VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
DEPARTMENT OF COMPUTATIONAL AND DATA MODELING

Master's final thesis

Artificial intelligence algorithms for processing of atomic force microscopy images

Done by:
Denys Kryvoshei signature

Supervisor:
prof., dr. Tadas Meškauskas

Vilnius
2023

Contents

Contractual glossary of terms

- AFM - atomic force microscopy.
- AI - artificial intelligence.
- ANN - artificial neural network.
- CNN - convolutional neural network.
- EIS - electrochemical impedance spectroscopy.
- FN - false negative.
- FP - false positive.
- mAP - mean average precision.
- MFT - Master’s Final Thesis.
- SRP - Scientific Research Project.
- tBLM - tethered bilayer lipid membrane.
- TN - true negative.
- TP - true positive.
- YOLO - you only look once.
- F/C - full-color.
- G/S - grayscale.

Abstract

The goal of this thesis is to use an artificial intelligence approach for defect detection in AFM images in order to preserve the clusterization parameters such as Voronoi entropy and Voronoi σ. The first chapters will cover the theoretical part by explaining scientific terms, describing the biological nature of tethered bilayer lipid membranes (tBLMs), defects and what's their danger to living organisms. In chapters 4 and 6, the reader will be introduced to Voronoi diagramming and the basics of digital image processing as well as methods of labeling AFM pictures. Besides that, there will be given a brief overview of feature extraction techniques such as the Hough circle transform and its performance on AFM images. But the main emphasis of this thesis is on the usage of a convolutional neural network for the detection of defects inside clusters because it may improve the prediction of electrochemical impedance spectroscopy response. A CNN model named YOLOv5 was selected as a basis for future modifications of the activation function. For the training process were prepared datasets with different combinations of channels and defect types. In the end, the model demonstrated a significant improvement over the author's research work by maintaining clusterization metrics relatively close to the original ones and improving the accuracy of defect detection within clusters. In addition, the model is able to detect both individual defects and defects within clusters, which can be considered a success.

Santrauka

Dirbtinio intelekto algoritmai atominės jėgos mikroskopijos vaizdų apdorojimui

Mikroskopijos technologija nuo pat jos išradimo labai patobulėjo ir yra skirstoma į tris pagrindinius tipus: optinę, krūvį turinčių dalelių (elektronų ir jonų) ir skenuojančio zondo. Šiame projekte daugiausia dėmesio skiriama atominės jėgos mikroskopijai (AFM), kuri laikoma skenuojančio zondo mikroskopijos šaka. Naudodami AJM mokslininkai gali gauti detalius, didelės skiriamosios gebos pririštų lipidų dvisluoksnių membranų vaizdus ir aptikti defektus. Šie defektai - tai sritys, kuriose membrana buvo pažeista dėl fizinio poveikio ląstelei arba dėl poras sudarančių toksinų poveikio. Kontaktas su šiais toksinais gali pažeisti ląstelių membranas. Tai sukelia sunkias ligas, pavyzdžiui, Parkinsono ligą. Defektai būna įvairaus dydžio ir formos, jie gali būti pilno apskritimo arba įvairių lankų formos. Kartais defektai koncentruojasi vienoje vietoje, sudarydami klasterius, todėl sunku atskirti kiekvieną atskirą defektą ir jo tipą. Be to, AJM vaizdai gali būti naudojami elektrocheminėje impedanso spektroskopijoje tiriant poras sudarančių toksinų elektrinį poveikį.

Pagrindinis šio darbo tikslas - taikyti dirbtinio intelekto metodą defektams AJM vaizduose aptikti, siekiant išlaikyti klasterizacijos parametrus, tokius kaip Voronoi entropija ir standartinis nuokrypis, kurie gali pagerinti pritvirtintų dvisluoksnių membranų EIS atsako prognozavimą.

Šio darbo pradžioje, skyriuose 1, 2 ir 3.3 skaitytojas ras trumpą susijusių straipsnių, darbų ir susijusios programinės įrangos analizę, po kurios bus pateikti teoriniai biologinių ląstelių, pririštų lipidų dvisluoksnių membranų, AJM veikimo principo, defektų pobūdžio ir galimų pasekmių nešikliui aprašymai. Skyriuje 4 apžvelgiamos Voronoi diagramos ir susijusios klasterizavimo metrikos, o skyriuje (5) pateikiama objektų aptikimo priemonių analizė, įskaitant požymių išskyrimo metodų ir dirbtinių neuronų tinklų palyginimą.

Požymių išskyrimo įrankiai nedavė jokių teigiamų rezultatų, nes jie veiksmingiausi, kai tiksliniai požymiai yra panašaus dydžio ir formos, tačiau defektai paprastai labai skiriasi.

Skyriuje 6 pateikiama informacija apie duomenų rinkinio parengimą, vaizdų apdorojimą, tyrimų duomenis ir įvairių anotacijų formatų, tokių kaip Pascal VOC ir YOLO, palyginimą. Skyrius 7 skirtas programavimo dalies įgyvendinimui, kompiuteriniams eksperimentams ir rezultatų analizei. Autorius aprašo programavimo aplinką, paaiškina, kodėl buvo nuspręsta naudoti CNN modelį YOLOv5 kaip dirbtinio intelekto metodą ir kaip buvo vykdomas pasirinkto modelio mokymas.

Atlikta daugiau kaip 15 kompiuterinių eksperimentų, naudojant daugiau kaip 10 duomenų rinkinių su skirtingais vaizdų ir defektų tipų deriniais. Apmokytas modelis reikšmingai patobulėjo, palyginti su autoriaus moksliniu tiriamojo darbo projektu (parašytu praėjusiame semestre), nes geriau išlaikė (palyginus su ekspertiškai sužymėtais AJM vaizdais) klasterizacijos metrikas, ir pagerino defektų aptikimo klasterių viduje tikslumą. Be to, modelis geba detektuoti tiek atskirus defektus, tiek defektus klasterių viduje, o tai galima laikyti sėkme.

Introduction

Ancient scientists always wondered about the body structure of both living and non-living things in the surrounding world. Step by step, science developed and people slowly but steadily came up with new technologies to examine nature. One of the most important inventions is surely the microscope. It was in the 1500ths years when the idea of the usage of magnifying lenses for object investigation came to European people's minds. Thanks to the new device, humanity gained the ability to take a closer look at how we are constructed. Since then, scientists hugely developed this technology which led to the appearance of new microscopes.

The human eye cannot perceive information from these modern microscopes because the data is in digital format. So, to process the output one can not do without special hardware and software. That means, it has been a long time ago, since natural science and information technology fields are closely tied up together, helping each other with different needs.

During the examination of particles of living organisms using microscopes it was discovered that they consist of cells. These cells act as fundamental building blocks in our bodies. With time and newer technologies scientists got to know the structure of the cells in more detail. For example, there is a tethered bilayer lipid membrane that separates the inner parts of the cell from the external environment and serves as a shield. Of course, these natural "bricks" can be damaged in different ways. One of the variations is so-called defects. They occur under different circumstances, either it is a normal process or some anomaly. For instance, defects can appear during cell aging or due to exposure to toxins. The enormous amount of defects can lead to extremely dangerous consequences for their carrier such as Parkinson's disease or other illnesses.

Atomic force microscopy allows one to take high-resolution pictures of a quite small area of a particle. Using these photos scientists can distinguish and point out present defects. Sometimes the number of such defects in one image reaches up to hundreds. Moreover, they can unite, creating aggregation clusters, where it is hard to distinguish individual defects. So, when there is a need to analyze thousands of atomic force microscopy photographs, it becomes a time-consuming task. Besides that, AFM pictures with known defect positions can assist in an investigation of the electrical effects of pore-forming toxins on surfaces using electrochemical impedance spectroscopy (EIS). That means finding a reliable way to automatically recognize the defects will make a positive impact on other studies.

So, the goal of this master's thesis is to estimate the efficiency of an artificial intelligence approach in preserving the clusterization parameters during defect detection in AFM images. Metrics such as Voronoi entropy and Voronoi σ will be used to measure the level of clusterization of an image. The images themselves are provided by the Life Sciences Center at Vilnius University and will be prepared for future manipulations, such as processing and labeling. Firstly, non-AI ways will be considered: feature extraction techniques, Hough circle transform in particular. After they prove themselves not suitable for this task, will be applied a CNN model YOLOv5. Different activation functions and dataset combinations will be tested to find the optimal configuration, which in the end will result in around 75% of precision on test images. Besides that, Voronoi diagramming will be applied to the model output in order to calculate clusterization metrics.

During the work, the author has written 300 lines of code while creating functions for calculating Voronoi metrics and working with datasets. More details are provided in section 7.1

The thesis can be roughly split into 3 major steps, so the approximate plan looks like this:

- Overview of available methods and techniques:

  - Voronoi diagramming;
  - Feature extraction tools and AI methods;
- Preparation of the dataset:
  - processing of raw images;
  - defects labeling;
  - formats conversion;
- Analysis of the CNN efficiency:
  - model modifications;
  - experiments with different data and parameters;
  - analysis of the results.

Work structure:

- Sections 1 and 2. An overview of related materials and software.
- Section 3. Theoretical information about the structure of tBLMs, how toxins create defects, and the operation of atomic force microscopy.
- Section 4. Overview of clusterization in AFM images and Voronoi diagramming.
- Section 5. Description of primitive object detections methods, such as Hough circle transformation algorithm and advanced AI tools starting from perceptron, ending with state-of-art CNN model YOLOv5.
- Section 4. Steps to organize the dataset for the CNN model training. Provided examples of different channels of AFM images, comparison of Pascal VOC and YOLO annotation formats, description of mosaic augmentation importance, and ways of splitting the dataset into training and testing parts.
- Section 5. Model training with different datasets and results analysis. Description of environment and used software.

The following chapters were used from the author's scientific research project [20]:

- The summary in English was used and updated.
- The summary in Lithuanian was used and updated.
- The introduction was used and updated.
- Chapter 1 (Related material) was used and updated.
- Chapter 3 (Biological membranes, defects and atomic force microscopy) was used.
- Chapter 5 (Object detection) was used and updated.
- Chapter 6 (Dataset preparation) was used and updated.

1 Related material

This section provides a general overview of related articles, web resources, and other literature. There is not much information about this particular topic of research, but some papers either contain intermediate steps and technical realization background or theoretical insights on the efficiency of different approaches. Analysis of the literature described in the list below may help to speed up the implementation process and improve the quality of this research work:

1. The latest paper from this list was published in 2022 and written by Tomas Raila et al., from the Faculty of Mathematics and Informatics at Vilnius University [33]. One of the objectives of this article was to investigate the main problems that occur during the automated recognition of defects in atomic force microscopy images and how to overcome them. Tomas et al. give a description of the detection algorithms' accuracy by describing precision, recall, and F1 prediction metrics.

   Besides just theory, the paper provides a comparison of Convolutional Neural Network (CNN) based models and TopoStats. The latter one demonstrated similar accuracy but significantly lower recall values. Tomas says that a great number of defects were not detected and suggests the next reason: the AFM images may have aggregations of defects (clusters, in other words) that become a problem for TopoStats and other non-AI-based detection methods.

2. Vytenis Navalinskas in 2020 defended a master's thesis on the topic "Defektu atpažinimas biologinese membranose" [27]. It is written in Lithuanian, but nevertheless contains a vast amount of theoretical information about the defects in biological cell membranes. Also, the author describes basic Deep Learning principles such as confusion matrices and metrics. The paper provides information on how to process images and prepare them for future digital processing.

   The Navalinskas' implementation of the detection system is based on the YOLOv3 pretrained CNN model. Moreover, the architecture of the YOLOv3 model is analyzed and explained in detail. In this master's work, the author managed to achieve some results in defect detection. The recognition is general, meaning only positions of the defects are detected without the separation into different classes.

3. A similar project was accomplished by Igor Vilghelm in the year 2021 [42]. The topic is called "Automatizuoti biologiniu membranu defektu atpažinimo algoritmai atomines jegos mikroskopo nuotraukose". In this paper, the author as well provides a deep theory on how atomic force microscopy operates, and the way images are being produced.

   The main difference compared to the master's thesis above is the usage of an improved convolutional neural network model for the next generation. A newer version of YOLO, v4 if being precise, offers improved precision of object recognition and decreased training time, making it possible to implement more accurate and advanced classification features in a shorter period of time.

   It is worth mentioning that Igor gives a comprehensive explanation of digital image processing algorithms and feature extraction techniques such as the Gaussian filter, Canny edge detector, Hough circles transformation, and other methods. The theory is followed by the application results, with groundings of the outcomes.

The author succeeded to implement a multi-class multi-label model, which is able to predict two things at the same time:

- Defects positioning.
- Classification, according to the shape of the defect.

The final model showed around 80% accuracy on test images. It indicates that migration to the newer version of the CNN model resulted in the improvement of the results.

4. A scientific research project done by Denys Kryvoshei [20] serves as a basis of this thesis and searches for new directions which were not investigated previously. The main emphases of the project were the examination of the correlation between the model precision metrics and the next parameters:

   - The size of the training dataset.
   - The number epochs.
   - Usage of raw/grayscale AFM images.

   Besides that, the research project introduced a new CNN model and a bit different approach to how the training is being done. The main idea of the method was to first train the model on a smaller dataset and save the obtained weights. Later on, these weights were used as a basis during training on the bigger datasets instead of starting from scratch. This approach demonstrated interesting results: the model precision first dropped due to new dataset entries but with epochs stabilized again.

   In the end, the best results were achieved after training the model for 1500 epochs on a dataset containing 34 grayscale images (considered as the bigger dataset in the boundaries of the project). For the training were used weights from previous experiments on a smaller grayscale dataset. Unlike the previously mentioned paper by Igor Vilghelm, the goal of the project was not to distinguish between different defect types or achieve the highest precision but to test the ground for future research.

5. Recent master's thesis by Vita Mačinskaitė on the topic "Mašininio mokymosi metodai biologinių membranų defektų klasterizacijos analizei" also provides some important information. Even though the main emphasis of her paper is a bit different from this project and directed mostly on the generation of artificial AFM images with some particular level of clusterization, Vita gave an overview of some methods which allow to calculate the clusterization metrics of AFM images. One of the presented approaches is building Voronoi diagrams based on the defect coordinates. In later chapters, this technique will be applied and considered in more detail.

2 Related software

This section contains information about the software that was applied to process the AFM images and some examples of usage.

Most of the advanced applications for handling microscopy images are neither available for free nor have trial versions making it harder to compare them. A brief overview of several solutions was given by Igor in his master's thesis [42]. In order to concentrate the efforts on the practical part rather than repeating comparison, it was decided to choose NanoScope Analysis as the tool for handling the AFM images because this software proved to have the most versatile functionality and the best user experience flow with an intuitively understandable interface, according to Igor's conclusions and observations.

NanoScope Analysis software is developed by the company Bruker, which is specialized in microscopy and nanoanalysis, producing a broad range of both physical and digital products for this field [5]. The user interface is shown in Figure 1.

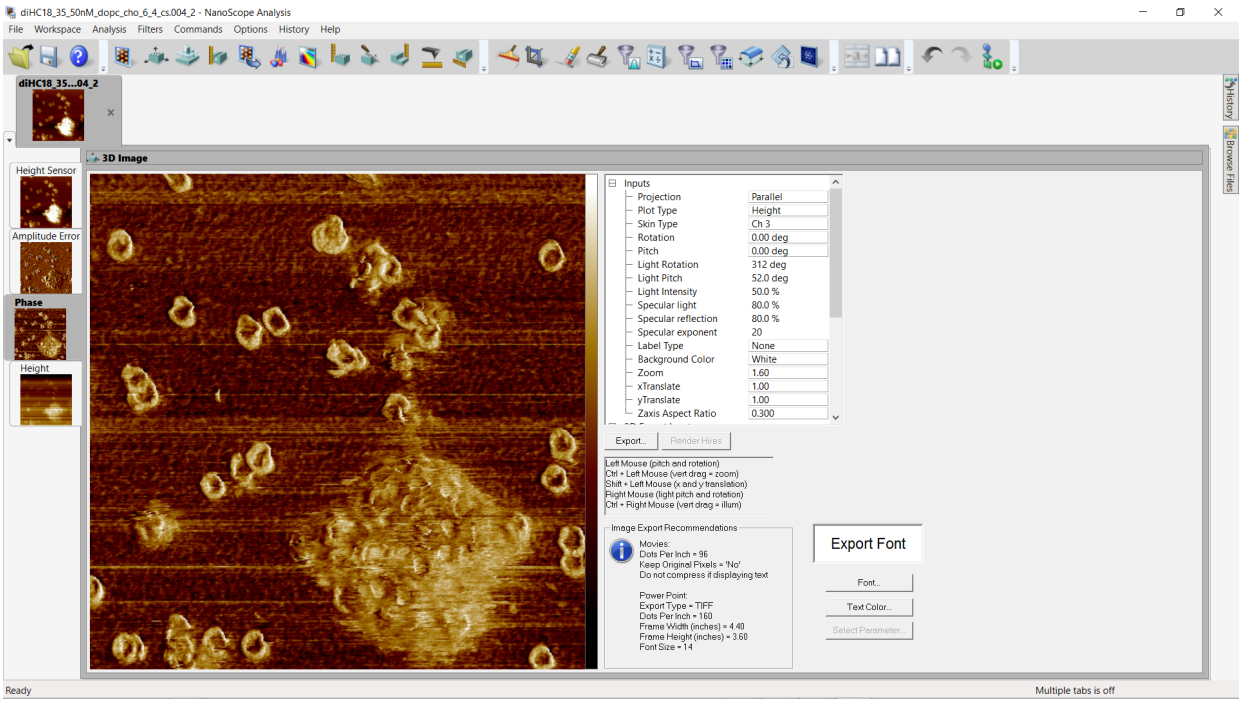
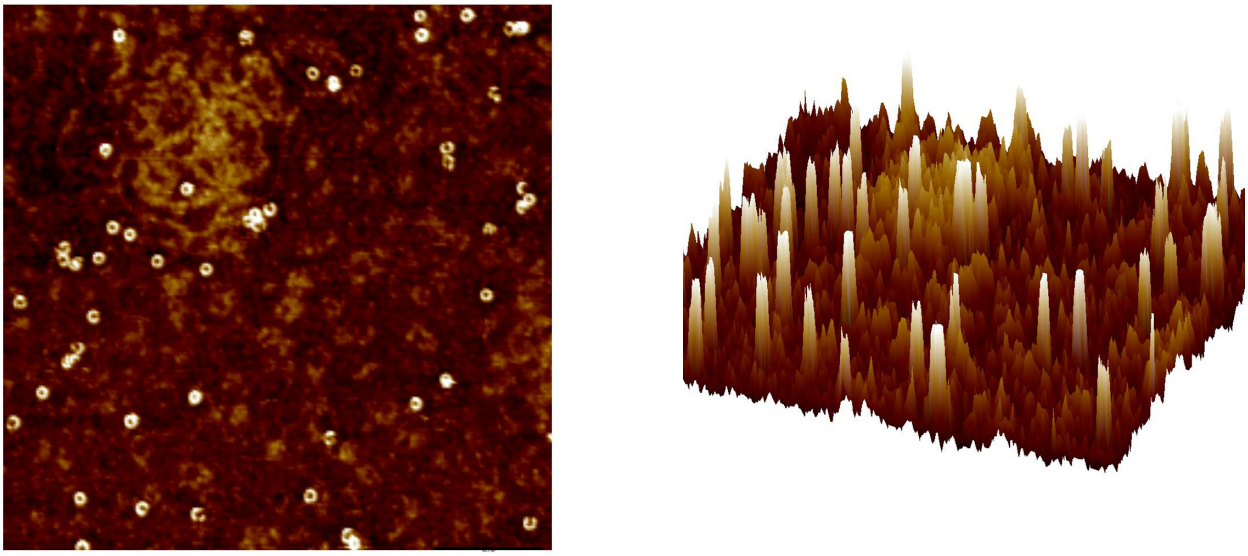


Figure 1. NanoScope Analysis user interface.

The possibilities of NanoScope Analysis are quite wide but in this project, the most important will be the next ones:

- Noise filtering.
- 3D viewing.
- Image processing filters such as Gaussian, low-pass, etc.
- Different channels viewing.

Figures 2a and 2b clearly demonstrate the difference between 2D and 3D perspective of the selected AFM image. Switching between these modes helps to understand the nature of the objects in the image and to see the results of applied manipulations.



(a) 2D (b) 3D

Figure 2. Different AFM image viewing modes.

More sophisticated usage of the software and image transformations such as flattening and different filters (Gaussian, median and others) which help to remove the noise from images will be described in more detail in a section 6, which is dedicated to the preparation of datasets.

3 Biological membranes, defects and atomic force microscopy

3.1 Biological membranes

All the living things around us are built up from the smallest structural units: biological cells. Some organisms consist of only one cell and are called unicellular, or single-celled, such as protozoans and most bacterial species. But multicellular ones hold the majority in nature and have a broad versatility among the species.

Scientists distinguish two types of membranes: animal and plant ones [37]. A brief comparison of their structures and differences can be seen in Figure 3. This project will concentrate on the biological cell membrane and its defects, in particular, so all the details about other cell components are omitted.

