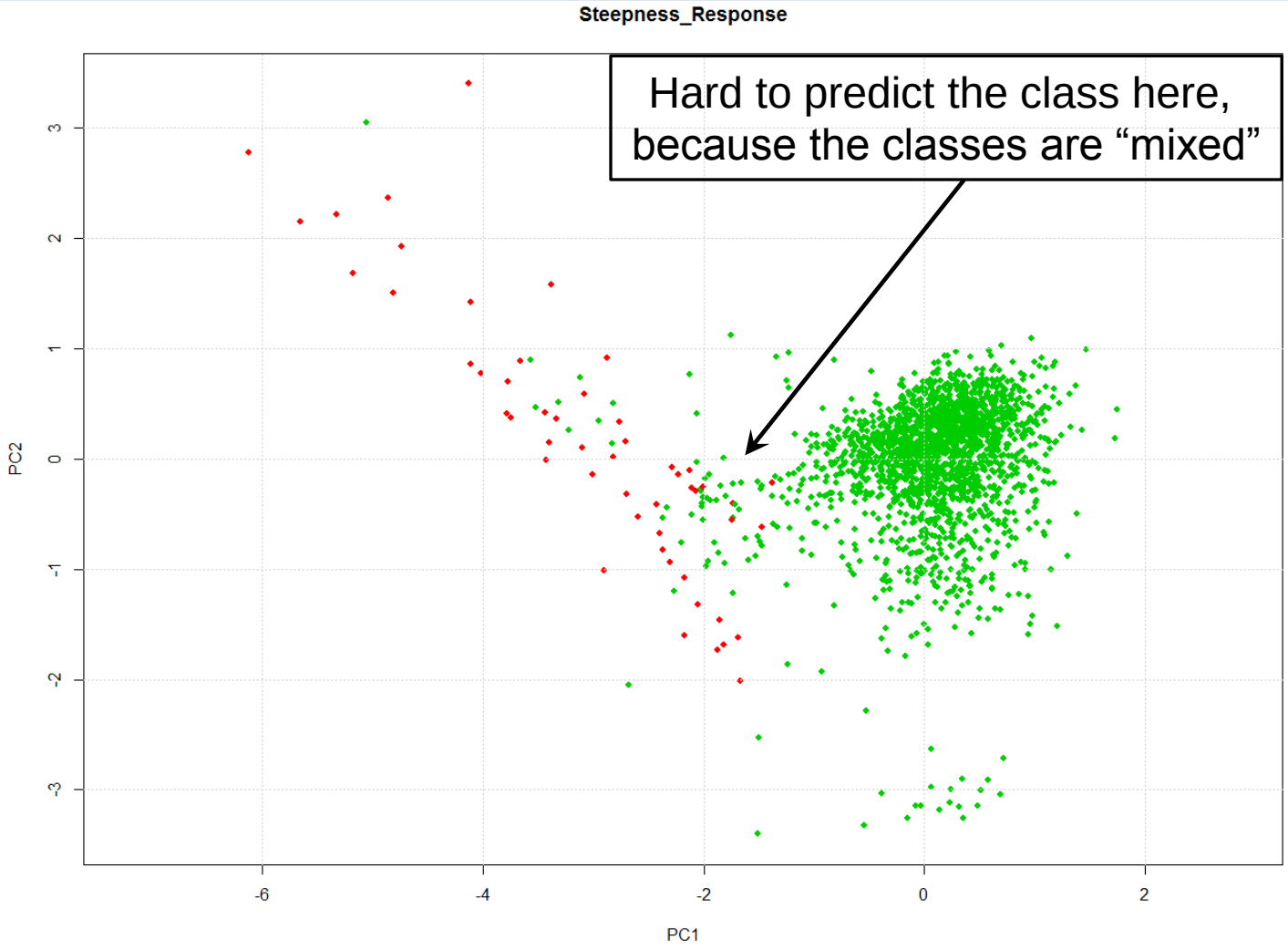
- The idea of the explorative analysis is to obtain an overview of the data, first *without looking at its labels*.
- Therefore we applied PCA to reduce the dimensions of the data (4096) and find some clusters which could later be attributed to the labels.
- Notice that producing high resolution images would lead to even greater number of variables, which is undesirable given the sample size.
- As usual in PCA, the data was standardized to have zero mean (per image) in order to avoid numerical problems.
- Afterwards various supervized learning methods were be considered.

Principal Components Analysis (PCA)



- According to one of the standard heuristic rules, about 660 Principal Components (transformed “pixels”) were enough to explain 80% of variability in the data.
- we attempt to plot it in the space of two most important Principal Components, although this may look like a significant loss of information.

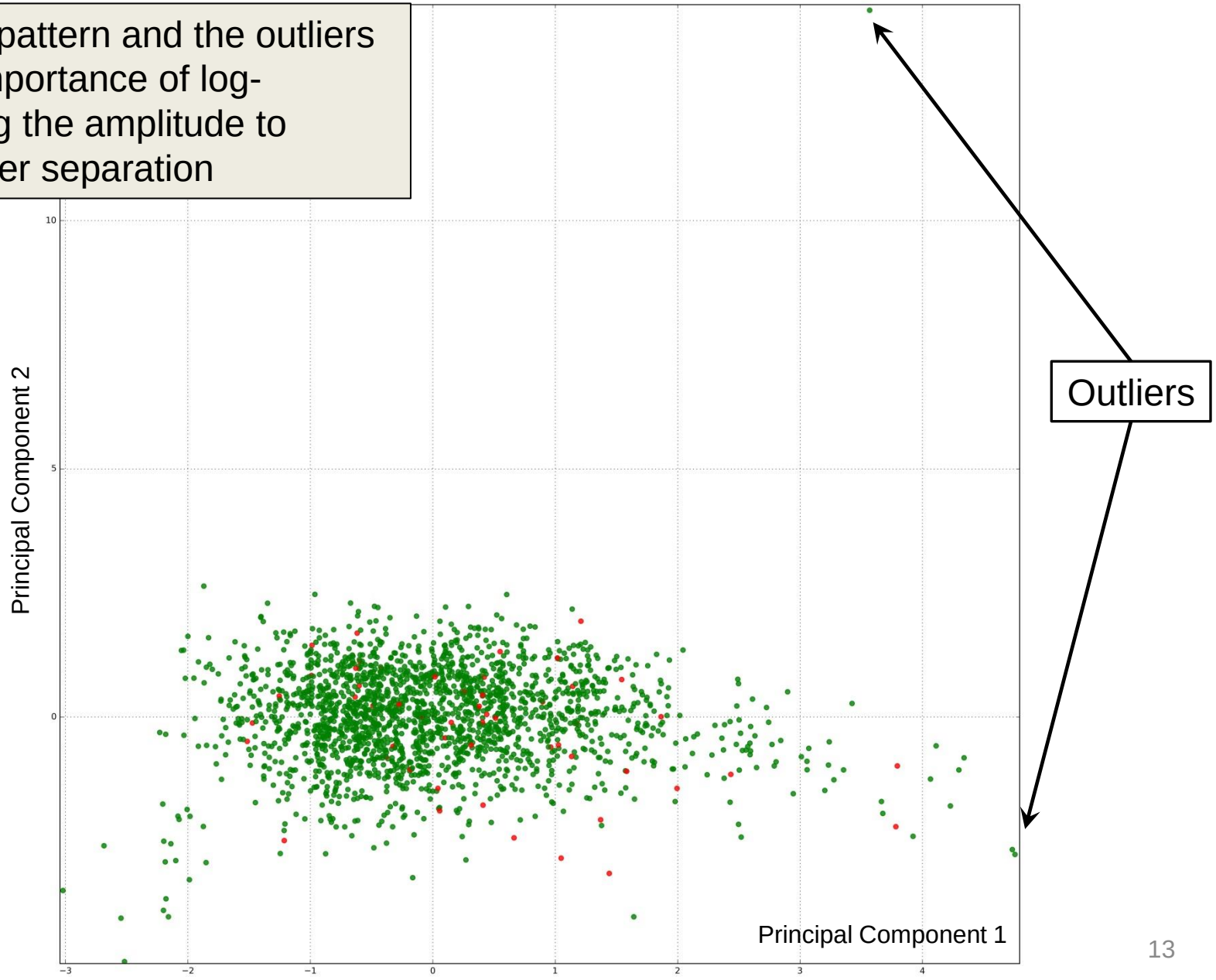
Principal Components Analysis (PCA)



- It’s already possible to distinguish the broken from the working classes relatively well.

PCA for untransformed variables

This mixed pattern and the outliers show the importance of log-transforming the amplitude to obtain clearer separation



Now that the Principal Components have been used to reduce noise in the data, let us experiment with one of the most successful dimensionality reduction techniques – **t-distributed Stochastic Neighbor Embedding (t-SNE)**.*

We will apply it to Principal Components and Variables(pixels) and observe how the results change when it is used on different number of PCs/variables.

* L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(Nov):2579-2605, 2008.

t-SNE

- High-level idea: In order to visualize high-dimensional data in 2 dimensions, model similar observations as close points and different observations as distant points in 2D space.
- Introduce similarity measure between the observations $\mathbf{x_1}$, ..., $\mathbf{x_N}$ in high-dim space:

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)},$$

Gaussian kernel

Normalize to get probabilities

- Center the Gaussian kernel at $\mathbf{x_i}$ and measure the density of all other points according to this kernel.
- Set the bandwidth σ_i for each point such that the conditional probability $p_{j|i}$ has a fixed perplexity ($2^{entropy}$) in order to adapt to different densities in the data space. It's analogous to choosing the effective number of neighbors.
- To obtain final similarities average the conditional densities:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

t-SNE

- Now let's introduce similarity measure between "projected" points $\boldsymbol{y_1}$, ..., $\boldsymbol{y_N}$ in the low-dimensional space:

$$q_{ij} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j)||^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + ||\mathbf{y}_k - \mathbf{y}_l)||^2)^{-1}}$$

Student t-kernel

Normalize to get probabilities

- The goal is to make q_{ij} as close as possible to p_{ij}
- Hence, minimize Kullback-Leibler divergence between the distributions:

$$KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right) \rightarrow \min_{\mathbf{y}_i}$$

- similar $\mathbf{x_i}$ and $\mathbf{x_j}$ ⇒ high p_{ij}, so q_{ij} must also be high, $\boldsymbol{y_i}$ and $\boldsymbol{y_j}$ close
- dissimilar $\mathbf{x_i}$ and $\mathbf{x_j}$ ⇒ low p_{ij}, so don't care about q_{ij}, $\boldsymbol{y_i}$ and $\boldsymbol{y_j}$ are far apart.
- In the end, preserve only the local similarity of the data and allow dissimilar points to be more distant in the low-dim space because of heavy-tailed Student's t-distribution (1)

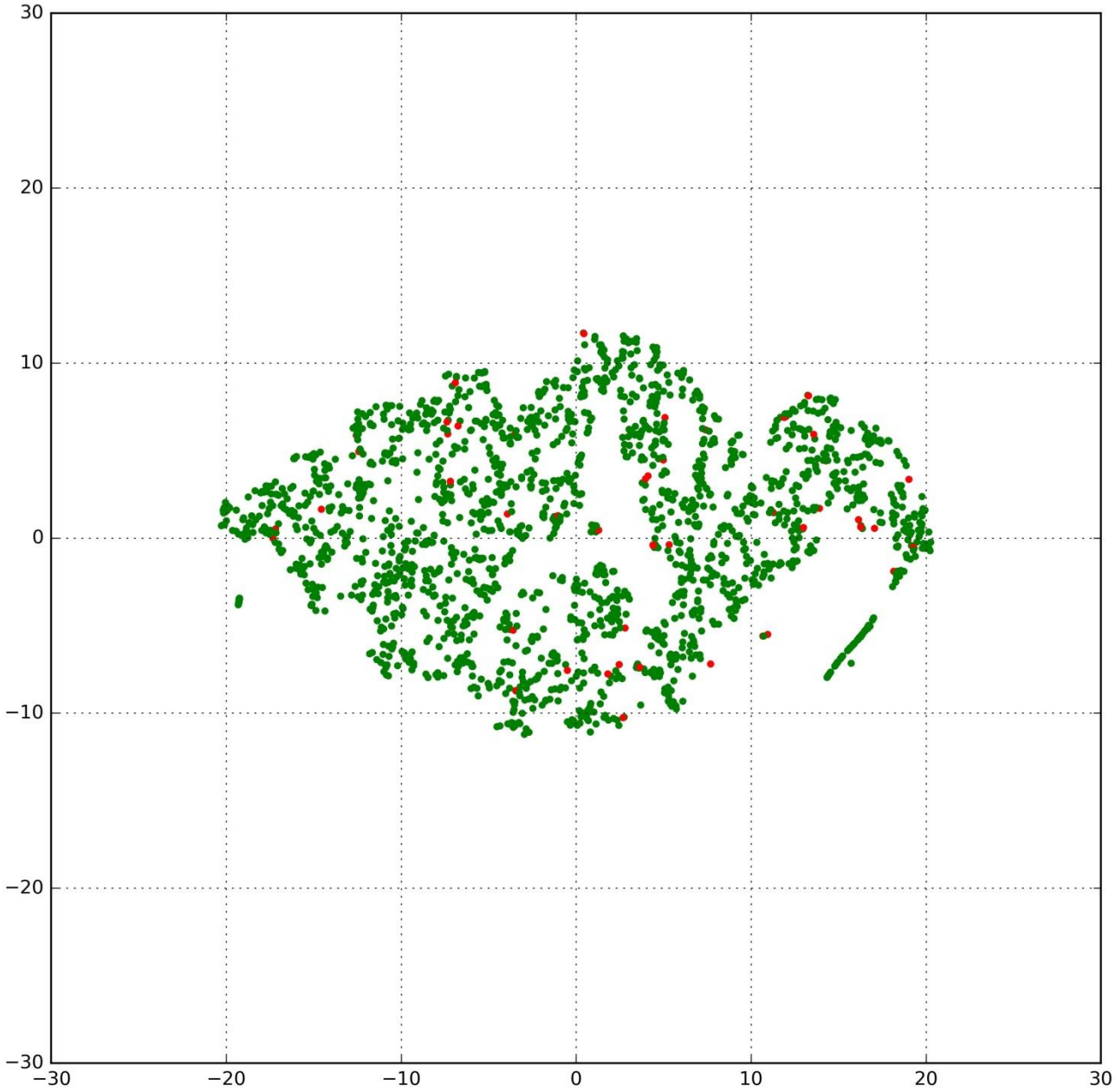
t-SNE

- Solve using gradient descent:

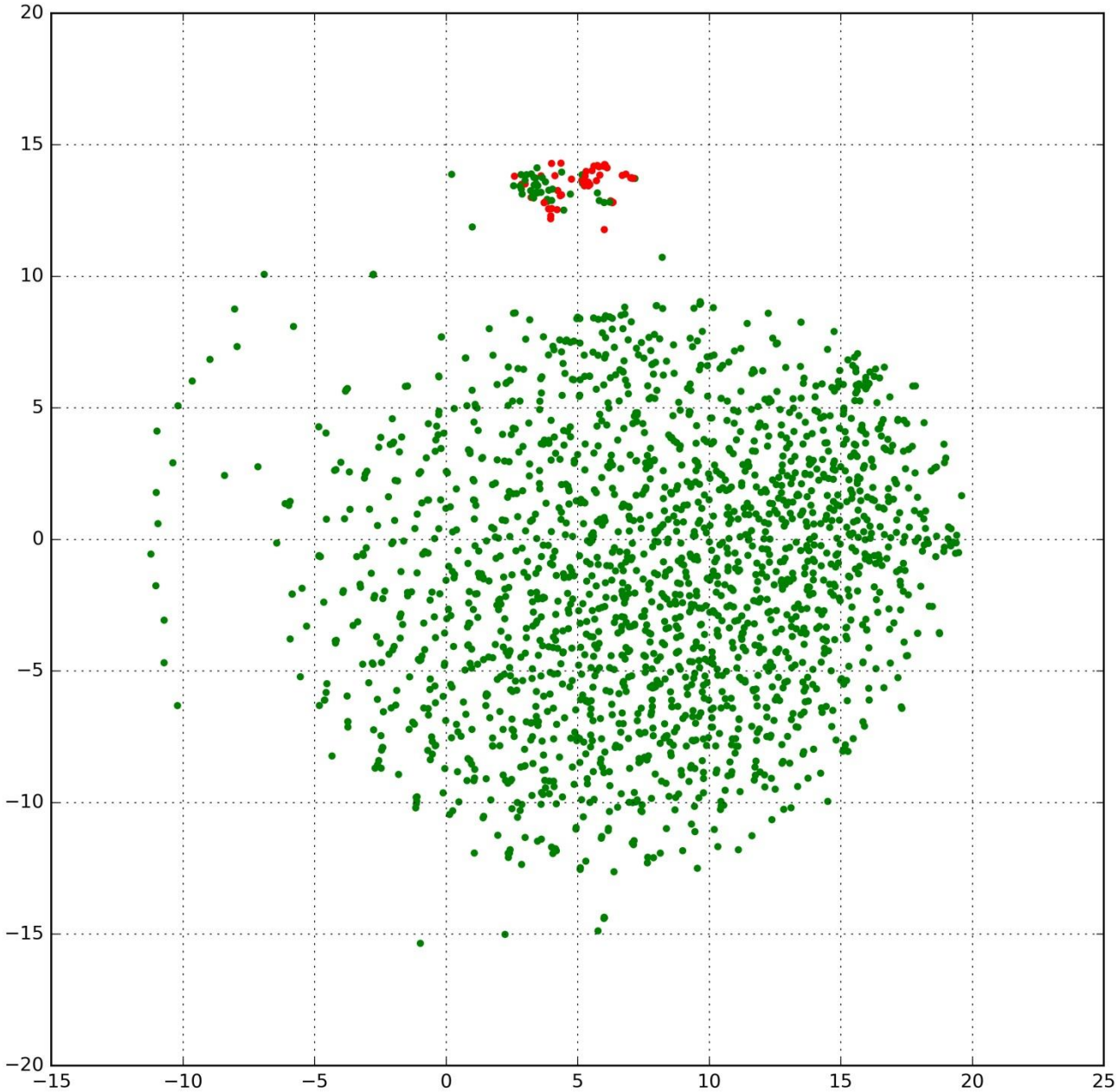
$$\frac{\partial}{\partial \mathbf{y}_i} KL(P||Q) = 4 \sum_j \underbrace{(p_{ij} - q_{ij})(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}_{\text{Compression of a spring}} \underbrace{(\mathbf{y}_i - \mathbf{y}_j)}_{\text{Spring between two points}}$$

- The gradient can be interpreted as a force along the spring between a point $\boldsymbol{y_i}$ and all other points.
- Thus, the gradient repels or attracts the points in low-dim space according to the similarity between points in the high-dim space.
- For big data: pairwise computation for all points is inefficient $O(N^2)$, hence pre-cluster close data points by their center of mass first, and then run gradient descent (Barnes-Hut approximation, $O(N \log N)$).

