Bachelor Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Physics

Break junction data clustering using supervised and unsupervised machine learning

Oliver Klimt

Supervisor: Ing. Ladislav Sieger, CSc.
Supervisor–specialist: RNDr. Jindřich Nejedlý, PhD.
May 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Klimt Oliver** Personal ID number: **499151**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Measurement**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Break junction data clustering using supervised and unsupervised machine learning

Bachelor’s thesis title in Czech:

Navrhněte metodu pro zpracování měření typu break junction

Guidelines:

1) Research machine learning methods for processing Break-Junction data (measurement of conductivity of organic molecules)
2) Choose a suitable algorithm for solving the specified task
3) Verify the suitability of the selected method on real measured data

Bibliography / sources:

1) SHUKLA, Nishant a FRICKLAS, Kenneth. Machine learning with TensorFlow. Shelter Island: Manning, [2018]. ISBN 978-1-61729-387-0.
2) GÉRON, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems. Second edition. Beijing: O'Reilly, 2019. ISBN 978-1-4920-3264-9.
3) Bro-Jørgensen, William – Hamill, Joseph M. – … Solomon, Gemma C. (2022): Trusting Our Machines: Validating Machine Learning Models for Single-Molecule Transport Experiments. Chemical Society Reviews, Roč. 51, Č. 16, s. 6875–6892, <https://doi.org/10.1039/d1cs00884f>.
4) Cabosart, Damien – El Abbassi, Maria – … Perrin, Mickael L. (2019): A Reference-Free Clustering Method for the Analysis of Molecular Break-Junction Measurements. Applied Physics Letters, Roč. 114, Č. 14, <https://doi.org/10.1063/1.5089198>.
5) Komoto, Yuki – Ryu, Jiho – Taniguchi, Masateru (2023): Machine Learning and Analytical Methods for Single-Molecule Conductance Measurements. Chemical Communications, Roč. 59, Č. 45, s. 6796–6810, <https://doi.org/10.1039/d3cc01570j>.

Name and workplace of bachelor’s thesis supervisor:

Ing. Ladislav Sieger, CSc. Department of Physics FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **25.01.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until:
by the end of summer semester 2024/2025

Ing. Ladislav Sieger, CSc. Supervisor’s signature	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____ Date of assignment receipt

_____ Student's signature

Acknowledgements

I would like to thank Doctor Ladislav Sieger for introducing me to this very topic. Doctor Sieger not only showed me how fascinating can molecular physics be but also introduced me to the great colleagues of Starý group.

I'd like to thank Doctor Jindřich Nejedlý for his guidance a patience. I learned chemistry in high school, but I came short on advance topics. Doctor Nejedlý took the time to introduce me into break junction experiments and related chemistry.

My thanks go to Doctor Jaroslav Vacek for allowing me to use data from the Breaker programme, a desktop application written exclusively for IOCB to analyse breaking curves.

And, of course, I want to thank Doctor Irena G. Stará and Doctor Ivo Starý for welcoming me into their group. And all members for a friendly atmosphere in their lab.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Abstract

Break junction experiments require us to collect many conductance traces. However, not every trace has a meaningful value. For example we can observe exponential decay of tunnelling current after a *snapback* event when no molecule is present or due to contamination; a trace can have a non-standard slope as electrodes move apart. We want to filter out traces with no molecule present and preserve traces that follow an expected trend. We have settled on using supervised and unsupervised machine learning techniques.

Keywords: Machine learning, supervised and unsupervised machine learning, data annotation, break junction, conductance traces, molecular electronics

Supervisor: Ing. Ladislav Sieger, CSc.
Technická 1902/2,
Praha,
místnost: B2-42

Abstrakt

Experimenty typu break junction vyžadují sběr velkých dat vodivostních křivek. Avšak ne všechny křivky mají pro nás výpovědní hodnotu. Kupříkladu se molekula nemusí na elektrody navázat a tím pádem po *snapback*u (odtržení) pozorujeme exponenciální útlum tunelovacího proudu nebo má vodivost při oddalování elektrod nestandardní trend, což může být zapříčiněno kontaminací roztoku. Naším cílem je rozlišit a vytřídit křivky bez molekuly a ty, které obsahují další význačný rys. Pro tento účel jsem se rozhodli využít nástrojů strojového učení typu unsupervised i supervised.

Klíčová slova: Strojové učení, strojové učení typu supervised a unsupervised, anotace dat, break junction, vodivostní křivky, molekulární elektronika

Překlad názvu: Navrhněte metodu pro zpracování měření typu break junction

Contents

Chapter 1

Motivation

Molecular electronics, in terms of studying electro-chemical, electro-sterical properties started in 70s, but is still a fascinating field of study.

Todays transistors are etched into silicon wafers. Diodes in those transistors take advantage of dissimilarity in materials where one side contains free-moving holes and the other free-moving electrons, which makes p-n junction; a key element in electronics.

Bulk silicon conducts electricity at certain energy levels (band-gap), but bulk metal conducts electricity for all levels [1]. This behavior changes for applications with few atoms, where only for some many atoms a band-gap will reappear, which makes it challenging to shrink transistor designs [2].

Single molecule electronics are systems of few nanometers and they are an alternative for anorganic silicon based electronics.

Molecule metal bonds, as two dissimilar substances, can act as p-n junction, thus creating molecular diode. First molecular diode was proposed by Aviram and Ratner in 1974 [3]. These diodes can be a foundation for molecular devices, such as switch [4].

Organic devices are nothing new, we can find organic light-emitting diodes (OLED) in screens of phones or televisions, there are organic field-effect transistors (OFET) that can be produced cheaply [5]. Another application of organic devices are various biosensors [6].

We want to examine how organic molecules conduct current in order to examine their potential for replacing or supplementing present silicon based technologies [7, 8].

Chapter 2
Introduction

We focus on analyzing big data from break junction experiment. Our datasets contains thousands of curves, millions of points and are several hundreds of megabytes large. That is quite a lot of data that needs to be processed in a acceptable time frame. Our programme should utilize all available resources, such as operating memory and CPU cores.

To accomplish this, we use both supervised and unsupervised machine learning techniques to gain a valuable insight of our datasets [9].

2.1 Understanding our data

A typical conductance trace can look like this (figure 2.1): It starts in a *bulk*, which is the golden wire itself, then we can distinguish one, two or three steps, that is an electrode tip forming, consisting from three to one atom. After that a *snapback* occurs. Snapback is a rapid loss of conductivity on a very short distance. At this very moment a wire broke into two distinct pieces and a metal-molecule-metal bond can be formed.

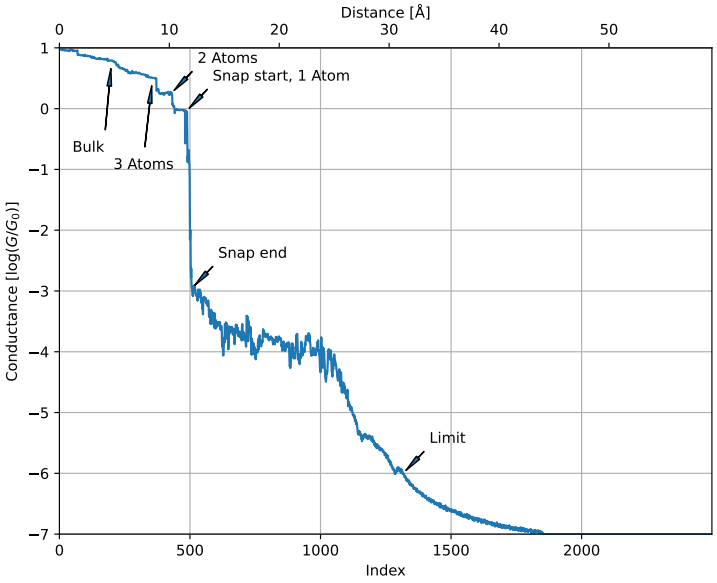


Figure 2.1: A typical conductance trace with a molecule present.

One can ask how do we determine whether a metal-molecule-metal bond has formed, how can one be sure that the conductance level is not a tunneling current or some contamination of solution.

We can plot a 1D histogram (see 2.2) and observe a peak at around $G \approx 10^{-4}\ G_0$, which corresponds with a plateau at 12 - 25 Å.

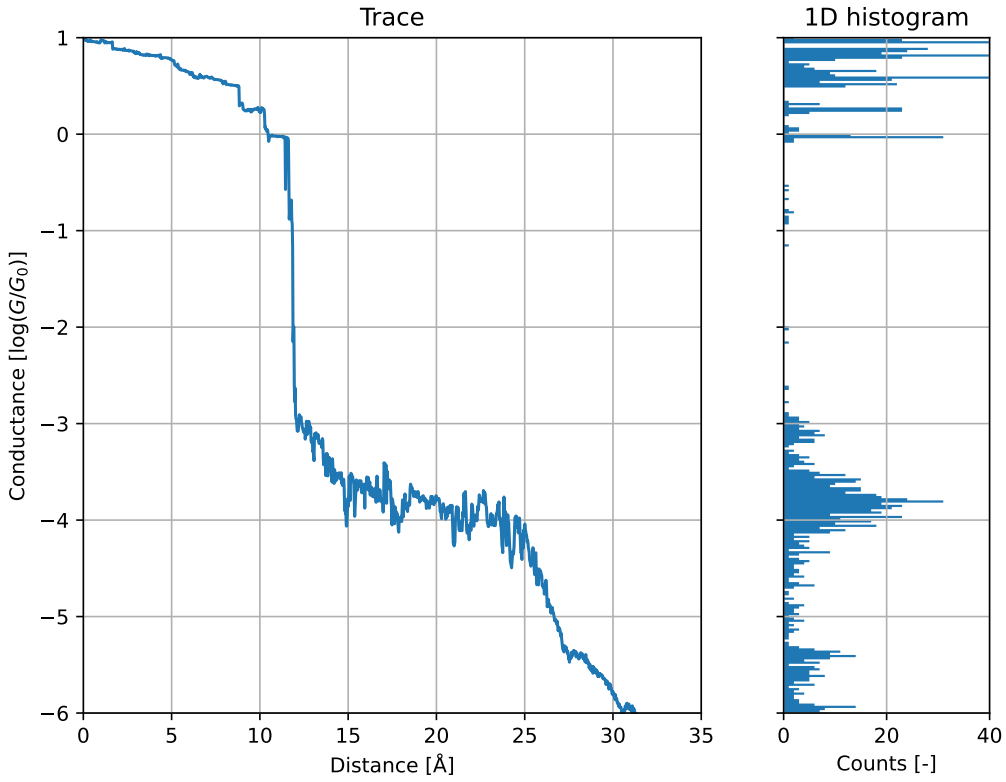


Figure 2.2: A typical conductance trace with a molecule present and a corresponding 1D histogram in logarithmic scale.

Our data from one experiment contains tens of thousands of traces, so visualizing one by one does not tells us much. Therefore, we can plot all traces at once using 2D histogram, as in figure 2.3. Both histograms are normalized per traces, meaning that bins are divided by total count of traces in dataset or cluster. In 2D histogram dark red means high count of traces, dark blue means low or no count of traces, white marks a half of the scale.

If we compare pictures 2.2 and 2.3, we are not able to locate traces with molecules in 2D histogram. There might not be enough of them or they can be so scattered around that 2D histogram is not very helpful.

For this very reason, we have to inspect our datasets with the help of machine learning in order to be able to find how many traces made metal-molecule-metal bridge and if the rest is unuseful data of tunneling current.