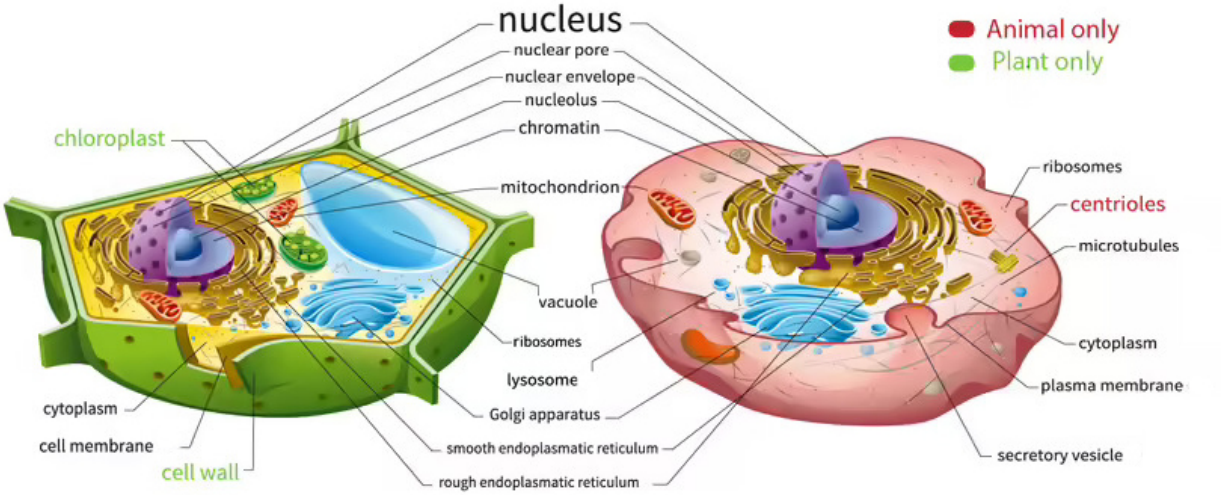


Figure 3. Animal vs plant cell structure [37].

Biological membranes, or as they are sometimes called cytoplasmic, are present in both types of cells and serve as a semi-permeable barrier between the outside environment and internal units such as ribosomes, lysosomes, etc. It is also responsible for the regulation of the material and interaction with surroundings by allowing a certain amount of nutrients to enter the cell and transport the waste products outside.

Scientists discovered the membrane in the 1890s years, and its components were identified later, in 1915. There were several attempts to generalize the structure and create a model of the plasma membrane, so the latest accepted model was proposed in 1972 by S.J. Singer and Garth L. Nicolson. This explanation was called the fluid mosaic model and evolved over time, but kept its initial idea: the components create some sort of a mosaic that determines a fluid character. Generally, the structure of the cytoplasmic membrane can be described as following [34]:

- Lipids (fats):
  - Phospholipids are the main fabric of the membrane. These molecules have hydrophilic and hydrophobic areas that allow them to form a bilayer membrane that separates fluid substances inside of the membrane itself from the internal cell matter. Therefore, the membrane is sometimes referred to as a phospholipid bilayer.

  - Cholesterol is found in the core of the membrane between phospholipids. The main function of this fat is to control the fluidity of the cell by mitigating the effects of temperature at the membrane.

- Proteins are a group of complex molecules that perform a number of crucial functions such as recognition of other cells and signal transduction. Moreover, they serve as a connection bridge between cells and are responsible for nutrient transportation. There present two types of membranes:

  - Integral proteins - are completely integrated into the membrane
  - Peripheral proteins - are proteins that can perform several roles depending on the specific situation. They are not permanently attached to the membrane, so can serve as enzymes, other protein activators, etc.

- Carbohydrates are the third major component. They are bound either to lipids, therefore constructing glycolipids or to proteins and form glycoproteins. These new molecules serve a role in cell-cell recognition by creating unique patterns on the membrane surface. Viruses tend to use this feature to hide from the immune cells.

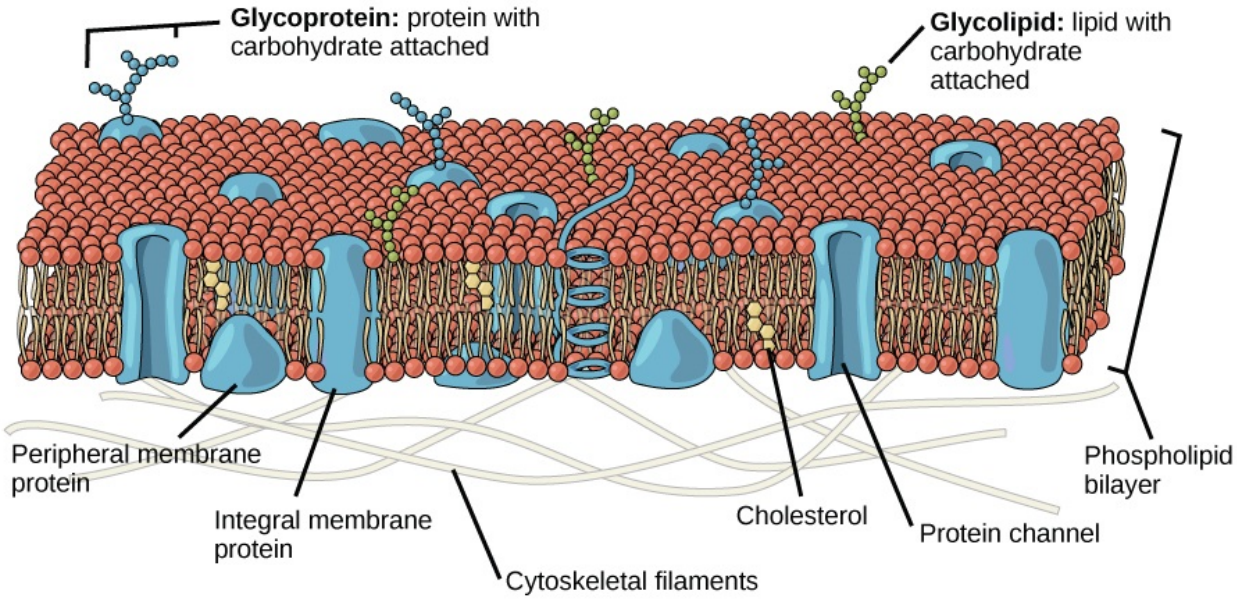
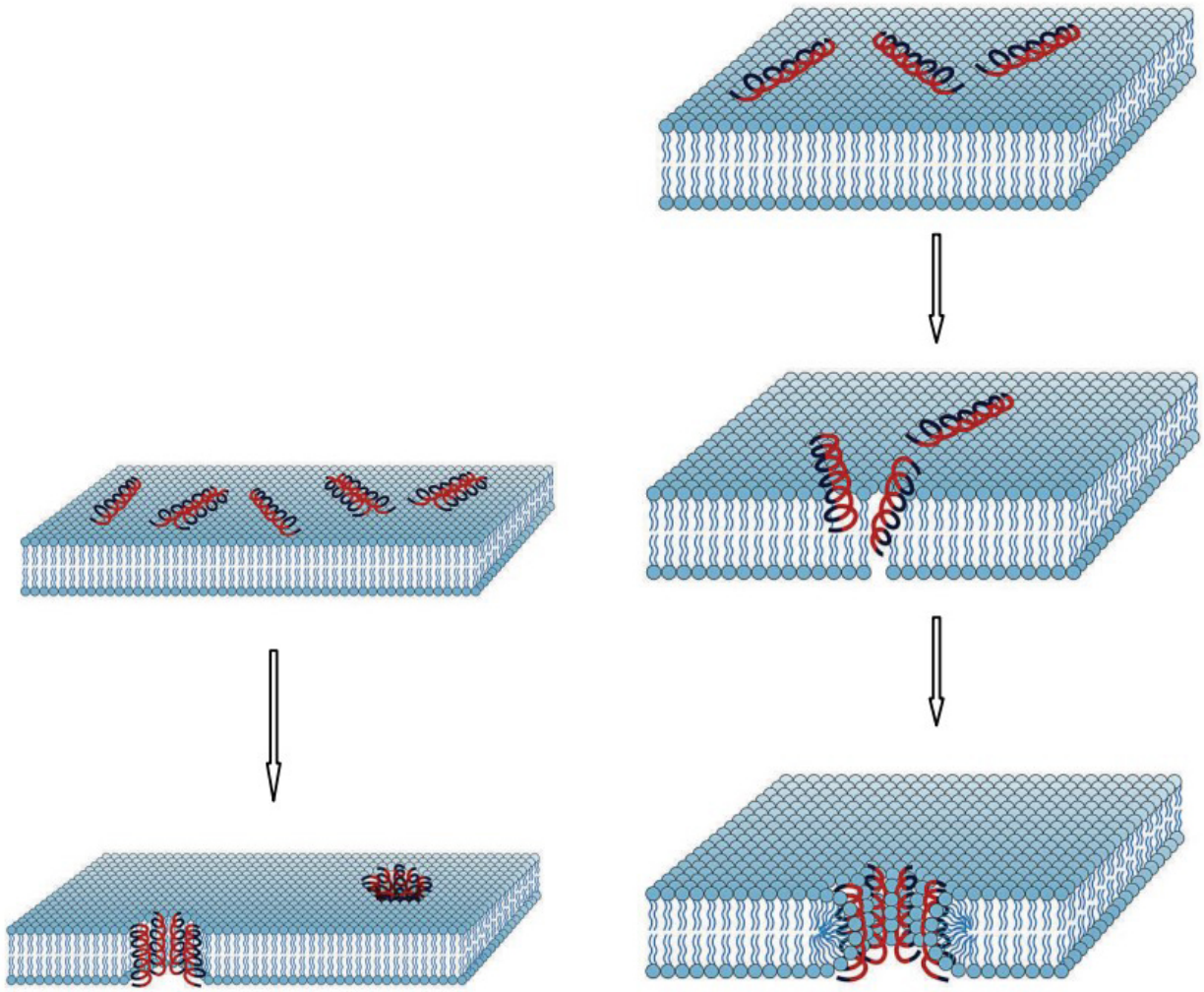


Figure 4. Cell membrane structure [34].

3.2 Defects in biological membranes

Throughout its existence, cells experience numerous interactions with other entities, and the cytoplasmic membrane is the first line of defense. When the phospholipid bilayer is exposed to some chemical reactions or mechanical impact - the membrane gets damaged. Usually, it is a normal process in the membrane lifecycle: phospholipids, due to their hydrophilic and hydrophobic zones tend to self-assembling and therefore fix these so-called defects. But when the amount of damage is too high - it can lead to irreversible consequences, such as the destruction of the whole membrane. As a result, toxins reach the nucleus and the cell dies.



(a) Barrel slave model

(b) Torroidal model

Figure 5. Models of peptide interaction with the cell membrane leading to defects [12].

Figures 5a and 5b illustrate two mechanisms of membrane pore formation, caused by foreign peptides. These properties are widely used in the antibiotics field. Antimicrobial peptides are considered to be a basis for the next-generation antibiotics due to holding great potential for combating bacterial resistance [21]. It is crucial to have technologies to detect these defects in membranes at the early stages because the consequences of the diseases causing such behavior can be fatal.

3.3 Atomic force microscopy

Microscopes allowed people to observe particles significantly smaller than the ones that can be seen with the naked eye. This technology has made a long-distance run from its initial invention in the XVII century to the devices present in modern laboratories nowadays. Here are some of the most outstanding discoveries:

- 1931 - transmission electron microscope.
- 1935 - scanning electron microscope.
- 1978 - confocal laser scanning microscope.
- 1980s - scanning probe microscope.

- 1986 - atomic force microscope.
- 2014 - stimulated emission depletion microscope.

This work concentrates on atomic force microscopy as a tool to investigate the cytoplasmic membrane. So it is preferable to know the main features of this technology and understand how this device works. As seen from the timeline above, AFM was invented relatively shortly after the scanning probe microscopy. It has a similar principle of operation, but without some significant drawbacks, such as imaging only conducting or semiconducting surfaces. Using this technology, scientists can obtain pictures with a horizontal resolution of 0.1 nm and a vertical resolution of up to 0.01 nm, which is much more precise than light and electron microscopes. [26, 10]

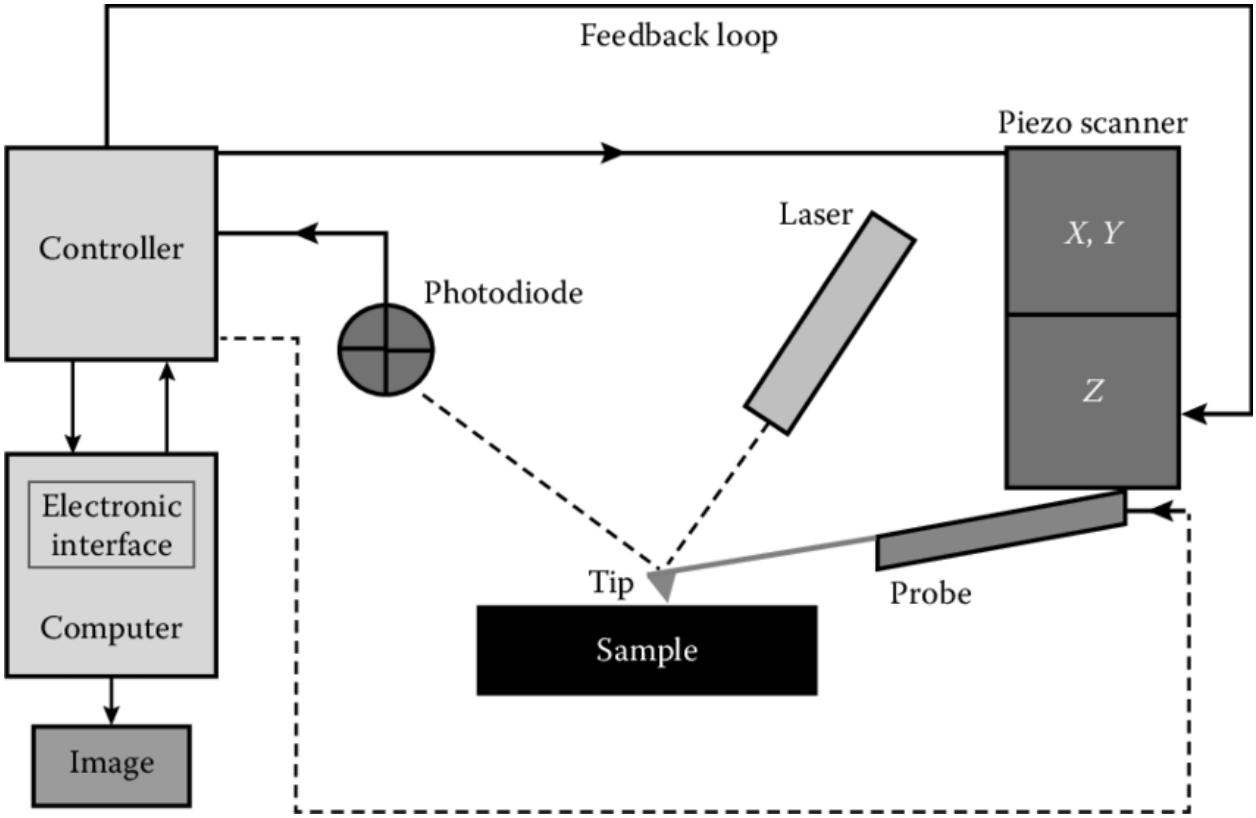
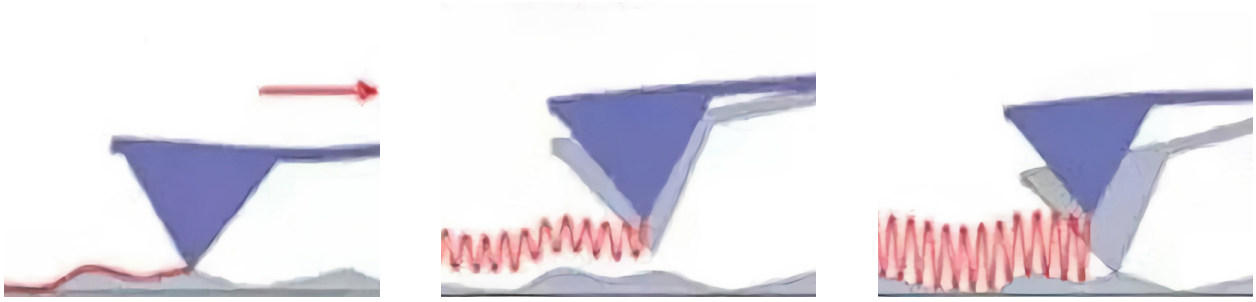


Figure 6. AFM operation [40].

Figure 6 illustrates the mechanism of an atomic force microscope. The sample is placed on the stage, under a sharp tip, which is usually manufactured from silicon or silicon nitride. During the scanning process, the laser falls on the back of the tip, which touches the sample and moves along its surface. Depending on the movement type, scientists distinguish three modes that will be described later. Then the laser gets reflected into the sensitive photodiode that transmits information to the controller. There exists a very weak force, which is either repulsive or attractive, between the tip of the probe and the atoms of the sample surface. When this force affects the tip — it changes the movement of the tip, the controller tracks these changes in the reflected beam and records them. A piezoelectric ceramic scanner controls the vertical and lateral positions of the AFM cantilever. A feedback loop, in its turn, controls the tip positioning, trying to maintain near-constant cantilever deflection. Then the computer generates an imaging map of the surface and produces an image. [26, 10].

Figures 7a, 7b, and 7c illustrate three different operational modes of an atomic force microscope. They are contact mode, non-contact mode, and tapping mode. Let's take a closer look at



(a) Contact mode (b) Non-contact mode (c) Tapping mode

Figure 7. AFM operational modes [1].

each of them [26, 10, 40, 1, 30]:

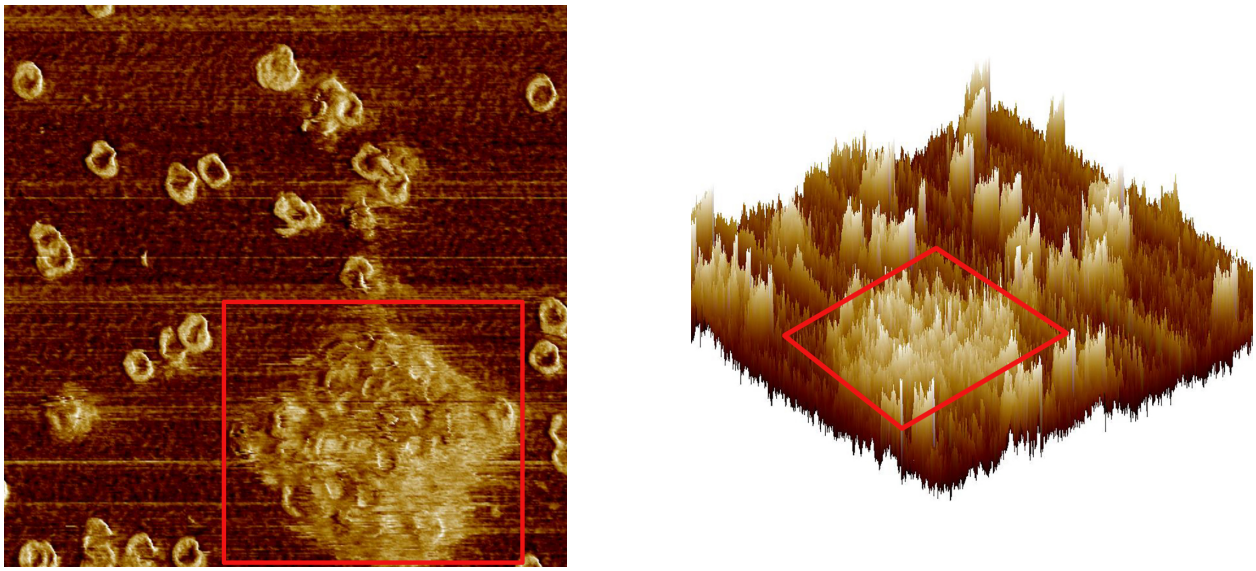
1. In contact mode the probe tip keeps constant contact with the surface while scanning it. In other words, the cantilever is being dragged along the sample. At the same time, the controlling system tries to maintain constant tip deflection, therefore interaction forces. The contact approach has several significant drawbacks:
   - One of the worst-case scenarios is sample damage. One should be aware of it when the number of samples is limited.
   - AFM tips are being worn faster compared to other modes. To overcome this downside, scientists started using lighter and softer cantilevers, but they are more susceptible to interaction forces.
   - The previous points lead to the degradation of the image quality.
2. Non-contact mode stands for its name: the cantilever is being kept at an extremely small distance (nm – several tens of nm) from the sample during the scanning process. In this mode, the image of the sample is created by measuring the attraction force between the tip and the sample. Here are the most noticeable advantages and some drawbacks:
   - Non-tapping mode results in extremely high-resolution images.
   - Less interaction between the tip and the sample, therefore no sample and tip damaging.
   - The ability to scan liquid substances because no contact is needed.
   - One of the disadvantages is that this mode requires more computing power to constantly keep the cantilever in the area of the attraction force.
3. Tapping mode tries to combine the benefits of the two methods mentioned above. It mitigates the drawbacks of the full contact mode by reducing contact with the surface. Also, this mode allows for to preservation of samples and brings improvement in the image quality of liquids. During scanning, the AFM probe is oscillated by a piezoelectric actuator. Usually, the frequency is between several tens and several hundred kHz. The main benefit of this approach is the reduction of lateral forces.

4 Clusterization and Voronoi diagrams

4.1 Defects clustering in AFM images

Previous chapters gave a brief overview of the defects' nature and the consequences they may lead to, but in real life, they oftentimes are found in groups, so-called clusters. Since the defects are holes in the cell membrane it means the clusters are more dangerous than single ones. That is because the damage to the cell membrane is more serious. During the scientific research work, it was discovered that a model which was trained on single defects is not capable of precisely detecting them inside of clusters. In the next chapters, the author will make several assumptions about the reasons why it is happening and what are the possible solutions.

Meanwhile, as discussed by Tomas Raila in his research, the density of the defects is strongly correlated with the electrochemical impedance spectroscopy response of tethered bilayer membranes. Also, Tomas has pointed out that the AI approach to AFM image analysis showed better results compared to Topostats. According to his observations, clusters are the stumbling block for non-AI methods. [33, 32]. That means finding a technique to precisely estimate clusterization will allow us to predict EIS spectra and explain the nature of the damage.