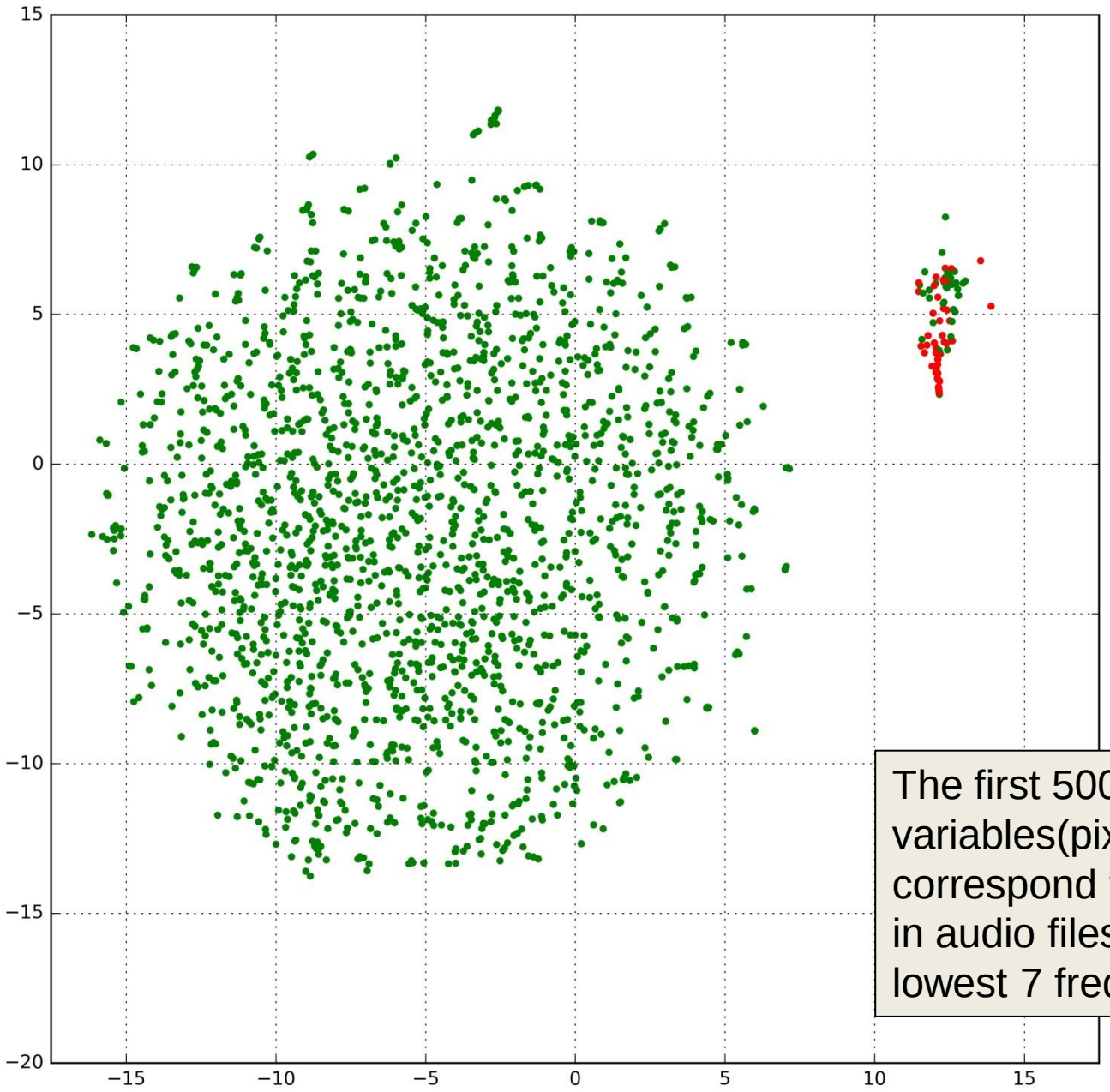
t-SNE clustering on all untransformed variables



t-SNE clustering on all transformed variables

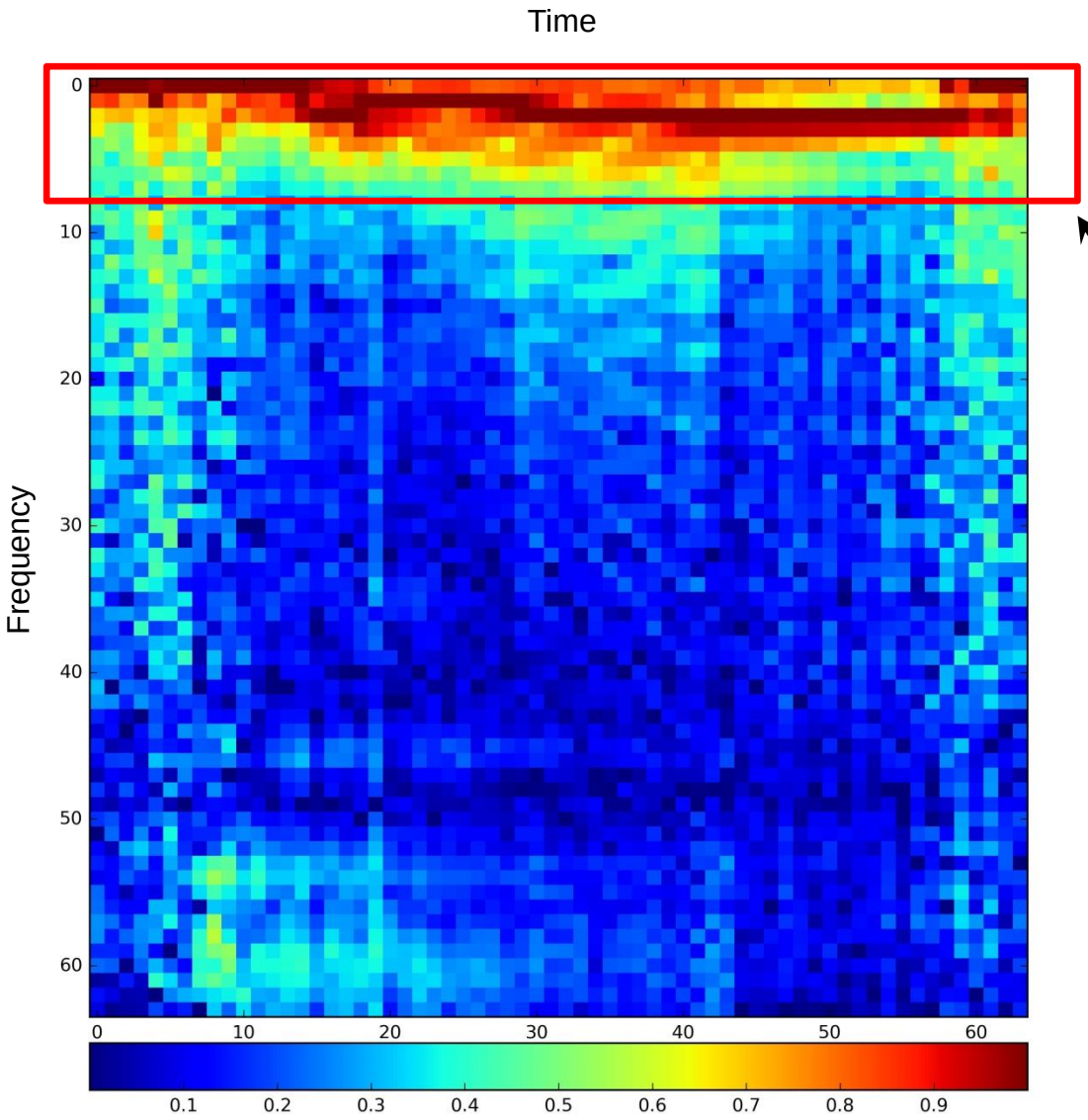


t-SNE clustering on first 500 transformed variables



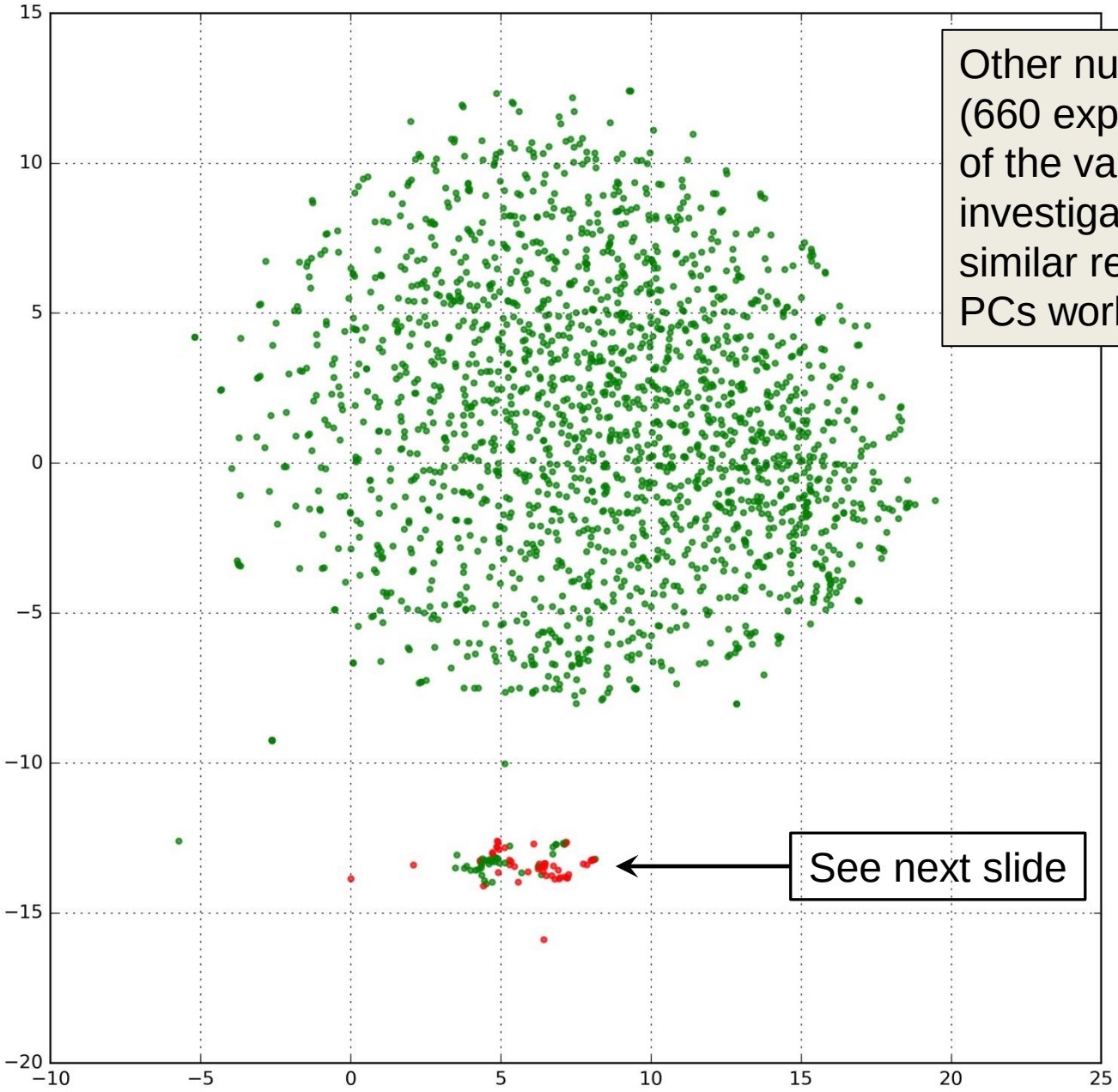
The first 500 variables(pixels) correspond to all times in audio files with the lowest 7 frequencies.

t-SNE clustering on first 500 transformed variables



The first 500 variables(pixels) correspond to all times in audio files with the lowest 7 frequencies (300 Hz - 5kHz). Human can hear 20Hz-20kHz.

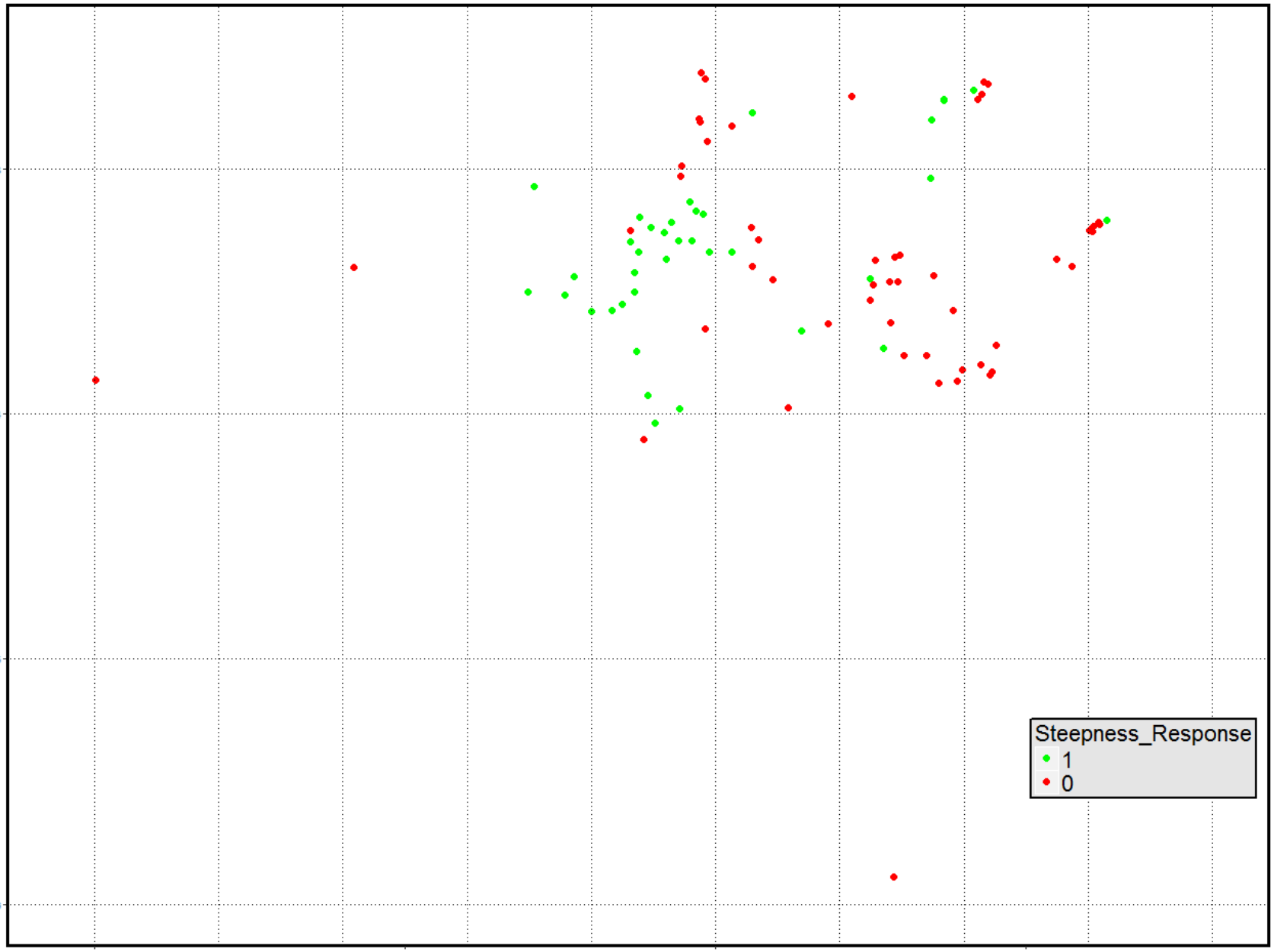
t-SNE clustering on first 500 PCs



Other number of PCs (660 explaining 80% of the variance) were investigated, yielding similar results, but 500 PCs worked best.

See next slide

t-SNE clustering on first 500 PCs (zoomed)



t-SNE clustering on first 500 PCs

- In the bottom cluster there are 37 “working” and 52 “broken” devices found.
- We are now able to capture all 52 broken emitters by eye inspection, and can thus verify how different supervised learning algorithms work.
- Even though the representation in 2 dimensions is not entirely correct, we have reasons to believe that these 37 working devices could be broken as well, as the next slides show.
- More generally, we suspected that some of the labels could be *misleading*, meaning that in fact some devices which were labeled as “working” were looking like “broken”.

Classification with Random Forest (RF)

Accuracy score: 0.9873 (+/- 0.0109)

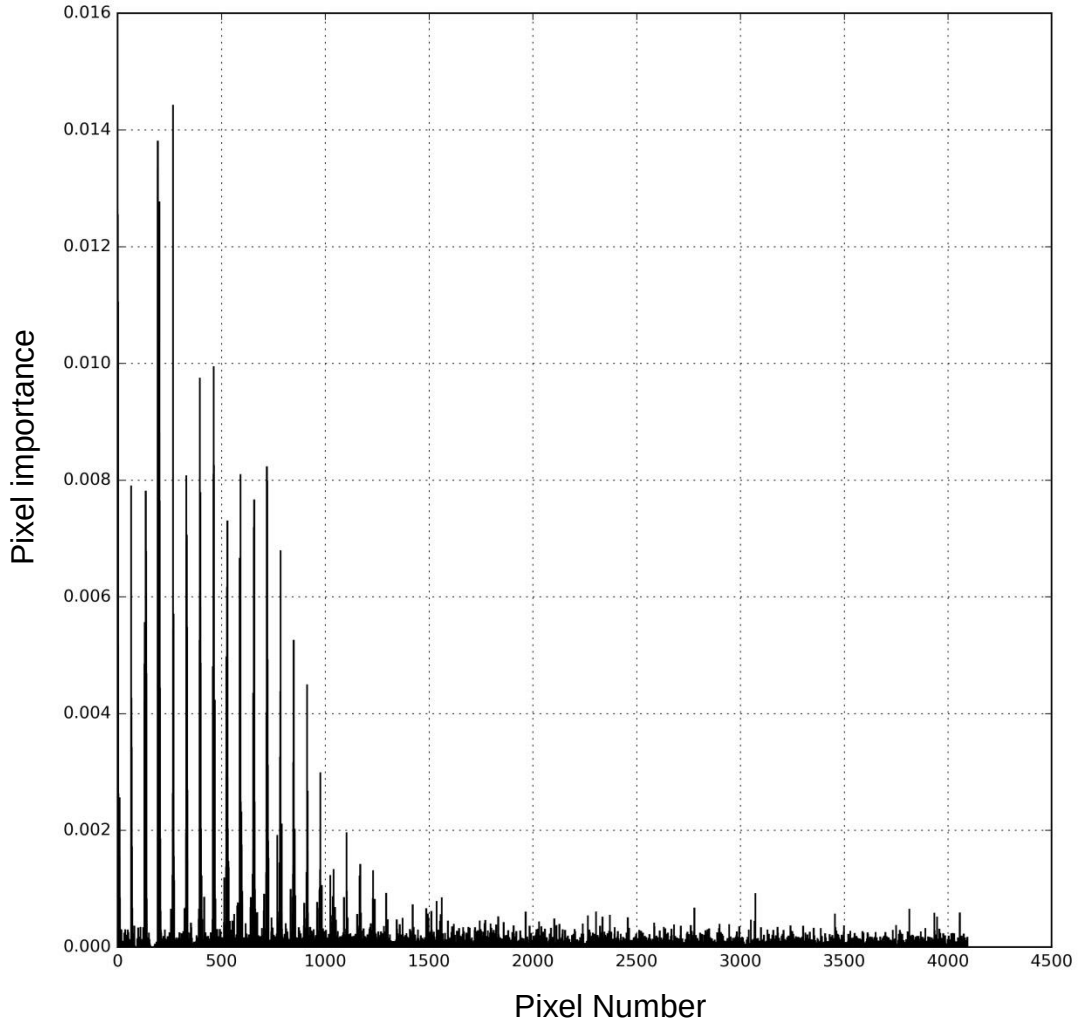
5-fold CV Predictions on training data (scarcity)

		True	
		broken (0)	working (1)
Predicted	broken (0)	32	20
	working (1)	6	1985

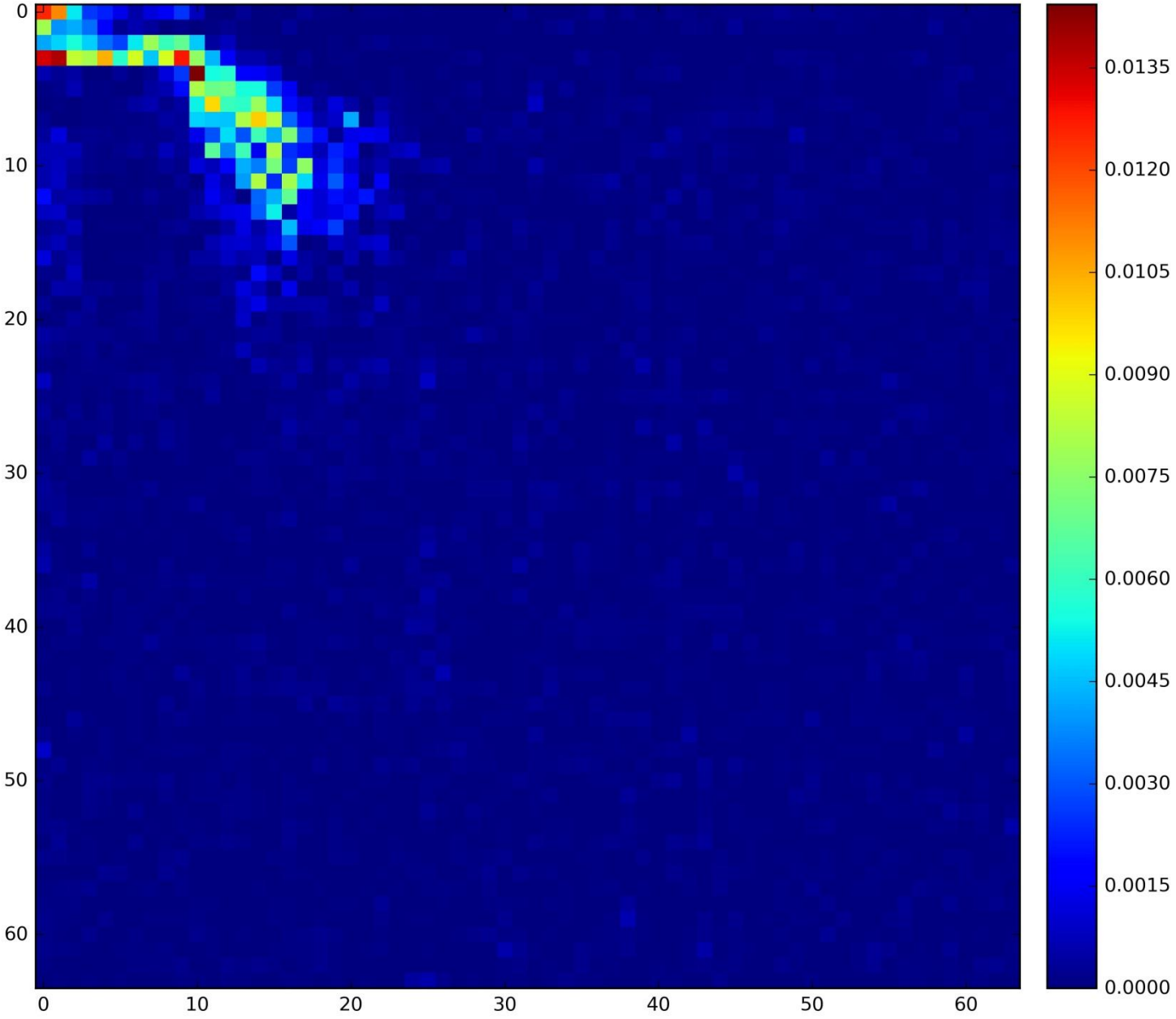
	precision	recall	f1-score	support
0	0.84	0.62	0.71	52
1	0.99	1.00	0.99	1991
avg / total	0.99	0.99	0.99	2043

Classification with Random Forest (RF)