On closer inspection, our curves should start at $G = 10$ that is 1 in logarithmic scale. Most of them does (see 2.3), but there are hundreds traces which does not and those traces need to

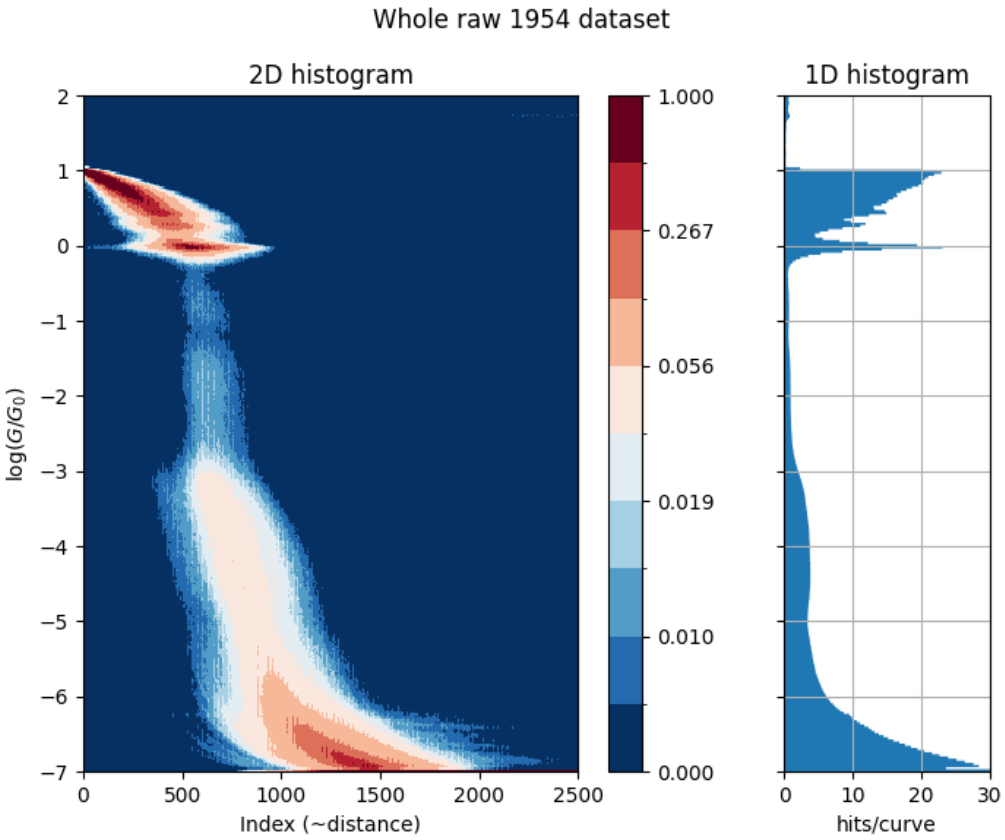


Figure 2.3: 2D and 1D histogram of traces from one dataset.

be filtered out. An example of such trace is in figure 2.4.

Discarding all bad traces, we are able to align curves in snapback and plot 2D histogram with distance in Å as in figure 2.5.

2.2 Clustering

When we filter out bad traces, we can inspect the rest of them using various clustering algorithms. Those algorithms will sort curves into different clusters, which gives us insights into our dataset.

2.3 Desired goals

We want to write application named **iCluto** to aid physicians and chemists with analyses of breaking curves. We want to improve clustering capabilities of CLUTO (Cluster Toolkit) [10], hence the 'i'.

Thanks to modern libraries, iCluto can recognize important events (start and end of snapback) using neural networks and can cluster utilizing all available CPU cores. Our programme is written solely in Python, which is very popular and actively used programming

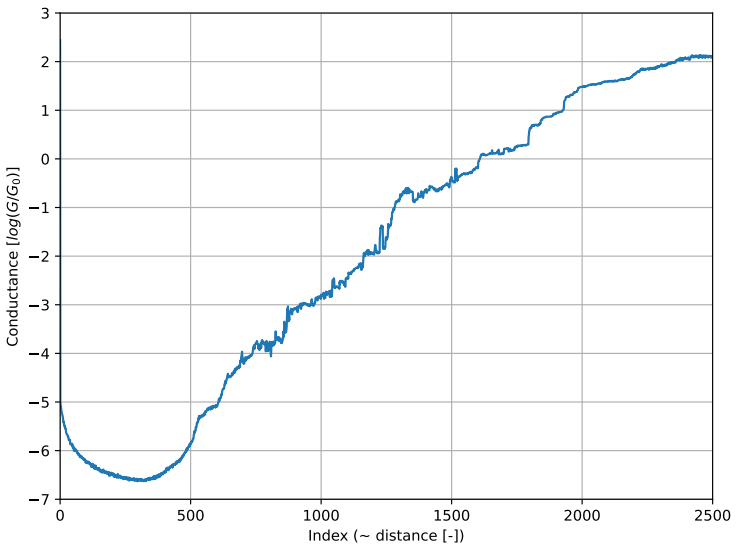


Figure 2.4: Example of a bad trace.

language. Python packages allow us to develop new functions and expand iCluto's possibilities. This means we can sort thousands of curves in a acceptable time frame.[1]

2.4 Related works

There are many studies to cluster traces into meaningful groups.

They all differ in selected feature space, whether they used unsupervised or supervised machine learning or what clustering algorithms they used [11–16].

Our data comes in a form of matrix, where each row represents one trace and each column of that row yields value G_i. There are 2500 points for each trace and roughly 35000 traces per measurement. That puts us into a similar situation as others.

To process this vast amount of data, one can make histograms for each trace and use this histogram as an input for clustering, reducing the input length further via principal component analyses [11]. Another paper proposes to assign each trace 5-tuple segment that encodes individual plateaus in a trace [12]. A straightforward method is to use a segment of trace after snapback and transform it into a $M \times N$ image, thus reducing data points needed [13]. Assigning each trace into a cluster can be done with various unsupervised clustering algorithms [14].

One can have their dataset labeled and use supervised machine learning where, a model has to be trained in order to cluster data properly. Such a model can be a recurrent neural network [15], deep neural network [16] or different machine learning architecture.

IOCB uses Breaker programme, a FORTRAN application written by Dr. Vacek, which uses only one CPU thread.

[1] Under an hour for $\sim$ 37000 traces.

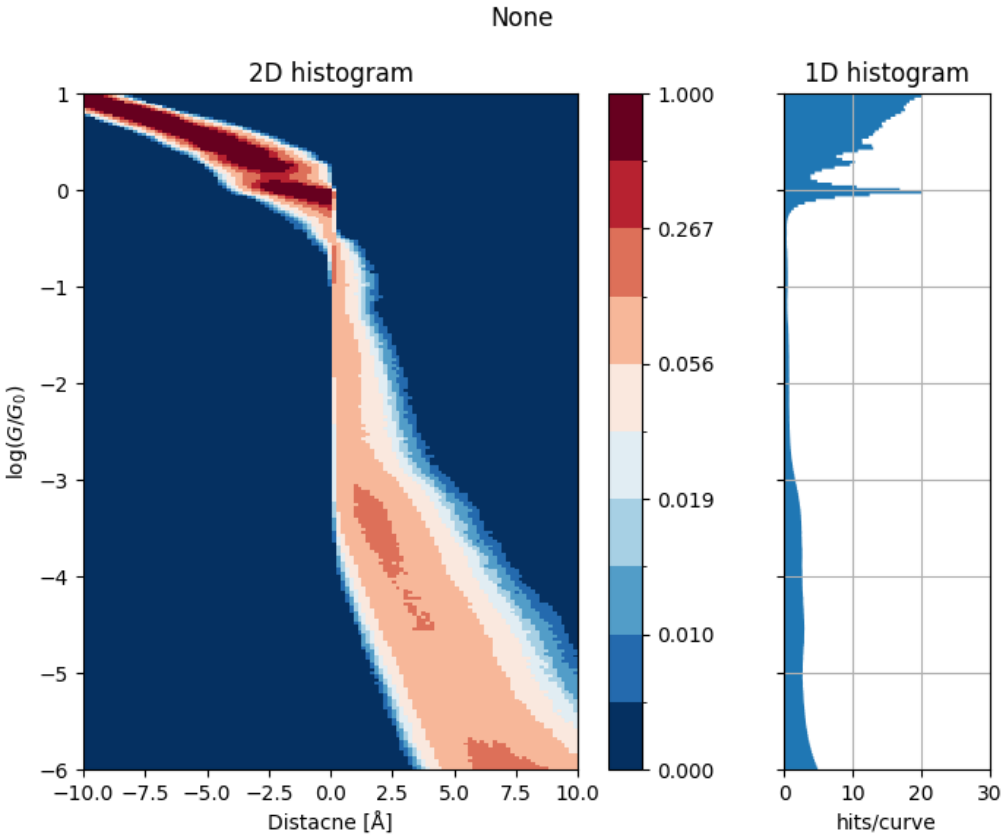


Figure 2.5: All traces aligned in snapback as our origin of distance.

Chapter 3

Break junction experiment

3.1 Principles of break junction experiments

Break junction (BJ) is a method of measuring conductivity at nanoscale level acquiring breaking curves. This experiment relies on the ductility and malleability of a thin metal wire, typically a gold wire, where we stretch it so much until it breaks apart. Those two parts with pointy tips work as electrodes, allowing a molecule to bond in between them. When a thin wire breaks we observe a huge loss of conductivity. But before it breaks, with high grade amplifiers, we do see discrete steps in conductivity. In general, we write conductivity as

$$G = \frac{I}{U}, \tag{3.1}$$

where G [S] is electrical conductivity, I [A] is electrical current and U [V] is voltage, but it is also defined as

$$G = \sigma \frac{A}{l}, \tag{3.2}$$

where σ [$\mathrm{S} \cdot \mathrm{m}^{-1}$] is electrical conductivity per meter, A [m^2] is a cross-sectional area and l [m] is a length of wire.

For very small A in equation 3.2 we observe discrete steps in conductance, because the cross-sectional area of our wire consists of $N \in \mathbb{N}$ Au atoms. Therefore equations 3.1-3.2 can be simplified for gold as

$$G = NG_0, \tag{3.3}$$

where N is number of gold atoms, and $G_0 = 2\frac{e^2}{h}$ [S] is von Klitzing[1] constant [17],where $e = 1.602 \cdot 10^{-19}$ C is elementary charge and $h = 6.626 \cdot 10^{-34}$Js is Planck's constant.

At $G = 1 \cdot G_0$ the gold wire breaks, forming two electrodes and allowing a tunneling current to flow between them. As the space between electrodes widens a molecule can bond. The possibility of molecule bonding relies on many factors, such as geometry of electrodes, used anchoring groups or shape of the molecule.