(a) 2D (b) 3D

Figure 8. Numerous defects gathered in one cluster.

Figures 8a and 8b are the same image shown from different perspectives. There are several defect aggregations on the image but the biggest cluster is pointed out with a red square. From the 3D perspective, it is clearly visible that defects protrude from the surface forming rings that look like volcanos with empty space inside.

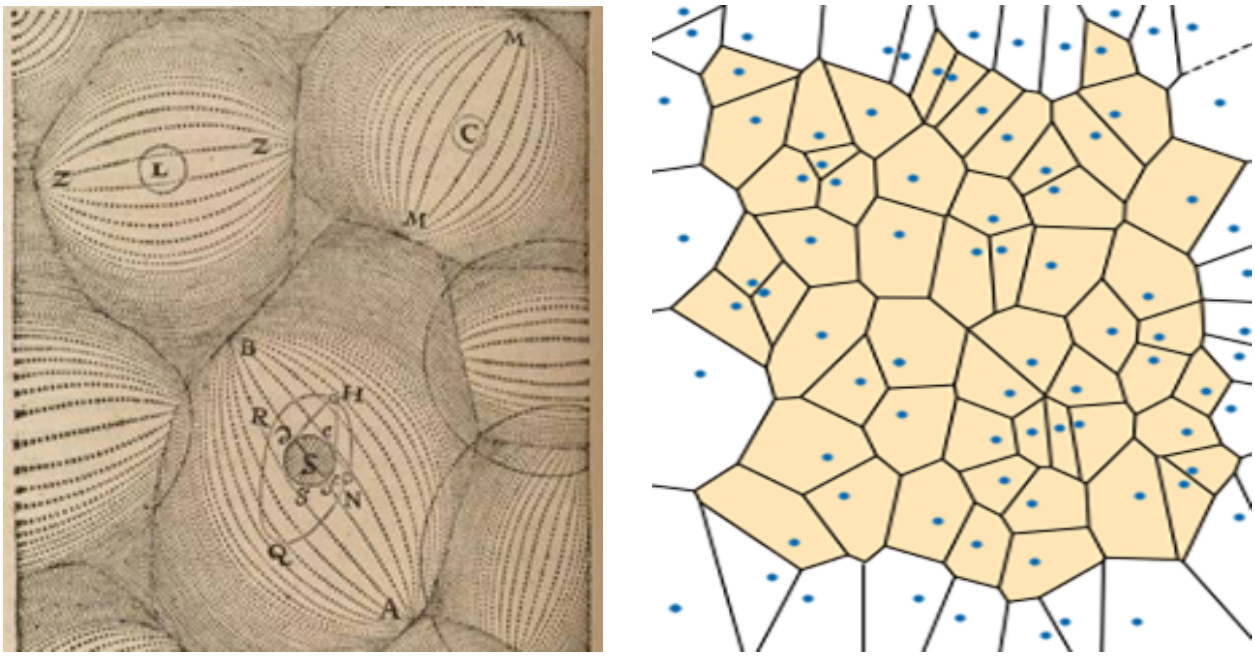
4.2 Voronoi diagrams

4.2.1 History and concept

Therefore arises a task - how to split the detected defects into clusters and estimate the clusterization. One of the approaches was introduced by Vita Mačinskaitė, a former Vilnius University

student. In her master's thesis, she used Voronoi diagrams as a tool to divide the space into regions and applied entropy with standard deviation as two main techniques to measure the level of an AFM image clusterization [23].

In the XVII century René Descartes attempted to explain some astronomical questions by introducing a theory of splitting space into regions (vortices) with a single star in the center, as shown in Figure 9a. This approach gained popularity among scientists and was used to explain different physical forces, for example, magnetism [22].



(a) Rene Descartes' vortices

(b) Voronoi diagram generated from an AFM image

Figure 9. Comparison of René Descartes' theoretical approach and developed Voronoi diagram (blue dots - objects of interest, orange polygons - inner ones).

The method of Voronoi diagrams (also known as Thiessen polygons or Dirichlet tesselation) was more formally developed from the research of Descartes only in the XIX century by scientists Dirichlet and Voronoi. The latter brought a great contribution and extended the studies to higher dimensions. The main idea is to break down the given space into polygons that contain only one object inside of itself. The regions are organized in such a way that any point of polygon A is closer to the object than any point from polygon B. Simply put, the Voronoi diagram allows us to find the closest vertices for any point in the space. In cases when some particular objects are more significant than others, a weighted Voronoi diagram is used.

One of the notable uses of a similar approach happened in 1854 during the cholera epidemic in London. The physician John Snow split the town into regions to correlate the number of deaths in a district to the closest pump.

4.2.2 Mathematical definition

For 2-dimensional Euclidean space $\mathbb{R}^2$, let P is the finite set of $n \geq 2$ points on the plane. $P = p_1, p_2, p_3, ..., p_n$. Then Voronoi polygon will be defined as following [19]:

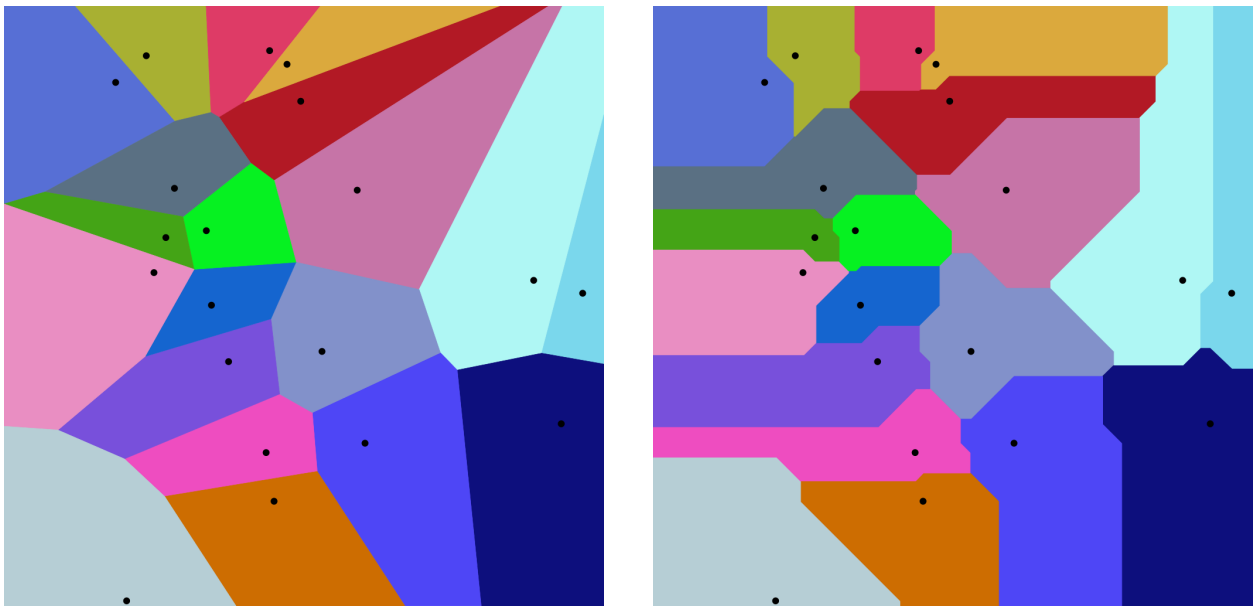
$$V(p_i) = r \in \mathbb{R}^2 : \|r - p_i\| \leq \|r - p_j\|, i \neq j, j = 1...n \tag{4.1}$$

where $d(p, x) = \|p, x\|$ is on of the next distance metrics:

- Euclidean: $\|p, x\| = \sqrt{(p_1 - x_1)^2 + (p_2 - x_2)^2}$, Figure 10a.
- Manhattan: $\|p, x\| = |p_1 - x_1| + |p_2 - x_2|$, Figure 10b.
- Chebyshev: $\|p, x\| = max((p_1 - x_1)^2 + (p_2 - x_2)^2)$.

In this paper for calculations will be used only Euclidean distance.
And Voronoi diagram for the finite set of points P will be defined as follows:

$$V(P) = \cup V(p_i), where\ p_i \in P \tag{4.2}$$



(a) Euclidean distance (b) Manhattan distance

Figure 10. Comparison of the Voronoi diagrams built using Euclidean (a) and Manhattan (b) distances [44, 45].

4.3 Clusterization metrics

To properly estimate clusterization, the square of each and every Voronoi polygon has to be normalized by multiplying the area of a polygon by a ratio of the number of polygons to the total area: $S_{norm} = S_P * \frac{N}{S_V}$, where S_P is the area of a polygon and S_V is the area of whole Voronoi diagram. This ensures that the later calculations will not depend on the number of defects. In this paper will be considered two metrics: entropy and standard deviation. These metrics will be used as additional measures to estimate the precision of the applied algorithms.

4.3.1 Voronoi entropy

When Voronoi diagrams started to get acknowledgment in the scientific world, there has arisen a task of calculating the similarity among different charts. Another problem was how to quantify the orderliness of the pattern. To solve this question was defined the Voronoi Entropy [3, 4]:

$$S_{vor} = -\sum_{i=1}^{n} P_i * \ln P_i, \tag{4.3}$$

where P_i is the fraction of polygons with i sides (so-called coordination number of the polygon), and n is the biggest number of edges that a polygon has in terms of the current Voronoi diagram.

As was mentioned above, this metric is used to estimate the orderliness, so next can be stated:

- When the diagram is generated as a fully random pattern, studies proved the value to be 1.71 [3, 4].
- In cases when the diagram is perfectly ordered (consists of polygons with the same amount of sides) - the entropy equals 0.
- The Voronoi entropy is an intensive metric, so it does not depend on the number of polygons in the chart.

Besides that, there were discussions on whether the Voronoi entropy is a true entropy and separate metric or just a special case of another measure. Some articles state that actually it represents the averaged Shannon measure of ordering for 2D patterns, but in terms of this paper this question will not be considered any further [2].

4.3.2 Standard deviation

The standard deviation (σ) is a common statistics metric that has a vast field of application in different fields. It shows how dispersed the set of values is: a high deviation is an indicator of objects being spread over a wide range, whereas a low value shows that the objects are clustered near the mean.

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}, \tag{4.4}$$

where x_i is the area of a polygon, $\overline{x}$ is the mean area of all polygons and N is the number of polygons on the diagram.

So this metric can be used to investigate how far the defects are dispersed from each other. For example, Tomas Raila et al. used standard deviation to describe the clusterization of defects in biological membranes [32]. In their studies, all the Voronoi regions were used for calculations, both complete (inner) and incomplete (outer). In this master's thesis, the standard deviation for Voronoi diagrams will be referred as Voronoi σ and for its calculation will be used only inner polygons, if it is not mentioned explicitly.

4.4 Implementation and experiments

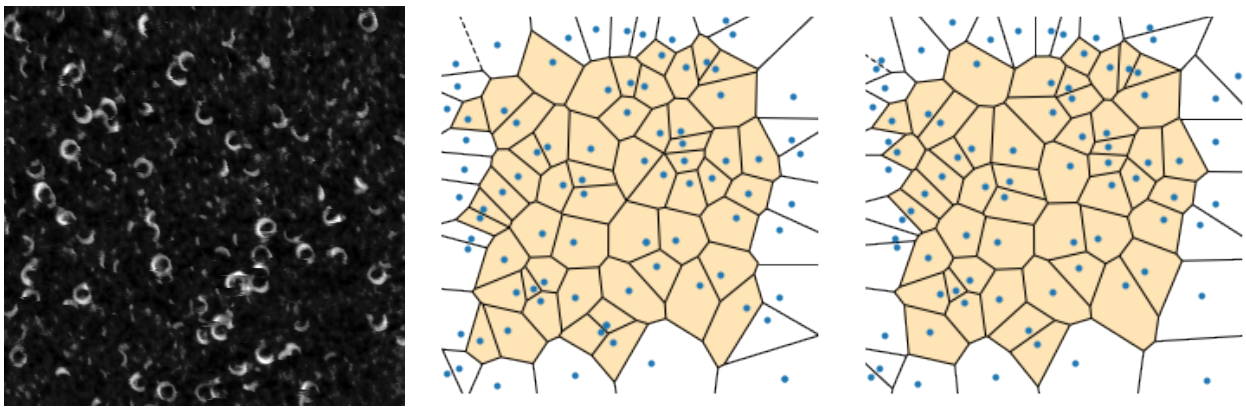
After some research, there was found no universal tool to build inner polygons of the diagram and automatically calculate the area. Even though Vita has used and modified some open-source solutions, but they turned out to be incompatible with the chosen environment, so it was decided to program new functions. As a basis for the custom Voronoi diagram representation and future metrics calculations was chosen *SciPy* library for *Python* language. From the package *spatial* were retrieved and used next items:

- *Voronoi* - a class that accepts coordinates of points and returns an object with properties of the resulting Voronoi diagram.

- *voronoi_plot_2d* - a function that accepts Voronoi object and build 2d plot.

Also, to calculate areas of the Voronoi polygons and re-build the diagram with only inner polygons, was applied a package named *Shapely*. Generally, this package is used for spatial analysis, manipulations of planar features and other computations in fields such as epidemiology, agriculture and others. So, the logic of the custom-written code for calculating entropy and Voronoi σ for given defects coordinates is following:

1. If the coordinates of the defects were not normalized before, divide each coordinate by image width to bring the values to a range from 0 to 1.
2. Feed normalized coordinates of the defects centers to *Voronoi* class (one should pay attention that SciPy accepts coordinates of points, not min/max values of the bounding box) and get an object containing all the characteristics of the resulting diagram.
3. Create a set of vertices that are considered outsiders. That means have any of *x, y* coordinates larger than 1 or smaller than 0.
4. Draw a plot with inner polygons, meaning the ones that do not contain vertices-outsiders derived in the previous step,
5. Calculate areas using *Shapely* and normalize them.
6. Calculate Voronoi entropy and σ.



(a) Original AFM image (b) Original coordinates (c) Self-mapped coordinates

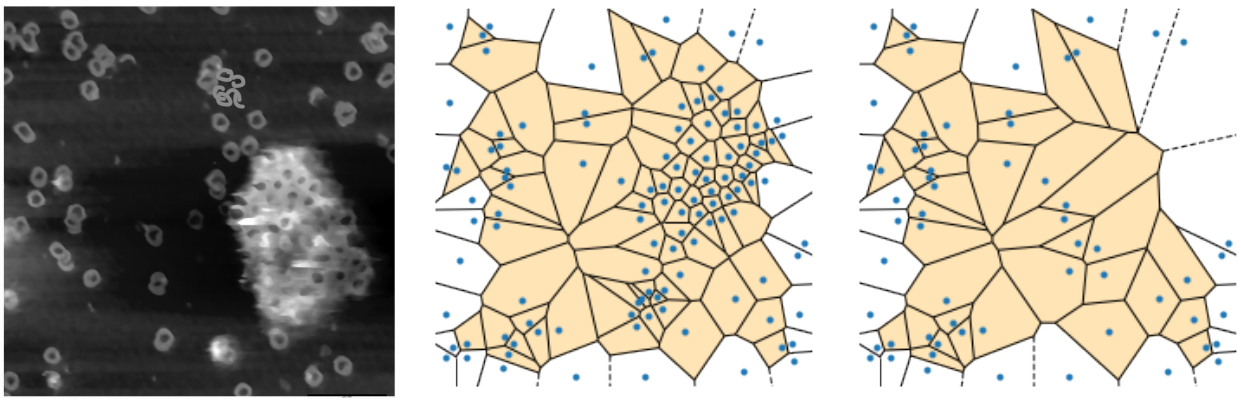
Figure 11. Comparison of Voronoi diagrams generated using the coordinates provided by VU Life Sciences Center, and the self-mapped coordinates.

One of the first conducted experiments was the comparison of Voronoi entropy and Voronoi σ between defect coordinates provided by Vilnius University Life Sciences Center and the ones, which were mapped by the author of this paper. The main goal of this experiment was to estimate the approximate precision of self-mapping because the major part of the images did not have cluster coordinates, so later on there may be a need to manually map defects during dataset preparation.

As shown in Figures 11b and 11c, even though the blue points (protein defects coordinates) are not in the exact same positions, the two Voronoi diagrams look almost identical and besides that demonstrated similar characteristics which are presented in the table 1. Turned out that self-mapped coordinates lack 4 defects that were missed during the labeling process. Also, based on the values derived from the table 1, the entropy differs by 0.14 units.

Metrics	Provided by VU	Self-mapped
№ of inner polygons	55	51
Voronoi entropy	1.505	1.643
Voronoi σ	0.422	0.463

Table 1. Voronoi diagram metrics built with self-mapped and provided by VU coordinates.



(a) Original AFM image (b) Single defects and clusters (c) Only single defects

Figure 12. Illustration of changes in the Voronoi diagram after removing the coordinates of clusters.

The goal of the next experiment was to check how much clusterization affects the metrics of the Voronoi diagram. For that purpose, there was selected an AFM image containing at least one big cluster of protein defects as it is shown in Figure 12a. Then the coordinates were modified by removing the defects inside clusters from the mapping. That is, two separate files were fed into the program: one contained the coordinates of all the present objects, whereas in the second one were present only single defects.

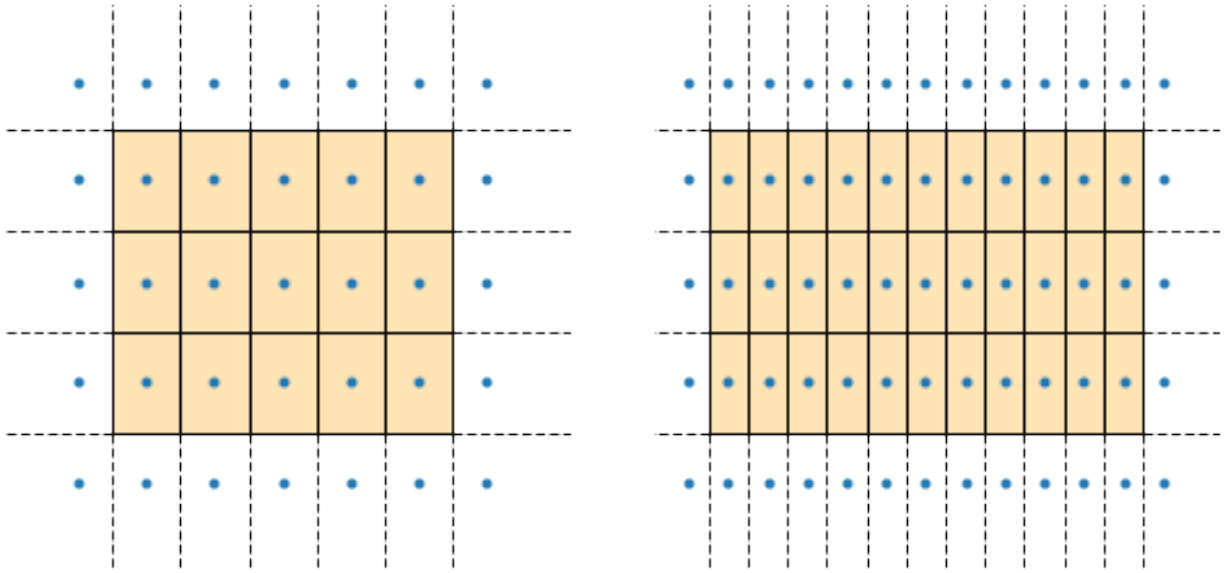
The results are listed in Table 2 and can seem unexpected at the first sight. After removing more than 50 polygons from the diagram, the entropy has almost not changed. But as was stated above in the section 4.3.1, Voronoi entropy is an intensive measure, so the amount of polygons does not affect the value. The entropy remained similar to the original one because it takes into account the fraction of polygons with the same amount of sides.

Metrics	All defects	Single defects
№ of inner polygons	88	37
Voronoi entropy	1.628	1.674
Voronoi σ	0.899	0.658

Table 2. Metrics of Voronoi diagrams generated using coordinates of all defects and only single ones.

At the same time, the Voronoi σ value has decreased in 1.3 times, from 0.899 to 0.658. It was discovered by T. Raila et al. that $\sigma \approx 0.54$ for random systems, and becomes greater than 0.54 for sparsely clustered patterns [32]. Therefore when more than 50 defects were removed from the Voronoi diagram, the value of Voronoi σ decreased as well, signifying that clusterization has declined.

To double-check the correctness of the self-implemented Voronoi diagramming methods, it was decided to conduct several more experiments. For example, as was mentioned above, for the systems that are close to perfect orderliness and mostly consist of a single type of polygons, Voronoi entropy should be close to 0. That is because the fraction of polygons with the same amount of vertices tends to be 1 and therefore the natural logarithm will be 0. To test this statement, there were manually created 2 perfectly ordered systems, where objects are at equal distances from each other. Since there were no AFM images where the defects were distributed uniformly, it was decided to generate artificial data. Voronoi diagramming methods were applied, and the results are presented below, in Figures 13a and 13b.