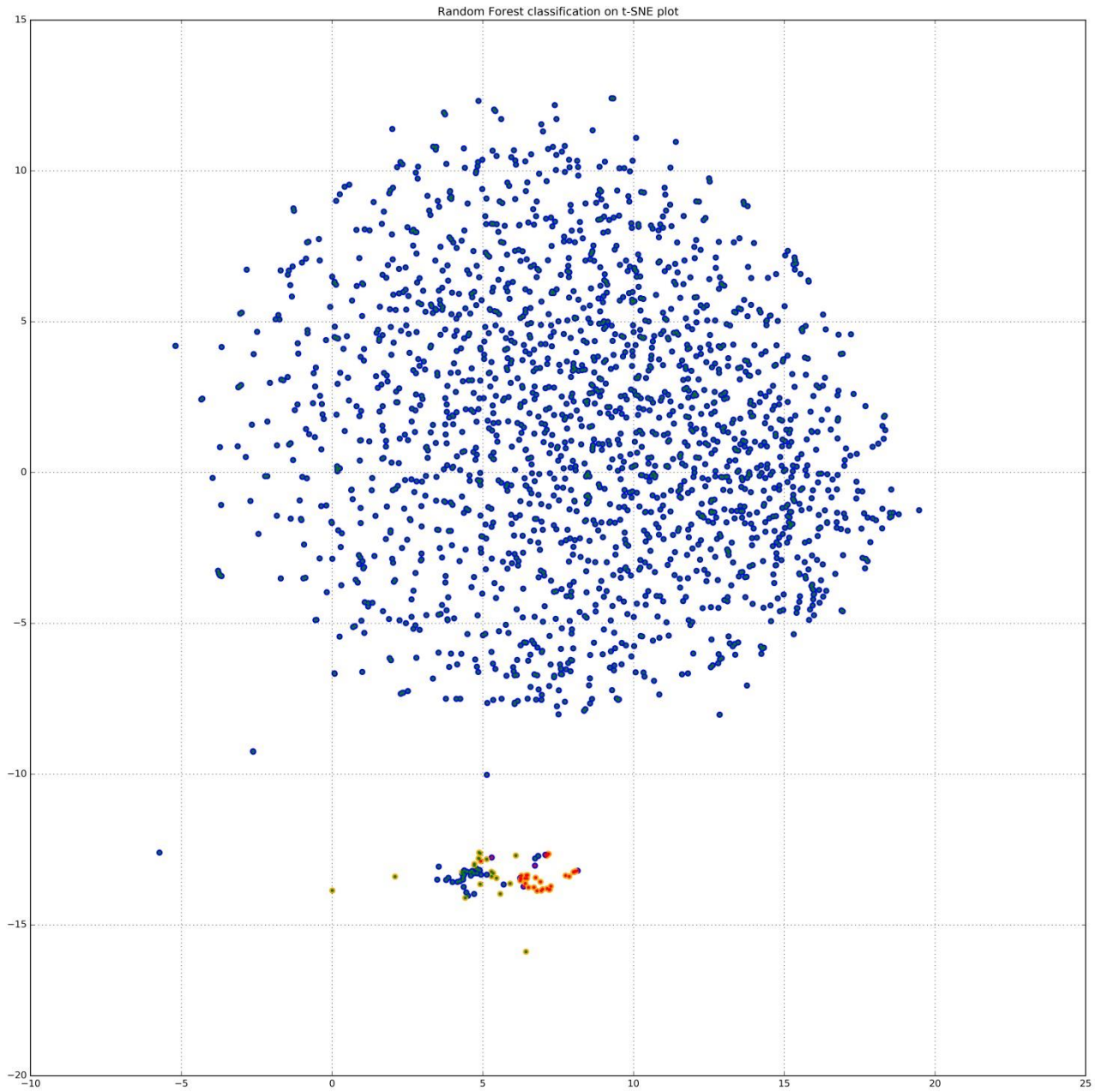
- We consider variable importance (how important is each pixel (variable) in predicting the quality class).



Which pixels are most important in classification?