[1] $G_0 \doteq 7.75 \cdot 10^{-5}$ S

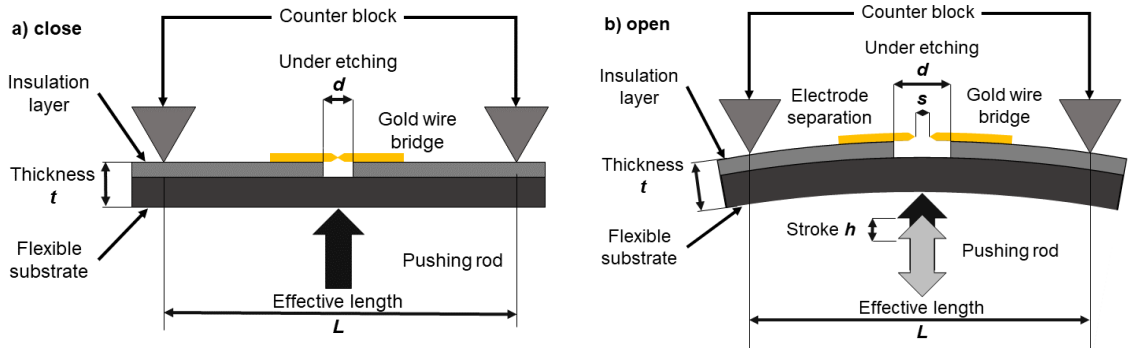
3.2 Execution of break junction experiment

There are two main types of BJ apparatuses; Mechanically Controllable Break Junction (MCBJ) and Scanning Tunneling Microscopy-Based Break Junction (STM-BJ). They differs in electrodes fabrication.

3.2.1 Mechanically Controllable Break Junction

Forming electrodes requires thin gold wire that is stretched until it breaks, see figure 3.1. When contracted, both tips join forming a single piece of wire again. The process repeats. Wire is oriented horizontally and the electrodes are symmetrical to each other.

This method was introduced by Moreland in 1985 [18] and improved by Muller in 1992, where it got its name "Mechanically Controllable" [19].



(a) : Closed bridge, the gold wire is still in one piece.

(b) : Open bridge, the gold wire broke into two electrodes.

Figure 3.1: Principles of MCBJ experiment. Taken from [7].

3.2.2 Scanning Tunneling Microscopy-Based Break Junction

Apart from MCBJ, in this method the electrodes are asymmetrical, there is only one pointy tip and the other electrode is a golden substrate. Electrodes are aligned vertically.

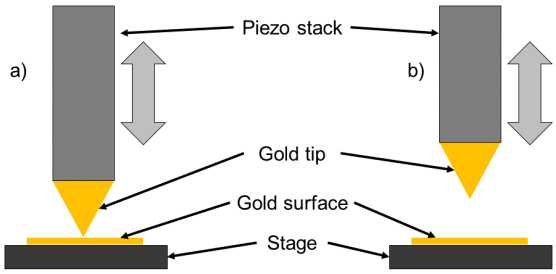
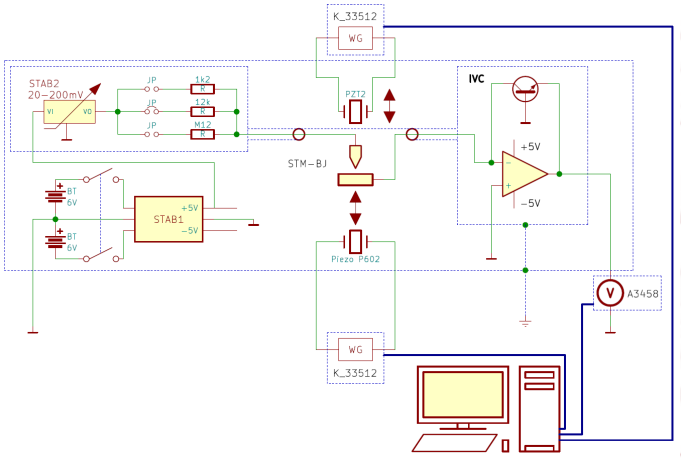


Figure 3.2: Schematic of STM-BJ. Taken from [7].

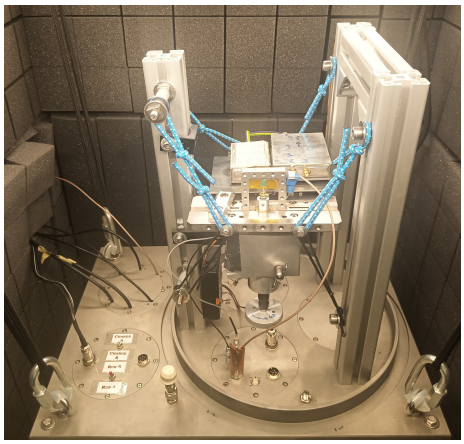
3.3 Technical parameters

Our apparatus is custom made because we have not found any commercially viable option. It was designed and built in collaboration with Prof. Josef Zicha (CTU) and Jiří Miletín (BMD, s.r.o Teplice). It is capable of detecting currents down to 100 fA, with a sampling rate of up to 200 kSa. Our apparatus can be configured for both MCBJ and STM-BJ types, but our datasets are mostly acquired by STM-BJ, therefore we will describe only this type.

In 3.3a at the center we can see that breaking is done by a pushing rod activated by a piezo crystal, that has pushing force of 100 N. Piezo crystal is activated using Keysight K_33512B wave generator. We are using two custom made voltage stabilizers (STAB1 and STAB2); the first one stabilize voltage from lead-acid batteries, and the second one is shielded (thin blue dashed line) and is used to offset measurement bias. Our custom current to voltage converter (IVC) utilizes logarithmic operational amplifier, that covers several orders of magnitude. We sense the output voltage with Agilent 3458 digital multimeter. Data acquisition is done with LabView software [20]. All sensitive parts are shielded and grounded [7].



(a) : Wiring diagram of our apparatus. Taken from [7].



(b) : Current version of our apparatus. Photo by Dr. Nejedlý.

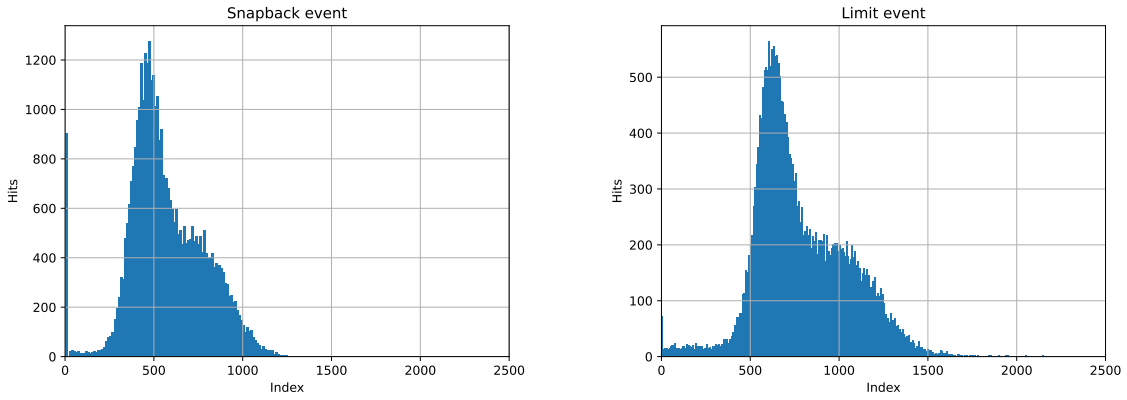
Figure 3.3: Apparatus for collecting breaking curves.

Chapter 4

Data processing

Data from break junction are not linear and are time variant, but mainly the data consists of GBs of data, therefore we need to preprocess our data. To do so, we will introduce several functions to handle our data better.

Our first attempt of clustering curves was simply focused on a specific interval. Apart from selecting range (length interval) and applying $\log_{10}$ element wise, no other modifications were made. From figure 2.3 we can determine that our desired interval starts at around 500 and ends roughly at 1000 (or 1200). That yields 500 points to work with. We applied several clustering algorithms such as K-Means, DBSCAN or HDBSCAN. Those mentioned algorithms separated curves into several groups and from those groups we could make another 2D and 1D histograms. We roughly estimated where snapback happens and where a limit is present. But that is only a rough approximation, because not all traces really do have snapback event at 500 and not all limits are at 1000 (see 4.1a-b).



(a) : Locations of snapback.

(b) : Locations of limits.

Figure 4.1: Even though most of the traces do snap at 500 the spread of snapback event is huge. The same can be said about limit event.

We track 2 main events in traces: *start of snapback* and *limit*. We also keep track of where snapback ends, which might be useful in future. We marked those annotations in 4.2.

It would be crucial to locate snapback and use it for calculating displacement of electrodes in Å and for alignment of traces in 2D histograms.

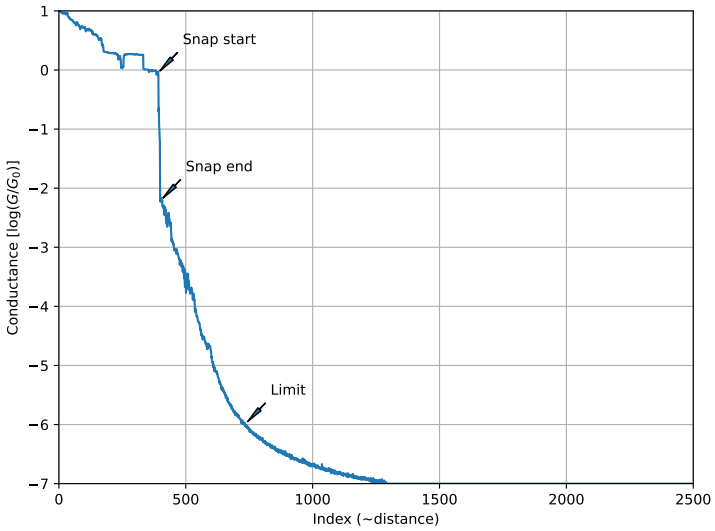


Figure 4.2: Annotated trace.

4.1 Neural networks

We utilized a Convolutional Neural Network (CNN) for snapback detection.

Detecting the snapback is crucial because this event allows us to align individual traces into our 2D histograms and give them a length scale (more in section 4.4).

4.1.1 Convolutional Neural Network for snapback detection

Our network consists of two convolution layers and one batch norm in between (figure 4.3). Kernel sizes are 128 and 64 for 1st and 2nd convolution respectively. First convolution takes 1 channel as input and outputs 16 channels that are, after batch normalization, fed into second convolution that again outputs single channel. We add padding to our trace, so after convolution the output has the exact dimensions as input. Argmax of our output vector represents an index of snapback, see figure 4.4.

It was crucial for us to utilize batch normalization; without it, our netowrk would not learn. This normalization happens in-between convolutional layers and for each input it performs:

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x]}}, \tag{4.1}$$

where x is our input vector (in this case 2500 point), E is arithmetic mean and Var is variance. Batch normalization allows using higher learning rates and reduce the need of Dropout or eliminating it completely [21].