(a) System a with 35 objects

(b) System b with 65 objects

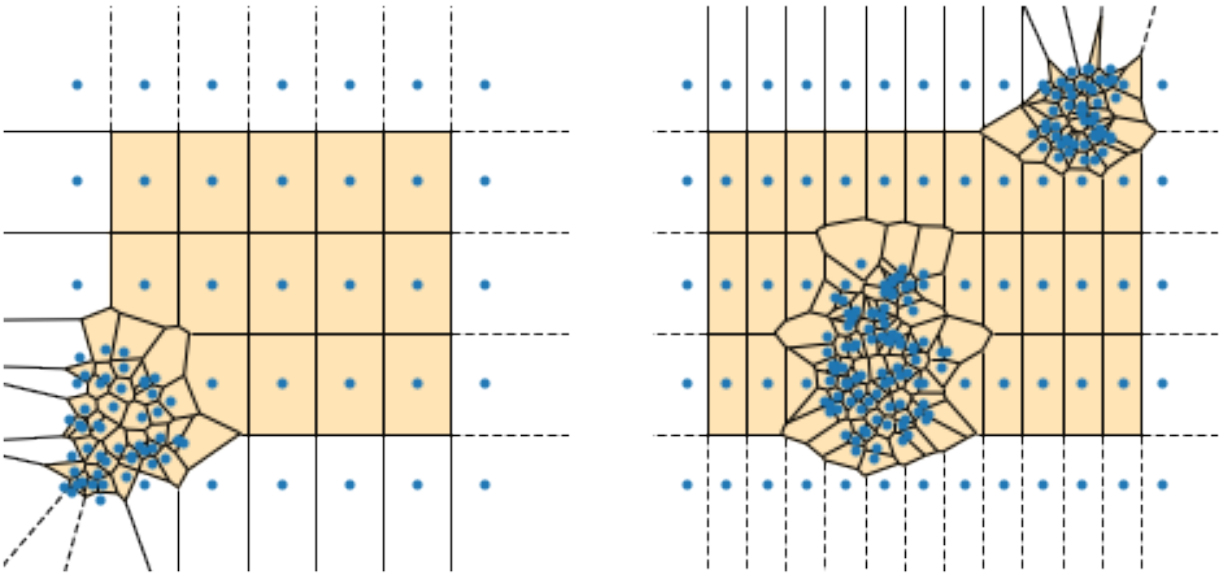
Figure 13. Voronoi diagrams of perfectly ordered systems.

As can be seen from Figures 13a and 13b, in both cases the grids consist of polygons with the same amount of vertices (4 in this particular example). Table 3 contains the numeric results of the experiment. As it was expected, the Voronoi entropy equals zero in both scenarios because the systems are perfectly ordered. The Voronoi σ is extremely close to zero, which indicates that the clusterization of the images is low. Indeed, the objects are evenly distributed over the image without any groupings.

Metrics	System a	System b
№ of inner polygons	15	33
Voronoi entropy	0.0	0.0
Voronoi σ	0.0096	0.00963

Table 3. Metrics of Voronoi diagrams generated for artificial perfectly ordered systems.

One more experiment was conducted to demonstrate the correlation between the Voronoi σ and aggregations of clusters. For this purpose were taken two systems from the previous experiment, but slightly modified. The first image was populated with one cluster, whereas the second one had two accumulations added. The resulting Voronoi diagrams are presented in Figures 14a and 14b.



(a) System a (b) System b

Figure 14. Voronoi diagrams of artificially generated systems with clusters.

Figures above clearly display that homogeneity of the polygons is disturbed and it will definitely affect the Voronoi entropy. Also, a hypothesis can be made that the Voronoi σ value will increase compared to the previous experiment, and it will be higher for the diagram presented in Figure 14b.

Metrics	System a	System b
№ of inner polygons	56	171
Voronoi entropy	1.579	1.592
Voronoi σ	1.3366	1.4872

Table 4. Metrics of Voronoi diagrams generated for artificial systems with clusters.

The values in Table 4 prove the assumptions made above: the entropy significantly increased. At the same time entropy tends to the values which were derived from the real data (around 1.63 as shown in table 2), but not reaches them because they are close to the limit of randomness (1.71), whereas current diagrams still have a significant fraction of homogenous polygons. As for the Voronoi σ, it has considerably changed as well compared to the homogenous systems. In this experiment, the metric is higher for system b, which is completely logical because there are two clusters on the diagram.

The results from the experiments listed above can lead to the next conclusion: the very exact precision of the coordinates of the defects does not matter to some extent because the clusterization metrics are not affected much by the positioning, but by the homogeneity and the even distribution of the objects on the Voronoi diagram. Therefore the following noteworthy things can be pointed out:

- If there will be not enough images labeled by professional biologists, it is possible to manually label as many AFM images as needed to prepare the datasets, and what is important is that the produced output should be trustworthy.

- The custom functions for calculating Voronoi diagram metrics operate properly.
- The percentage of the detected defects makes a big difference to the results.
- Voronoi σ does reflect the clusterization of the defects.
- From the previous statement derives a need to be able to detect defects inside of clusters so a proper Voronoi diagram could be generated.

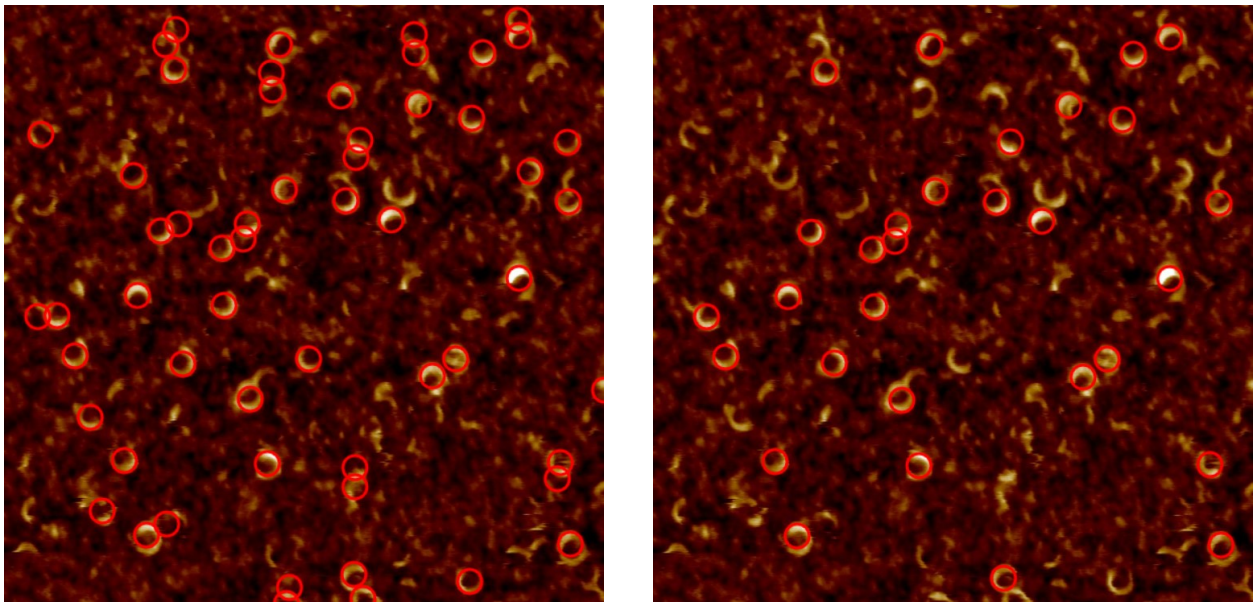
5 Object detection

5.1 Feature extraction techniques

The papers from the last years clearly stated that the application of simple feature extraction techniques alone is not sufficient for solving this problem [42]. For example, the Hough transform is commonly used for identifying the positions of ellipses or circles in digital images.

The author of the last year's master thesis tried to adopt the Hough method and achieved around 65% of the accuracy in the defects detection [42]. As described by the author himself, this result is fair only for "rare case" images, where the defects are similar to each other by geometric shape and size (circle radius). Also, for successful application of the Hough transform, there should be as less clusters as possible, and the defects should have full circle shapes. If the objects differ too much, the selection of parameters becomes a big challenge.

Moreover, the author has described that it is literally impossible to have pre-configured values fitting any random image. To solve this problem on the go, it was suggested to use the grid search method. Another issue is the intersection of detections: sometimes the Hough method produces double or triple circles where only one defect should be present. This task is too complex for feature extraction methods due to the versatility of the possible data. In addition, the Hough Circle Transformation method doesn't allow to classify the detected objects into types.



(a) Low threshold value

(b) High threshold value

Figure 15. Illustration of Hough Circle transform results obtained with an open-source software.

On Figures 15a and 15b one can observe the issues arising during Hough Circle Transformation usage. Even with a low threshold value that allows passing more doubtful results, lots of defects are not detected either because their shape is not similar to a circle according to the Hough method or the contour is not clear enough. When the threshold parameter is increased as in Figure 15b, this issue becomes even more noticeable with a huge number of defects being skipped. These images are obtained using software available for free [13] and presented only as an illustration of the Hough method's drawbacks showing why it is not suitable for this task.

So, in the work it was highly recommended to use neural networks for defect recognition because those models are self-taught and demonstrated much better accuracy.

5.2 Neural networks

Biological neural networks are combinations of neurons that communicate with each other by nerve impulses. Each neuron receives information from others and depending on these signals it decides where to transmit the impulse to other neurons or stay silent. When the aggregated input (excitation and inhibition signals) reaches some threshold - the neuron fires up and passes its impulse to others. [41]

5.2.1 Perceptron

Perceptron is an algorithm presented as a mathematical model in 1943 by McCulloch and Pitts and later on implemented by F. Rosenblatt in 1958. [24]. The main idea of this method is to mimic the behavior of a biological neuron. It is presented in Figure 16 and consists of the next steps:

1. the perceptron receives data from other perceptron or directly from a source;
2. inputs are multiplied by weights (coefficients);
3. multiplication results are summed up;
4. activation function decides whether the output is 0 or 1;

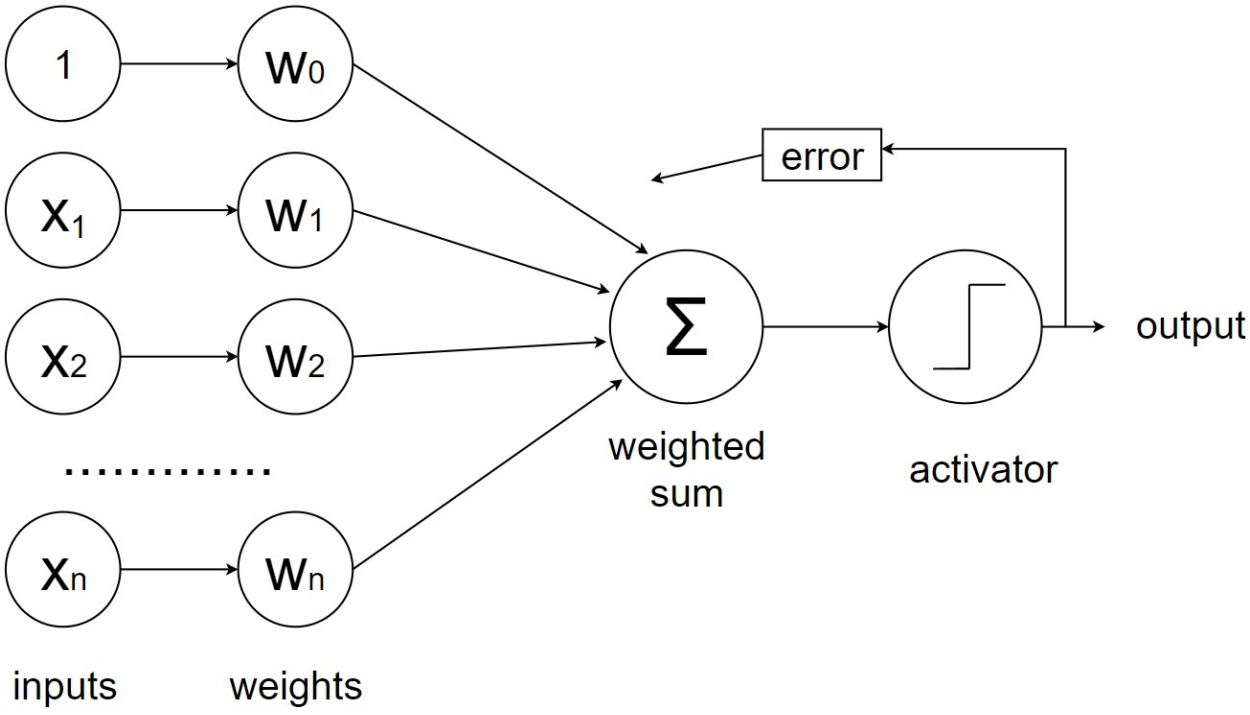


Figure 16. Schema of a perceptron.

The first input is always 1, and w_0 weight is called bias, so the first input is always 1. Perceptron is also referred to as a "binary classifier" because the output is either 0 or 1 (true or false). One can notice that the perceptron is similar to linear regression. Indeed, the core function for both methods is the same: $y = w_0 + \sum_{n=1}^{N} x_n * w_n$. One of the differences is that regression produces real numbers, whereas the output of perceptron is binary.

The final component is the activator, it determines the output of the whole perceptron. There are different types of activation functions such as step, sign, sigmoid, ReLu, and others. Sigmoid is one of the most used because this function brings input to a range from 0 to 1:

$$\sigma(x) = \frac{1}{1+e^{-z}}, where\ z = w_0 + \sum_{j=1}^{d} w_j x_j \tag{5.1}$$

5.2.2 Artificial neural networks

Artificial neural network (ANN) is a collection of connected artificial neurons (nodes), perceptrons for instance. The simplest neural networks (models) are single-layered, but their efficiency is limited. So more advanced networks are organized into a multiple-layer structure. The first layer accepts external raw data and is called the input layer. The last layer is the output layer because it produces the final result of the network. All the other layers are called hidden, they receive data from the previous layer and the output is the input for the next one.

Most of the ANN has only forward flow, which means the data moves only in one direction: from the input layer to the external. These networks are also referred to as feedforward ANN. Another approach is called backpropagation - it is the movement from output to input, which allows calculating the error of each node and adjusting the weights accordingly [9, 15].

5.2.3 Convolutional neural networks

One of the ANN classes is convolution neural networks (CNN), which are widely applied in digital image analysis and deep learning. CNNs are particularly effective at processing data that has a grid-like structure, such as an image, because they are designed to take advantage of the spatial structure of the data.

Typically, CNN models are organized in a structure with three main components:

- Convolution layers or feature extractor: at this step CNNs apply a set of filters (also called kernels or weights) to the input data and produce a set of output maps (also called feature maps). These filters are small, usually 3x3 or 5x5, and are designed to detect specific patterns or features in the input data. For example, a filter might be designed to detect edges or corners in an image. So the purpose of this part is to bring the input into a format that is understandable for the model.

- Pooling layers or downsampling - at this stage the model reduces the dimensionality of the features. Data is aggregated with max or average pooling. Even though a lot of important information is lost at this step, the pooling layer reduces the complexity, thus making the model faster.

- Fully connected (FC) layers are responsible for performing the tasks of classification and regression based on the features, extracted at the previous steps.

In a similar work from the last year, there was made an overview of different CNN models and the preference was given to models from the YOLO family [42]. There was released a new version of the model, following the YOLO operating principle, therefore it was decided to check the difference in the performance compared to the previous version.

5.3 YOLOv5

YOLO abbreviation stands for 'You Only Look Once', and the main principle of the algorithm is to divide the input image into a grid, where each cell is responsible for detecting objects within itself. One of the features of YOLO-family models is that they require only a single forward propagation during image recognition. Also, it is the first object detector that connected the prediction of class labels with bounding boxes in an end-to-end differentiable network. The architecture of the image is presented in Figure 17

Recently, there was done an independent comparison of the YOLOv3, YOLOv4, and YOLOv5 models. The results state that YOLOv5 slightly outperforms its predecessors in precision and recall metrics. YOLOv5 is based on the PyTorch framework, whereas both YOLOv3 and YOLOv4 are developed in the Darknet framework, written in C programming language [28].

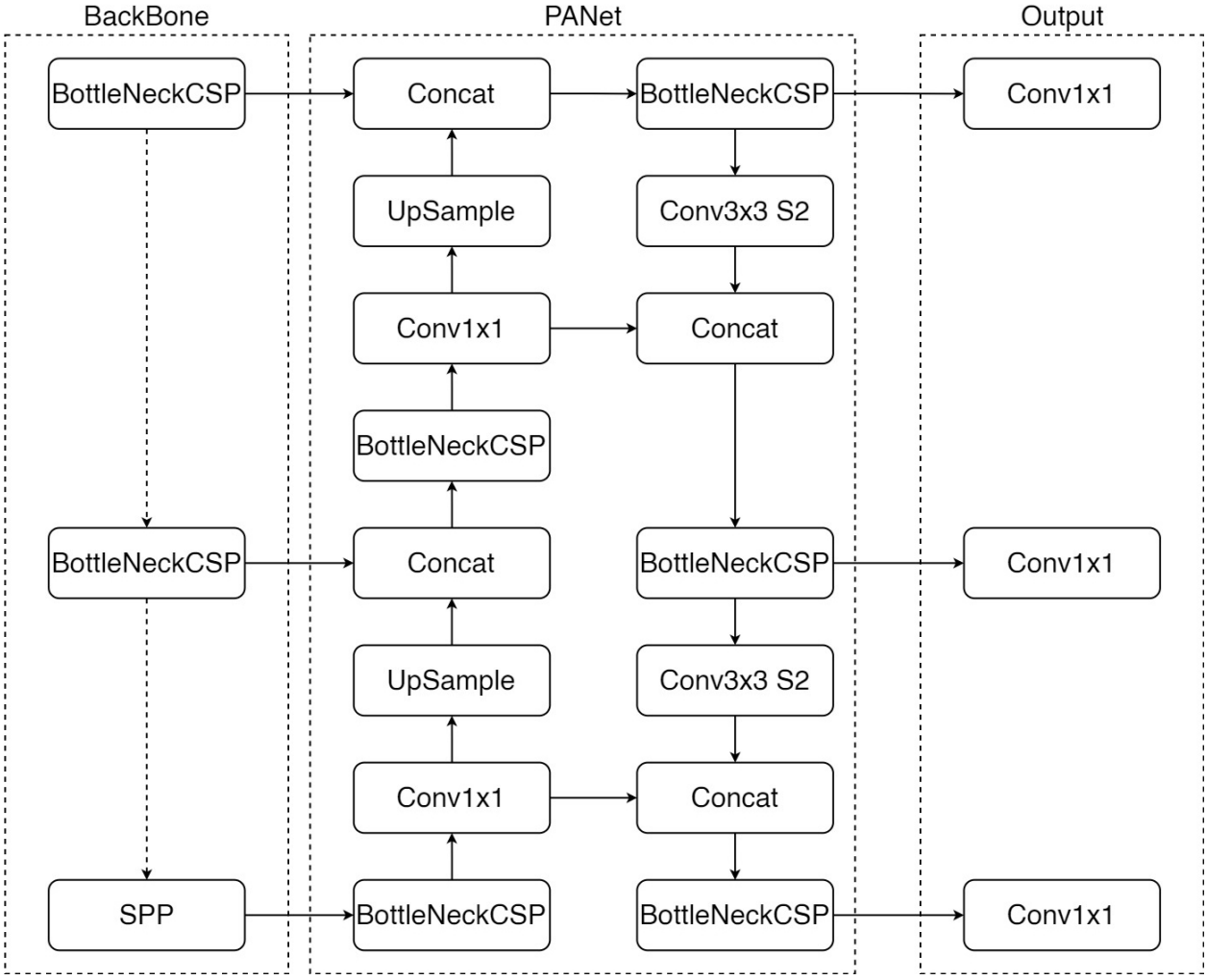


Figure 17. YOLOv5 architecture [[17]].

The classification process can be generalized as follows:

- Divide the image into a grid.
- Generate a class probability map.
- Generate bounding boxes.
- Apply non-max suppression to remove excess bounding boxes.

In the scientific research project the author used a pre-trained model YOLOv5s, which is the second smallest and mostly used on mobile devices as the middle ground between speed and size. Since the main emphasis of this thesis is on the detection of defects inside clusters, which is a more complicated task compared to the SRP, it was decided to use a more complex variant, medium-sized YOLOv5m with greater depth and width. So, in all the following experiments will be used YOLOv5m if it is not stated otherwise. Worth mentioning, that YOLOv5 follows the EfficientDet compound scaling model, so all the YOLOv5 models have the same structure, but use different compound scaling constants for scaling. [16]

6 Dataset preparation

This section describes the process of the dataset preparation for the YOLOv5 model training. Provided information about digital image processing, annotation formats conversion, and splitting the dataset into training and validation parts.

6.1 Research data

Life Sciences Center at Vilnius University has collected and provided a dataset containing more than 100 AFM images. Besides that for some samples, there was information about clusters or results of experiments that were previously conducted in the laboratories.

The resolution of the images is either 1024x1024 or 512x512 pixels, whereas the actual size of the displayed samples is measured in micrometers and is either 1µm or 2µm. The most important for the current research are raw files that can be opened with NanoScope Analysis, as it was described above in the chapter 2. The major part of the images are already converted to one of the widely used formats (TIFF, JPG) and presented in several different channels [6]:

- Height - the Z piezo voltage set by the feedback calculation in the Digital Signal Processor (DSP) is displayed.
- Height sensor - displays the signal from the Z position sensor in the closed loop head. Standard imaging of topography.
- Amplitude - the RMS of the cantilever amplitude signal is displayed and captured.
- Phase - phase data from oscillating TappingMode tips are displayed and captured.

The issue with the AFM images which are already processed and converted by the Life Sciences Center is that they are not properly organized: some files are transformed to the height channel, others to phase and third are represented in the amplitude one. It was not an issue in the scientific research work, because those datasets were relatively small and included data prepared by other students for their research, Igor Vilghelm in particular [42]. But in terms of this project, such inconsistency can make it impossible to prepare one big dataset, which will be required for more thorough CNN model training.