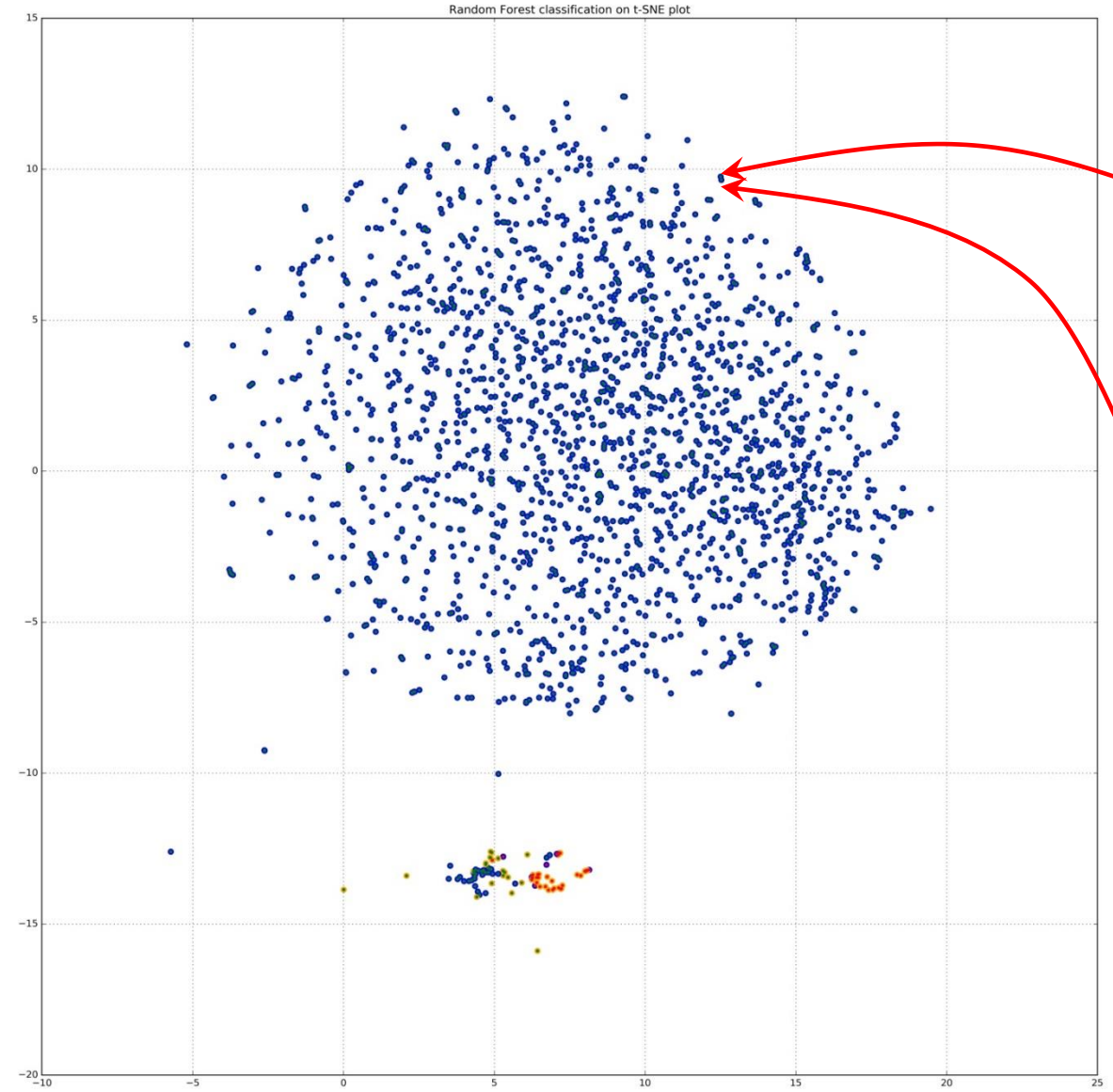


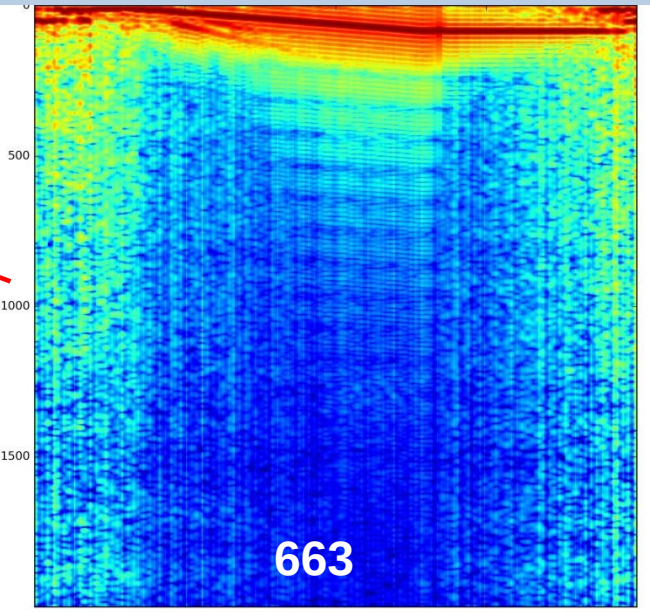
Overlay RF prediction on t-SNE

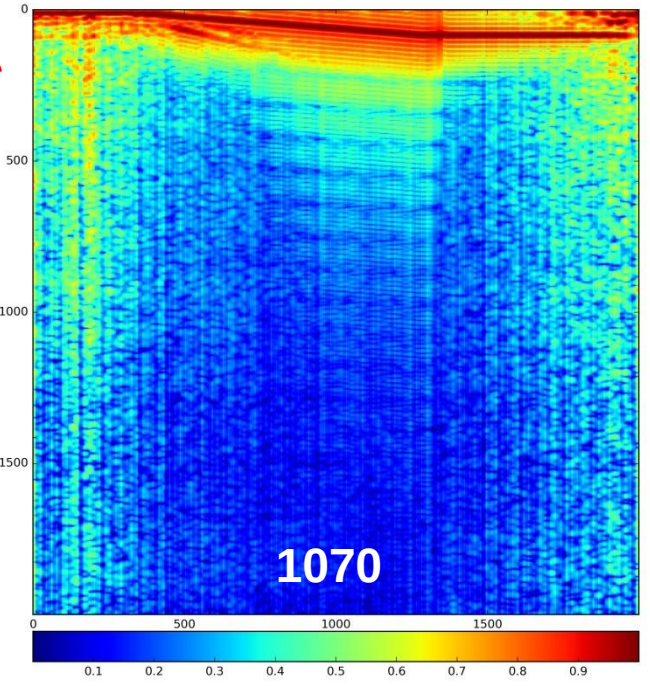


		True	
		0	1
Predicted	0		
	1		

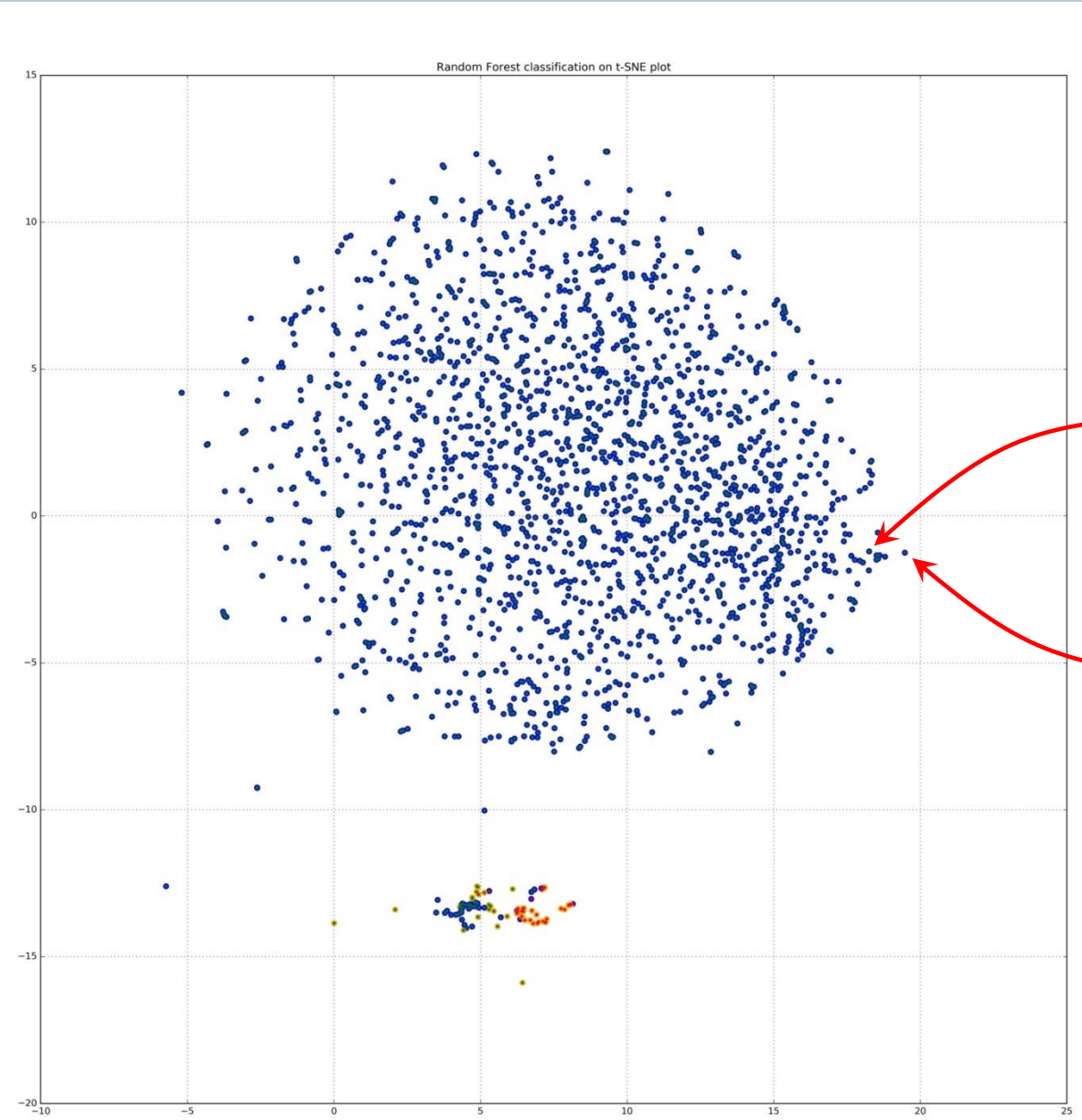
Overlay RF prediction on t-SNE

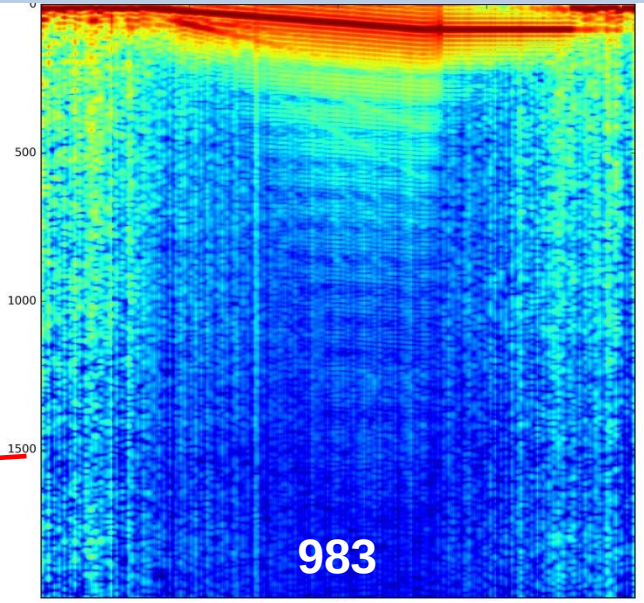


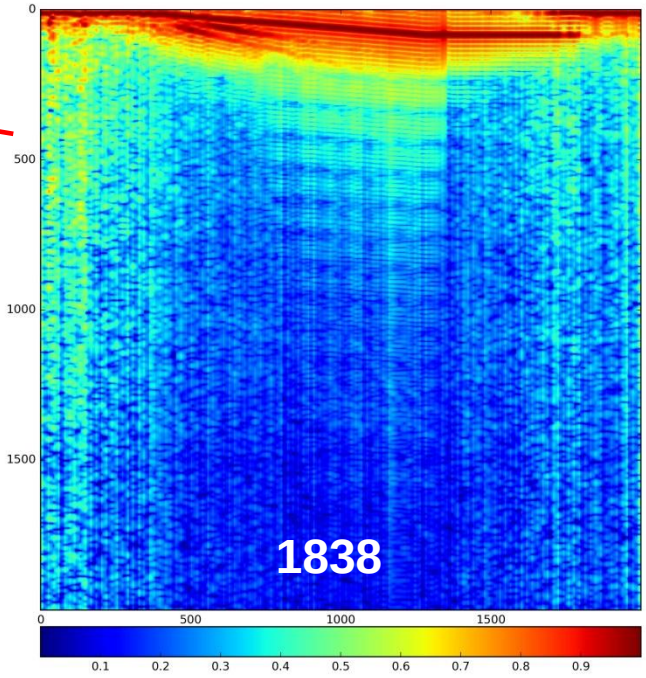




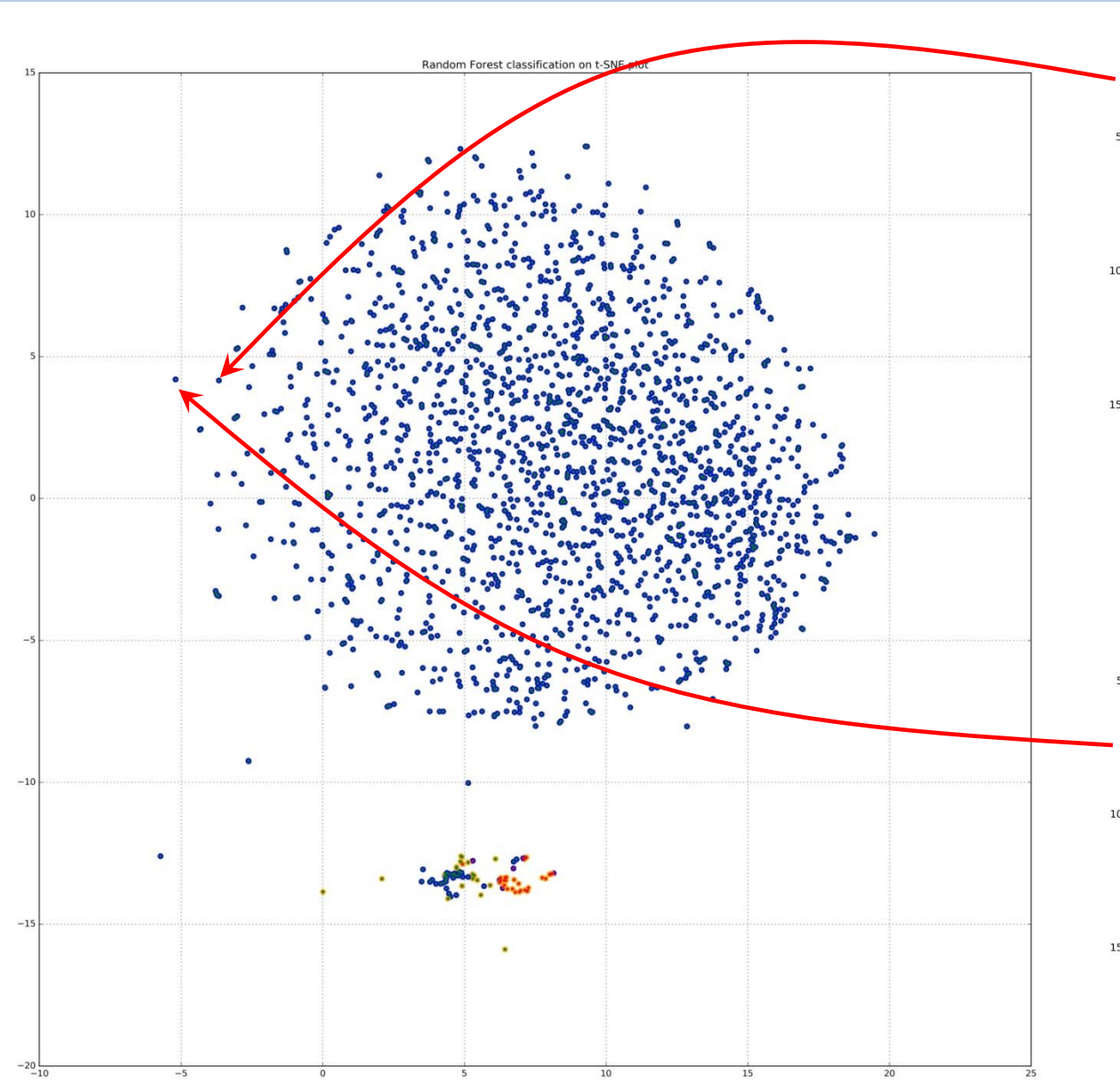
Overlay RF prediction on t-SNE

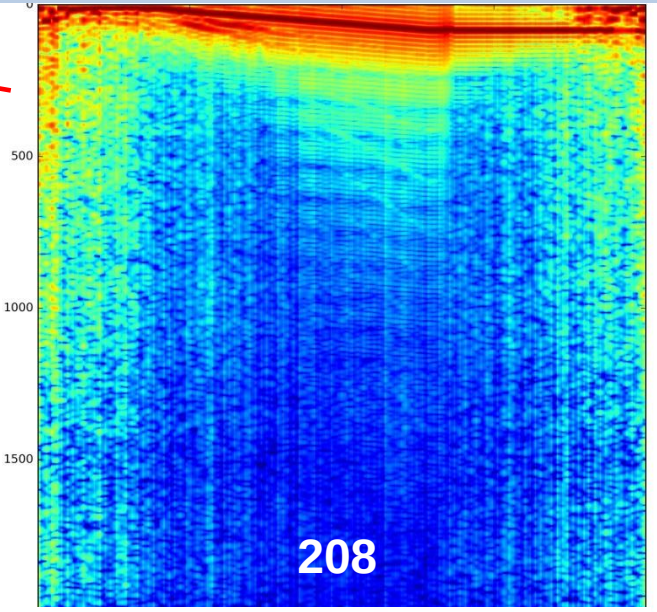


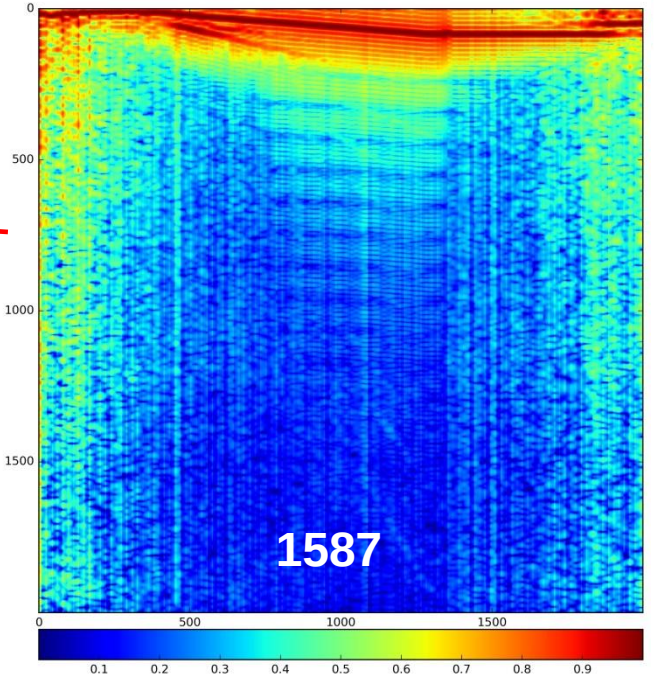




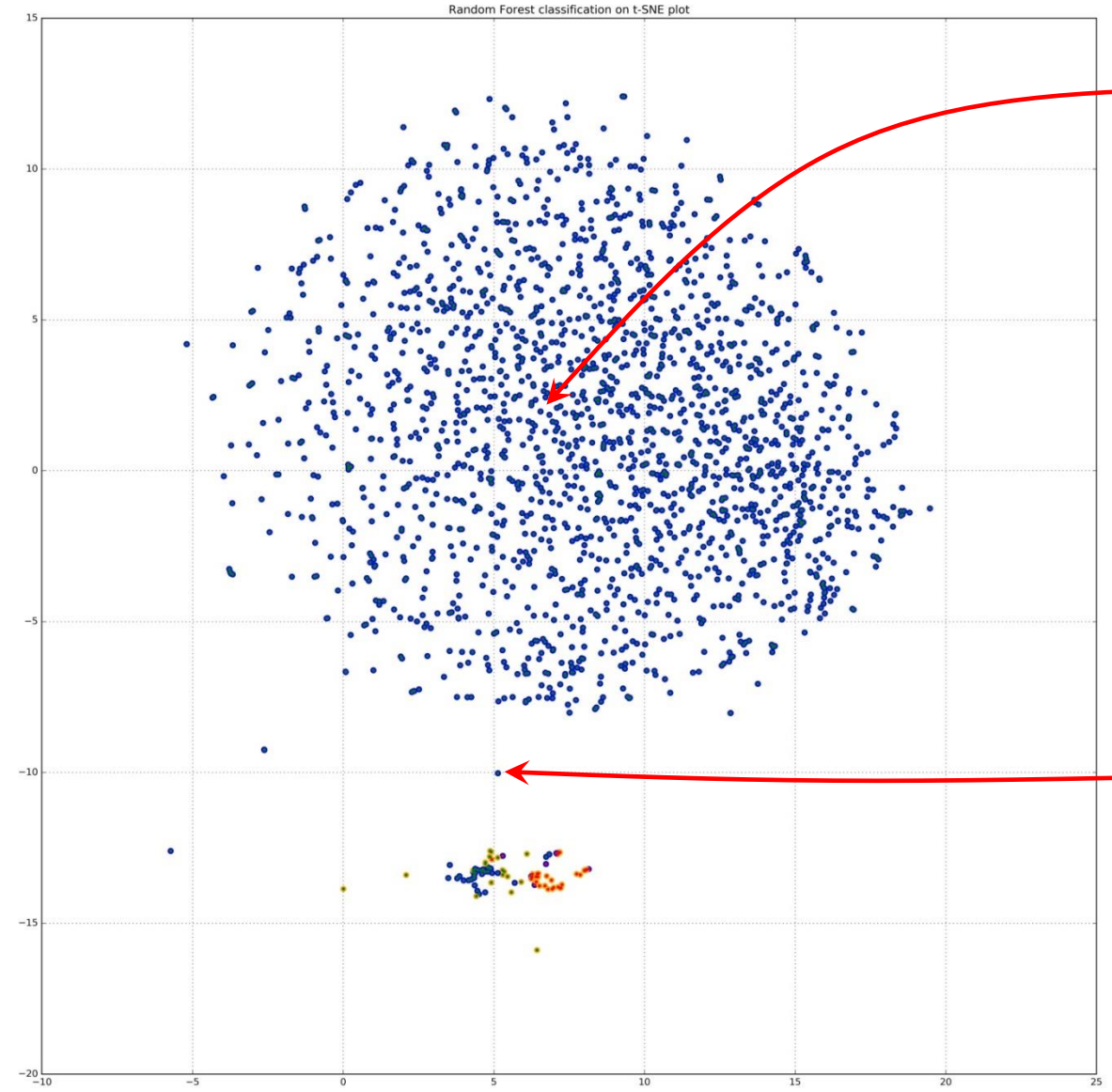
Overlay RF prediction on t-SNE

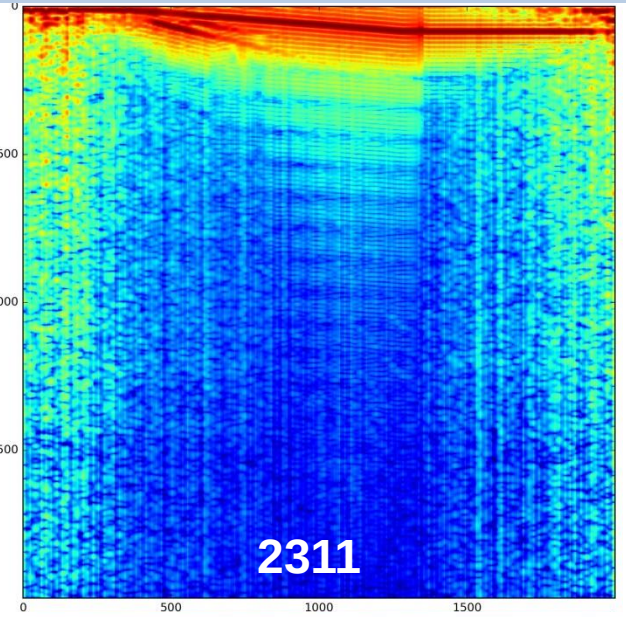


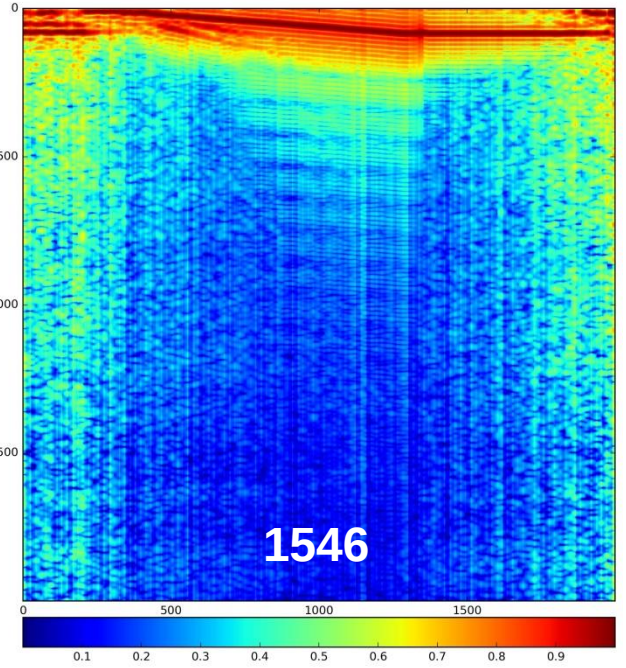




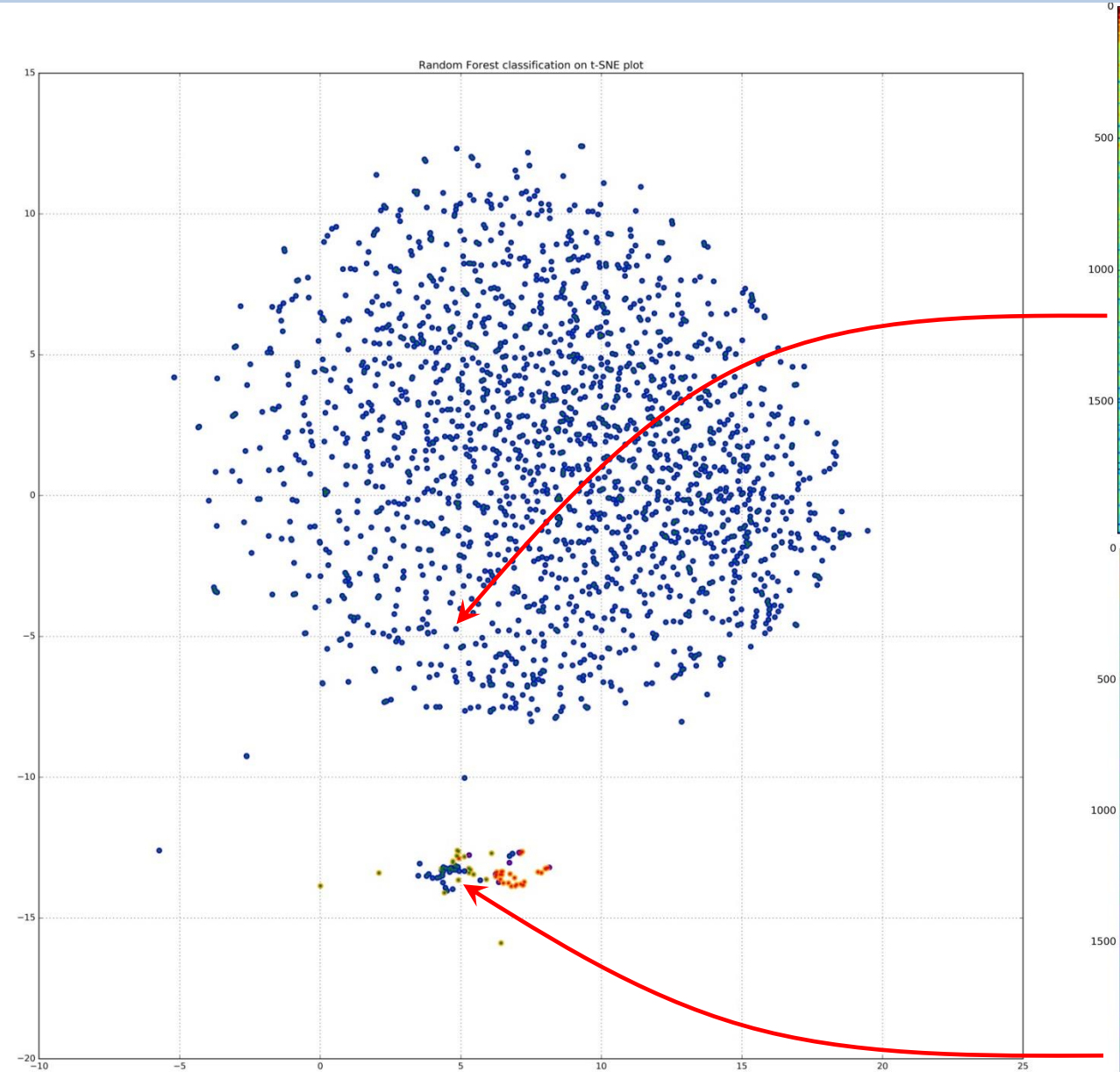
Overlay RF prediction on t-SNE

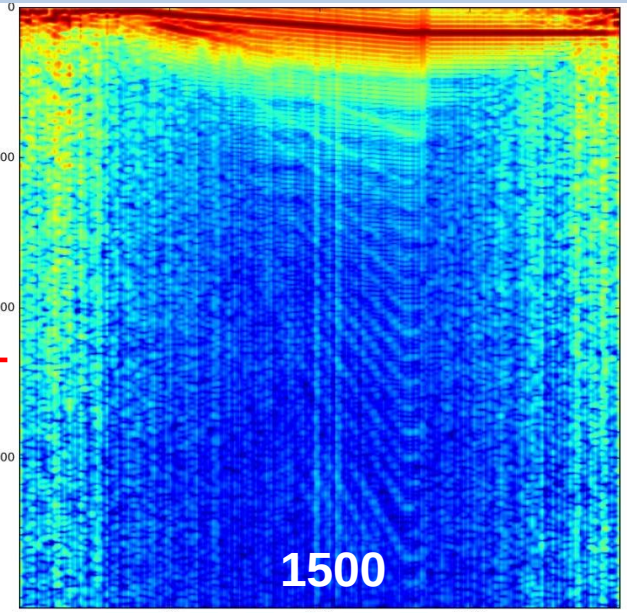


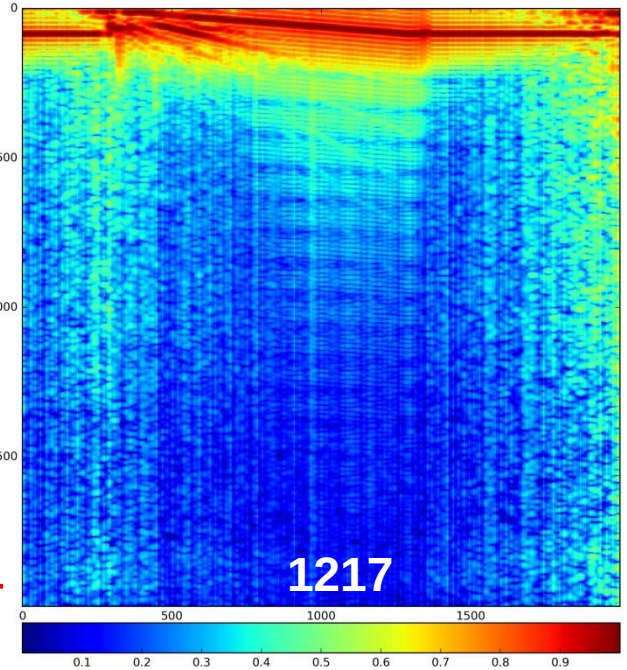




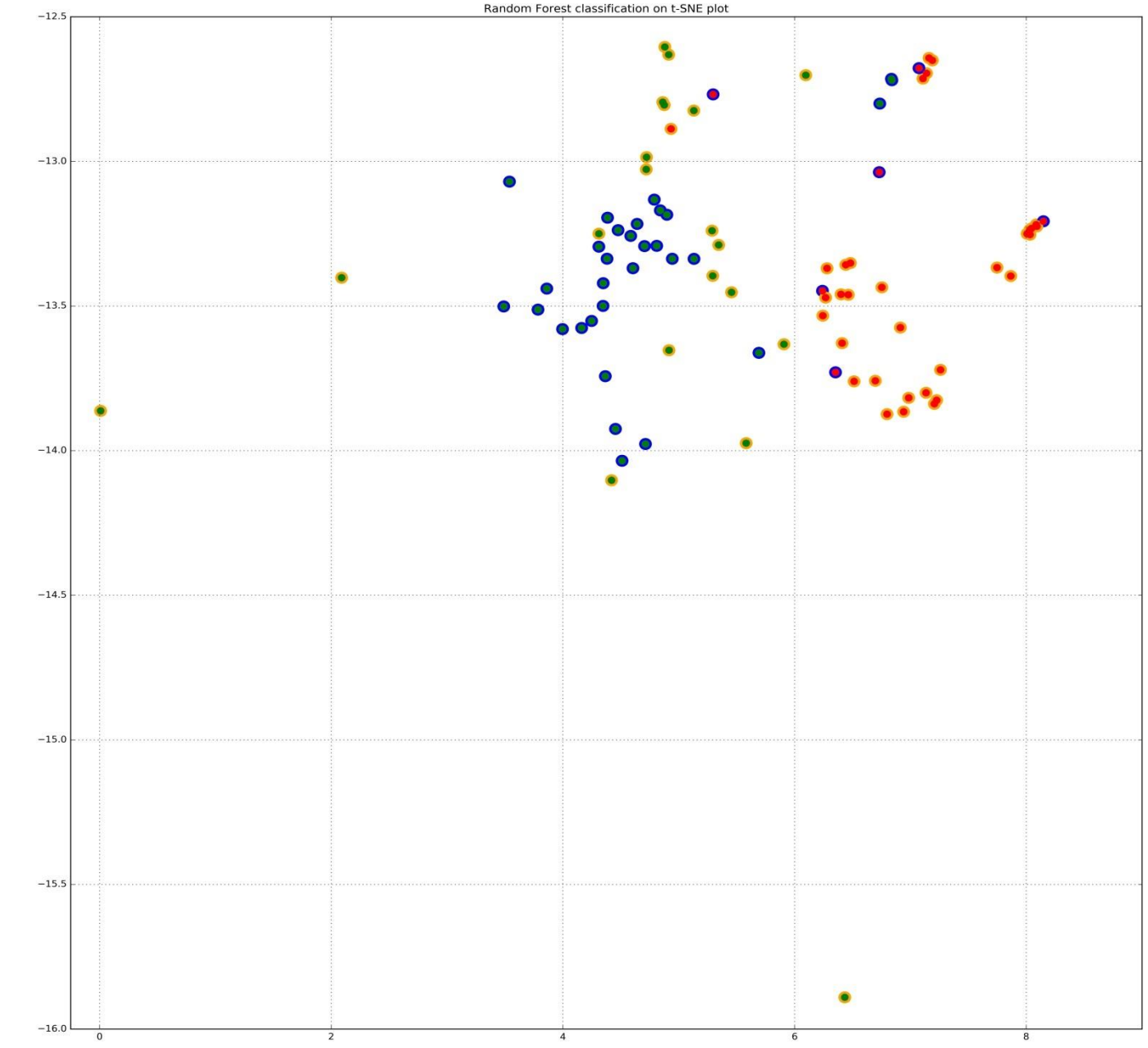
Overlay RF prediction on t-SNE





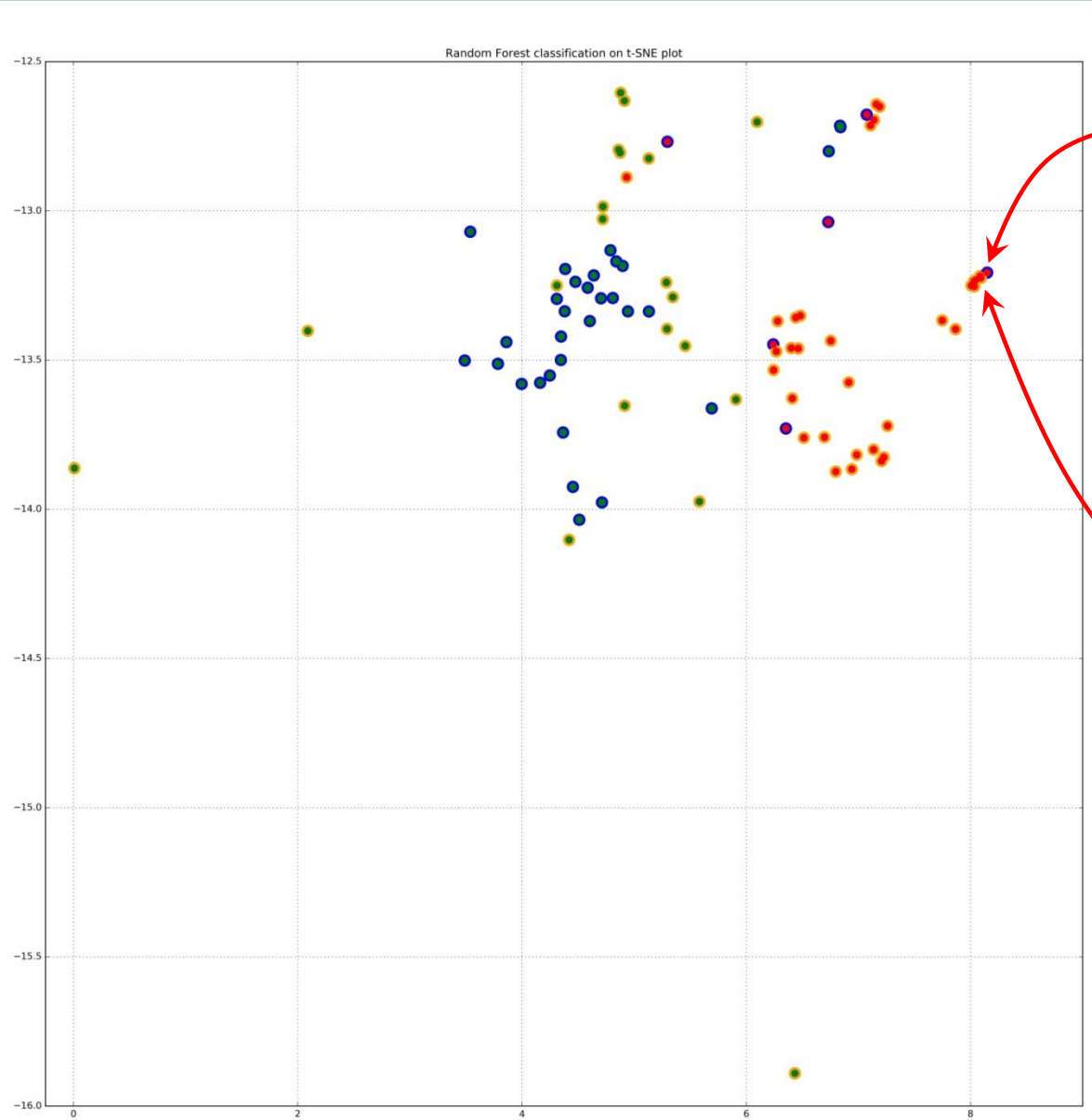


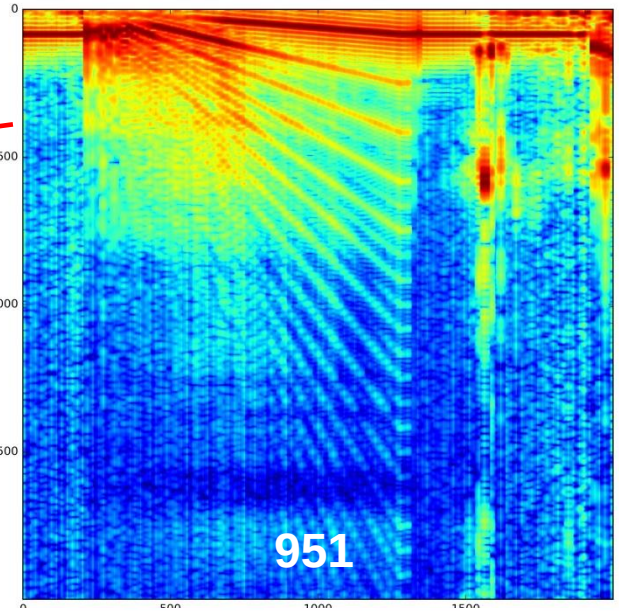
Overlay RF prediction on t-SNE (zoomed)