4.1.2 Curating output of CNN

Our CNN was accurate in only 82 % of our testing dataset. We found that it detected snaps at deficient levels. Noted that snapback should occur at level of $G = 1\ G_0$, we can force our

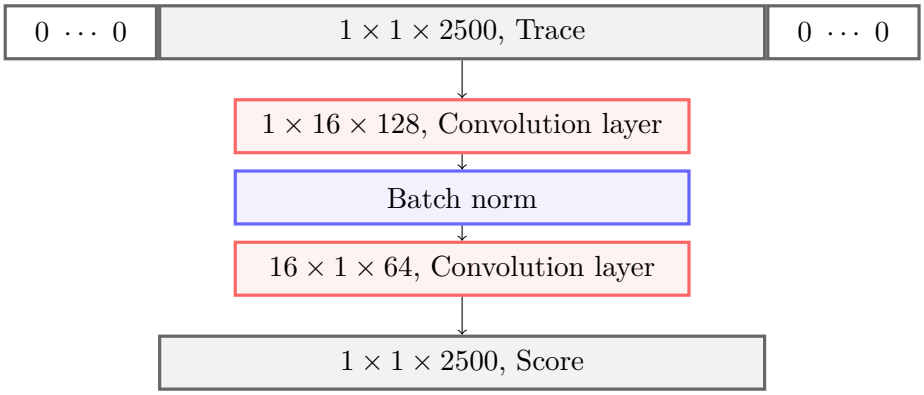
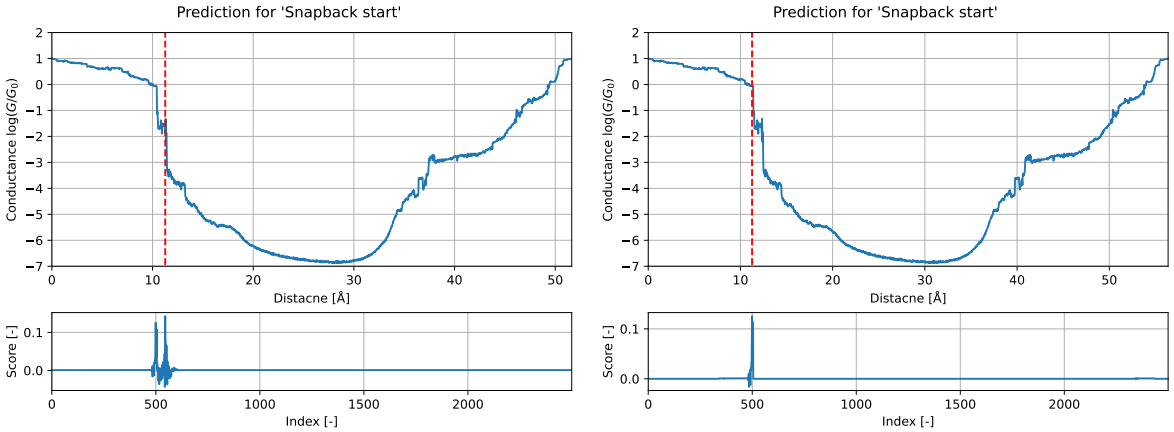


Figure 4.3: Convolutional neural network for snapback detection. Padding is symbolized as block of zeros, convolution layers are defined as *input channels* × *output channels* × *kernel size* and input and output vectors have one channel.

network to be active at this level and zero the score elsewhere. By doing so, we have improved its accuracy to 99 %.



(a) : A raw output from CNN.

(b) : Curated output.

Figure 4.4: Our model is not robust enough, in some cases it flags snapback in very low conductance levels. We can curate our model so it performs more appropriate.

This simple step can be considered as physics-informed machine learning. We can curate our data on many levels. Physics-informed machine learning means handpicking a criteria that our system should follow and inserting it into machine learning pipeline [22]. When modeling an AI model, we need to decide on:

1. Dataset,
2. Model,
3. Loss function,
4. Optimizer.

And at each level we can "bake" our requirements, in our case snap specific interval. For example choosing only traces that satisfies our interval (10 G_0, 0.1 G_0) will change the training

dataset and thus influencing the rest of ML learning process. Or as in our case, annulling the score for conductance outside our specified range is considered as informing our model; such a thing can also be done when designing a loss function. Loss function is used prior calculating backpropagation and parameter tuning so that it can penalize more for scores outside our range.

4.2 Detecting limit

There is no need for neural networks in detecting the limit of our traces. Values that are lower than $G = 10^{-6}\ G_0$ are near the limit of our OPAMP. So we can take the first occurrence of $G < 10^{-6}\ G_0$ and mark it as a limit. This "hardcoded" approach is better, because we can omit labeling traces and training another neural network, which saves time and it serves the same purpose.

4.3 Extracting useful data from traces

From observations of individual traces, we can mark $G = 10^{-6}\ G_0$ as the limit and set the rest of the trace to this value. This helps us clear traces that have clear snapback but after the limit, they rise (as depicted in figure 4.4).

The same can be done at the start of our trace. Values before snapback are bulk[1], which means that we can get rid of them and shorten the trace, reducing input vector for clustering algorithms.

4.4 Indifactor

Indifactor denotes start of our coordination system. It is calculated per trace, hence indi as individual, and it yields scale factor in Angström (10^{-10} m or Å).

For indifactor, we write

$$f = \frac{g_1}{j_0 s} + \frac{1}{2}\frac{g_2}{j_2 s}, \tag{4.2}$$

where $s = 1.28$ is slope and was chosen empirically, j_0 is index of snapback starting point, j_2 lies in the middle of indices 0 and j_0, g_2 is a value at index j_2 and $g_1 \approx 10$ is first value of a trace.[2]

It is convenient to assign origin (0 Å) at the beginning of snapback, because electrode separation starts at this very point. We do this for all 2D histograms.

[1] By bulk we mean a thin wire with more than 20 atoms in diameter.
[2] All our measurements for individual traces roughly starts at $g_1 \approx 10$.

4.5 Principal component analyses

Principal component analyses (PCA) is a common technique for dimensionality reduction. It is a method for finding subspaces for a given dataset [23].

Let $\mathbf{A} = [\mathbf{a_1} \cdots \mathbf{a_n}]$ be matrix of traces $\mathbf{a_i}$, then

$$\mathbf{A}\mathbf{A}^T = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \tag{4.3}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_\mathbf{1}\ \lambda_\mathbf{2}\ \cdots)$ is a matrix of eigenvalues $\lambda_\mathbf{i}$ such that $\lambda_\mathbf{1} > \lambda_\mathbf{2} > \cdots$ and $\mathbf{V} = [\mathbf{X}\ \mathbf{Y}]$ is orthogonal matrix with submatrix $\mathbf{X}$ that gives us basis vectors to our subspace [24]. We get reduced (or compressed) data by simple

$$\mathbf{B} = \mathbf{X}^T\mathbf{A}. \tag{4.4}$$

The advantage is that we can fit our transformation on one dataset and use it on another. This might help to separate traces with molecules and without them even on noisy datasets.

4.6 Filters

We mentioned example of bad trace (see figure 2.4). In order to mark a trace as bad, it needs to meet one of these following criteria:

- start of the trace is $G > 30\ G_0$,
- at index 80, the conductivity is $G < 0.1\ G_0$,
- prediction of snapback is at index 0,
- prediction of limit precedes prediction of snapback,
- indifactor is higher than 400 Å,
- prediction of snapback and limit are more then 600 points apart,
- there is $G = 0\ G_0$ or $G = \pm\infty\ G_0$ in a trace.

The first three points should eliminate very short traces and traces that are false in general. The following three points aim to filter out traces that are too noisy or too long, and the last criteria is for eliminating traces that do not make sense, because infinite or no conductivity is impossible.

Chapter 5

Clustering algorithms and clusters

We have chosen to work with SciKit learn [25], a Python's scientific library, for its wide adoption and active development.

iCluto can cluster using several algorithms. We were successful with some of them. We aimed to cluster "out of the box"; meaning that we wanted to use SciKit learn's default parameters for a given algorithm.

In general there are two main families of clustering algorithms; density based and centroid based.

5.1 Data pipeline

We chained previously mentioned methods (chapter 4) and created a data processing pipeline (figure 5.1).

First we load all traces, then we select only *good traces* (criteria described in 4.6), we choose what feature we want to cluster. A feature can be histogram of trace or section of trace after snapback. Than we select principal components and feed it into clustering algorithm.

Figure 5.1: iCluto's data processing pipeline.

Histograms are generated from snapback to limit, so only a relevant part of trace is processed. Our feature *hist32* is a 350 points long histogram that was compressed with PCA to 32 points.

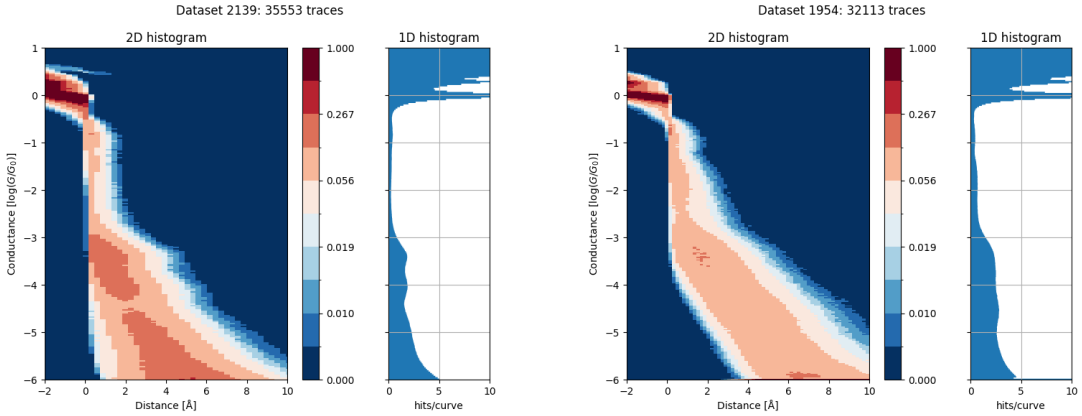
In case of *trace64* feature, we took 600 consecutive points after snapback and compressed them with PCA to 64 points.

We have always fitted PCA on 2139 dataset, because we noticed that it helped with discovering new clusters and 2D histogram of clusters were clearer.

We are only considering clusters with at least 700 members, that is 2 % of traces of our datasets, this percentage was chosen arbitrary and is based on long-term experience with these types of measurements . All clusters with fewer members are merged into cluster labeled -1.

We ran clustering on two datasets. One that contains 4,4'-Bipyridine, which is used as standard example [26], because at $G_0 \approx 10^{-3.5}$ and $G_0 \approx 10^{-4}$ we observe clear peaks at 1D histogram. We refer to it by number 2139 (figure 5.2a). The other dataset, with reference

number 1954, is noisy and is intended as a benchmark, because at previously mentioned conductance levels the peaks are not clear (figure 5.2b).



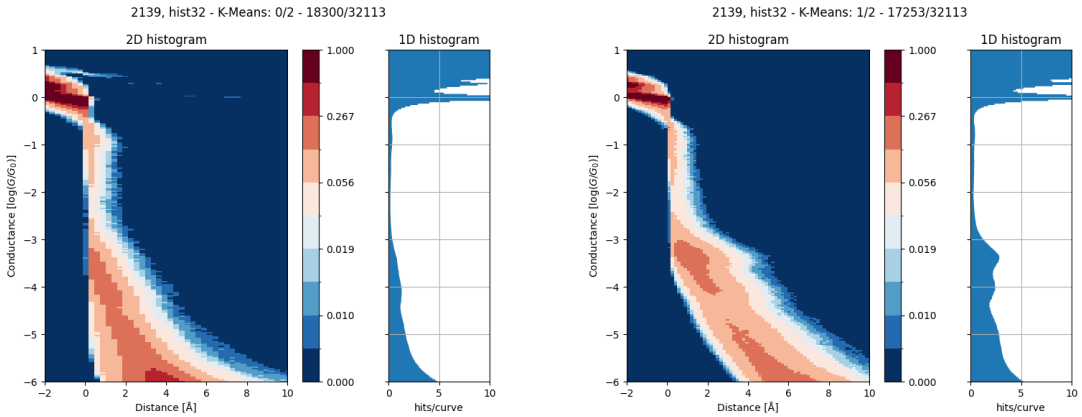
(a) : Filtered 2139 dataset. **(b) :** Filtered 1954 dataset.