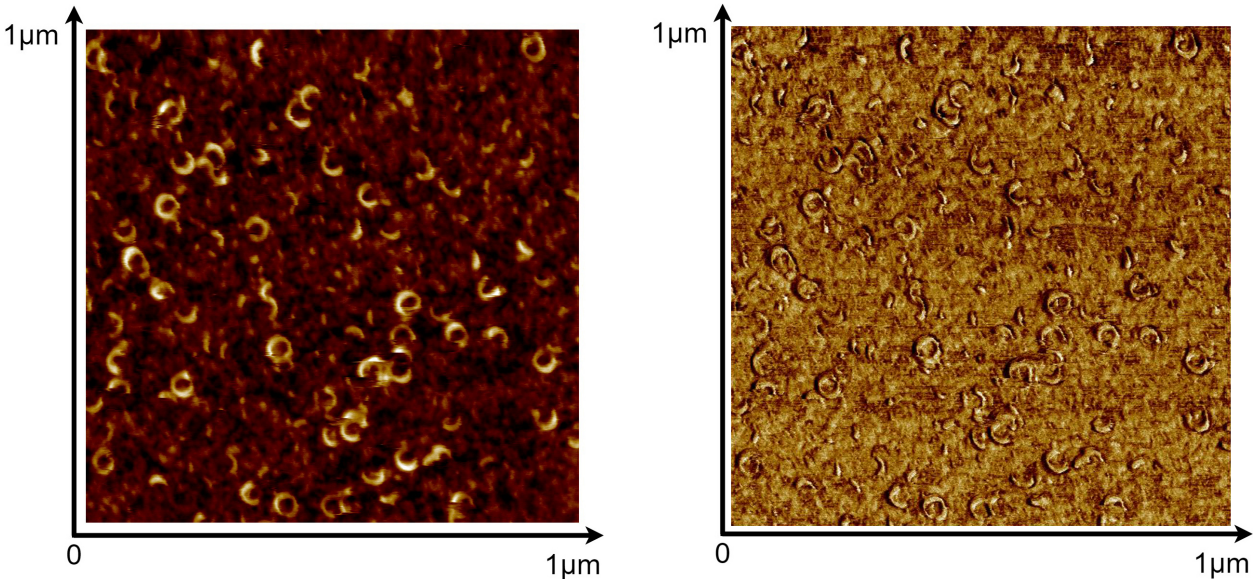
Examples of different channels with size axes are provided in Figures 18b and 18a.

6.2 Image processing

Due to the reasons listed above, it was decided to take advantage of using the raw images because it allows us to freely manipulate them with the help of NanoScope Analysis and prepare datasets with images of any available channel.

6.2.1 Manipulation of raw AFM images

Initially, the images come with some background noise, which if unprocessed will complicate the training and detection processes. Figure 19a displays an unprocessed raw representation of the height sensor channel. One can notice how the color changes from deep brown at the bottom to light beige at the top. Lighter color means that the tip of the microscope went up at those sections of the surface, whereas in dark ones - went lower. There are also slightly noticeable rings, which



(a) Height channel (b) Phase channel

Figure 18. Difference between AFM image channels.

are hard to distinguish and classify even for a human. In the best-case scenario, neural networks should not be trained on such images, because the targets are not clearly seen and blend with the background.

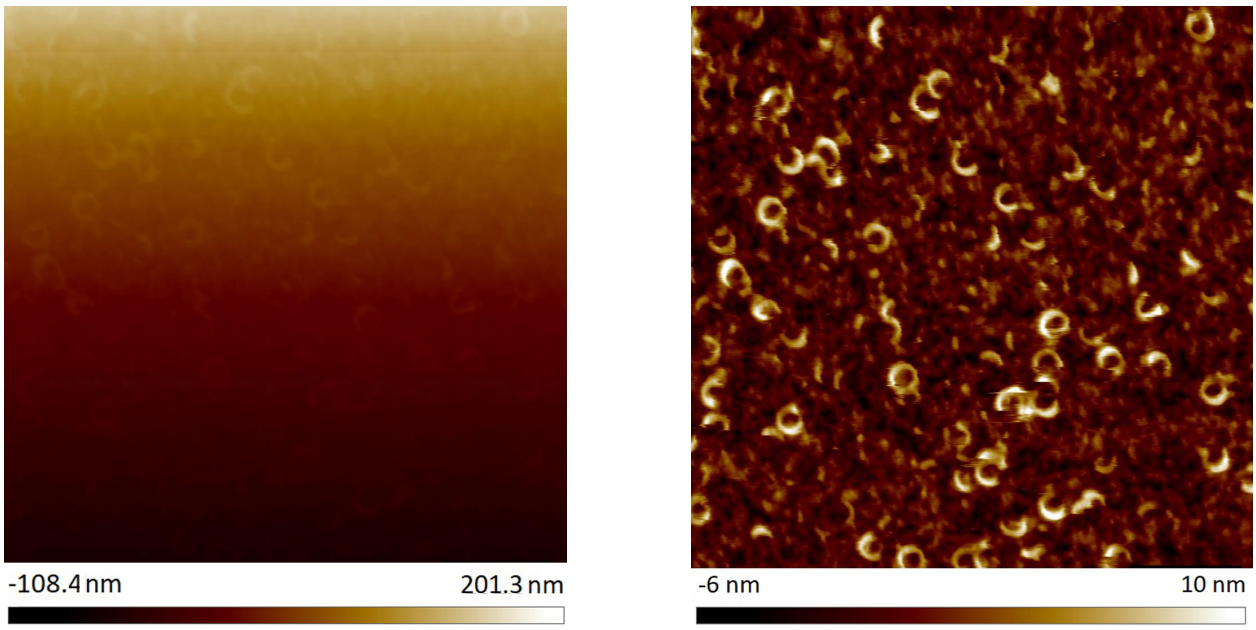
NanoScope provides a command called *Flatten* to eliminate unwanted features before the main analysis. As the official documentation states, this procedure is a filter that removes low-frequency noise and tilt from the AFM image. Also, the software allows one to choose one out of 4 various flattening orders, each will produce different output. Briefly, each option removes Z-offset between scan lines, but does it in a different way [7]:

- 0th - subtracts average Z value.
- 1th - calculates 1st order of least squares fit + removes tilt.
- 2th - calculates 2nd order of least squares fit + removes tilt and bow.
- 3th - calculates 3rd order of least squares fit + removes tilt and bow.

Besides that, there is an option to specify a concrete threshold or flatten only certain parts of the image based on the threshold height (Z offset). It allows the preservation of the original data in some sections if there is such a need.

Beneath both Figures 19a and 19b presented a scale, showing how the color correlates with the relative height. In the original image, the tip had a minimum offset of -108.2 nm and a maximum of 201.3 nm. After applying the flattening with 1st order, the offset drastically decreased to a range from -6 nm to 10 nm. Moreover, the features have become clearly distinguishable, revealing single clusters and aggregations. Such an image can already serve as training material for a neural network. Even though some original images have a less pronounced Z offset difference, such a procedure of flattening still has to be performed on all of the samples to ensure that most of the noise is removed.

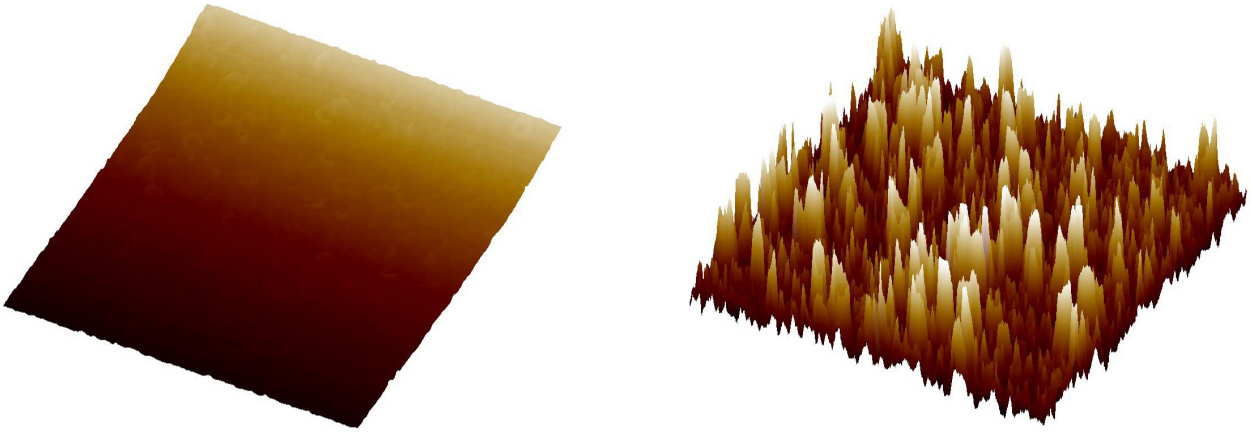
Figures 20a and 20b depict a different perspective of already discussed process. Both samples are observed from the same position, zoom and angle. The non-flattened image observed from



(a) Non-flattened Height Sensor channel

(b) Flattened Height Sensor channel

Figure 19. The effect of flattening operation.



(a) Non-flattened Height Sensor channel in 3D

(b) Flattened Height Sensor channel in 3D

Figure 20. The effect of flattening in the 3D perspective.

another viewpoint looks like a tilted flat plane without any features. Meanwhile, after performing the flattening, the picture is completely different: the surface does not look like a hill anymore. The defects no longer merge with the surface and become lighter as they pull over, with their tip becoming almost white. There are also deepenings that are darker. It proves that the light color corresponds to higher points and vice versa, as was stated in previous paragraphs.

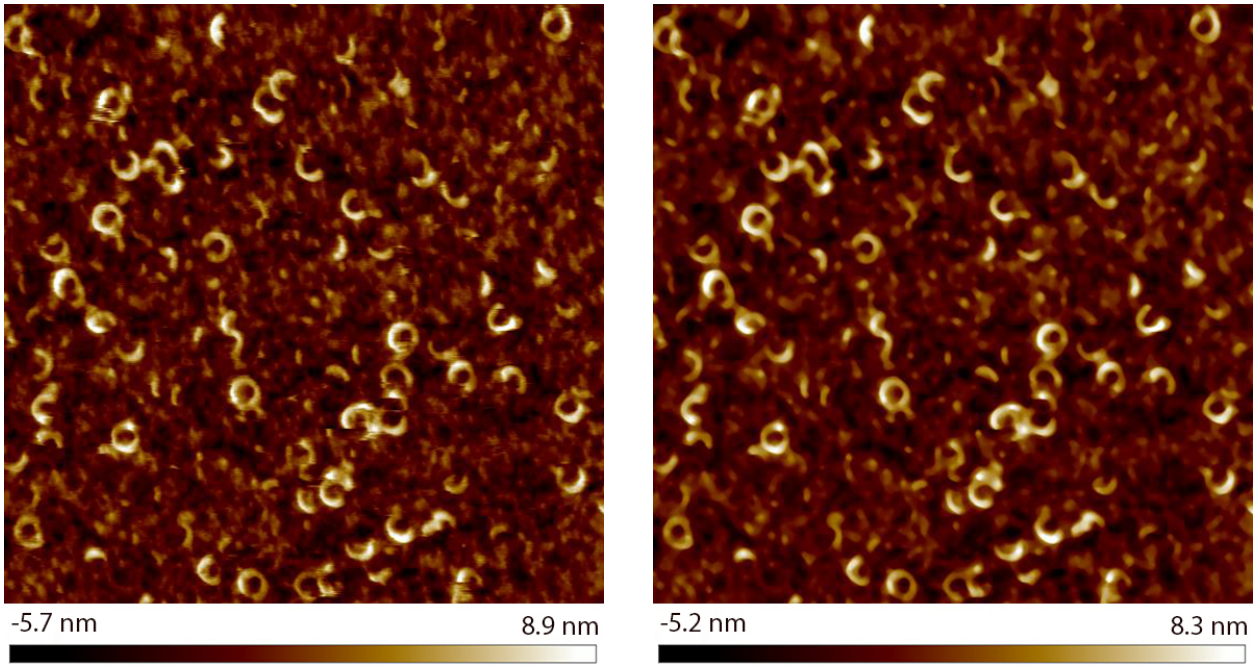
Among other features offered by NanoScope Analysis, filters are one of the most useful when it comes to image processing. There is a choice of 3 different options:

- Gaussian - a classic filter based on the Gaussian curve, which can be run either on the Y axis or X. Provides several options for customization such as the size of the scanned line, axis and type: lowpass/highpass.
- Lowpass - uses a 3x3 kernel of pixels to recalculate the pixels in the center by averaging the surrounding values. The developers of NanoScope Analysis agreed that the naming

"Average" suits more, but left it as it is due to compatibility.

- Median - similar to the previous one, but allows the user to choose a kernel of the recalculation. Available options are 3x3, 5x5, 7x7, 9x9, or 11x11.

The practical effect of all three filters is similar and results in slight image blurring, removal of sharp noise and loss of details. To simplify the image preparation process and save time it was decided to apply the Median filter because this filter does not require any complex configuration compared to the Gaussian but still allows to adjust the kernel according to the situation. Figure 21a depicts the image before applying any filter, while figure 21b shows the results of using the Median filter twice with the kernel 11x11. Slight effects are noticeable: the features are not that sharp anymore, but at the same time background noise has blended with the surface. The results of applying other filters are provided in the appendices.



(a) Before filtering

(b) After filtering

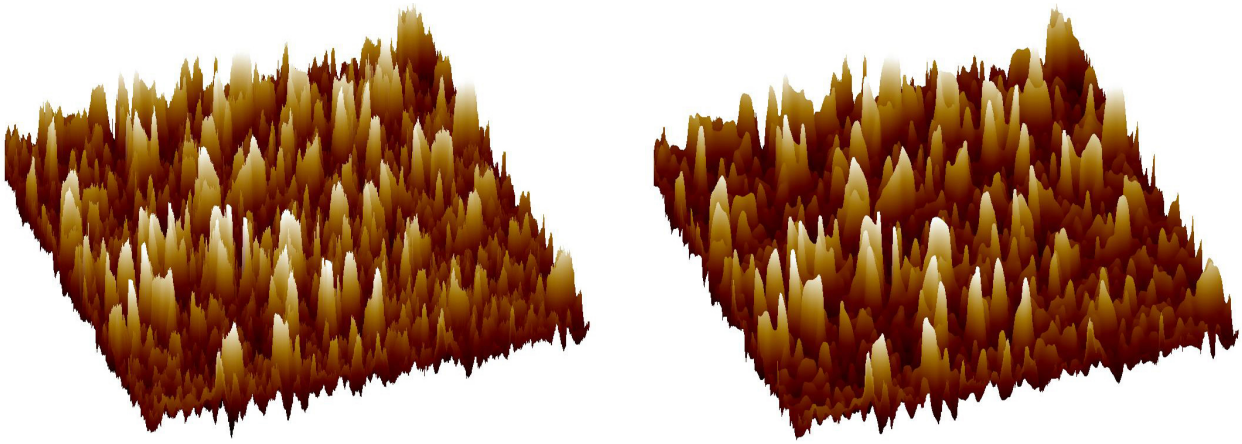
Figure 21. The results of applying the Median filter twice with the kernel 11x11.

One can pay attention that the scale provided beneath Figures 21a and 21b has changed its range. Indeed, filtering has smoothed all the features. As illustrated in Figures 22a and 22b from the 3D perspective, the spikes are not that sharp. In addition, smaller thorns are less distinguishable as well as the deepenings.

These are the basic but the most useful functions, which help to extract important features from the raw images. After all the manipulations, the image is exported to JPG format and is ready for the next processing steps.

6.2.2 Grayscale format

All the previously considered AFM images had a brown background. NanoScope Analysis allows changing of the color scheme, meaning what color would represent a certain Z-offset of the tip. Also, there is an option to adjust the RGB channels. For example, Figure is obtained through NanoScope Analysis. The noise is almost completely removed, but the issue with this approach is



(a) Before filtering (b) After filtering

Figure 22. The results of applying the Median filter twice with the kernel 11x11, 3D perspective.

that each image has to be considered and adjusted individually. From the personal experience of the author, it takes much more time than applying a filter, for example.

Conversion to grayscale format has already been considered by Igor Vilghelm in his master thesis, but it has never really been used for comparison, meaning that the model was still trained on the images from the height channel. Igor has described a few methods to manipulate RGB channels with different proportions, but in terms of this work will be used only a standard function. Red, green and blue channels are extracted from each image and multiplied by coefficients, as in the next formula:

$$P_{x,y} = 0.2989R_{x,y} + 0.5870G_{x,y} + 0.1440B_{x,y}, \tag{6.1}$$

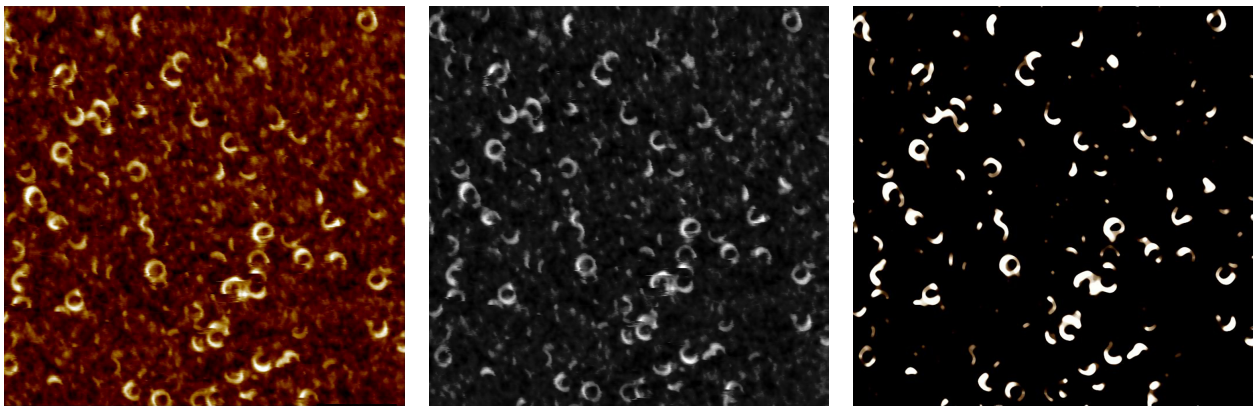
where $P_{x,y}$ is the new value of the pixel channels, $R_{x,y}, G_{x,y}, B_{x,y}$ are original channel values, and x, y are pixel coordinates on the plane.

Such manipulation may reduce the amount of noise in the images or blend it a bit with the background, therefore making the objects in the picture more distinguishable. As a result, there is a chance that the CNN model can make more precise predictions and reduce the error rate. As the main goal of this project is to perform analysis, it was decided to reuse an existing tool for conversion to the grayscale format, written by Igor Vilghelm during his master's thesis [42] (results is shown in Figure 23). It will bring all the images to the same format and save time compared to individual processing in NanoScope Analysis, considered above.

6.3 Image labeling

To perform regression tasks, such as object detection, the model has to be trained on known data. Usually, the information about the object positions is stored in annotation files alongside the image it describes. One of the popular annotation types is the bounding box. Bounding boxes are rectangular boxes defining the location of the object. They are determined either by top left and lower right corner coordinates or by specifying the center, width and height of the box.

For some images the Life Sciences Center provided coordinates of the spotted defects in Pascal VOC format. Pascal VOC is an XML file that contains information about the image it describes, such as dimensions, name, relative path, and present objects. Each object has several key features:



(a) Colorful image (b) Processed by Igor's tool (c) Processed in NanoScope

Figure 23. An AFM image in different color schemas.

- Name - for classification of the detected objects.
- Truncated - 0 or 1 value, indicating whether the bounding box is full or truncated.
- Difficult - 0 or 1 value, showing if the object is hard to detect.
- Bndbox - contains 4 child nodes with the top left and lower right corner coordinates.

YOLOv5 model accepts annotation in a format different from Pascal VOC. It should be a txt file where each row is responsible for a separate object. The row should contain 5 numbers: image class index, object center coordinates, width, and height. All the coordinates are normalized according to the image dimensions, therefore are in the range from 0 to 1. This format is called YOLO Darknet TXT. To convert the present Pascal VOC annotations to Yolo, it was decided to write a script in Python programming language. Since these two formats store coordinates in different ways, the algorithm for coordinates conversion of one object is provided in 1:

Algorithm 1. Pascal VOC to YOLO Darknet TXT

Require: $x_{min}, y_{min}, x_{max}, y_{max}, w_{image}, h_{image}$
Ensure: $x_{center}, y_{center}, w_{bbx}, h_{bbx}$
1: $x_{center} = (x_{min} + x_{max})/2/w_{image}$
2: $y_{center} = (y_{min} + y_{max})/2/h_{image}$
3: $w_{bbx} = (x_{max} - x_{min})/w_{image}$
4: $h_{bbx} = (y_{max} - y_{min})/h_{image}$
5: **return** $x_{center}, y_{center}, w_{bbx}, h_{bbx}$

6.4 Mosaic augmentations

When the amount of images in the dataset is low, it is beneficial to perform such image transformations as rotating, cropping, changing values of the color channels, etc. Usually, all these transformations are generalized by word augmentations. The application of these methods allows to create an illusion that the dataset is more varied. At the same time, this approach lowers the chances of the model overfitting, because the neural network considers that the training is being done on different pictures.

If one of the model goals is to perform a regression task, such as the detection of object position on the image - the programmer faces the issue of transforming bounding boxes at the same time with the picture to preserve their meaningfulness in regards to the position on the image. One can try to use masking: when the image augmentation process starts, the mask of the same dimension as the image is created, with bounding box coordinates put on the mask. Then the mask is treated in the same way as the image: cropped, flipped, etc. After the transformations, the mask is converted back to the bounding box coordinates.

There are other approaches to this task and already prepared solutions on the market (open-source libraries), but the problems with integration and the human error factor still remain. To relieve the programmers from the burden of developing their own augmentation solution or applying 3rd party ones, the YOLOv5 model has a built-in image augmentation module, which automatically applies the transformations during training. One of the features that stands out the most is called mosaic augmentation.

The main idea of this feature is that before feeding an original image to the model, the algorithm automatically combines multiple different pictures from the dataset in a dispersed way, creating a grid with transformed images. Each cell of the grid contains parts of several modified images with some sections overlapped, hidden or passed through some color filters. The developers of YOLOv5 claim that the model never sees the same image twice. An example of the mosaic image augmentation algorithm result is provided in Figure 24.