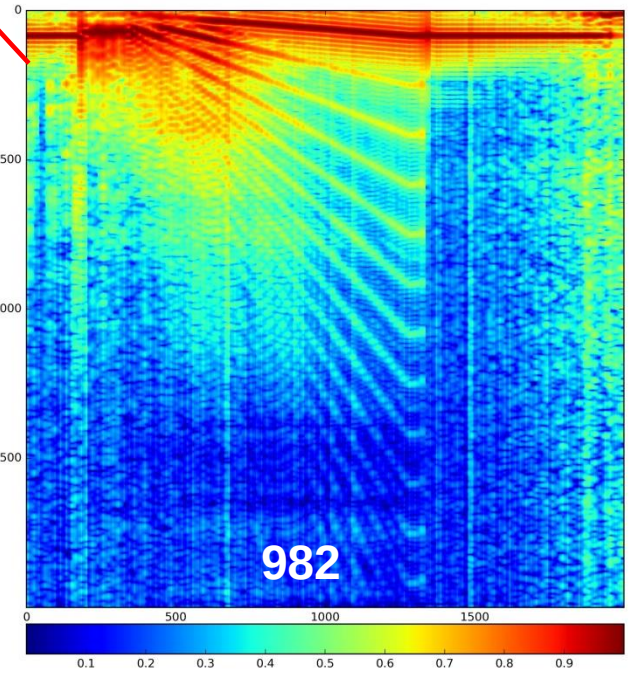


		True	
		0	1
Predicted	0		
	1		

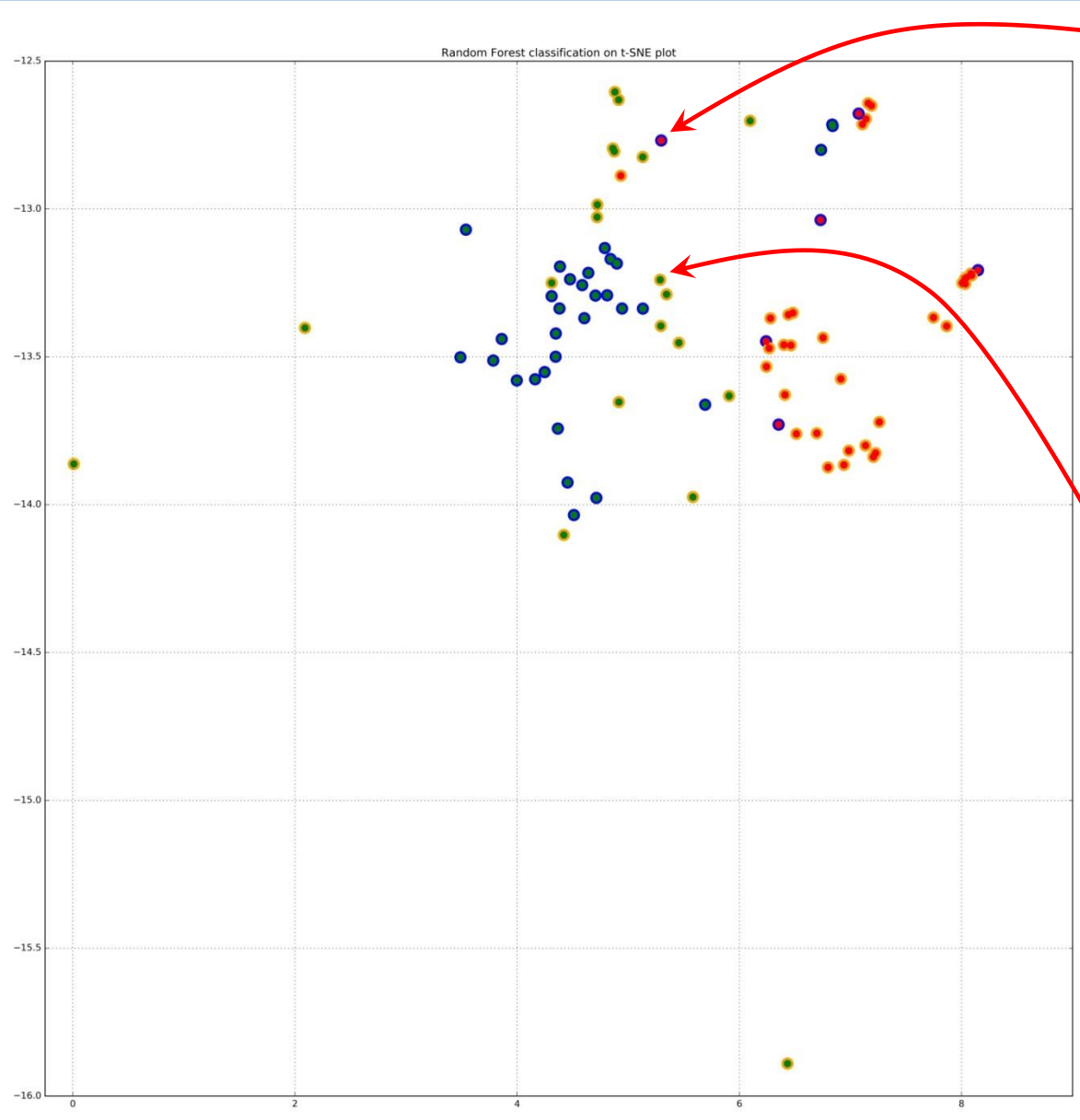
Overlay RF prediction on t-SNE (zoomed)

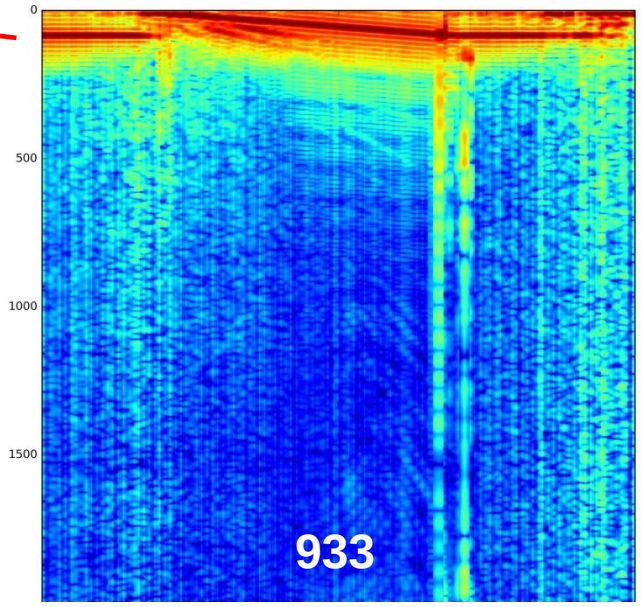


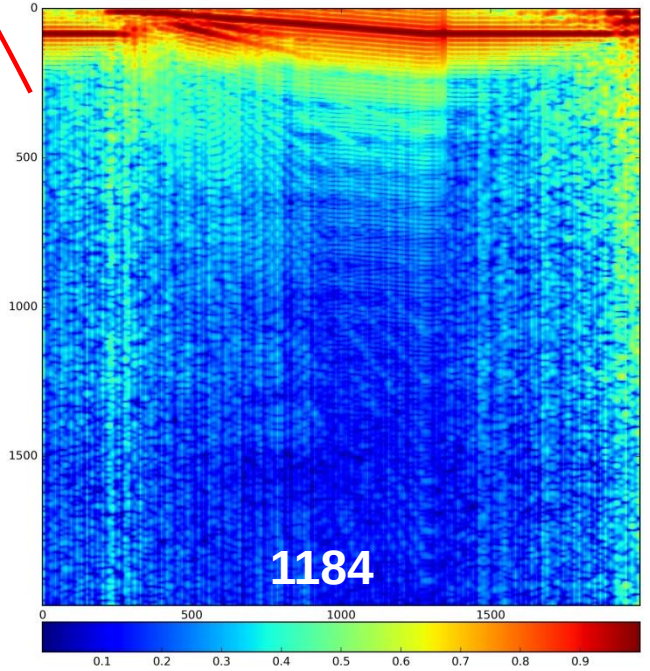




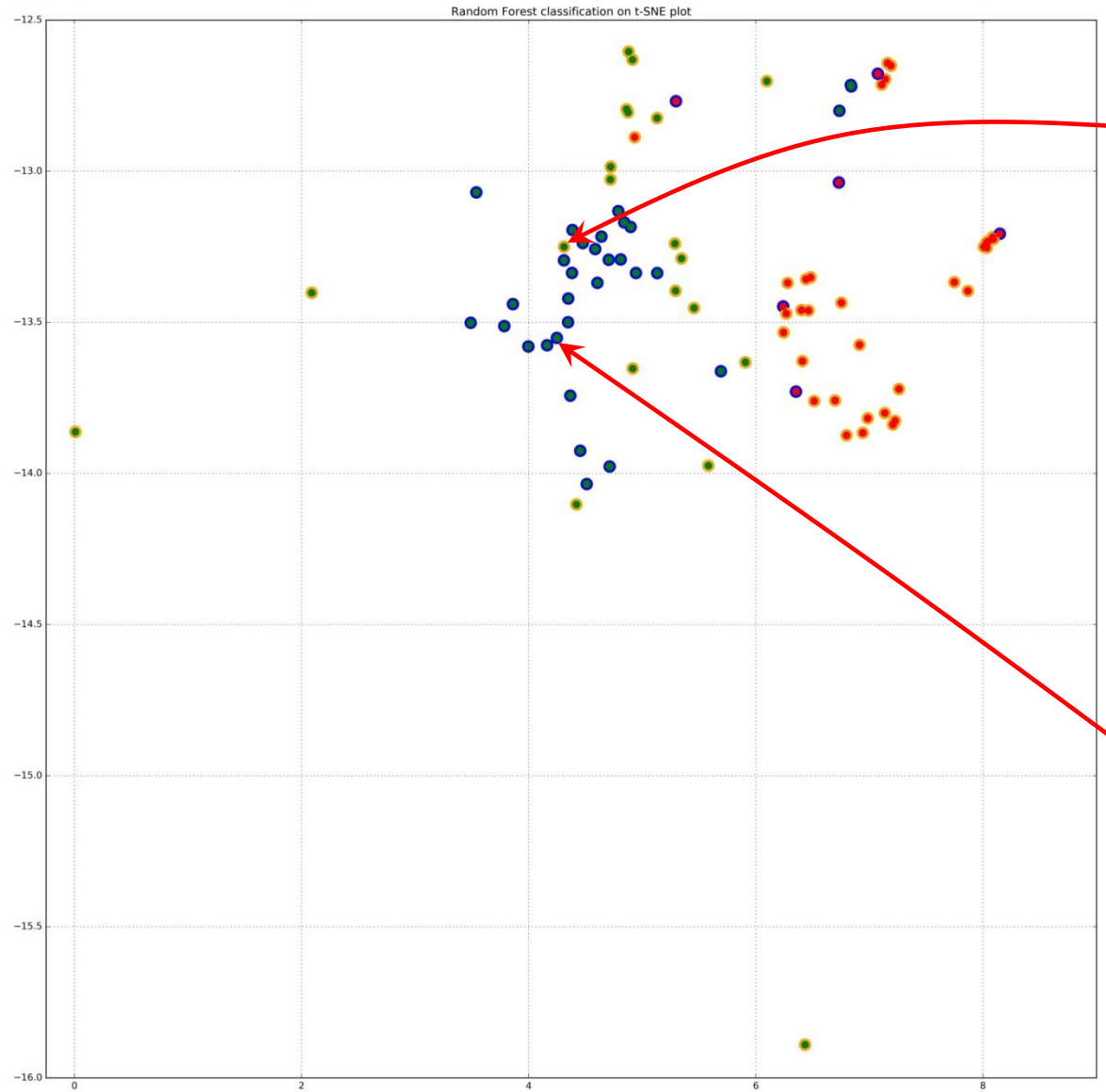
Overlay RF prediction on t-SNE (zoomed)

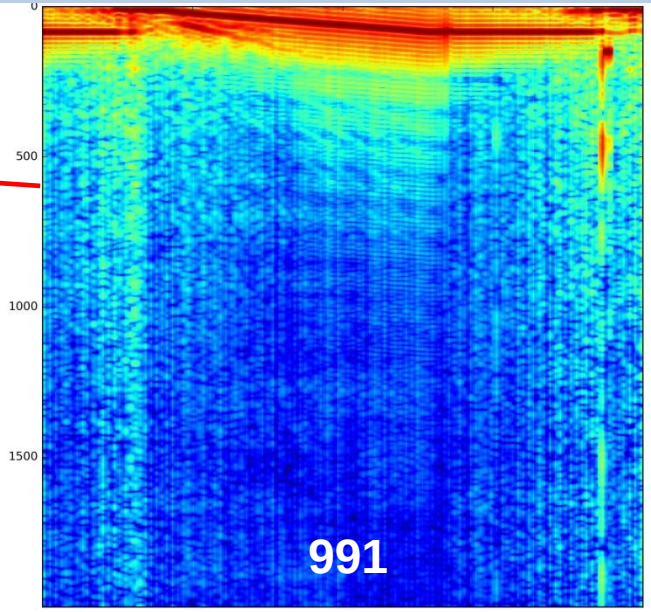


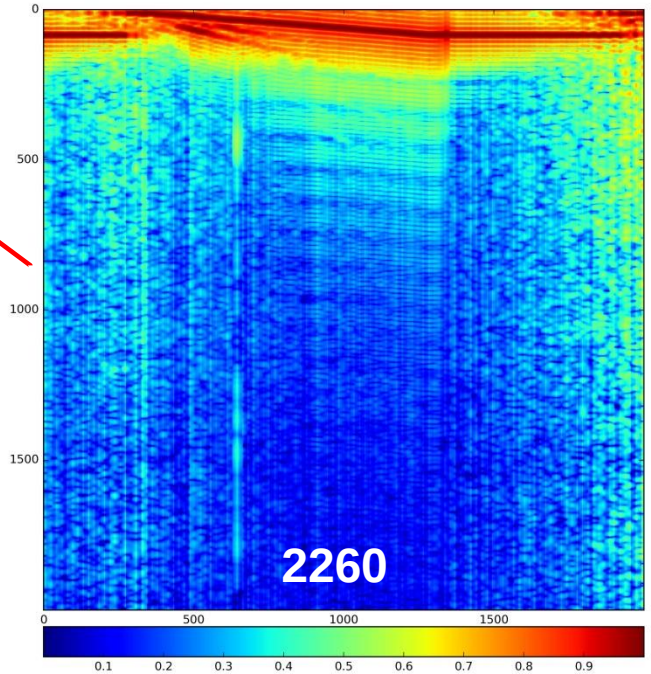




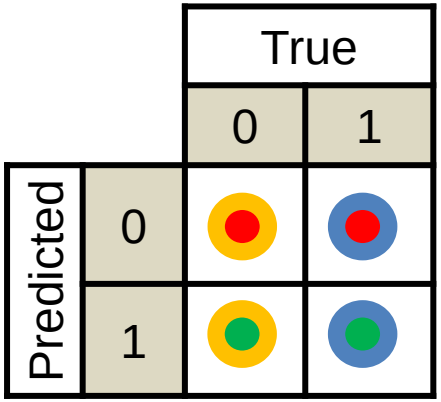
Overlay RF prediction on t-SNE (zoomed)



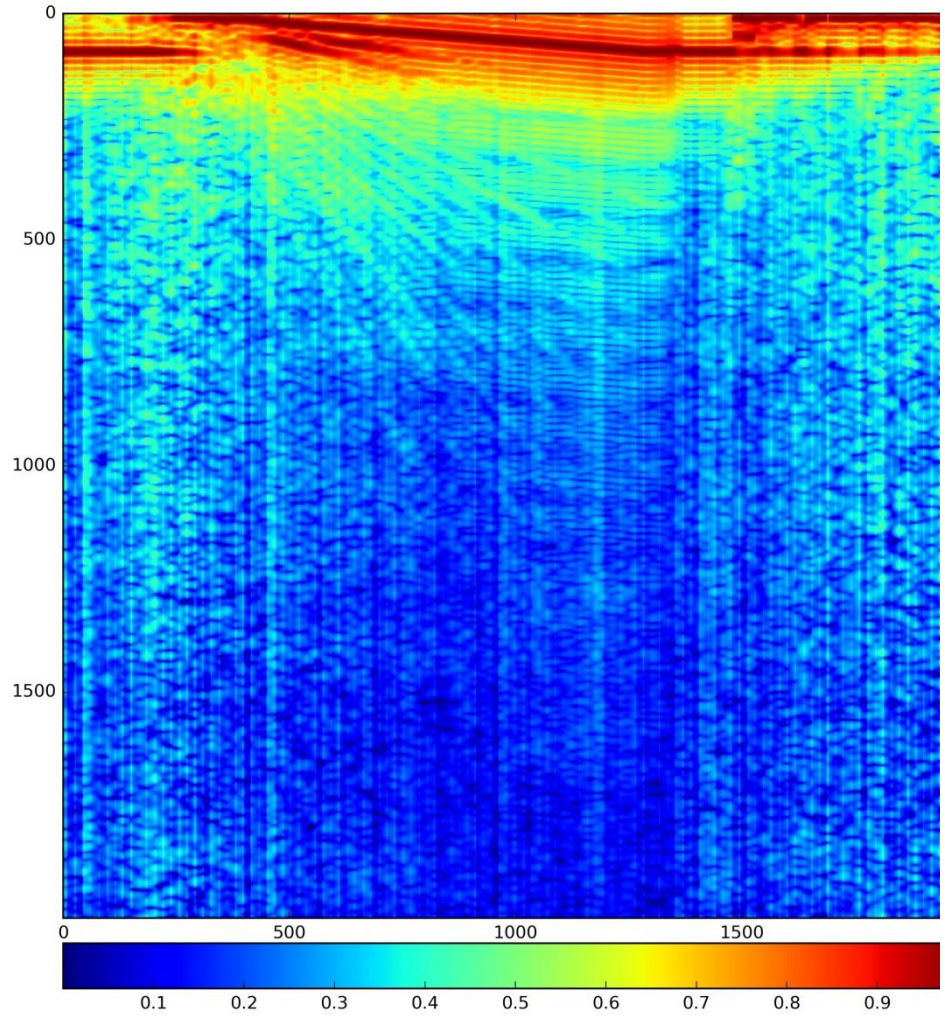




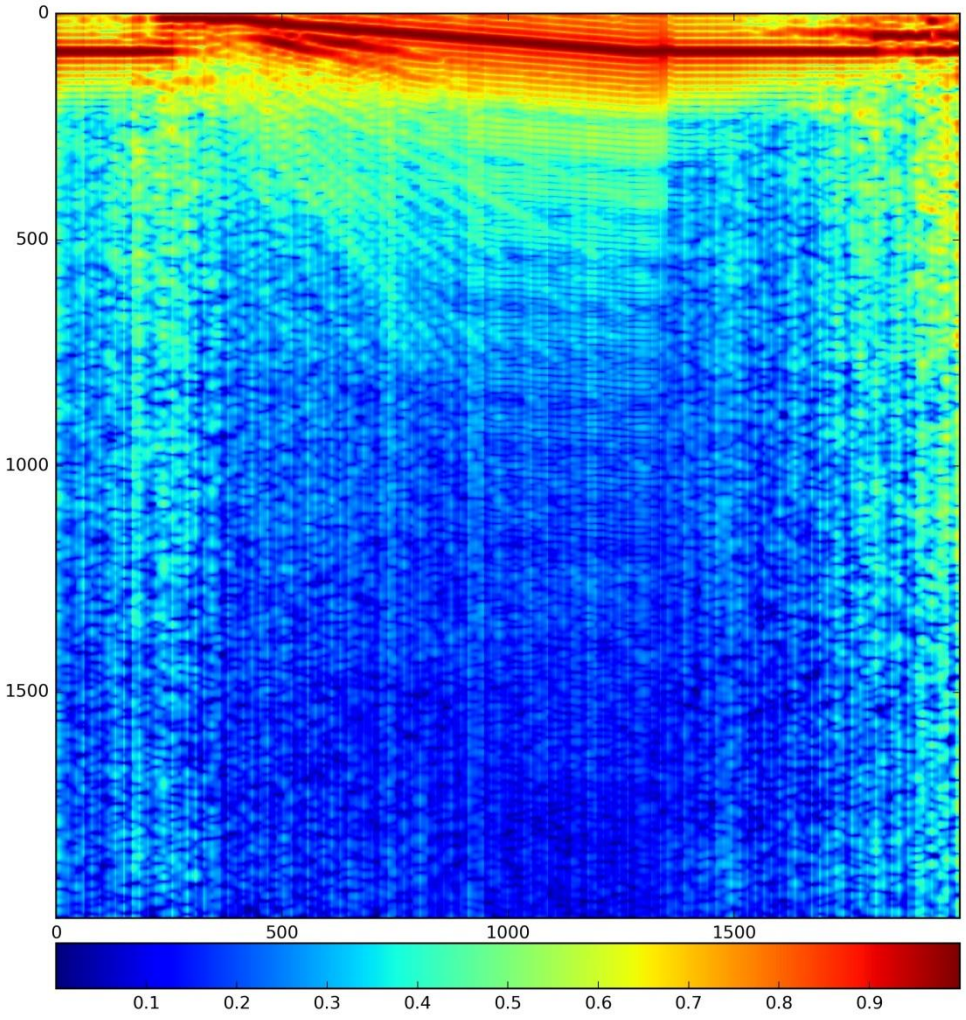
Now let us have consider spectrograms from each class, and if the class was predicted correctly.