Figure 5.2: Histograms of datasets after Good/Bad filtration.

5.2 K–Means

K–Means algorithm [27] uses iterative refinement technique. It initialize N cluster centroids randomly and assigns nearest traces. Then it recomputes centroids by taking the mean of all traces in a cluster. Next, traces are assigned to cluster based on their proximity to cluster centroid. The process repeats until convergence. Because the algorithm is not guaranteed to find the optimum, SciKit learn repeats random initialization 10 times [25].

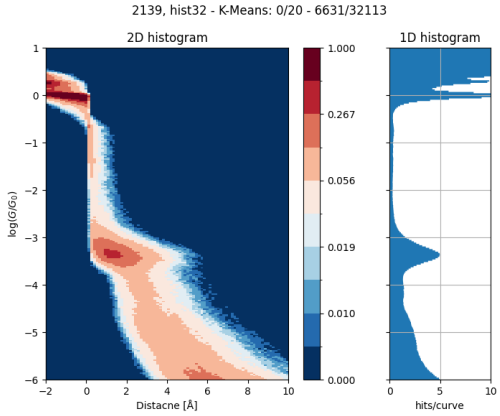
The number of clusters is the only parameter to set, which is hard to guess [28]. If the number of clusters is set to low, we might get some clusters merged into one (see figure 5.3). On the other hand, too many clusters means that some clusters may become splitted.



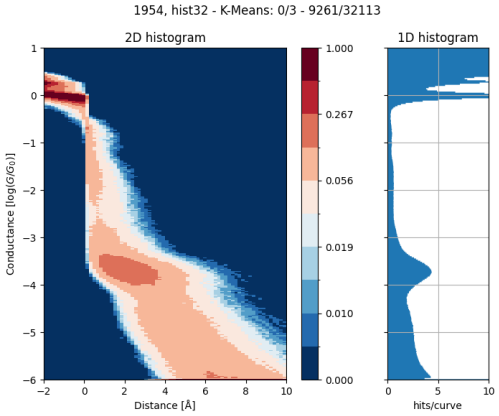
(a) : Cluster of tunneling current. **(b) :** Molecule present.

Figure 5.3: K–Means can differentiate between tunneling current and molecule. But there are two possible configurations of molecule bonding to gold atom, therefore 3 clusters would be more suitable.

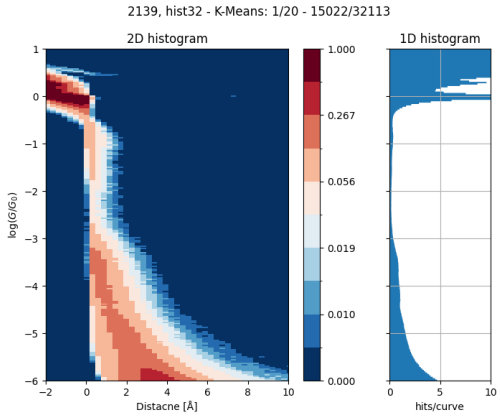
Having 3 groups, we can distinguish between tunneling current and 2 configurations of molecule bonding (figure 5.4).



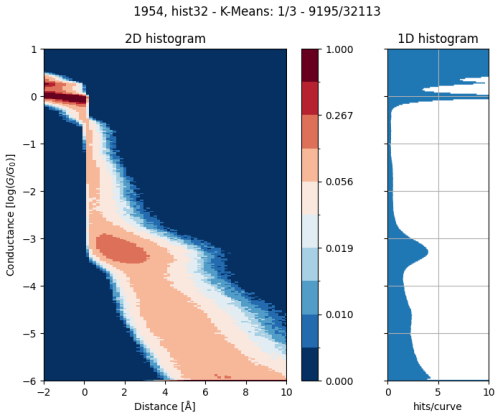
(a) : First configuration.



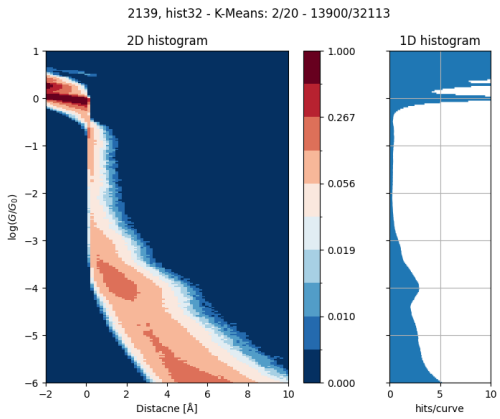
(b) : Second configuration.



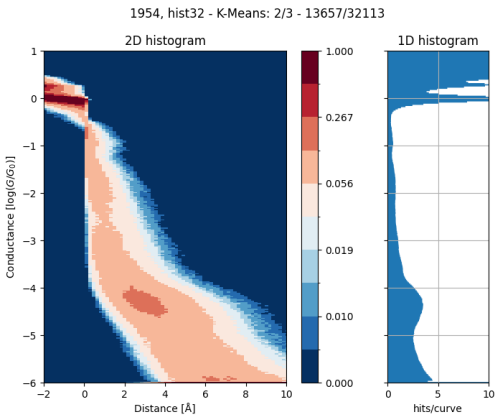
(c) : Tunneling current.



(d) : First configuration.



(e) : Second configuration.



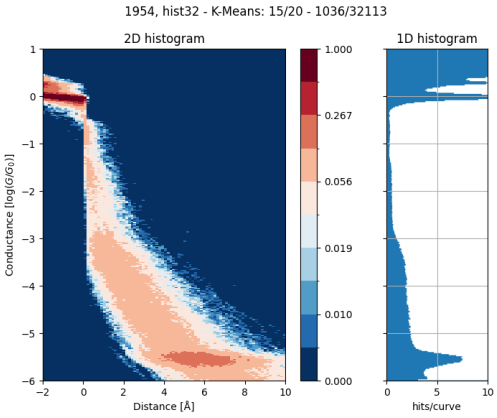
(f) : Faulty cluster.

Figure 5.4: Clustered datasets 1954 and 2139 by K–Means into 3 groups. Only 1954 clustered well.

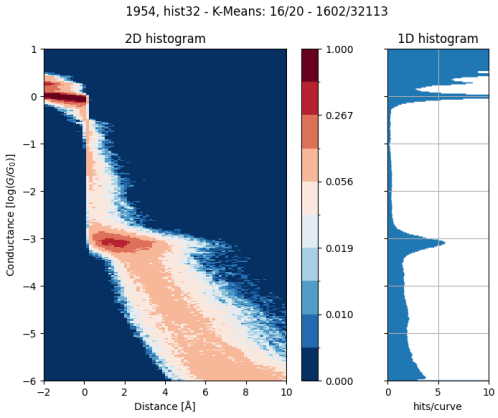
We also ran K–Means for 20 clusters. We used both histograms and traces as features. For this many clusters K–Means starts to be very sensitive for niche variance in groups.

For example using histograms as input, K–Means was able to group noisy traces together (see figure 5.5a), which can be marked as bad, because from physical perspective these traces do not make sense, because peaks in 1D histogram are too low.

Yet we were able to separate clusters with molecule even in our noisy 1954 dataset (figure 5.5b).



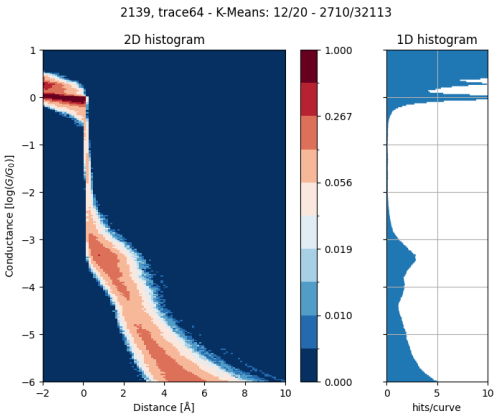
(a) : Bad traces with histogram peaks near OPAMP limit.



(b) : Cluster with molecule.

Figure 5.5: Examples of bad and good cluster from K-Means; selection out of 20 clusters.

We were not able to distinguish between two configurations of molecule bonding with *trace64* feature (figure 5.6), but with this feature space we were able to distinguish between traces with molecule and without.



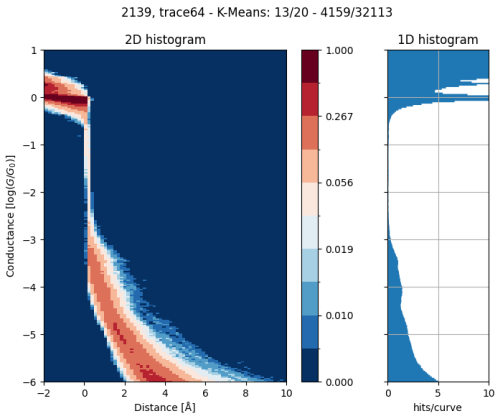


Figure 5.6: Clustered with K-Means with *trace64* feature space.

5.3 Balanced Iterative Reducing and Clustering using Hierarchies

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [29] is a centroid based clustering algorithm that builds a feature tree from given data. Branching factor B and threshold T are the two parameters involved in tree creation. Each leaf of that tree is called Clustering Feature (CF) and it is essentially a triple

$$CF = (N, \vec{LS}, SS), \tag{5.1}$$

where N is number of entries in a node, linear sum $\vec{LS} = \sum_{i=1}^{N} \vec{X}_i$ and squared sum $SS = \sum_{i=1}^{N} \vec{X}_i^{\,2}$.

Branching factor sets the maximum number of subclusters in a node and threshold defines the maximum $||CF||$ of leafs. Each non-leaf node contains up to B entries in form of $(CF_i, child_i)$, where $i = 1, \ldots, B$ and $child_i$ is i-th pointer to another node. We insert a new sample X_i into the root of CF tree and recursively merge with a node of smallest $||CF||$ after merging, leaf nodes should satisfy $||CF|| < T$, if not the leaf is splitted.

After CF tree creation we use AgglomerativeClustering algorithm [25] to obtain final K clusters.

SciKit learn sets the number of clusters to 3 by default, it also sets branching factor to 50 and threshold to 0.5 [25].

We opted for *hist32* feature space, because *trace64* feature space failed to sort traces with and without molecule. (figure 5.7).