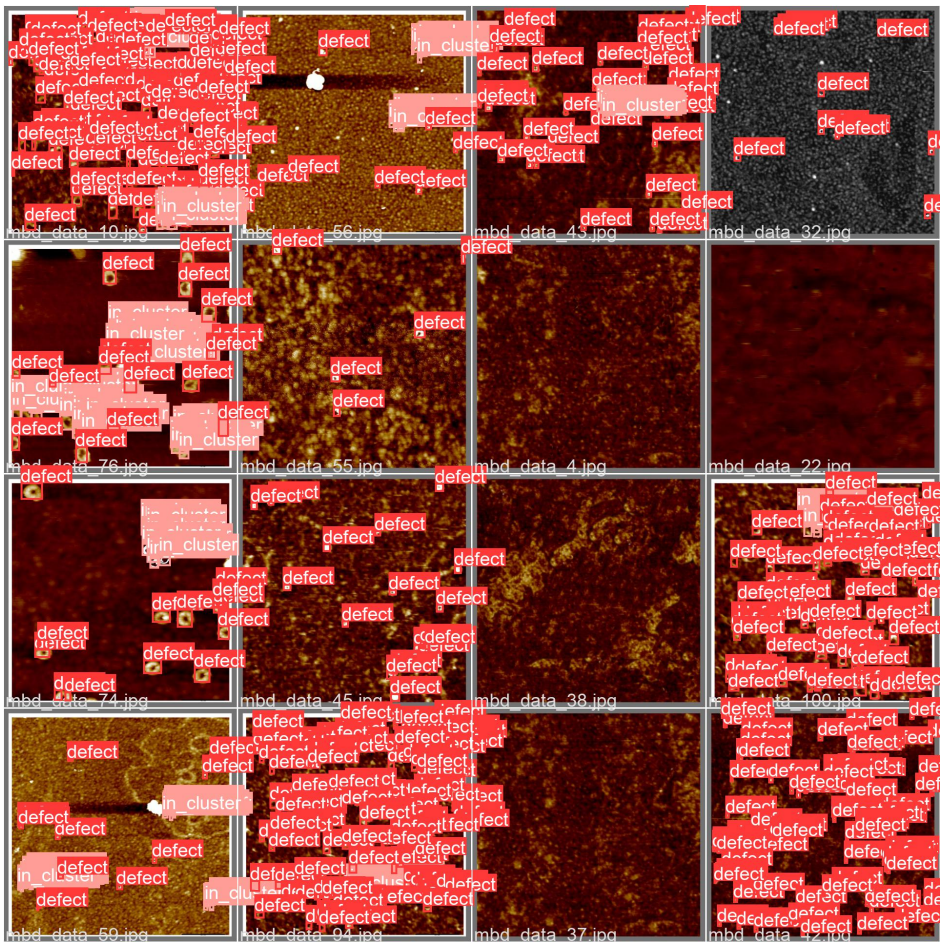


Figure 24. Mosaic image augmentation.

6.5 Splitting the dataset into test and training parts

One of the most time-consuming tasks throughout this thesis was processing, cropping and exporting data from NanoScope Analysis with further manual labeling. It was decided to split images into training and testing parts using the holdout approach. That means, that one segment of the dataset was used to train the model, whereas the remainder was applied for checking how well the model predicts defects on other images.

Here and later, if the dataset contains the suffix *_raw* in its name, that means the dataset contains images with a preserved original color scheme, whereas the suffix *_gray* means that images were converted to the grayscale format.

Also, if not specified differently, all the datasets contain height channel images exported from raw files. The main reason behind this is that the height channel is the only one that is present in most of the raw files, therefore allowing the preparation of larger datasets.

For the CNN model training were prepared datasets with the following code names:

- *single_defects_raw* - images with only single defects labeled, and without clusters being present on the image at all, 45 images in total;
- *single_defects_gray* - same as previous, but in the gray color scheme;
- *defects_in_clusters_height_raw_29* - height channel images with only defects inside clusters were labeled, 29 images in total
- *defects_in_clusters_height_gray_29* - same as previous, but in the gray color scheme;
- *defects_in_clusters_amplitude_raw_29* - same as previous, but the amplitude channel was used instead of the height one;
- *defects_in_clusters_amplitude_gray_29* - same as previous, but in the gray color scheme;
- *defects_in_clusters_height_raw* - images with only defects inside clusters labeled, 50 images in total;
- *defects_in_clusters_height_gray* - same as previous, but in the gray color scheme;
- *single_and_inside_clusters_raw* - the biggest dataset with labeled both single defects and the one inside clusters, 89 images in total;
- *single_and_inside_clusters_gray* - same as previous, but in the gray color scheme.

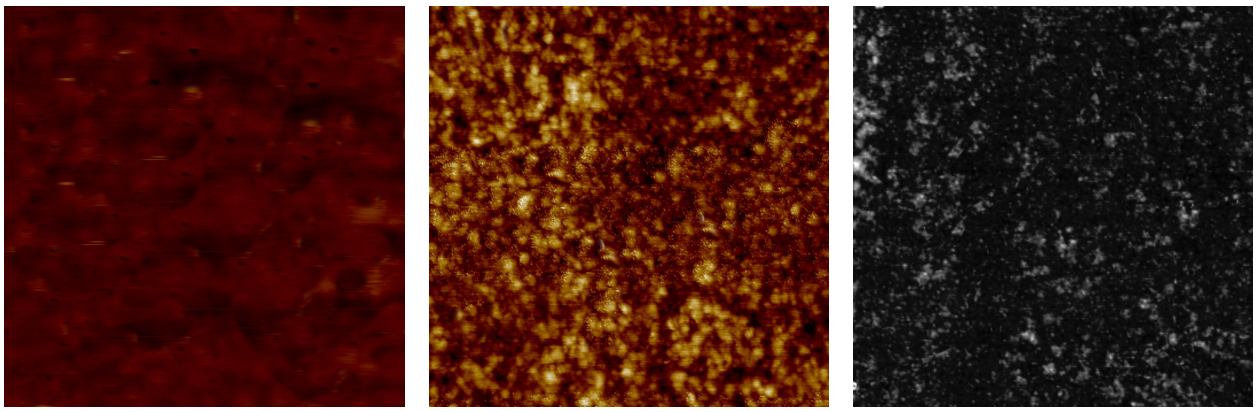
Also, from the author's research work [20], were reused 4 datasets where all the defects are labeled, but images mostly contain only single defects. Initially, there was a classification into 4 types, but for sake of simplicity, it was removed. Worth mentioning that these datasets were split into 80%/20% parts:

- *small_all_raw* - 23 images in total;
- *small_all_gray* - same as previous, but in the gray color scheme;
- *big_all_raw* - 34 images in total;
- *big_all_gray* - same as previous, but in the gray color scheme.

To prevent duplicates during splitting images into training/test groups, there was developed a custom script using Python programming language which automates this process by accepting the collection of images and labels and then randomly putting them into separate folders based on entered percentage.

An important detail is that images with labeled clusters were rare among the data provided by the VU Life Sciences Center, so some clusters were manually drawn or generated by combining several images into one, therefore it may affect the precision of defects detection in real examples.

One of the key differences between the datasets in this thesis and the author's scientific research project is that the new datasets exclude non-thematic true negative images with no defects by introducing pictures with the membrane surface. This may assist the model algorithms to better distinguish between the background and target objects. These images were obtained either by cropping out some parts of raw files or by completely blurring out the defects. Examples of thematic true negative images are provided in Figure 25:



(a) (b) (c)

Figure 25. Thematic true negative images without defects.

7 Model training and results analysis

7.1 Programming environment and written code

All the project parts were implemented with Python programming language (v 3.9.11). The image preprocessing and annotation formats conversion were done on the local machine. The specifications of the local environment are the following:

- Windows 10 Pro (v 19044.1706).
- CPU AMD Ryzen 7 5800H 3.20 GHz.
- RAM 16.0 GB.
- GPU Nvidia GeForce RTX 3050 Ti 4 GB.

For the model, training was used Google Colaboratory environment [8], which provides free-to-use computing resources such as CPU and GPU. Besides that, next Python libraries and technologies were used during the final master's thesis:

- PyTorch (v 1.11) [31] - a library for machine learning.
- NumPy (v 1.22.0) [29] - a library for array manipulations.
- Shapely (v 1.22.0) [36] - a package for manipulation of planar geometric objects.
- SciPy (v 1.22.0) [35] - a library with fundamental algorithms for scientific computing.
- Jupyter (v 6.4.11) [18] - a platform for working with Python notebooks.
- Weights and Biases (v 0.12.10) [43] - a platform with model performance tracking tools.

The author has written functions for the calculation of Voronoi entropy and sigma, splitting the dataset into test/train parts, conversion of Pascal VOC to YOLO labeling format and functions to combine the detection of the defects with the calculation of Voronoi metric creating better user experience with the easier flow. Overall, 300 lines of code were written.

7.2 Statistical metrics

Before proceeding to the experiments, it is important to introduce several main statistical measures: F1 score, recall and mean average precision (mAP) are the evaluation metrics that are commonly used to measure the performance of object detection algorithms.

Recall is the number of true positive detections divided by the total number of objects in the test set (true positives plus false negatives). It is a measure of how many of the objects in the test set the model was able to detect. A high recall indicates that the model is able to detect a large proportion of the objects in the test set.

$$Recall = \frac{TP}{TP + FN} \tag{7.1}$$

Precision is used to measure the ability of the classifier to correctly identify instances of a particular class. A high precision indicates that the classifier is making few false positive detections and is therefore more accurate:

$$Precision = \frac{TP}{TP + FP} \tag{7.2}$$

Mean average precision (mAP) is a metric that measures the average precision of the model across multiple classes. Precision is calculated for each class and then averaged across all classes. mAP is often used to evaluate the performance of object detection algorithms because it takes into account both the precision and the recall of the model. Average Precision is calculated as:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} AP_i, \text{where } AP_i \text{ is the average precision for class i} \tag{7.3}$$

The metrics listed above can be reliable only if the dataset is class-balanced. Even though most of the prepared datasets contain a single class, the main one contains both single defects and in-cluster ones. The latter one may hold a minority, introducing imbalance (as stated in the section 6). For such cases F1 score is used alongside with mAP and recall, which is the harmonic mean of precision and recall and is calculated as:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{7.4}$$

7.3 Review of author's SRP results

The author's Scientific Research Project can serve as a background for future experiments and is worth to be briefly reviewed. The main goal was to check if YOLOv5 is suitable for the task of defects detection and whether it can produce satisfying results with relatively small datasets. During the process of training was used a model YOLOv5s (pretrained), that is holding a middle-ground between speed and efficiency as claimed by the developers. The task was not only to predict the positions of the defects but to classify them into 4 groups according to their shape. The results are provided in the table 5.

There were held 6 experiments with 4 datasets described in the section 6.5. Obviously, the model demonstrated better results in all 3 statistical metrics when it was provided with larger datasets. But the main point is that the model was able to show more than 80% of the F1 score, meaning it has potential for future research.

Dataset	Epochs	mAP	$Recall$	$F1$
small_all_raw	750	84.54%	78.04%	81.16%
small_all_gray	750	71.08%	64.68%	67.73%
small_all_raw	750+1500	85.8%	78.52%	81.99%
small_all_gray	750+1500	72.03%	65.12%	68.4%
big_all_raw	750+1500	86.49%	79.8%	83.01%
big_all_gray	750+1500	89.8%	83.3%	86.43%

Table 5. Scientific research project results.

During the SRP was made an important observation. When the model was trained for more than 2000 epochs using the small datasets, it started showing signs of overfitting: the charts of the statistical metrics did not change over a long amount of epochs and the model started to perform poorly on unseen data.

7.4 Model modifications

7.4.1 Activation functions

As it was briefly discussed in the section 5.2.1, the activation function is a key component of convolutional neural networks that decides whether the output from the current neuron is important or not. It transforms the summed and weighted inputs into a value that will be provided to the next layer later. Choosing the appropriate activation function for a CNN can have a significant impact on the performance of the network. They can be divided into three groups [11]:

- binary: $f(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1250, & \text{if } x \geq 0 \end{cases}$;
- linear: $f(x) = k * x$ - they simply return the input value as the output, without applying any non-linear transformations;
- non-linear

The latter one adds complexity and turns the perceptron into a non-linear function. Without it the neural network would behave as a linear regression model, therefore making the process of solving any complex task almost impossible. In some cases, a model can have a combination of linear and non-linear activators in different hidden layers.

There is no universal activation function that will produce the best results in every problem. Each task requires unique investigation and comparison in order to detect the most useful one. Default activation function in YOLOv5 is SiLU - Sigmoid-Weighted Linear Units (more expanded version is called Swish), which is relatively new and was introduced by Google researchers in 2017. It proved to perform well with popular datasets such as MS COCO. Swish has also been shown to be more computationally efficient than other activation functions, such as ReLU, because it does not require setting negative input values to zero. This can make training deep neural networks faster. The mathematical definition is the next one [39, 38]:

$$Swish(x) = x * \sigma(\beta * x), \text{where } \beta = 1 \text{ for Swish-1 and SiLU, and } \sigma = sigmoid \text{ for SiLU} \quad (7.5)$$

It was decided to check whether a different activation function can improve the results achieved in the author's scientific work where the model was run with the default one [20]. For this purpose, the model was trained 4 times using each of the same 4 datasets from the SRP, the mAP chart is presented in Figure 26. Besides default SiLU, were chosen the next 3 activation functions [25, 14]:

- Mish - a non-monotonic activation function that combines the properties of ReLU (Rectified Linear Unit) and tanh (hyperbolic tangent).
  $Mish(x) = x * tanh(ln(1 + e^x))$.
- Flexible Rectified Linear Unit (FReLU) - a variant of ReLu, which is designed to be more flexible than just cutting the negative part. According to the creator of YOLOv5, FReLU may be suitable for smaller versions, but in larger ones causes earlier overfitting.
  $FReLU(x) = x * g(x)$, where $0 \leq g(x) \leq 1$.
- HardSwish - an activation function that is similar to Swish, but has a slightly different form and sometimes is used as an alternative to ReLU. It has a smooth gradient and is less prone to saturating than other activation functions.
  $Hardswish(x) = x * sigmoid(x + 3)/3$.

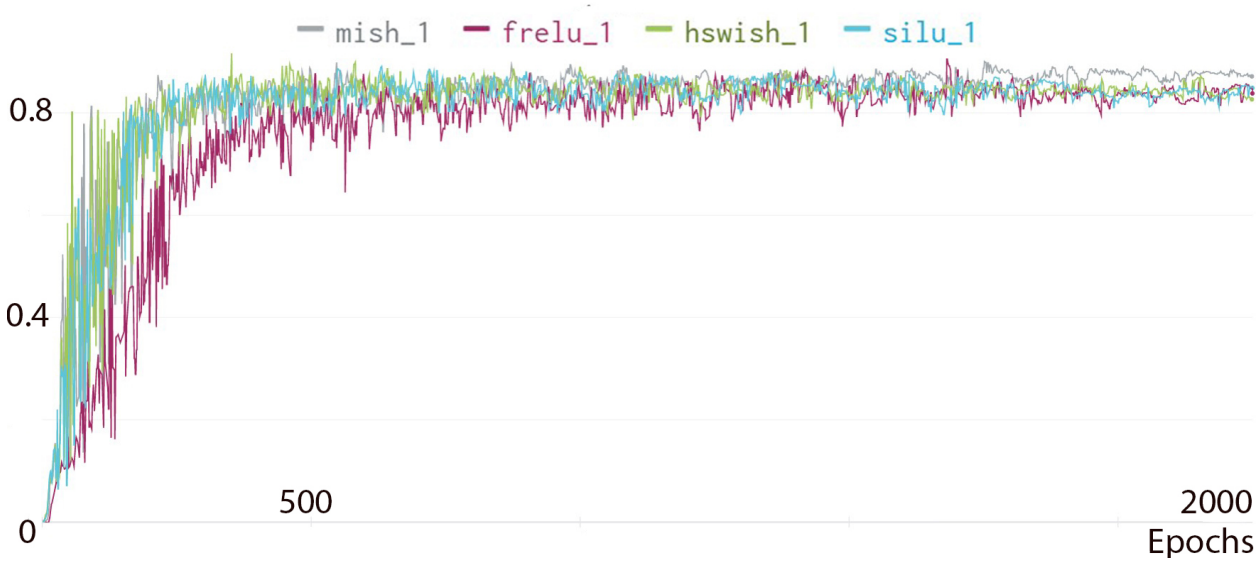


Figure 26. mAP chart for an experiment with *small_all_raw* dataset,

Activation \ Dataset	*small_all_raw*	*small_all_gray*	*big_all_raw*	*big_all_gray*
Mish	0.85	0.67	0.85	0.83
FReLU	0.84	0.66	0.86	0.80
HardSwish	0.84	0.71	0.86	0.83
SiLu	0.85	0.68	0.87	0.83

Table 6. mAP for different combinations of datasets and activation functions.

According to the table 6, FReLU performed not as well as other activation functions, demonstrating the lowest mAP in 3 out 4 experiments. Taking into consideration the developer's note that it may cause issues on bigger models, FReLU definitely will not be used in this thesis. Mish also did not showcase any improvements in precision, whereas models with HardSwish and SiLu activation functions had similar outputs and are candidates for being used in future experiments.

Activation \ Dataset	*small_all_raw*	*small_all_gray*	*big_all_raw*	*big_all_gray*
Mish	51m 29s	32m 37s	1h 58m 55s	1h 57m 25s
FReLU	49m 44s	34m 2s	1h 55m 59s	1h 55m 13s
HardSwish	57m 9s	35m 0s	1h 57m 31s	2h 02m 37s
SiLu	40m 30s	30m 27s	1h 54m 15s	1h 50m 24s

Table 7. Model training time for different combinations of datasets and activation functions.

Besides mAP there is also one more extremely important criterion — training time. For instance, if the activation function can give a 1% boost in mAP but the execution time will increase by 30% — the feasibility of such a change is highly questionable. Table 7 contains data on how much time each training has taken. Based on that it can be seen that all the experiments with HardSwish, which was considered for future usage, took the longest compared to all others. Mish and FreLu did not provide any positive impact either on training time or on mAP, therefore will not be used anymore as well. The only left option is SiLu activation function, usage of which

showcased both the fastest speed and the highest mAP among considered options. Therefore, in future experiments will be used SiLu.

7.5 Model training

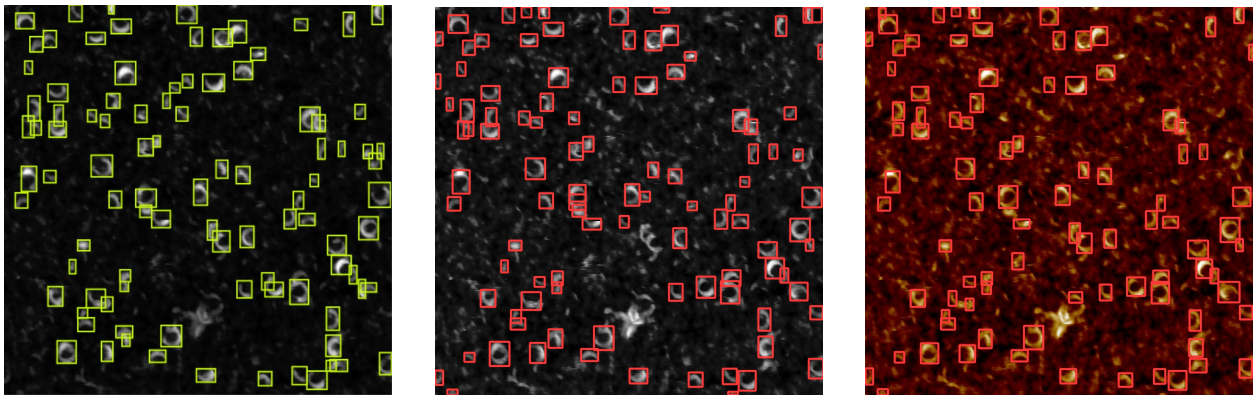
Training of a CNN model is the process of showing the network a large number of examples and adjusting the network's internal parameters (called weights and biases) so that it can accurately classify or predict the output for a given input. The AFM defects detection can be classified as supervised learning because the model will already have examples of correct predictions - the bounding boxes in annotation files.

To estimate the clusterization parameters will be used Voronoi entropy and σ. The following subsections will contain examples of YOLOv5 detections, to simplify the captions of figures, the color scheme in the name of the datasets might be replaced with three dots. For example, the sentence "trained using *single_defects_raw* and/or *single_defects_gray* datasets" will be shortened to "trained using *single_defects_...* datasets".

7.5.1 Detection of single defects

The goal of the first two experiments was to train the models to detect only single defects without classifying them into any types, only position prediction. For this purpose were used datasets *single_defects_raw* and *single_defects_gray* which mostly contain images without distinguishable clusters and the present ones are not labeled.

In both cases the models were trained for 500 epochs, the results of predictions are illustrated on the pictures below. During the detection, the confidence score was set to 0.5. All the examples are provided with a greater scale in the appendices.



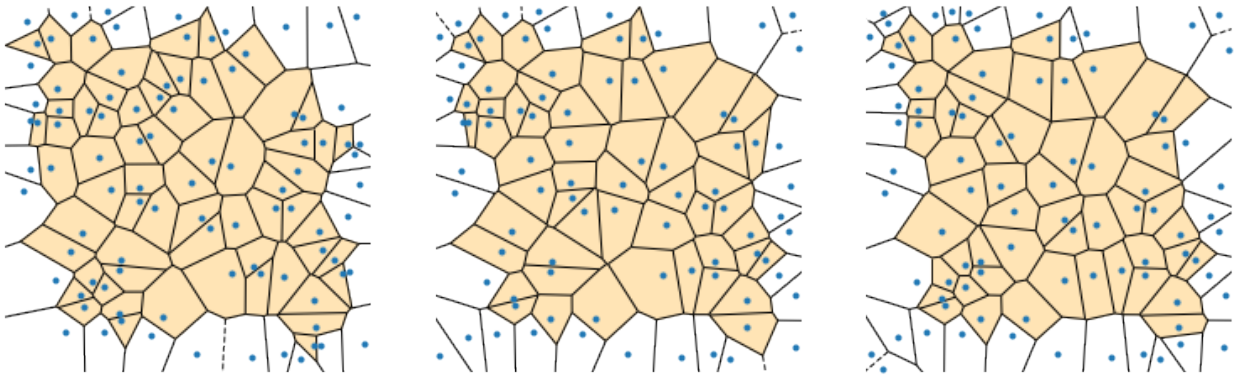
(a) Original labels (b) Grayscale trained model (c) Full-color trained model

Figure 27. Detections of models trained using *single_defects_...* datasets.