RF predictions vs. labels. Predicted = 0, true = 0

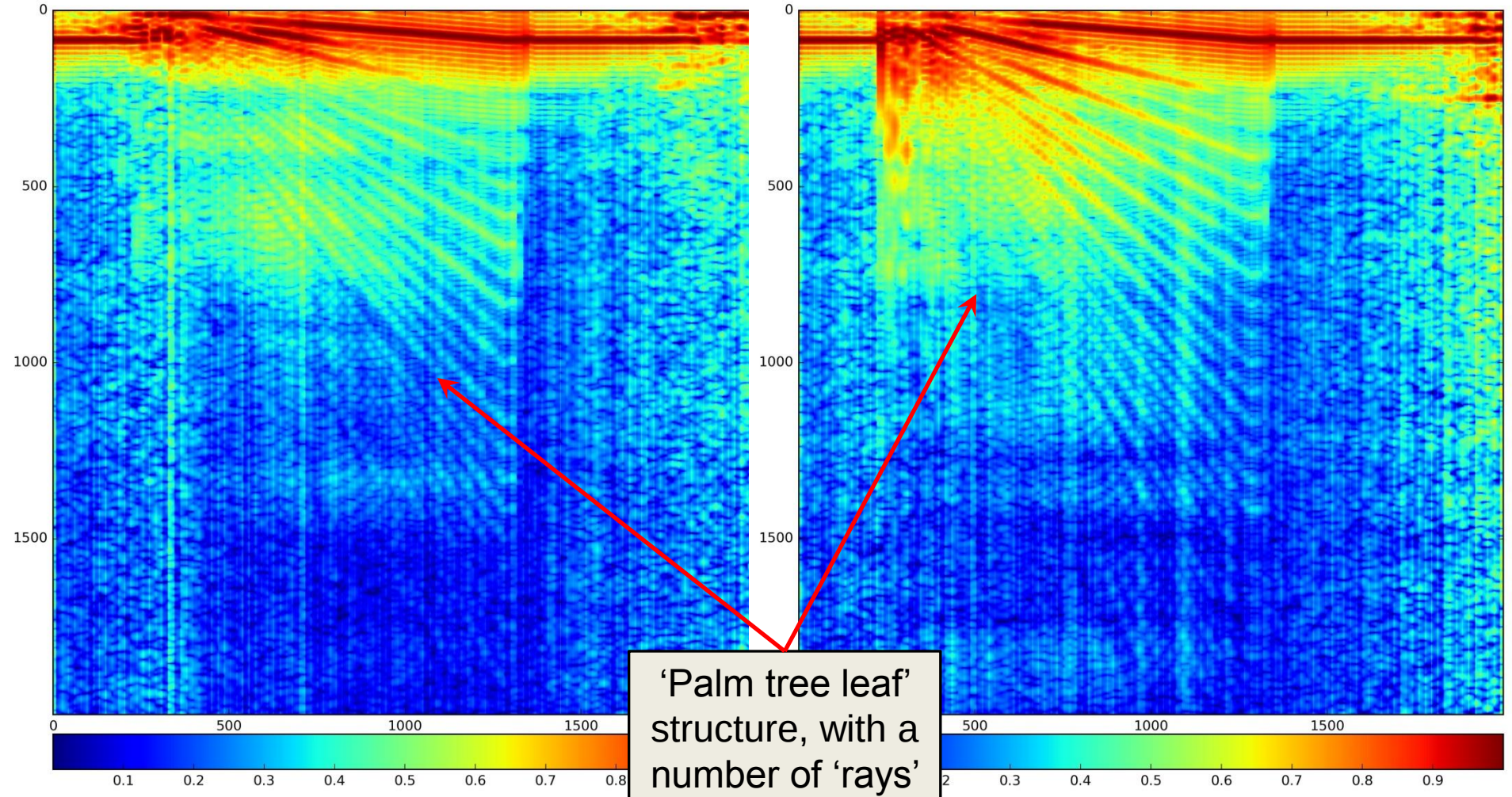


58.wav



59.wav

RF predictions vs. labels. Predicted = 0, true = 0

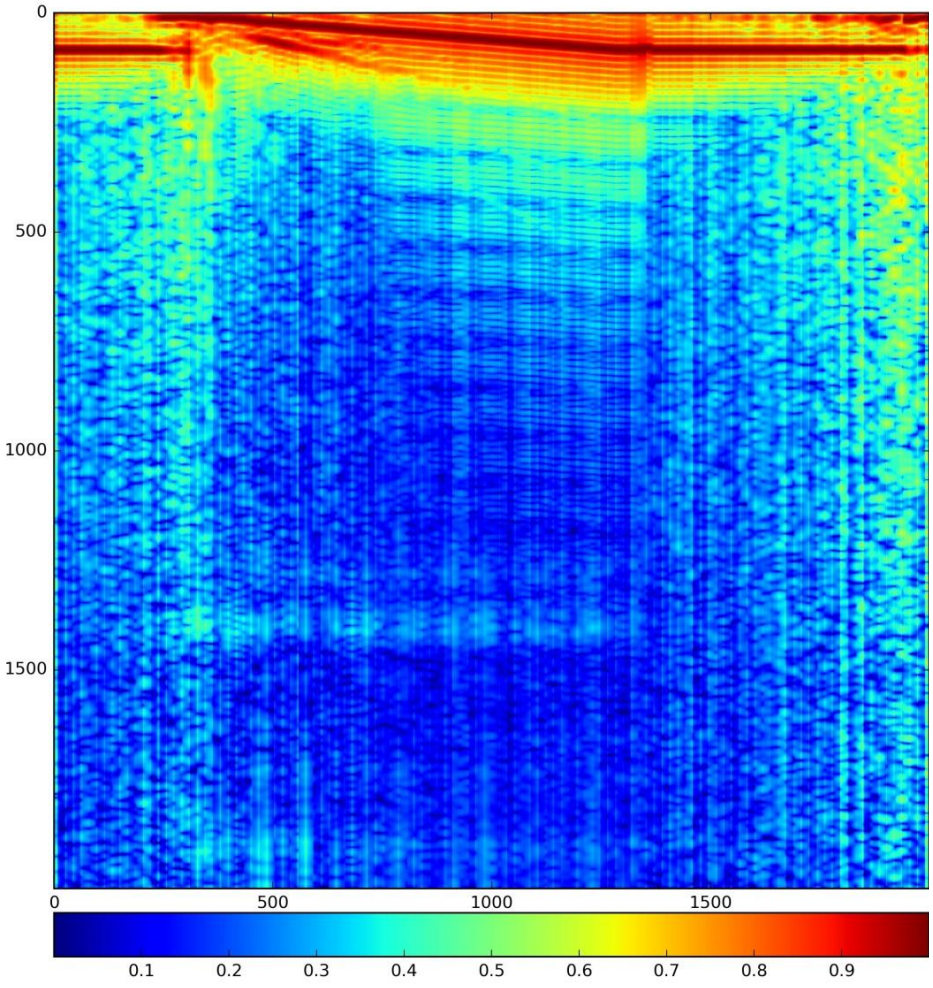


66.wav

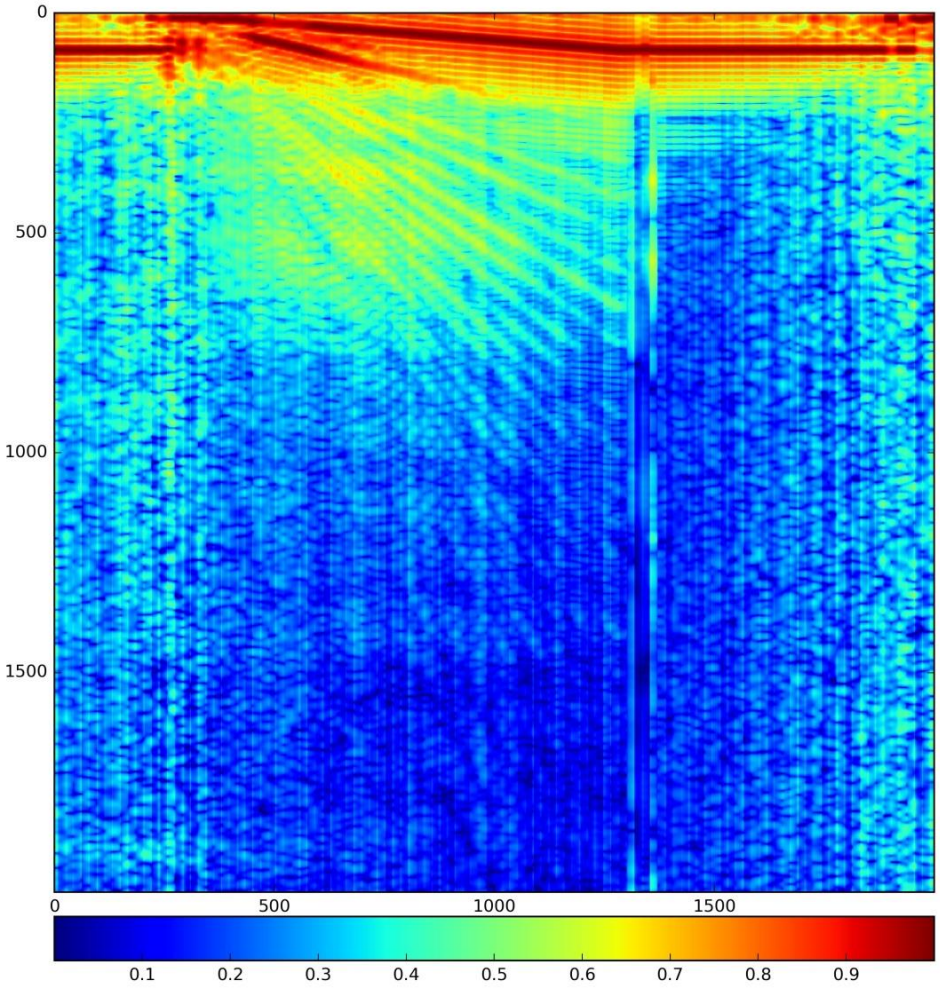
‘Palm tree leaf’ structure, with a number of ‘rays’ spanning many frequencies over longer times

473.wav

RF predictions vs. labels. Predicted = 0, true = 0

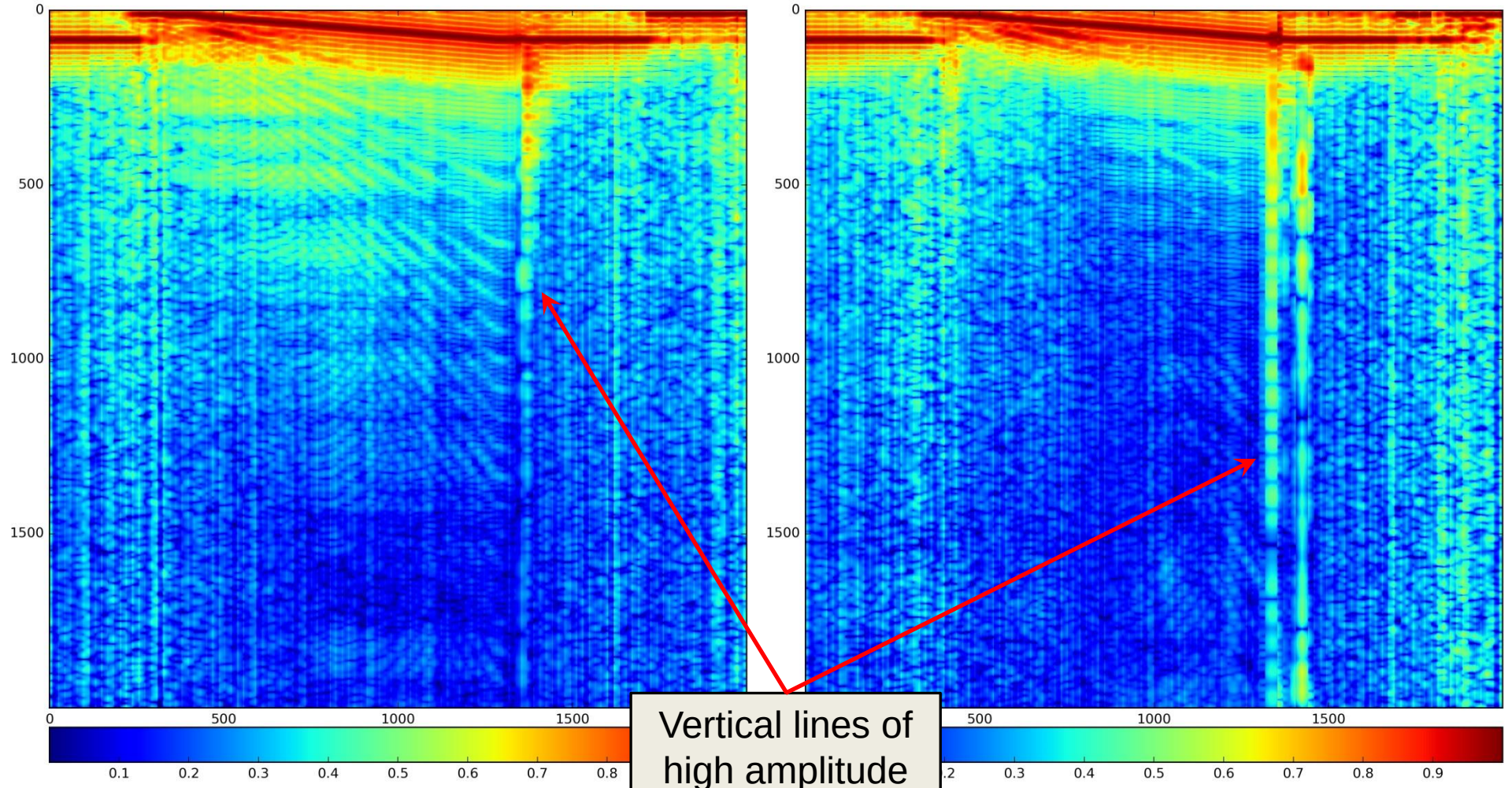


819.wav



830.wav

RF predictions vs. labels. Predicted = 0, true = 1

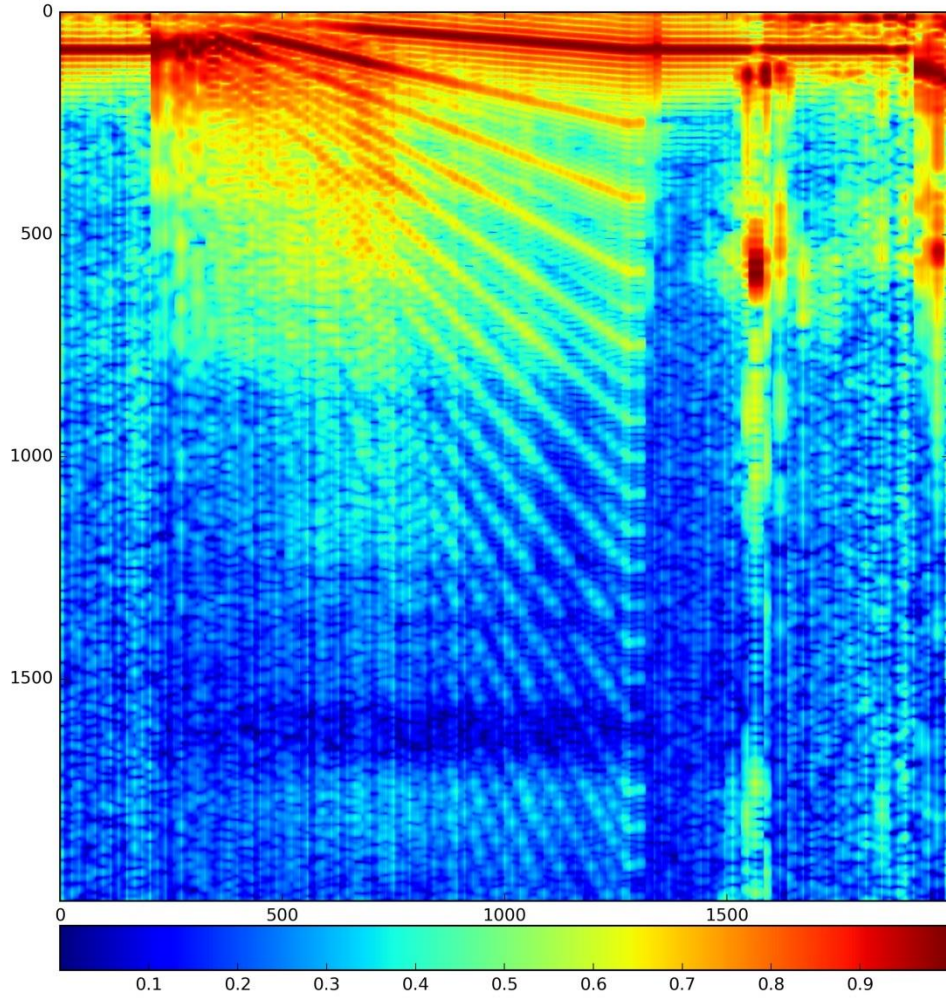


381.wav

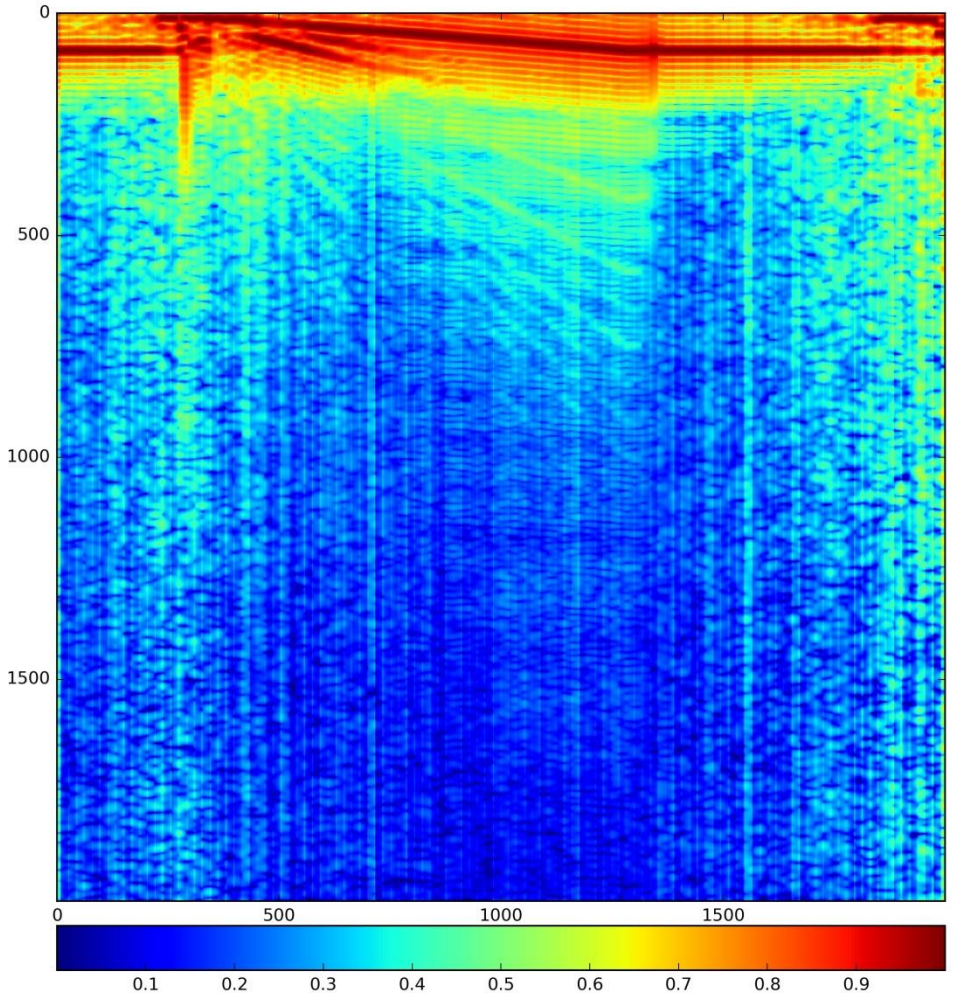
Vertical lines of high amplitude extending for all frequencies over short times