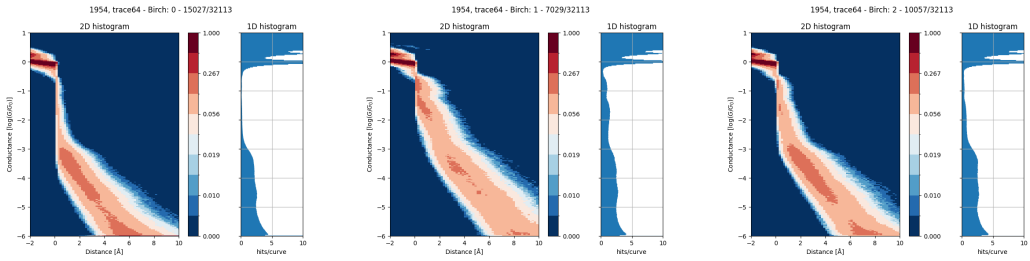
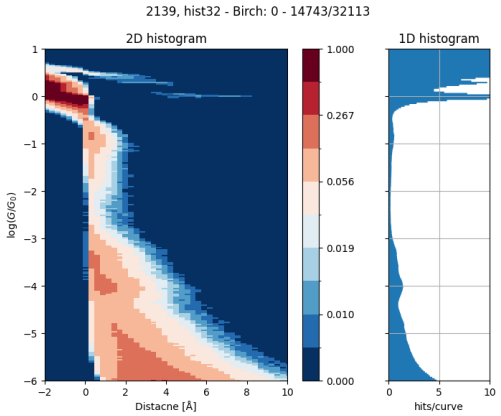
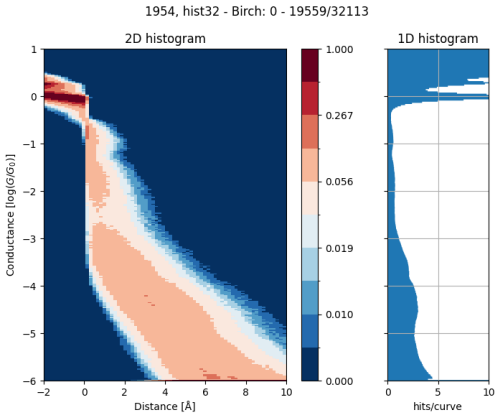


Figure 5.7: BIRCH failed clustering with *trace64* feature space. There is no considerable peak in 1D histogram in any cluster.

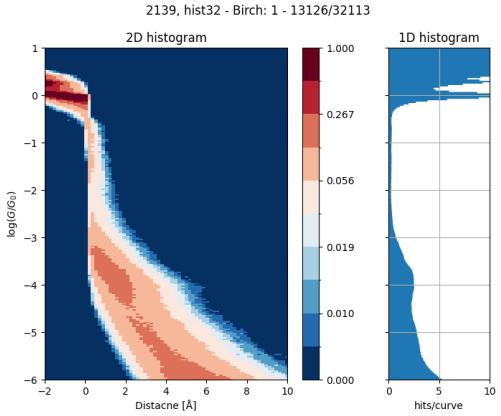
BIRCH was able to discover both geometry configurations in 1954 dataset (see figures 5.8d and 5.8f). But the distinction between one geometry group and tunneling current in 2139 dataset is not that clear (figures 5.8a and 5.8c).



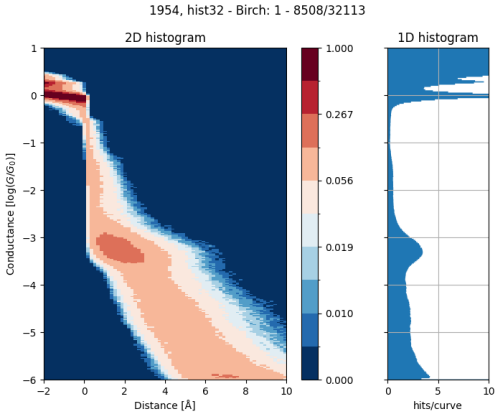
(a) : Molecule in second configuration.



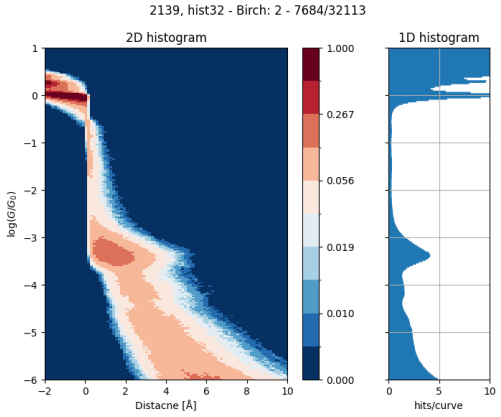
(b) : Noisy tunneling current.



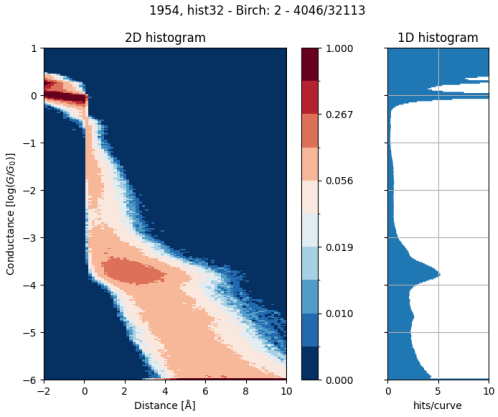
(c) : Tunneling current.



(d) : Molecule in first configuration.



(e) : Molecule in first configuration.



(f) : Molecule in second configuration.

Figure 5.8: Clusters from BIRCH.

5.4 Unsuitable algorithms

We were not successful with other clustering algorithms. We tried algorithms as follows. Some of them created many small clusters and failed to recognize traces with and without molecule. Others failed to find any other cluster and returned only one cluster.

We tried both *hist32* and *trace64* feature spaces.

5.4.1 Density based clustering

Density Based Spatial Clustering of Applications with Noise (DBSCAN). DBSCAN is a deterministic algorithm for discovering clusters based their density. It is able to discover non-convex clusters, its only parameter is *eps*, lower the *eps* means higher density clusters. [30]

Parameter *eps* is a neighborhood ε. If distance between two samples satisfies $||x_i - x_j|| < \varepsilon$, those samples are candidates for cluster (see figure 5.9). If there is more than 700 samples satisfying this condition we assign these samples into a cluster.

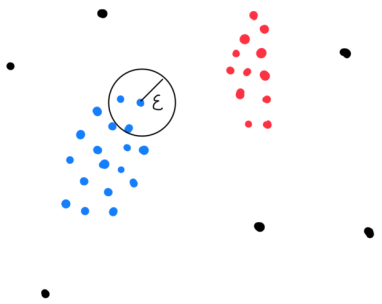


Figure 5.9: Example of DBSCAN clustering on a 2D plane with ε visualized, where blue and red dots are members of cluster and black dots represent noise.

We had to fine-tune *eps* parameter. We iteratively changed *eps* in range of 1 to 50. We settled down on $eps = 15$, which divided our dataset into two parts. Albeit this division does not yield meaningful clusters (figure 5.10). Cluster labeled -1 means that those traces do not belong into a cluster, they are outliers. Cluster 0 contains traces with and without molecule, therefore we conclude that DBSCAN is not a suitable candidate for break junction data clustering. For $eps < 15$ the outlier cluster -1 was too big and based on K–Means and BIRCH clusters we were not observing clusters with molecule present.

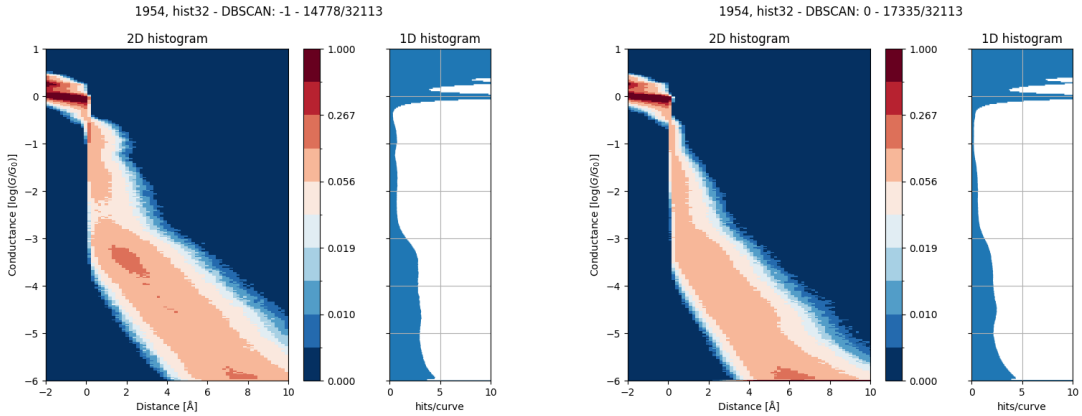


Figure 5.10: Clusters from DBSCAN. Cluster labeled -1 contains outliers, which is similar to cluster 1.

Hierarchy Density Based Spatial Clustering of Applications with Noise. Hierarchy Density Based Spatial Clustering of Applications with Noise (HDBSCAN) [31] is based on previously mentioned DBSCAN. It builds a weighted reachability graph across all samples and than it greedily removes edges with highest weight. A cluster is formed if it contains more than k connected components.

The algorithm with $k = 700$ did not find any clusters for neither *hist32* and *trace64* feature spaces.

Ordering Points to Identify the Clustering Structure. Ordering Points to Identify the Clustering Structure (OPTICS) [32] is similar to DBSCAN, it uses a range of ε instead of a fixed value.

Samples of dataset are processed in such order, that the closest points become neighbors in the ordering.

We were not able to obtain any cluster from OPTICS algorithm. But fine-tuning parameters or different feature spaces [12] might give us meaningful clusters.

5.4.2 Centroid based clustering

Mean Shift. Mean Shift algorithm [33] discovers blobs in feature space, it selects centroids based on density gradient and assign samples to cluster based on centroids proximity.

We were not able to discover any clusters with Mean Shift.

Affinity propagation. Affinity propagation [34] treats each sample as centroid candidate. It sends availability and reachability messages between samples and based on message traffic centroids are discovered.

The algorithm returned hundreds of clusters with very similar traces, but no cluster contained more then 700 members, which does not meet our criteria of minimal cluster size.

Chapter 6

iCluto programme

Previous histograms and graphs were generated by iCluto.

iCluto expands the capability of CLUTO (Clustering Toolkit) from Karypis [10], a general clustering toolkit. The need to manage our datasets led us to develop our own set of tools. iCluto aims to connect unsupervised and supervised methods and incorporate parallelization of data processing.

Used language and libraries. iCluto is written in Python 3 [35]; it requires Python 3.8 and higher. It is packaged by Poetry [36], which allows us to deploy iCluto on desktops and servers easily. We use Torch library [37] for designing and training snapback model (see 4.1.1) and SciKit Learn [25], a library for scientific data analysis, for all clustering algorithms. All plots are generated with Matplotlib [38].

iCluto is not a stand-alone application, it is more of a set of tools that one needs to analyze breaking traces. We will describe each tool in following sections.

6.1 Dataset annotation

In section 4.1.1 we came up with CNN. In order to learn and validate our models, we need labelled data. This task is very monotonous and time-consuming. For this reason we developed a GUI application for data annotation. One can notice that our interface (see figure 6.1) can label many events: bulk, 3,2 and 1 atom, snap start, snap end, limit, presence of molecule, quality of trace.

Bulk. This is an area of high conductivity, where the change is continuous function and not discrete, it precedes all other labels. It should be placed some distance before atom plateaus. This label is reserved for future use; therefore, its specifications are not exact.

Atoms. In section 3.1 we introduced a simplified equation for conductivity (equation 3.3). We label 3 atoms, 2 atoms and 1 atom are for $N = 3$, $N = 2$ and $N = 1$ respectively. Label 1 atom is in most cases redundant with Snap start label, but its intended use was for training a neural network for atom plateaus detection. Atom plateaus are not obvious in some traces, therefore 3, 2 and 1 atom labels cannot be placed for every trace in our datasets. We put these labels at the end of the atom plateau.

Snapback. Snapback event is significant, as mentioned in section 4.1. Our first attempt to detect snapback was done at the end of the event, later on, we have decided to mark the start.

Presence of molecule. Molecule can bridge electrodes only after snapback, we mark its presence with buttons *Molecule Yes* and its absence with *Molecule No*. At the current version of iCluto we do not use this label.