Figures 27a, 27b, 27c represent the results of the first two experiments. The models have not seen these images during training, so it can be considered real-life testing. As can be seen, the models have correctly detected most of the single defects, without any major differences. The model trained on the full-color images succeeded to predict a few more defects in the center.

To investigate more details for each of the images above were built Voronoi diagrams and calculated metrics to estimate clusterization (Figures 28a, 28b, 28c). From the table 8 it can be seen that both predictions missed around 10 defects, meanwhile, the Voronoi σ value remained almost the same, decreasing from 0.49 in original labels to 0.479 in labels predicted by the model

trained on grayscale images, and decreasing up to 0.469 for labels predicted by the full-color trained model. The Voronoi entropy changed as well, indicating that the orderliness changed.



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

Figure 28. Voronoi diagrams generated using coordinates from the experiments in Figure 27.

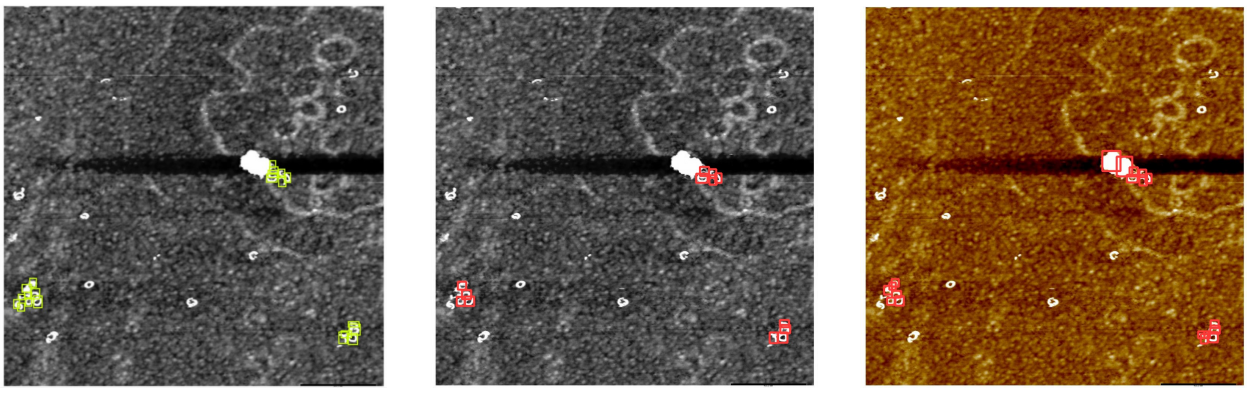
Metrics	Original labels	Grayscale trained model	Full-color trained model
№ of inner polygons	66	54	53
Voronoi entropy	1.631	1.33	1.493
Voronoi σ	0.49	0.479	0.469

Table 8. Metrics of Voronoi diagrams displayed in Figure 28.

7.5.2 Detection of defects inside clusters

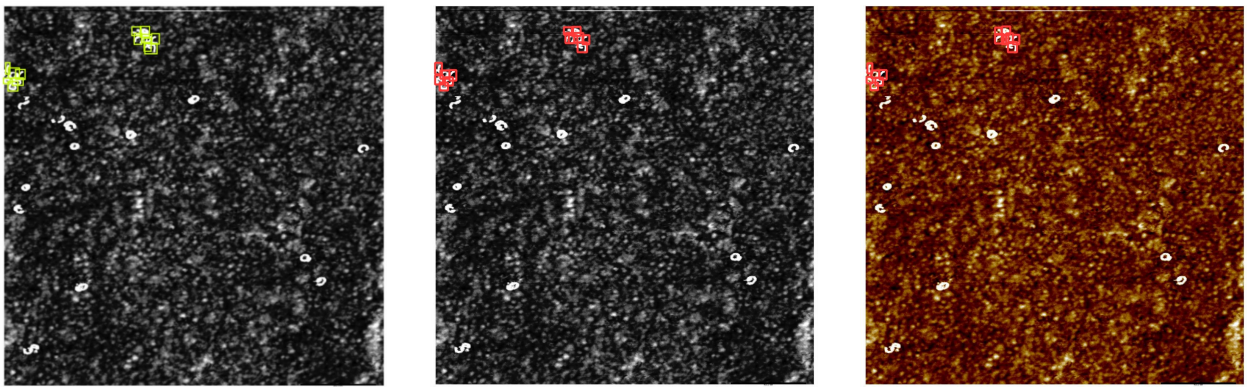
This section is dedicated to the detection of the defects inside clusters. Firstly it was decided to test if the usage of a different channel can bring any benefits. Therefore was selected the amplitude channel because it is the second most common in the raw images. To make the comparison with the height channel fair, there were prepared two equal datasets with 24 images each, as described in chapter 6.5. The datasets contain such few images because the amplitude channel was not present in all the raw files and the manual generation of the clusters was a too tedious task.

The model was trained for 500 epochs using datasets *defects_in_clusters_height_..._29*. Results are presented in Figures 29 and 30



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

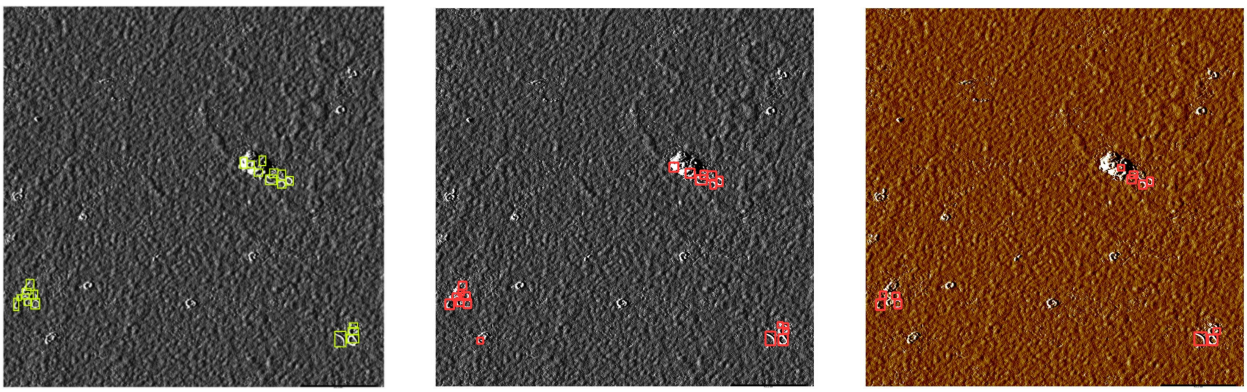
Figure 29. Detections of models trained using *defects_in_clusters_height_..._29* datasets. Test 1.



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

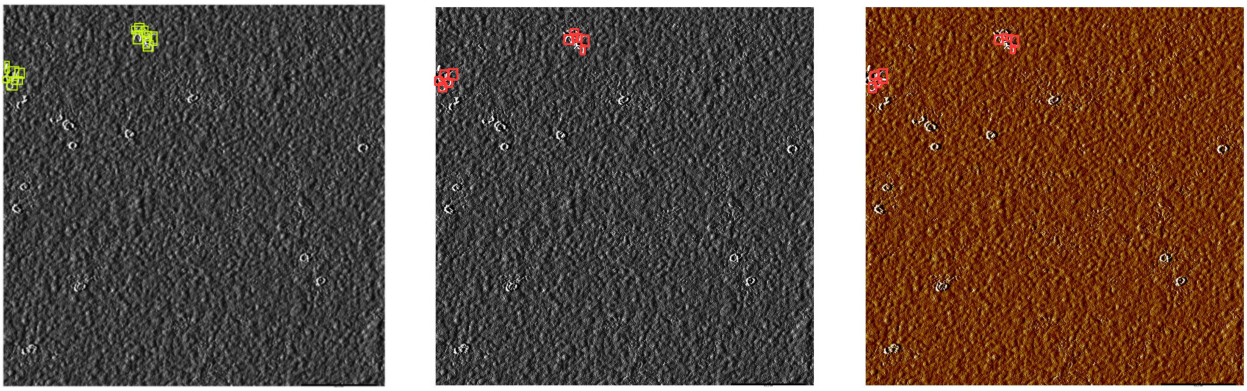
Figure 30. Detections of models trained using *defects_in_clusters_height_..._29* datasets. Test 2.

Datasets *defects_in_clusters_amplitude_raw_29* and *defects_in_clusters_amplitude_gray_29* were fed into the model during the experiments with the amplitude channel. Training again was held for 500 epochs, and the results of the detections can be seen in Figures 31 and 32. The exact same pictures were used during the training on the height channel. Building Voronoi diagrams for these predictions will not bring any meaningful insights because the amount of defects in these Figures is too less.



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

Figure 31. Detections of models trained on *defects_in_clusters_amplitude_..._29* datasets. Test 1.



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

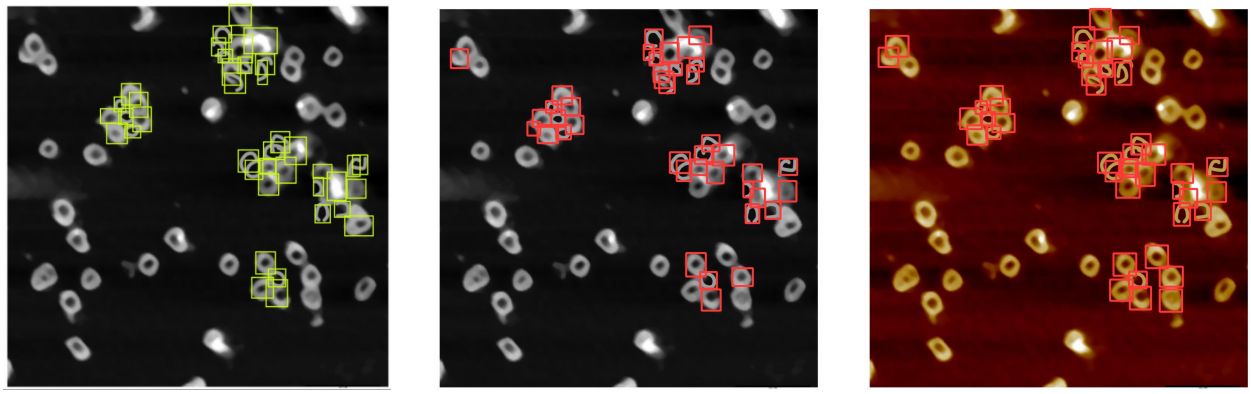
Figure 32. Detections of models trained on *defects_in_clusters_amplitude_..._29* datasets. Test 2.

Table 9 contains statistical measurements from the experiments above. As it was expected, the accuracy is relatively low, because the task is more complex than detecting single defects and the model was supplied with few images. In both cases, the training on the amplitude datasets resulted in worse results compared to the training on the height channel images. And after training on both datasets with gray images, the model demonstrated poor results compared to full-color images. The highest precision and F1 score were achieved during the training on *defects_in_clusters_height_raw_29* dataset, therefore in the next experiment will be used datasets with height channel.

Metric / Dataset	Recall	mAP	F1
defects_in_clusters_height_raw_29	0.63	0.7	0.66
defects_in_clusters_height_gray_29	0.58	0.6	0.59
defects_in_clusters_amplitude_raw_29	0.55	0.59	0.57
defects_in_clusters_amplitude_gray_29	0.53	0.52	0.52

Table 9. Statistical metrics of the trainings performed with datasets *defects_in_clusters_..._29*.

Next two experiments showcase the performance of a model trained on large height channel datasets containing only defects. The examples of detections are provided in Figure 33. Both models, trained on gray and full-color images marked defects in the top left corner of Figures 33b and 33c as clusters. It leads to the question of whether those defects should have been initially marked as a cluster. So, as it has already been stated several times: during manual labeling by a non-professional, some details might be not noticed resulting in such situations.

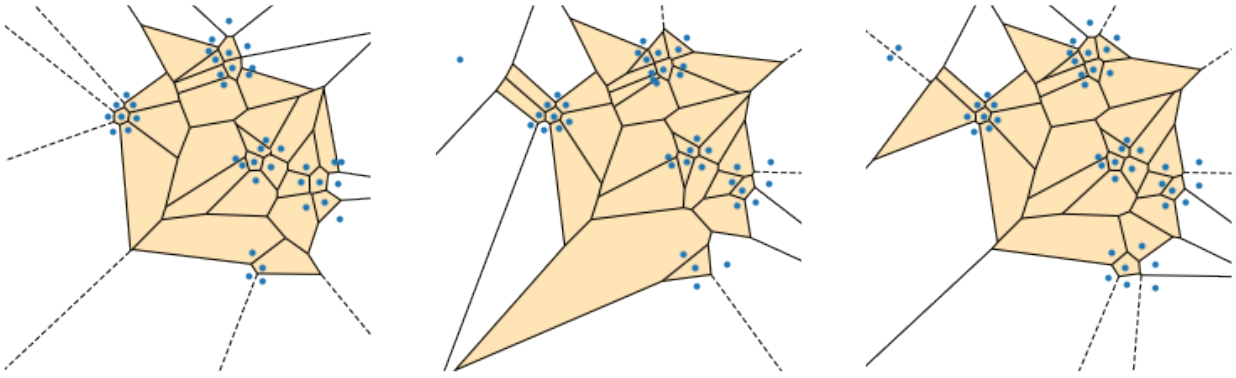


(a) Original labels (b) Grayscale trained model (c) Full-color trained model

Figure 33. Detections of models trained on *defects_in_clusters_height_...* datasets. Test 1.

As the predictions in Figure 33 contain more defects, it might make sense to build a Voronoi diagram for assessing clusterization parameters. The diagrams are presented in Figure 34, and Table 10 contains the metrics. The first thing that catches the eye, is that all three diagrams have noticeable differences, even though the number of predicted defects is close to the ground truth. Figure 34b contains one less defect in the bottom aggregation, which resulted in the huge inner polygon that has a large area significantly affecting the Voronoi σ: original is 0.854, whereas the predicted is almost twice larger —1.5. Similarly, since the defects in the top left corner are not present in the original image, the inner polygons differ. In such cases, slight changes may affect

the whole picture, which is why it is important to have a broad choice of samples allowing to test as many cases as it is possible.



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

Figure 34. Voronoi diagrams generated using coordinates from the experiments in Figure 33.

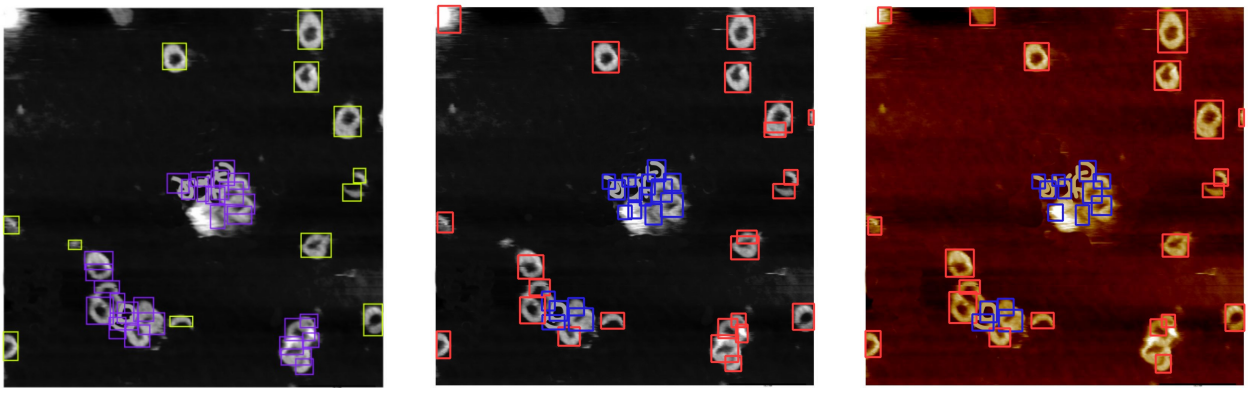
Metrics	Original labels	Grayscale trained model	Full-color trained model
№ of inner polygons	27	29	29
Voronoi σ	0.854	1.5	0.819

Table 10. Metrics of Voronoi diagrams displayed in Figure 34.

7.5.3 Detection of single defects and defects inside clusters

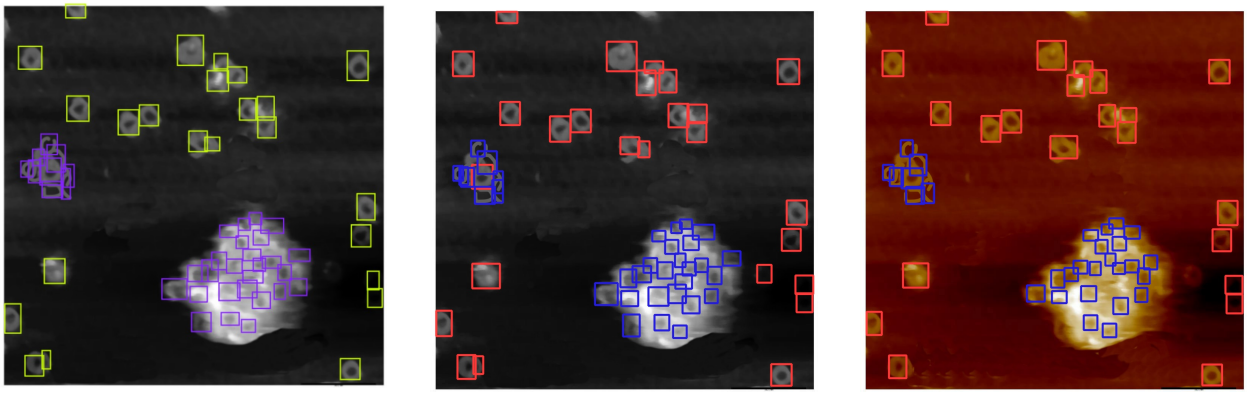
This section is dedicated to training the model for the detection of all the defects present in the AFM images. The models were trained for 250 epochs using *single_and_inside_clusters_gray* and *single_and_inside_clusters_raw* datasets, which are the largest ones and contain 89 images with defects classified into two types: *single* and *in_cluster*. The author considered training the model for a larger amount of epochs, but the training for 250 epochs was taking almost 2 full days in Google Colab, therefore the author decided to do not increase the training time.

Examples of detections are provided in Figures 35 and 36. In Figures 35a and 36a single defects are marked in yellow, and clusters are inside violet boxes. In the Figures containing predictions, single defects are outlined with red color, and clusters are blue.



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

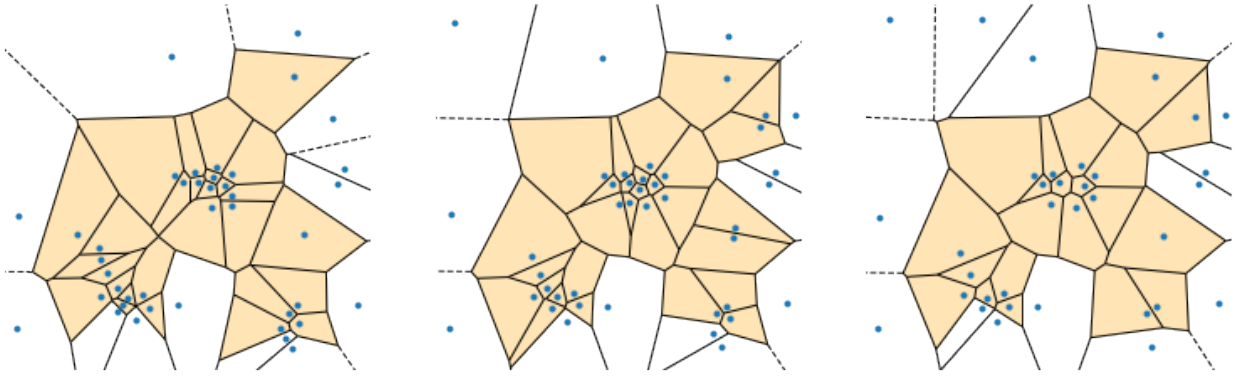
Figure 35. Detections of models trained using *single_and_inside_clusters_...* datasets. Test 1.



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

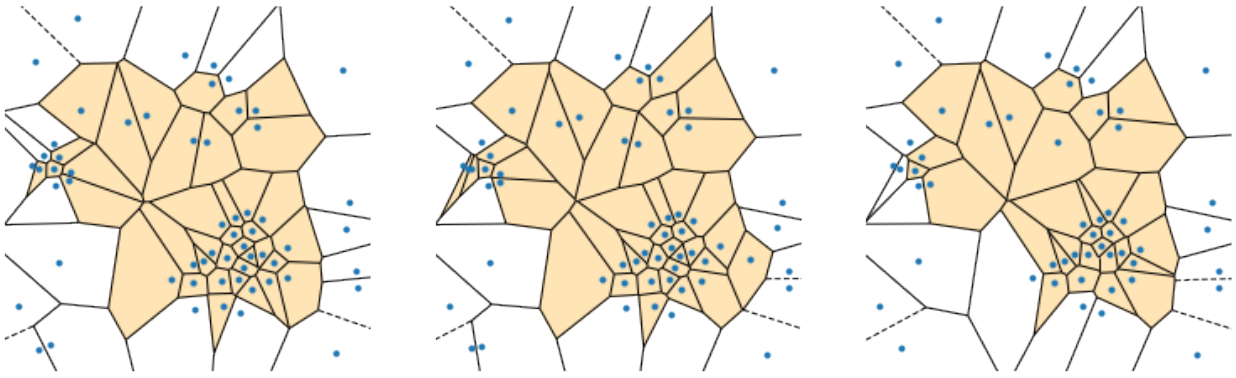
Figure 36. Detections of models trained using *single_and_inside_clusters_...* datasets. Test 2.