933.wav

RF predictions vs. labels. Predicted = 0, true = 1

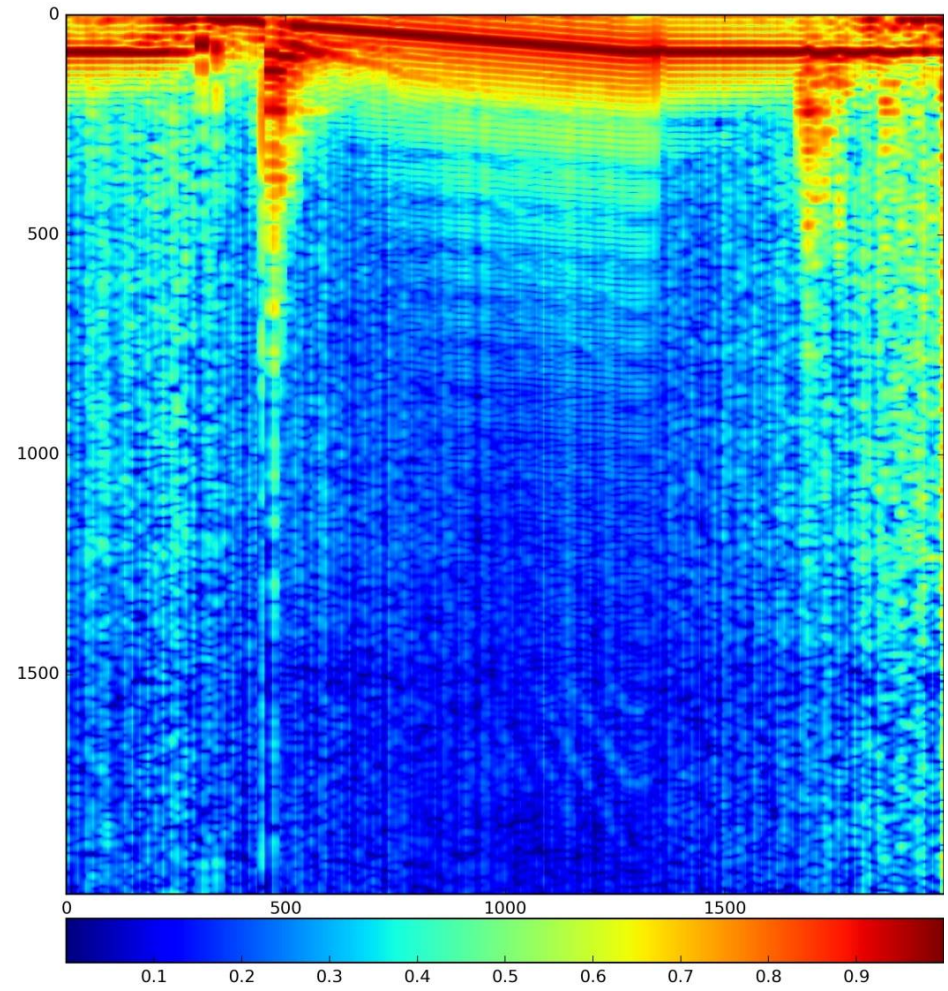


951.wav

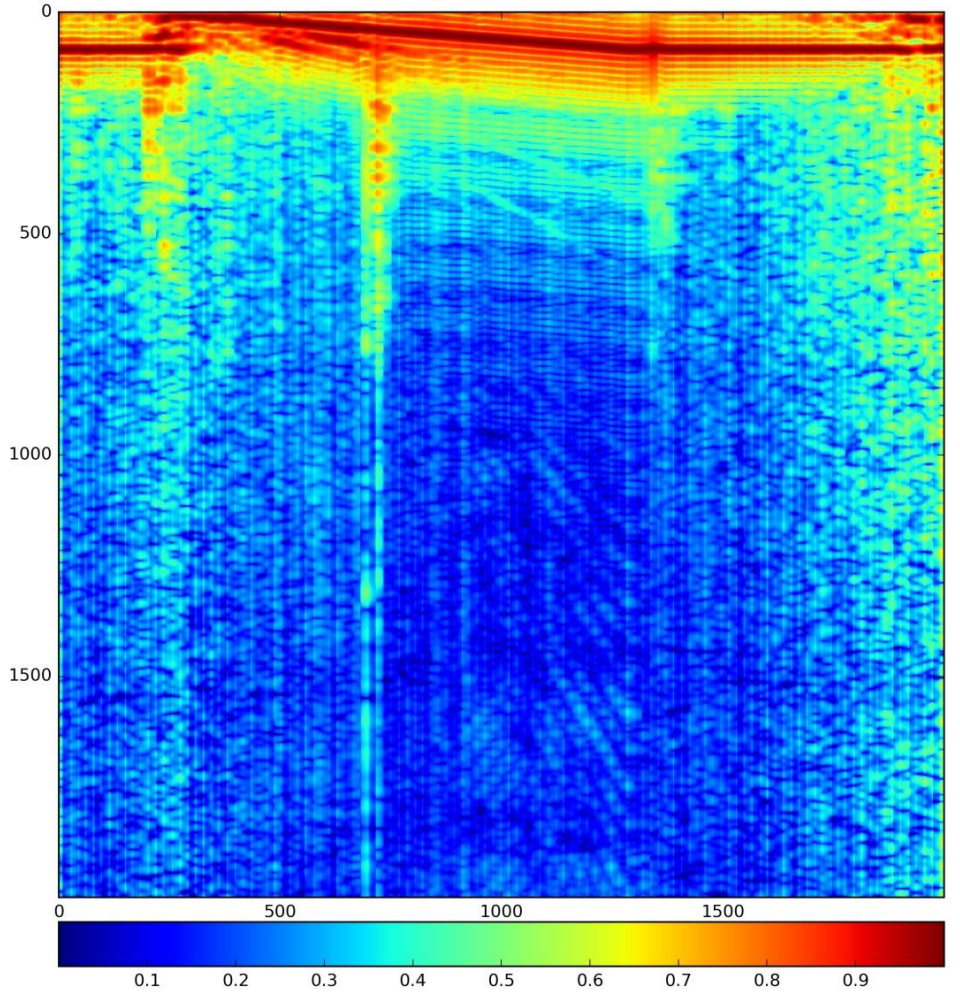


1145.wav

RF predictions vs. labels. Predicted = 1, true = 0

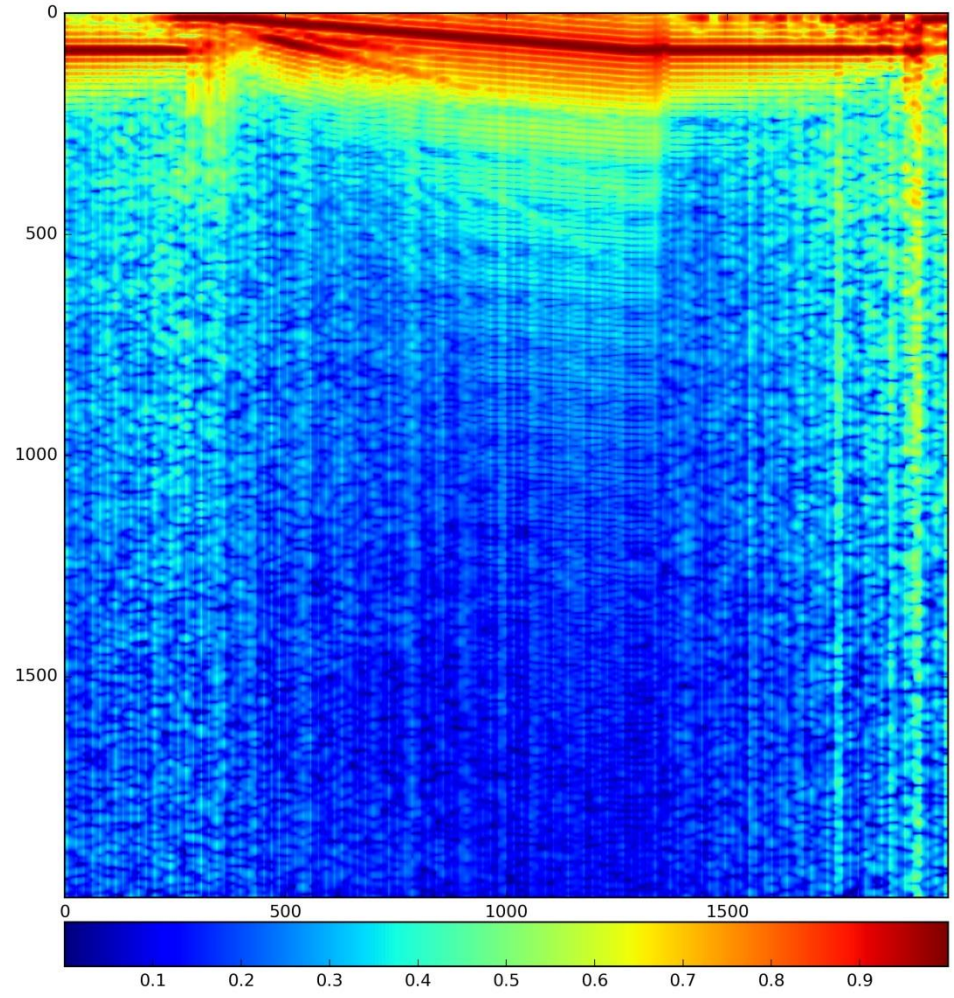


466.wav

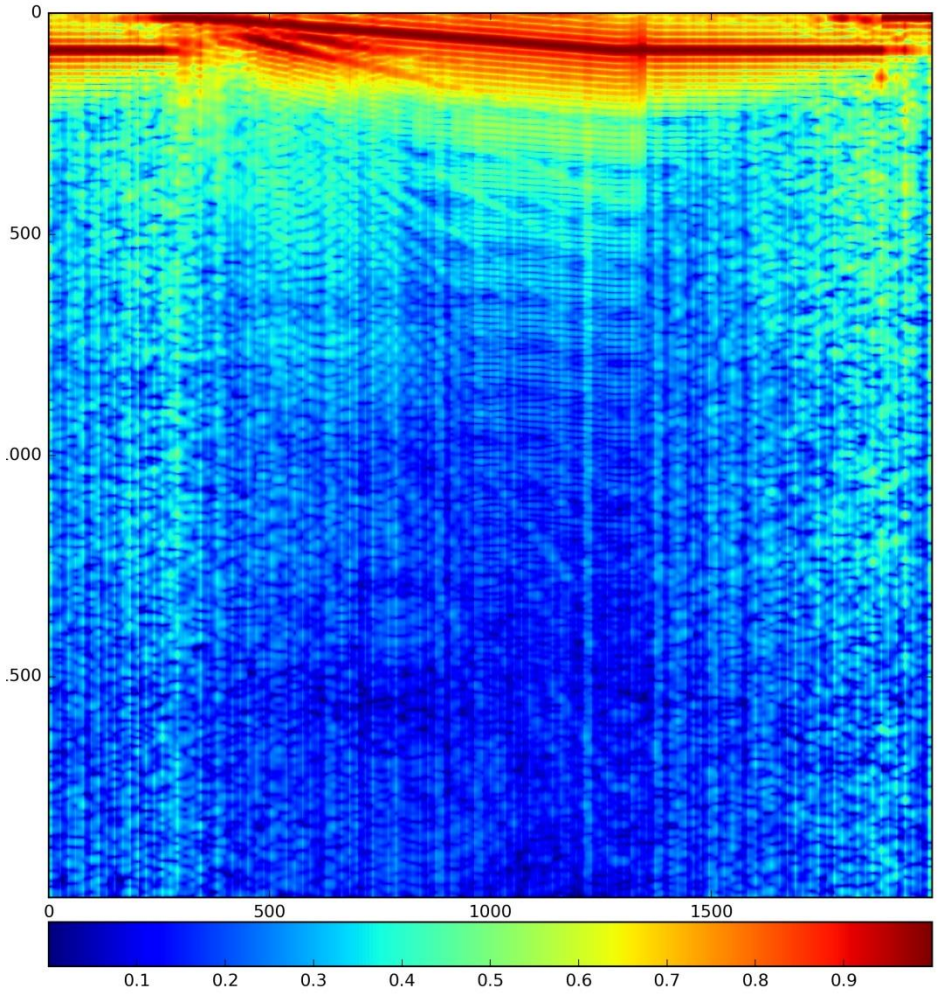


924.wav

RF predictions vs. labels. Predicted = 1, true = 0

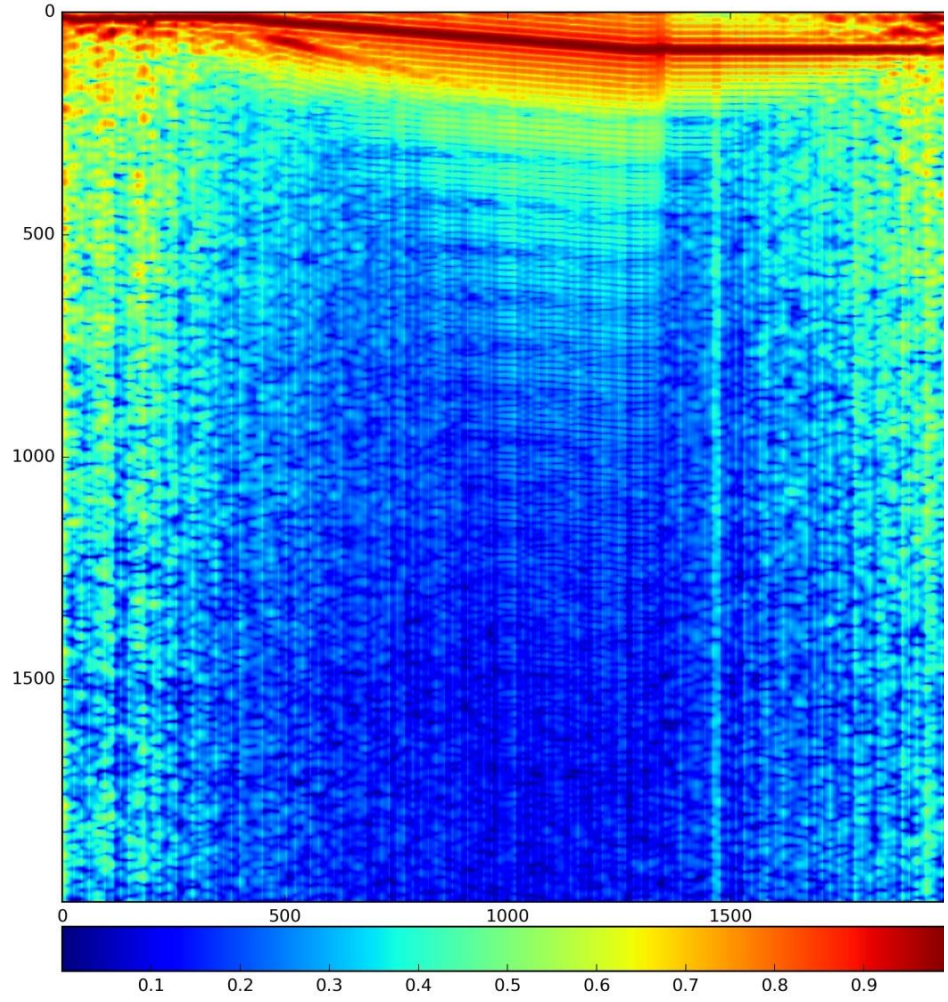


1292.wav

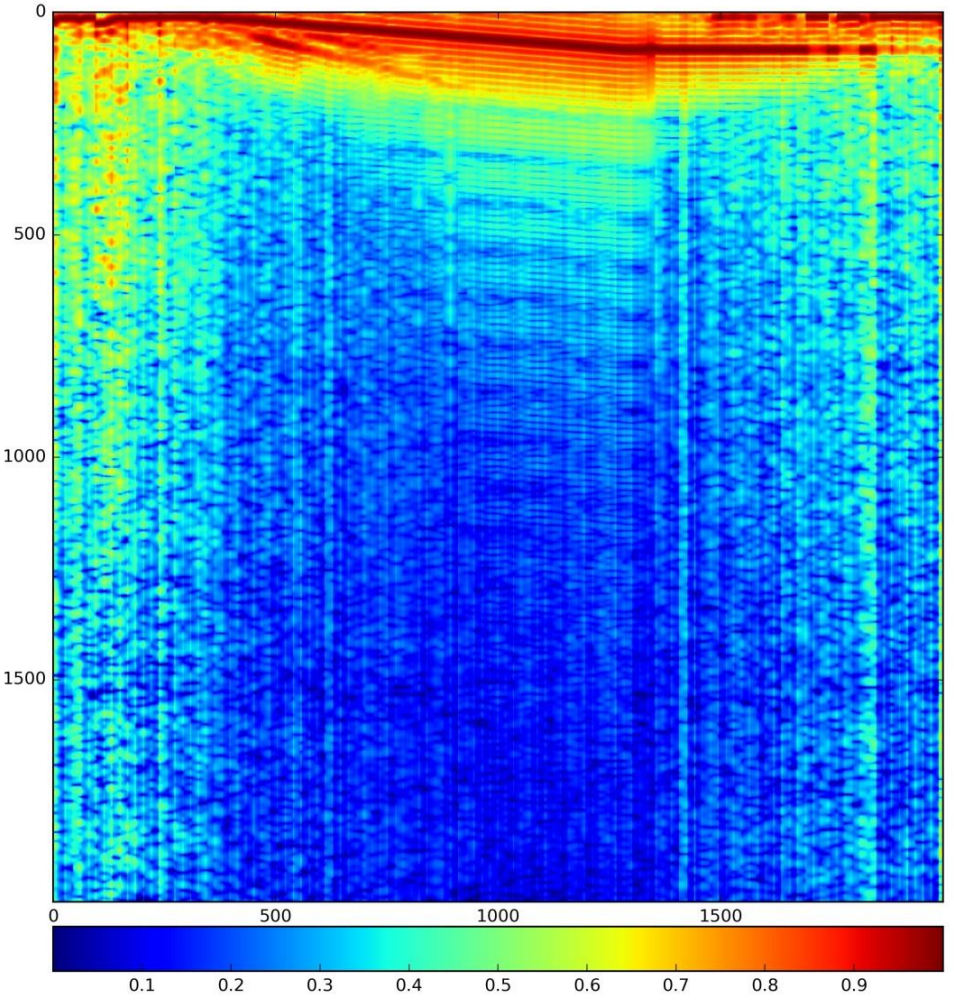


1285.wav

RF predictions vs. labels. Predicted = 1, true = 1

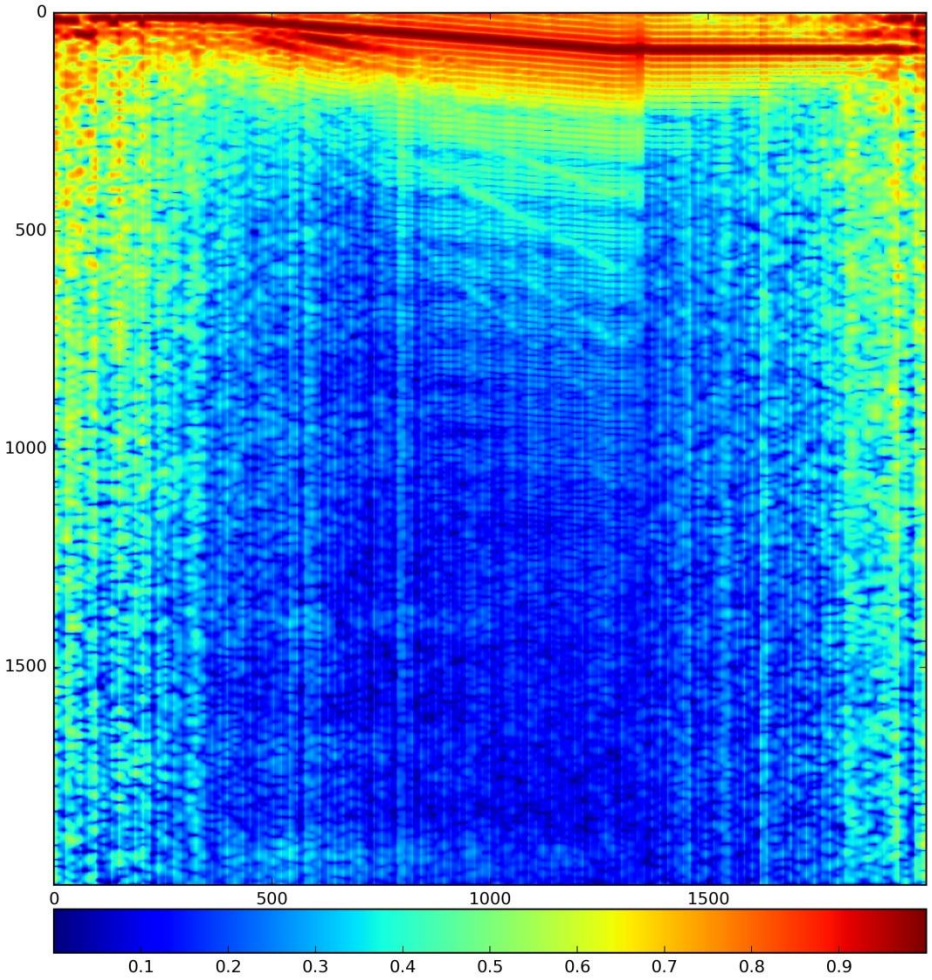


58.wav

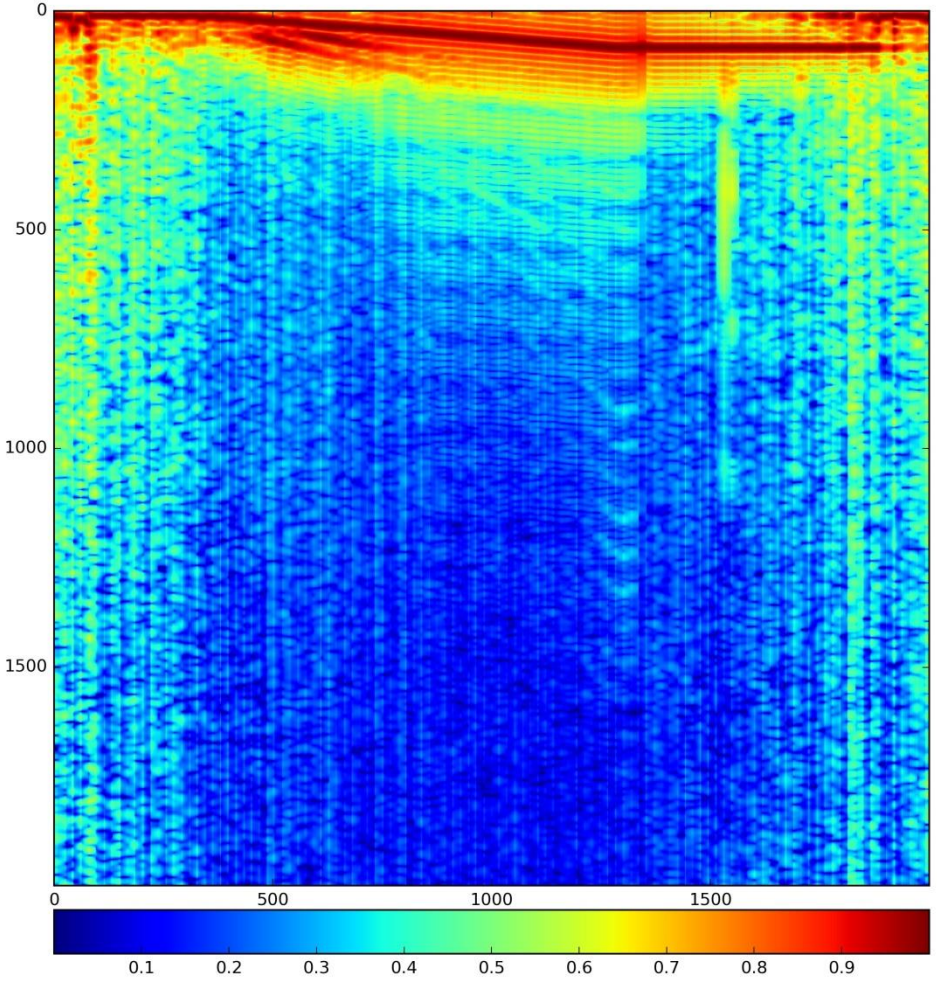


108.wav

RF predictions vs. labels. Predicted = 1, true = 1



700.wav



1228.wav

Conclusions from RF classification

1. Although it is hard to identify exact distinguishing patterns by eye immediately, some similarities/differences could be found.

2. Spectrograms corresponding to the broken emitters have a typical *'palm tree leaf'* structure, with a number of *'rays'* spanning many frequencies over longer times from the origin.

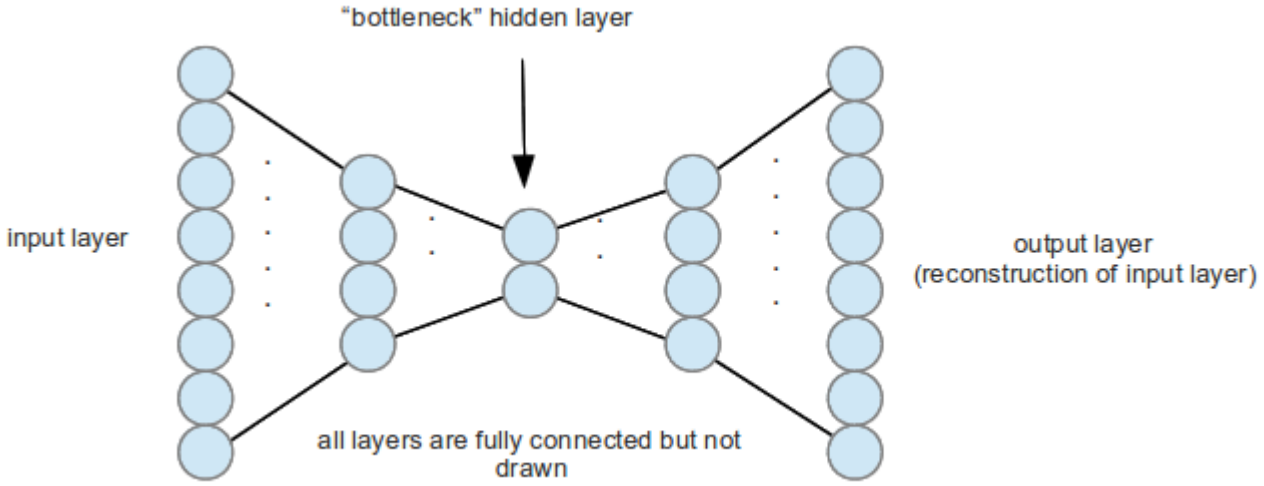
3. Another phenomenon associated with broken emitters are *vertical lines* of high amplitude extending for all frequencies over short times. Sometimes, this occurs for later times away from the origin.

4. Working devices, which were identified as such, display a more continuous pattern for lower frequencies with few or none of the extending 'rays'.

5. RF may also fail to diagnose a broken device when a vertical line (3) occurs for earlier times closer to the origin, with continuous pattern afterwards.

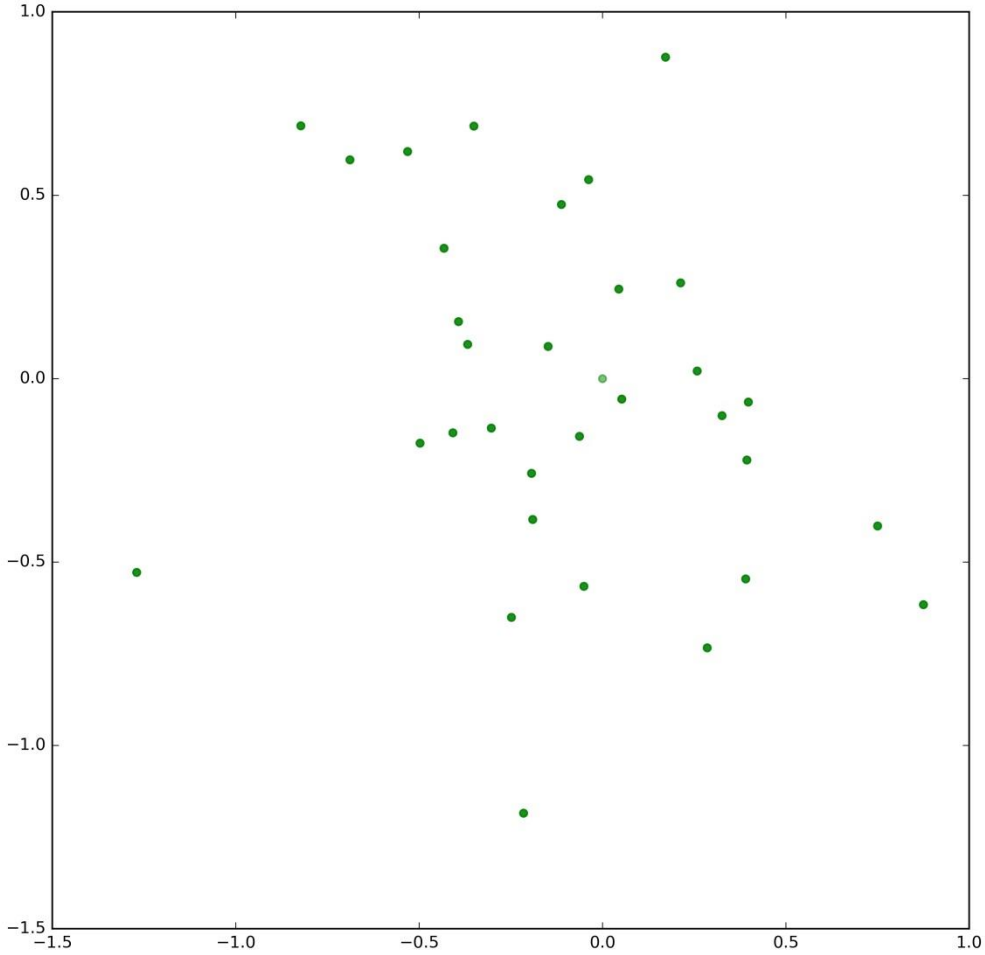
Autoencoder