Quality of trace. This label marks whether we can extract valuable data from the trace. Traces similar to 2.4 are marked as *bad*, while the rest are marked *good*. Most of the bad traces are filtered out thanks to filters described in section 4.6. We use this label for verifying whether our criteria for good traces are met.

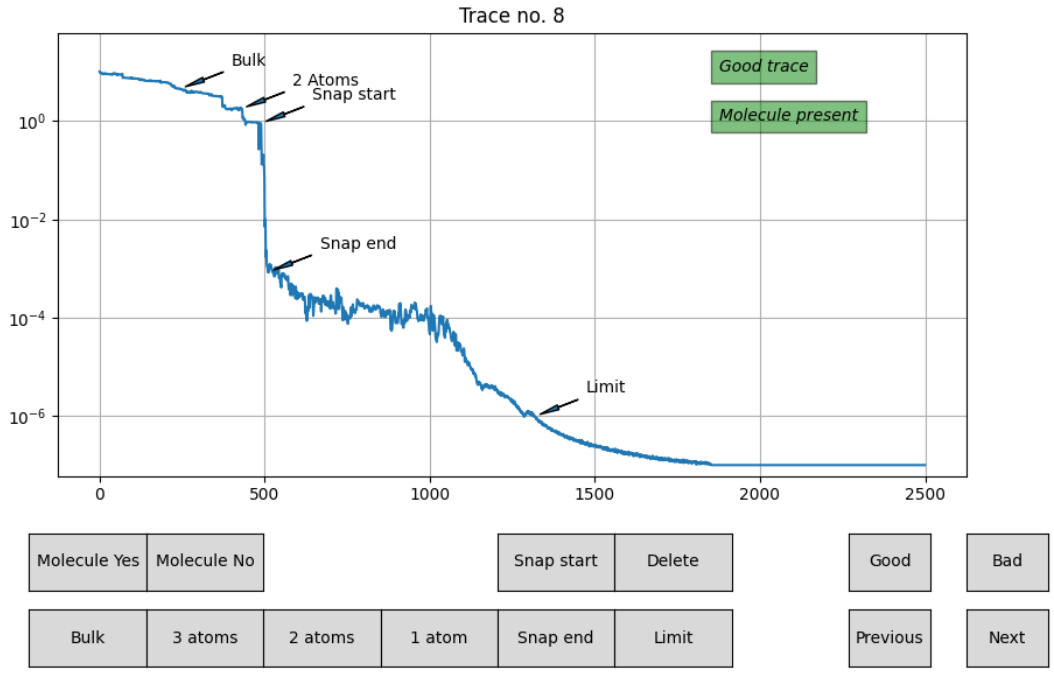


Figure 6.1: GUI for labeling traces.

Currently, we are using only labels for the start and end of snapback, but we have decided to include other labels for future use. Even though it is easy to add different labels to our GUI, having them already implemented will speed up future development.

Our GUI exports labels as JSON files [39]. This format was chosen for its simplicity and readability. It is also a part of Python's standard libraries. If needed, we can easily change any value in a text editor.

Listing 6.1: Example of labels for trace number 8 from .JSON file.

```
"8": {
    "Molecule": 1,
    "Quality": 1,
    "Snap start": 488,
    "Snap end": 511,
    "Limit": 1318,
    "Bulk": 244,
    "2 Atoms": 429
    },
```

6.2 Model validation

The first validation of our models is done on the testing dataset. We split our dataset into training and testing sets; during training, we feed training data into our model, and we let it do the forward pass and back propagation so it can adjust its weights. Then, we validate its accuracy on a testing dataset. We have labelled ~ 600 traces so far, which means our training dataset has ~ 480 traces, and the testing dataset has ~ 120 traces.

We have developed another GUI for inspecting traces one by one (see figure 6.2).

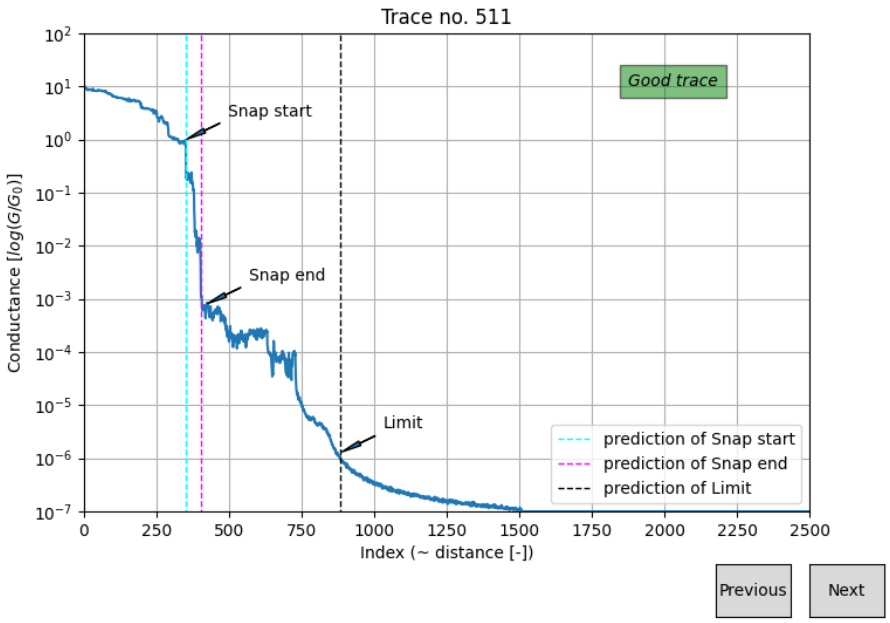


Figure 6.2: Inspecting models described in section 4.1.

6.3 Clustering and Cluster inspection

Clustering. Clustering is done mainly in Jupyter notebooks [40]. This allows us to quickly change features and clustering parameters without loading traces again. It saves us time while keeping clustering pipeline organized.

Cluster inspection. The last GUI (figure 6.3) was developed for inspecting individual clusters trace by trace. This is important; as only from histograms we cannot tell if there are only traces with molecule present and if some anomaly is present in the cluster.

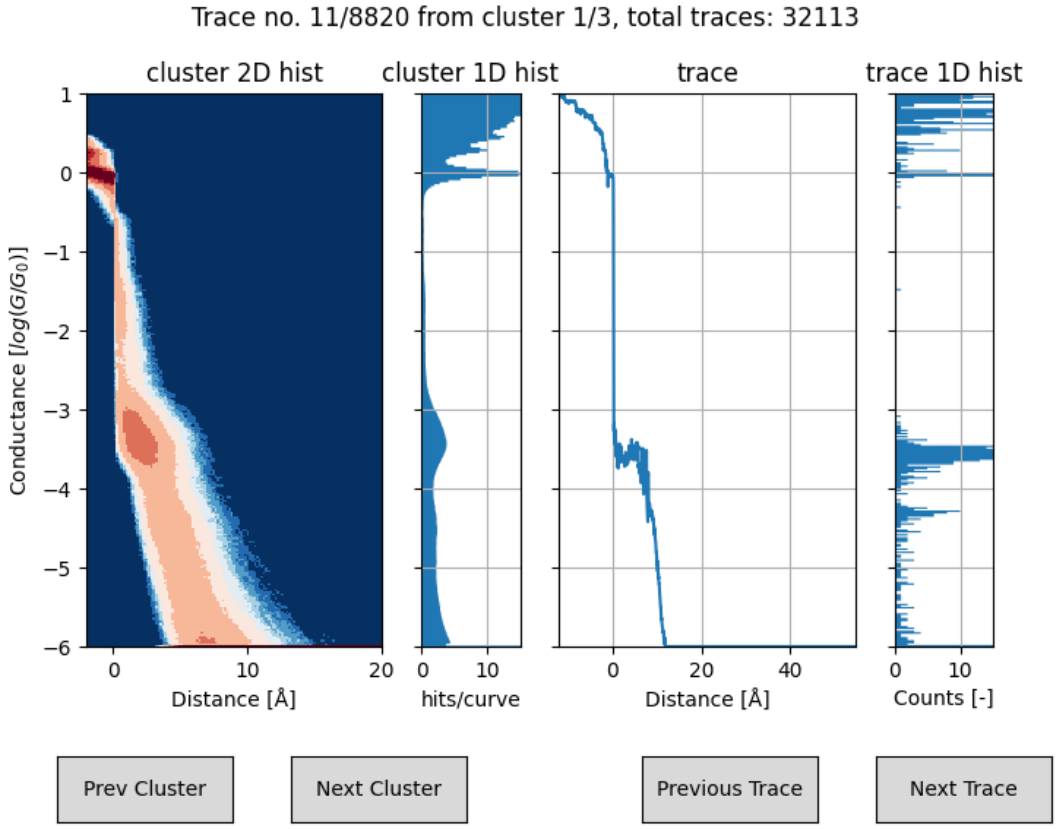


Figure 6.3: Inspecting cluster and its traces. Two left windows show 2D and 1D histograms of whole cluster, two right windows show current trace and its 1D histogram. User can list through clusters and their traces using buttons.

Chapter 7

Conclusion

In this thesis we report on successful concept, design and creation of program for data clustering of conductance traces from break junction experiment.

Molecule bridging electrodes is a pseudo-random phenomena, therefore we need large quantity of data, files consisting of several GBs, which needs to be analyzed statistically. Only a fraction of these traces are usable for statistical interpretation.

We have developed iCluto that is capable of clustering large datasets efficiently. Our toolkit uses modern programming language, it runs fast and it utilize available resources economically. Clustering algorithms spread the workload across all available CPU cores. Filtering of 36000 traces is done under 2 minutes, clustering, depending on used algorithm, ranges from 30 seconds to 120 seconds. This is an improvement over CLUTO and Breaker programs, where clustering took hours.

Our development of graphical user interfaces for data annotation and cluster inspection allows us to inspect more datasets quickly. iCluto's GUIs can be accessed remotely on a server or locally on Linux, MacOS or Windows. Both labeling and inspection tools are user friendly.

We proposed a clustering pipeline, that contains filtration and processing blocks. We ran various clustering algorithms with success, capable separating traces with and without molecule.

K-Means clustering for 3 clusters is suitable for standard dataset. It is capable of sorting for two geometry groups and tunneling current for not noisy dataset, but fails for noisy one. For high number of clusters it can find molecules in both geometry groups even in noisy datasets.

We have demonstrated, that BIRCH algorithm is suitable for noisy datasets, because it can distinguish between two geometry groups and tunneling current, yet it was not able to detect them in a standard dataset.

A combination of the two mentioned algorithms might cover the majority of datasets.

We ran iCluto's clustering pipeline only on two datasets, which limits our tests and validations. We are still relying on knowledgeable personnel to determine presence of molecule.

From a clustering application iCluto has grown into a break junction research ecosystem. We sped up clustering by a significant margin and we are able to process big data.

Chapter 8

Future work

This thesis serves as foundation for future research. The possibilities for further development are new feature spaces, more labels or fine tuning implemented algorithms.

Features. We discussed *hist32* and *trace64* feature spaces, but there are more features spaces to consider. For example we can train encoder and decoder for traces and use the middle hidden layer as feature space (figure 8.1) [41].