- An autoencoder is a neural network built with the aim to reconstruct the input images and to estimate the representation for a set of data. It's an unsupervised learning method employed to find clusters in the data. It has the same number of output nodes as the input nodes and one usually considers how well the output images are reconstructed compared with the input images.



Autoencoder

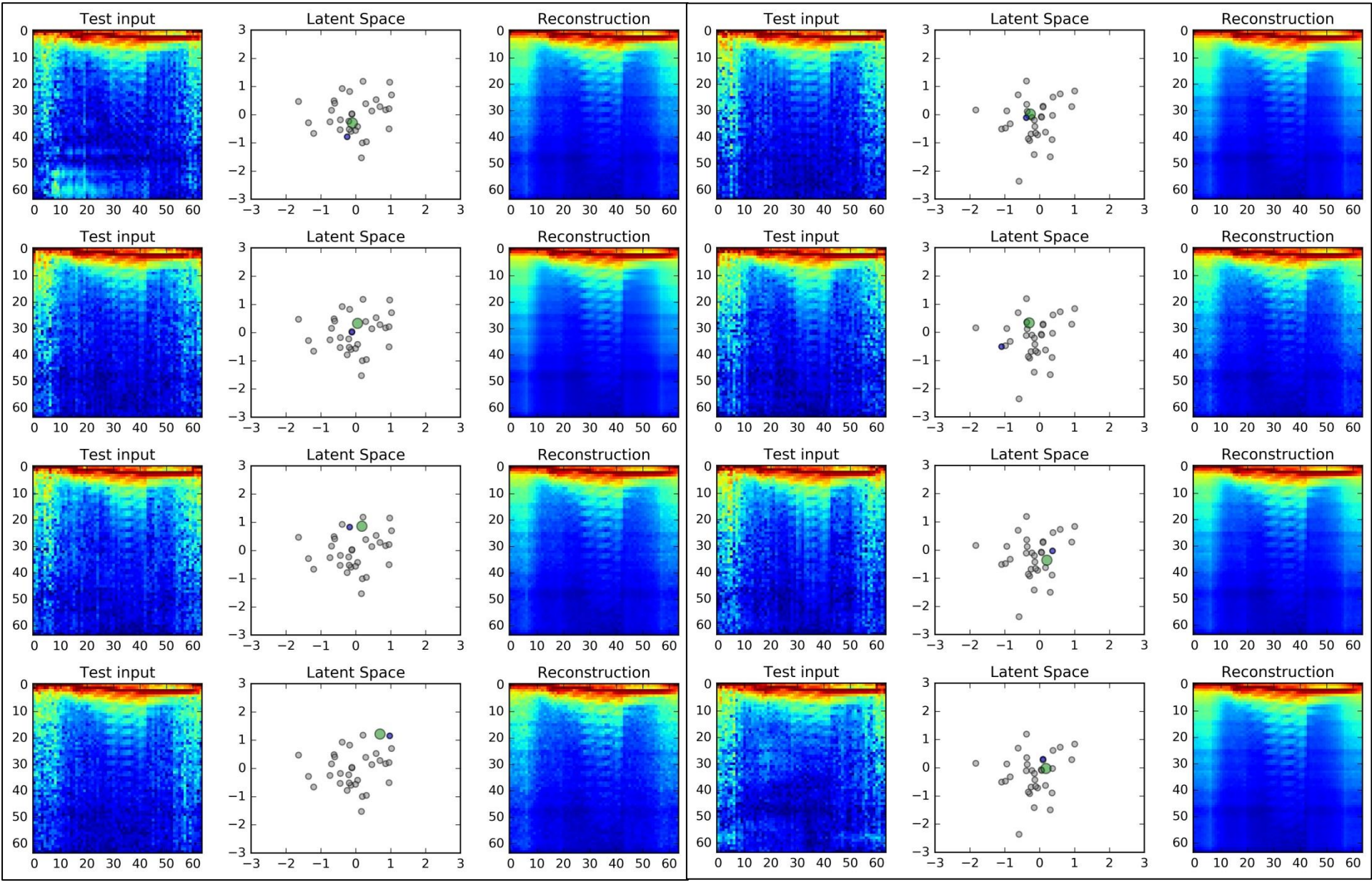
- A Variational Autoencoder with 2 hidden layers (with 501 hidden nodes each) both in the encoder and the decoder and 2D - Gaussian prior was investigated. The aim was to produce a 2 dimensional latent space to identify outliers in this space.



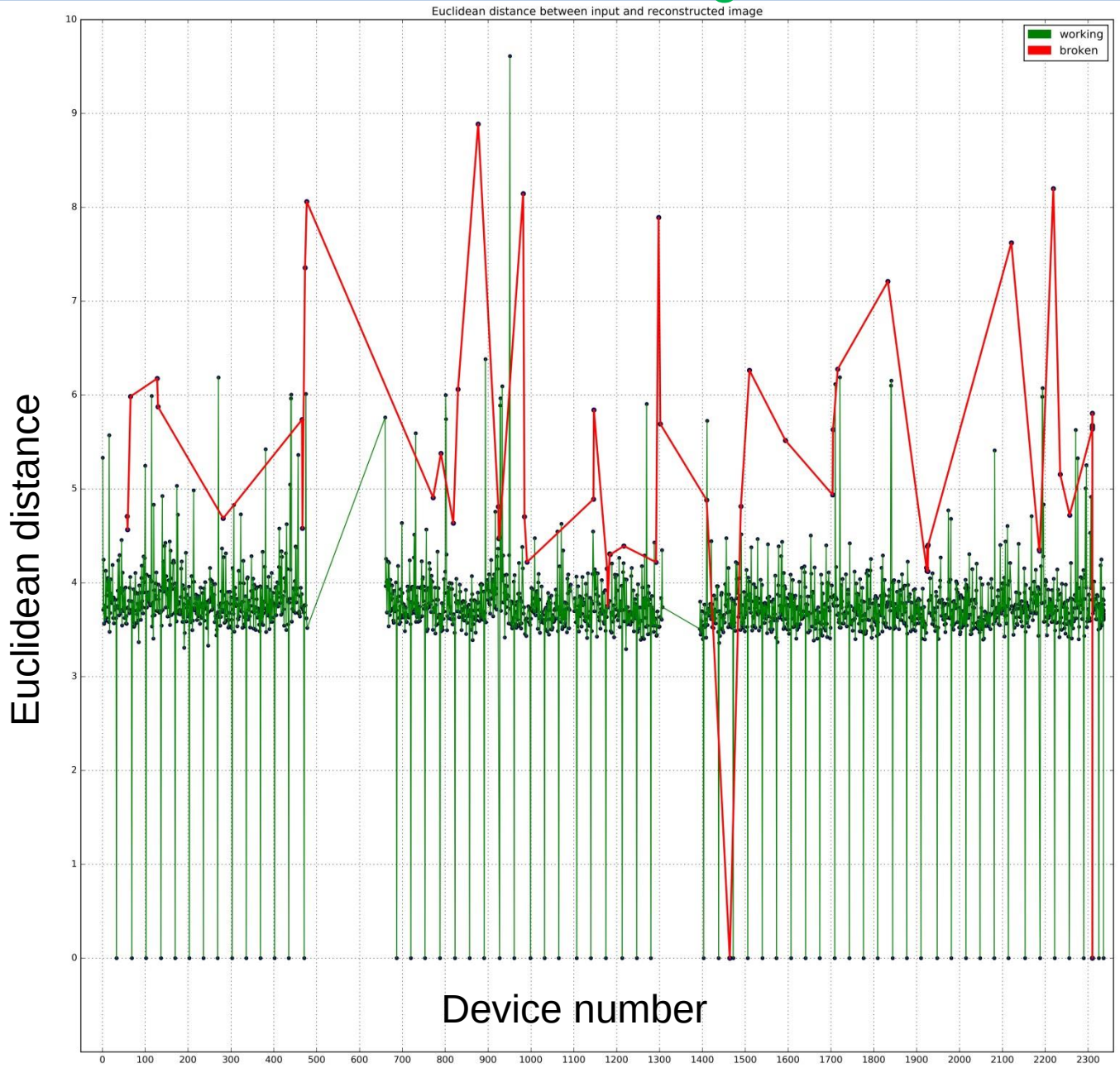
Autoencoder

- Since no distinct patterns were seen in the latent space after a number of experiments, the standard methods (e.g. RF) were tested and succeeded.
- An autoencoder could also provide a useful approximation to the functional relation between the working emitters and the broken ones separately.
- Even though it reproduced the input spectrograms well, it was still impossible to identify broken devices. An autoencoder was first fitted on the working devices only, and then attempted to reconstruct the broken ones.
- If more data had been provided, autoencoders could have outperformed the standard methods.
- In comparing the inputted and reconstructed spectrograms a Euclidean distance per pixel was used.

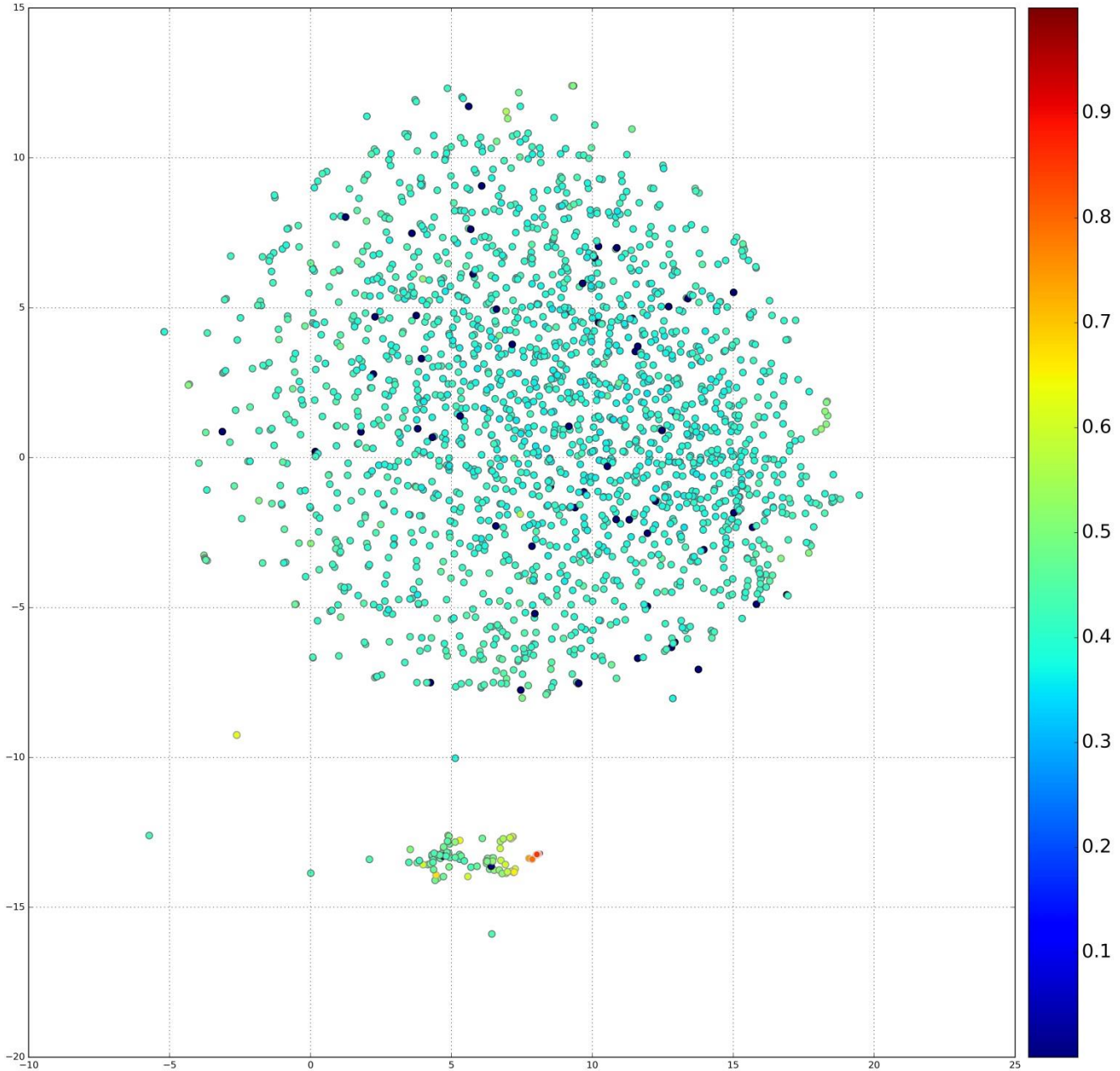
Autoencoder’s reconstruction performance



Autoencoder's reconstruction performance on broken and working devices



Autoencoder's reconstruction performance on t-SNE clusters



Thank you for your attention!