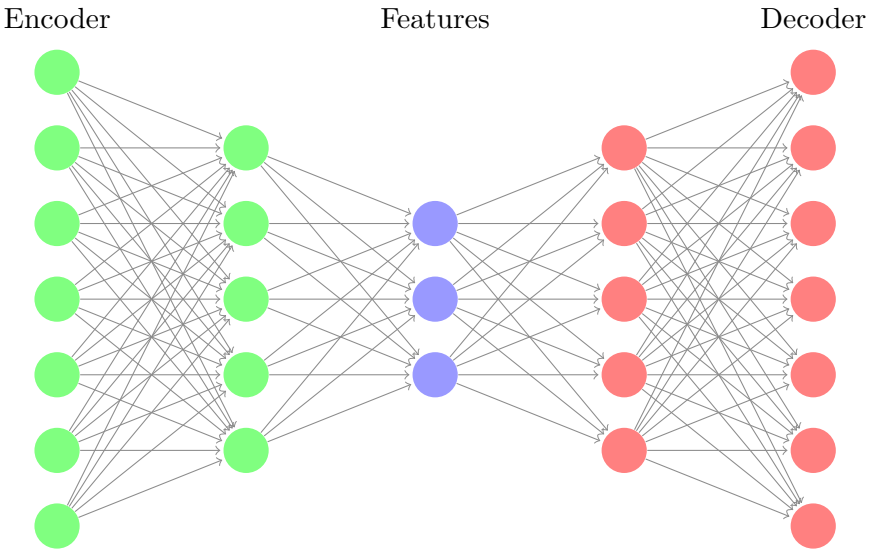


Figure 8.1: Encoder and decoder in neural network.

Labels. Obtaining labels is very time consuming. Once we annotate our data, we treat them as ground truth, which means we use them for training and validation of machine learning models without questioning their correctness. We rely on persons labeling these data, which means we want as many as possible knowledgeable researches to participate in data labeling.

Fine tuning. In section 5.3 we noticed that BIRCH can cluster well noisy 1954 dataset but fails on 2139, this might require fine tuning its branching and threshold parameters. The same applies for the other algorithms mentioned in section 5.4.

Supervised clustering. Labeled data would allow us to use Support Vector Machine (SVM) or an outlier detection, thus adding another filter to our toolkit.

We can train classifiers for not only filtering, but also for molecule recognition. If trained well, our classifiers could predict the type of molecules in solution.

Appendix A

Index of terms

BJ. Break junction is an apparatus for obtaining breaking curvers and measuring conductivity on a nanoscale level. 3, 9

breaking trace. Breaking trace or conductance traces (or curve) is function of conductivity versus displacement. 3

bulk. Bulk is a high conductance state, where the change is continuous function not discrete. In case of wire it is at least 20 atoms in diameter. 3

snapback. Snapback is an event in breaking trace where a sudden loss of conductivity happens when a wire breaks. 3

FORTRAN. FORTRAN is an acronym for Formula Translator; a programming language. 6.

MCBJ. MCBJ is an acronym for Mechanically Controllable Break Junction. 10

STM-BJ. STM-BJ as an acronym for Scanning Tunneling Microscopy-Based Break Junction. 10

GB. GB is giga byte. 13,31

limit. Limit is an event in conductance trace at $G = 10^{-6}$. 13

NN. NN is an acronym for Neural network. 14

CNN. CNN is an acronym for Convolutional neural network. 14

Padding. Padding means adding zeros to input vector. 14

Dropout. Dropout is a technique in learning phase of neural network, where we randomly zeroes elements of NN. 14

indifactor. A scaling factor for distance in Å. 16

OPAMP. OPAMP is an acronym for Operational Amplifier. 16

PCA. Principal component analasys is a common dimensionality reduction technique. 17

K–Means. K–Means is a centroid based clustering algorithm. 20

BIRCH. BIRCH is an acronym for Balanced Iterative Reducing and Clustering using Hierarchies; a centroid based clustering algorithm.23

CF. CF is an acronym for Clustering feature. 23

DBSCAN, HDBSCAN. (H)DBSCAN is an acronym for (Hierarchy) Density Based Spatial Clustering of Applications with Noise; a density based clustering algorithm. 25

OPTICS. OPTICS is an acronym for Ordering Points to Identify the Clustering Structure; a density based clustering algorithm. 26

Mean Shift. Mean Shift is a centroid based clustering algorithm. 26

Affinity propagation. Affinity propagation is a centroid based clustering algorithm. 26

GUI. GUI is an acronym for Graphical user interface. 28, 29

SVM. SVM is an acronym for Support vector machine; a classification method in machine learning. 34

Bibliography

[1] M. Bednařík, *Studjní texty k předmětu Fyzika 2.* Katedra fyziky, Fakulta elektrotechnická, České vysoké učení technické v Praze, 2020.

[2] M. Kamenetska, *Single molecule junction conductance and binding geometry.* Columbia University, 2012.

[3] A. Aviram and M. A. Ratner, "Molecular rectifiers," *Chemical Physics Letters*, vol. 29, no. 2, pp. 277–283, 1974. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0009261474850311

[4] T. A. Su, M. Neupane, M. L. Steigerwald, L. Venkataraman, and C. Nuckolls, "Chemical principles of single-molecule electronics," *Nature Reviews Materials*, vol. 1, no. 3, pp. 1–15, 2016.

[5] H. Sirringhaus, "Device physics of solution-processed organic field-effect transistors," *Advanced Materials*, vol. 17, no. 20, pp. 2411–2425, 2005. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.200501152

[6] H. A. Alhadrami, "Biosensors: Classifications, medical applications, and future prospective," *Biotechnology and Applied Biochemistry*, vol. 65, no. 3, pp. 497–508, 2018. [Online]. Available: https://iubmb.onlinelibrary.wiley.com/doi/abs/10.1002/bab.1621

[7] J. Nejedlý, "The synthesis of π-electron systems suitable for transfer and retention of charges," 2021.

[8] J. R. Widawsky, *Probing Electronic and Thermoelectric Properties of Single Molecule Junctions.* Columbia University, 2013.

[9] W. Bro-Jørgensen, J. M. Hamill, R. Bro, and G. C. Solomon, "Trusting our machines: validating machine learning models for single-molecule transport experiments," *Chemical Society Reviews*, vol. 51, no. 16, pp. 6875–6892, 2022.

[10] G. Karypis, "Cluto-a clustering toolkit," 2002.

[11] J. M. Hamill, X. T. Zhao, G. Mészáros, M. R. Bryce, and M. Arenz, "Fast data sorting with modified principal component analysis to distinguish unique single molecular break junction trajectories," *Phys. Rev. Lett.*, vol. 120, p. 016601, Jan 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.120.016601

[12] N. D. Bamberger, J. A. Ivie, K. N. Parida, D. V. McGrath, and O. L. A. Monti, "Unsupervised segmentation-based machine learning as an advanced analysis tool for single molecule break junction data," *The Journal of Physical Chemistry C*, vol. 124, no. 33, pp. 18 302–18 315, 2020. [Online]. Available: https://doi.org/10.1021/acs.jpcc.0c03612

[13] D. Cabosart, M. El Abbassi, D. Stefani, R. Frisenda, M. Calame, H. S. Van der Zant, and M. L. Perrin, "A reference-free clustering method for the analysis of molecular break-junction measurements," *Applied Physics Letters*, vol. 114, no. 14, 2019.

[14] M. El Abbassi, J. Overbeck, O. Braun, M. Calame, H. S. van der Zant, and M. L. Perrin, "Benchmark and application of unsupervised classification approaches for univariate data," *Communications Physics*, vol. 4, no. 1, p. 50, 2021.

[15] K. P. Lauritzen, A. Magyarkuti, Z. Balogh, A. Halbritter, and G. C. Solomon, "Classification of conductance traces with recurrent neural networks," *The Journal of Chemical Physics*, vol. 148, no. 8, p. 084111, 02 2018. [Online]. Available: https://doi.org/10.1063/1.5012514

[16] T. Fu, Y. Zang, Q. Zou, C. Nuckolls, and L. Venkataraman, "Using deep learning to identify molecular junction characteristics," *Nano Letters*, vol. 20, no. 5, pp. 3320–3325, 2020, pMID: 32242671. [Online]. Available: https://doi.org/10.1021/acs.nanolett.0c00198

[17] J. Ulrich, D. Esrail, W. Pontius, L. Venkataraman, D. Millar, and L. H. Doerrer, "Variability of conductance in molecular junctions," *The Journal of Physical Chemistry B*, vol. 110, no. 6, pp. 2462–2466, 2006.

[18] J. Moreland and J. W. Ekin, "Electron tunneling experiments using Nb-Sn "break" junctions," *Journal of Applied Physics*, vol. 58, no. 10, pp. 3888–3895, 11 1985. [Online]. Available: https://doi.org/10.1063/1.335608

[19] C. Muller, J. van Ruitenbeek, and L. de Jongh, "Experimental observation of the transition from weak link to tunnel junction," *Physica C: Superconductivity*, vol. 191, no. 3, pp. 485–504, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/092145349290947B

[20] "What is NI LabVIEW? Graphical Programming for Test & Measurement — ni.com," https://www.ni.com/en/shop/labview.html, [Accessed 29-04-2024].

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[22] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, 1st ed. USA: Cambridge University Press, 2019.

[23] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow.* " O'Reilly Media, Inc.", 2022.

[24] T. Werner, *Optimalizace.* Katedra kybernetiky, Fakulta elektrotechnická, České vysoké učení technické v Praze, 2023.

[25] 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.

[26] A. Borges, E.-D. Fung, F. Ng, L. Venkataraman, and G. C. Solomon, "Probing the conductance of the σ-system of bipyridine using destructive interference," *The Journal of Physical Chemistry Letters*, vol. 7, no. 23, pp. 4825–4829, 2016, pMID: 27934052. [Online]. Available: https://doi.org/10.1021/acs.jpclett.6b02494

[27] J. MacQueen, "Some methods for classification and analysis of multivariate observations," 1967.

[28] D. J. KETCHEN and C. L. SHOOK, "The application of cluster analysis in strategic management research: An analysis and critique," *Strategic Management Journal*, vol. 17, no. 6, pp. 441–458, 1996. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-0266%28199606%2917%3A6%3C441%3A%3AAID-SMJ819%3E3.0.CO%3B2-G

[29] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," *SIGMOD Rec.*, vol. 25, no. 2, p. 103–114, jun 1996. [Online]. Available: https://doi.org/10.1145/235968.233324

[30] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[31] L. McInnes and J. Healy, "Accelerated hierarchical density based clustering," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017, pp. 33–42.

[32] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," in *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '99. New York, NY, USA: Association for Computing Machinery, 1999, p. 49–60. [Online]. Available: https://doi.org/10.1145/304182.304187

[33] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[34] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1136800

[35] "Welcome to Python.org," Apr. 2024, [Accessed 28-04-2024]. [Online]. Available: https://www.python.org/

[36] S. Eustace and The Poetry contributors, "Poetry: Python packaging and dependency management made easy." [Online]. Available: https://github.com/python-poetry/poetry

[37] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, and S. Chintala, "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation," in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, Apr. 2024. [Online]. Available: https://pytorch.org/assets/pytorch2-2.pdf

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

[39] "JSON — json.org," https://www.json.org/json-en.html, [Accessed 29-04-2024].

[40] "Project Jupyter — jupyter.org," https://jupyter.org, [Accessed 28-04-2024].

[41] Y. Komoto, J. Ryu, and M. Taniguchi, "Machine learning and analytical methods for single-molecule conductance measurements," *Chem. Commun.*, vol. 59, pp. 6796–6810, 2023. [Online]. Available: http://dx.doi.org/10.1039/D3CC01570J