For estimation of clusterization metrics were generated Voronoi diagrams, displayed in Figures 37 and 38. At the first sight, there are noticeable differences in the top right and left corners of Figures 37a, 37b and 37c: the models detected more defects than it was originally mapped, resulting in a bit different structure of the polygons, whereas in Figure 38 the diagrams look similar except the absence of a big inner polygon in the middle of Figure 38c.



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

Figure 37. Voronoi diagrams generated using coordinates from the experiments in Figure 35.



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

Figure 38. Voronoi diagrams generated using coordinates from the experiments in Figure 36.

Tables 11 and 12 contain values of the clusterization measurements. Voronoi entropy is included as well because these are the main experiments of the thesis with a relatively high number of defects. In both experiments, the model trained on full-color images detected 10 fewer defects, which resulted in different Voronoi diagrams as described in the previous paragraph.

Metrics	Original labels	Grayscale trained model	Full-color trained model
№ of inner polygons	29	29	21
Voronoi entropy	1.434	1.502	1.356
Voronoi σ	0.944	0.899	0.757

Table 11. Metrics of Voronoi diagrams displayed in Figure 37.

Metrics	Original labels	Grayscale trained model	Full-color trained model
№ of inner polygons	38	40	30
Voronoi entropy	1.511	1.445	1.574
Voronoi σ	0.902	0.874	0.829

Table 12. Metrics of Voronoi diagrams displayed in Figure 38.

In the first experiment, Voronoi σ of the diagram built using the predictions of the grayscale-trained model is relatively close to the original value with a difference of 0.045 down, meanwhile, the full-color trained model result is less by almost 0.2. In the second experiment, the results are similar: Voronoi σ value is higher when the defects were predicted more precisely (0.874 for the grayscale-trained model), and is less when the model predictions of defects inside clusters are less accurate (0.829 for the full-color trained model).

Changes of the Voronoi entropy value are not following the logic of the Voronoi σ. Based on the values from both experiments, it appears that the entropy randomly changes compared to the Voronoi σ and the number of polygons. In the first experiment, the entropy value of the grayscale-trained model predictions is higher than the original one, while σ is less, and for the full-color trained model, both Voronoi entropy and σ are the lowest ones. But in the second experiment, the entropy of the full-color trained model predictions is higher than the original value, while the σ is the lowest of the three samples, and the number of predicted defects is almost 30% lower than the ground truth.

These observations once again prove that Voronoi entropy is not suitable for the estimation of the clusterization level of the defects in AFM images and any objects in general. This metric just shows how uniformly objects are distributed over the plane, and how homogenous are the Voronoi polygons. Meaning that the diagram in Figure 37c is less disordered than 37a, and even more homogenous than 37b.

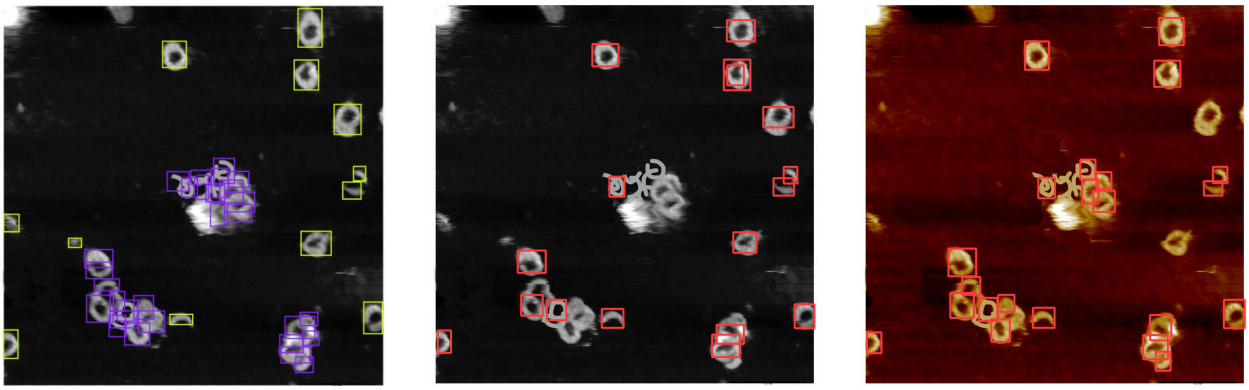
Table 13 contains the statistical performance metrics of the model trained using 2 different datasets. According to the data in the table, the models reached the mAP value 0.74 in both experiments, but the recall was higher during training with *single_and_inside_clusters_gray* dataset, meaning that the model trained on grayscale images is better at identifying positive samples, therefore F1 score is higher as well. Taking into consideration the conclusions from the Voronoi diagrams analysis, it can be stated that the model trained with *single_and_inside_clusters_gray* dataset produces better results, predicting the defects positions and the clusterization metrics more precisely.

Dataset \ Metric	Recall	mAP	F1
single_and_inside_clusters_raw	0.68	0.74	0.71
single_and_inside_clusters_gray	0.72	0.74	0.73

Table 13. Metrics of the trainings performed with *single_and_inside_clusters_...* datasets.

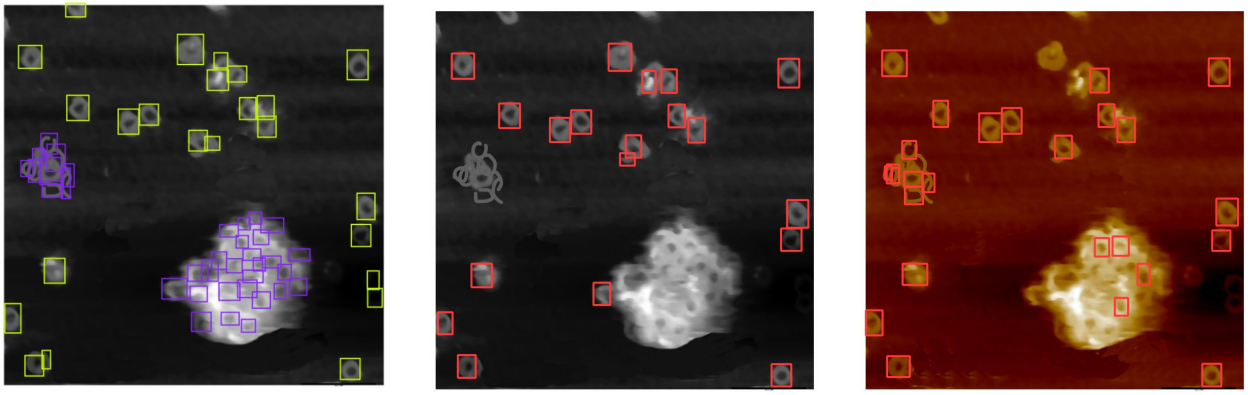
7.5.4 Comparison with the results of SRW

As the concluding experiment of this thesis, the author decided to compare the performance of the models trained during his scientific research project and the ones prepared in this work. For the experiments were selected the same two images as in the previous section, to make the comparison clearer. As demonstrated in Figures 39 and 40, both models struggle to detect defects inside clusters, but after the visual analysis, it is clear that the model trained on full-color images managed to perform better.



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

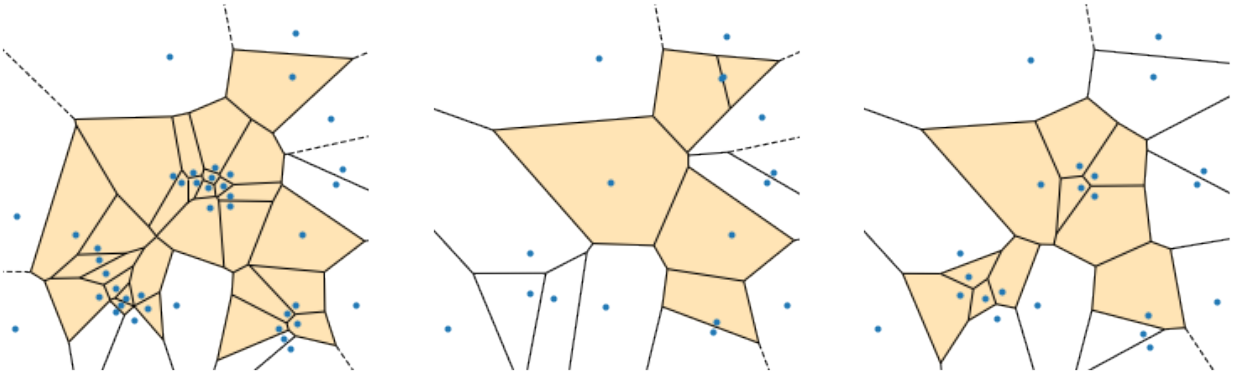
Figure 39. Detections of models trained during SRW. Test 1.



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

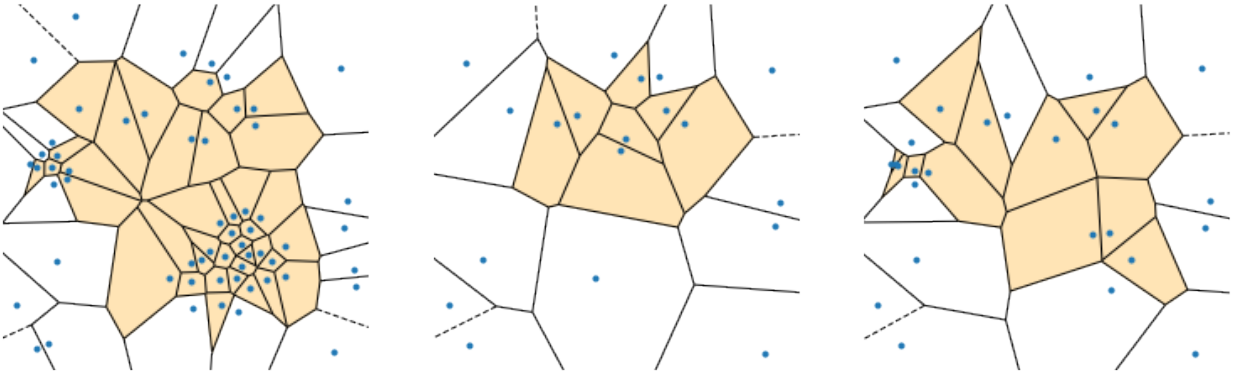
Figure 40. Detections of models trained during SRW. Test 2.

To analyze the predictions in-depth, Voronoi diagrams were built as in the previous experiments. Figures 41 and 42 demonstrate the huge difference between the original Voronoi diagrams and the ones generated using the predictions. Both models lack significant parts of defects and failed to reproduce at least a similar pattern.



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

Figure 41. Voronoi diagrams generated using coordinates from the experiments in Figure 39.



(a) Original labels (b) Grayscale trained model (c) Full-color trained model

Figure 42. Voronoi diagrams generated using coordinates from the experiments in Figure 40.

Tables 14 and 15 show Voronoi metrics for original labels and defects predicted by SRW and MFT models. SRW models detected significantly fewer defects compared to the ground truth. For example, the G/S SRW model only predicted 17-18% of defects, while the F/C SRW model predicted around 30%. This is reflected in the Voronoi σ: the original in the second case is 0.902, while the G/S and F/C SRW models predicted 0.591 and 0.685, respectively. However, G/S and F/C MFT models demonstrate improvement, predicting 0.874 and 0.829, respectively.

Metrics	Original labels	G/S SRW	G/S MFT	F/C SRW	F/C MFT
№ of inner polygons	29	5	29	10	21
Voronoi σ	0.944	0.691	0.899	0.748	0.757

Table 14. Metrics of Voronoi diagrams displayed in Figure 41.

Metrics	Original labels	G/S SRW	G/S MFT	F/C SRW	F/C MFT
№ of inner polygons	38	7	40	12	30
Voronoi σ	0.902	0.591	0.874	0.685	0.829

Table 15. Metrics of Voronoi diagrams displayed in Figure 42.

Conclusions and Recommendations

After more than 15 experiments with different datasets and activation functions, the best results were obtained when the model was trained for 250 epochs with the dataset containing 72 training images and 17 validation ones. The reader should pay attention that during the most successful computer experiment the author used SiLU activation function, medium-sized YOLOv5m model and height channel AFM images converted to grayscale.

To sum up this paper, the next conclusions can be derived:

- According to the tables 11 and 12 the model managed to detect defects preserving the clusterization metrics which may improve the prediction of EIS response.
- The exact coordinates of the defects are not crucial, the clusterization metrics are not highly impacted by this factor if the amount and the positioning of defects remain relatively similar to the original.
- Compared to the author's scientific research work, the models demonstrated significant improvement in the results as it is shown in the tables 14 and 15:
  - the amount of the detected defects is closer to the ground truth;
  - the Voronoi σ value is preserved relatively close to the original.
- Even though the detection of the defects inside clusters may not be highly precise, Voronoi σ showed that clusterization parameters are preserved.
- Voronoi entropy value is not reliable as a clusterization metric because it mostly indicates the homogeneity and orderliness of the defects' positioning.
- Training with different activation functions produced similar results, the difference was not significant, up to 2%.
- Built-in augmentations module saves time during the model configuration and eliminates the errors which could occur during manual bounding box transformation.

As the goal of this paper was to preserve the clusterization metrics and improve the precision of the model detections inside clusters, it can be stated that the goal was achieved. The model showcased significant progress producing results close to the original ones.

Guidelines for future research

Even though the model shows promising results, there is room for improvement, and it is suggested to consider next statements for the future studies:

- Include real EIS response modeling instead of clusterization metrics.
- Distinguish between different types of defects by shape.
- Develop a CNN network that will be dedicated exactly to the task of defects detection;.
- Prepare bigger datasets with channels different from the height one.
- Train a CNN model on several channels at the same time.
- Apply different activation functions or optimizers for the detection of defects inside clusters.
- Apply different AI methods to detect defects inside clusters.

References

[1] Ramazan Asmatulu and Waseem S. Khan. Chapter 13 - characterization of electrospun nanofibers. In *Synthesis and Applications of Electrospun Nanofibers*, Micro and Nano Technologies, pages 257--281. Elsevier, 2019.

[2] Edward Bormashenko, Mark Frenkel, and Irina Legchenkova. Is the Voronoi Entropy a True Entropy? Comments on "Entropy, Shannon's Measure of Information and Boltzmann's H-Theorem", Entropy 2017, 19, 48. *Entropy*, 21(3), 2019. https://www.mdpi.com/1099-4300/21/3/251.

[3] Edward Bormashenko, Mark Frenkel, Alla Vilk, Irina Legchenkova, Alexander A. Fedorets, Nurken E. Aktaev, Leonid A. Dombrovsky, and Michael Nosonovsky. Characterization of Self-Assembled 2D Patterns with Voronoi Entropy. *Entropy*, 20(12), 2018. https://www.mdpi.com/1099-4300/20/12/956.

[4] Edward Bormashenko, Irina Legchenkova, and Mark Frenkel. Symmetry and Shannon Measure of Ordering: Paradoxes of Voronoi Tessellation. *Entropy*, 21(5), 2019. https://www.mdpi.com/1099-4300/21/5/452.

[5] High-Value Life Science and Material Research and Diagnostics Solutions | Bruker. https://www.bruker.com/.

[6] Bruker Corporation. NanoScope 8.15 Software: User Guide. Channels Interface. https://www.nanophys.kth.se/nanolab/afm/icon/bruker-help/Content/SoftwareGuide/Realtime/Views/ChannelsInterface.htm.

[7] Bruker Corporation. NanoScope 8.15 Software: User Guide. Flatten. https://www.nanophys.kth.se/nanolab/afm/icon/bruker-help/Content/SoftwareGuide/Offline/ModifyCommands/Flatten.htm.

[8] Google Colaboratory platform. https://colab.research.google.com/.

[9] Roza Dastres and Mohsen Soori. Artificial neural network systems. *AZoNanoInternational Journal of Imaging and Robotics (IJIR)*, 2021.

[10] Xiangying Deng, Fang Xiong, Xiayu Li, Bo Xiang, Zheng Li, Xu Wu, Can Guo, Xiaoling Li, Yong Li, Guiyuan Li, Wei Xiong, and Zhaoyang Zeng. Application of atomic force microscopy in cancer research. *Journal of Nanobiotechnology*, 16(1):102, Dec 2018.

[11] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 2022. https://www.sciencedirect.com/science/article/pii/S0925231222008426.

[12] Thomas Ebenhan, Olivier Gheysens, Hendrick G Kruger, Zeevaart Jan Rijn, and Mike M. Sathekge. Antimicrobial peptides: Their role as infection-selective tracers for molecular imaging. *BioMed Research International*, 2014.

[13] Fiveko, hough circle detection. https://fiveko.com/online-tools/hough-circle-detection-demo/.

[14] Hard swish explained | papers with code. https://paperswithcode.com/method/hard-swish.

[15] IBM Cloud Education. Neural networks. https://www.ibm.com/cloud/learn/neural-networks.

[16] Glenn Jocher. Differences between yolov5 models. https://github.com/ultralytics/yolov5/issues/7152.

[17] Glenn Jocher. Yolov5m architecture. https://github.com/ultralytics/yolov5/issues/6094.

[18] Jupyter platform. https://jupyter.org/.

[19] Elena Kiseleva and Larisa Koryashkina. The theory of continuous optimal set partitioning problems as a universal mathematical formalism for constructing the voronoi diagram and its generalizations. i. theoretical foundations. *Cybernetics and systems analysis*, 2015.

[20] Denys Kryvoshei. Automatic algorithms for detection of defects in atomic force microscopy images. *Vilnius University, Mathematics and Informatics Faculty*, 2022.

[21] J Li, J-J Koh, S Liu, Lakshminarayanan R, CS Verma, and RW Beuerman. Membrane active antimicrobial peptides: Translating mechanistic insights to design. *Front. Neurosci.*, 2017.

[22] Library of Congress. Physical astronomy for the mechanistic universe. https://www.loc.gov/collections/finding-our-place-in-the-cosmos-with-carl-sagan/articles-and-essays/modeling-the-cosmos/physical-astronomy-for-the-mechanistic-universe.

[23] Vita Mačinskaitė. Mašininio mokymosi metodai biologinių membranų defektųklasterizacijos analizei. Master's thesis, Vilnius University, Mathematics and Informatics Faculty, 2022.

[24] Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127--147, 1943.

[25] Mish explained | papers with code. https://paperswithcode.com/method/mish.

[26] NanoAndMore GmbH. What is atomic force microscopy (afm). https://nanoandmore.com/what-is-atomic-force-microscopy.

[27] Vytenis Navalinskas. Defektu atpažinimas biologinese membranose. Master's thesis, Vilnius University, Mathematics and Informatics Faculty, 2020.

[28] Upesh Nepal and Hossein Eslamiat. Comparing yolov3, yolov4 and yolov5 for autonomous landing spot detection in faulty uavs. *Sensors*, 22(2), 2022. https://www.mdpi.com/1424-8220/22/2/464.

[29] Python Numpy library. https://numpy.org/.

[30] Park Systems. Using true non-contact mode afm for the imaging of plasmids in liquids. *AZoNano*, May 2022.

[31] Python PyTorch library. https://pytorch.org/.

[32] Tomas Raila, Filipas Ambrulevičius, Tadas Penkauskas, Marija Jankunec, Tadas Meškauskas, David J. Vanderah, and Gintaras Valincius. Clusters of protein pores in phospholipid bilayer membranes can be identified and characterized by electrochemical impedance spectroscopy. *Electrochimica Acta*, 364:137179, 2020.

[33] Tomas Raila, Tadas Penkauskas, Filipas Ambrulevičius, Marija Jankunec, Tadas Meškauskas, and Gintaras Valinčius. Ai-based atomic force microscopy image analysis allows to predict electrochemical impedance spectra of defects in tethered bilayer membranes, January 2022. https://www.nature.com/articles/s41598-022-04853-4.

[34] Connie Rye, Robert Wise, Vladimir Jurukovski, Jean DeSaix, Jung Choi, and Yael Avissar. Biology, October 2018. https://openstax.org/books/biology/pages/5-1-components-and-structure.

[35] SciPy. https://scipy.org/.

[36] Shapely. https://shapely.readthedocs.io/en/stable/index.html.

[37] Jesslyn Shields. Here's how plant and animal cells are different, March 2021. https://science.howstuffworks.com/life/cellular-microscopic/plant-cells-animal-cells.htm.

[38] Silu explained | papers with code. https://paperswithcode.com/method/silu.

[39] Swish explained | papers with code. https://paperswithcode.com/method/swish.

[40] Saeideh Tasharrofi, Sedigheh Sadegh Hassani, and Zahra Sobat. *Encyclopedia of Nanotechnology*. 05 2015.

[41] The University of Queensland. How do neurons work? https://qbi.uq.edu.au/brain-basics/brain/brain-physiology/how-do-neurons-work.

[42] Igor Vilghelm. Automatizuoti biologiniu membranu defektu atpažinimo algoritmai atominės jegos mikroskopo nuotraukose. Master's thesis, Vilnius University, Mathematics and Informatics Faculty, 2021.

[43] Weights and Biases platform. https://wandb.ai/.

[44] Wikimedia Commons contributors. Euclidean voronoi diagram. https://commons.wikimedia.org/w/index.php?title=File:Euclidean_Voronoi_diagram.svg&oldid=486001345.

[45] Wikimedia Commons contributors. Manhattan voronoi diagram. https://commons.wikimedia.org/w/index.php?title=File:Manhattan_Voronoi_Diagram.svg&oldid=462362754.

Appendices

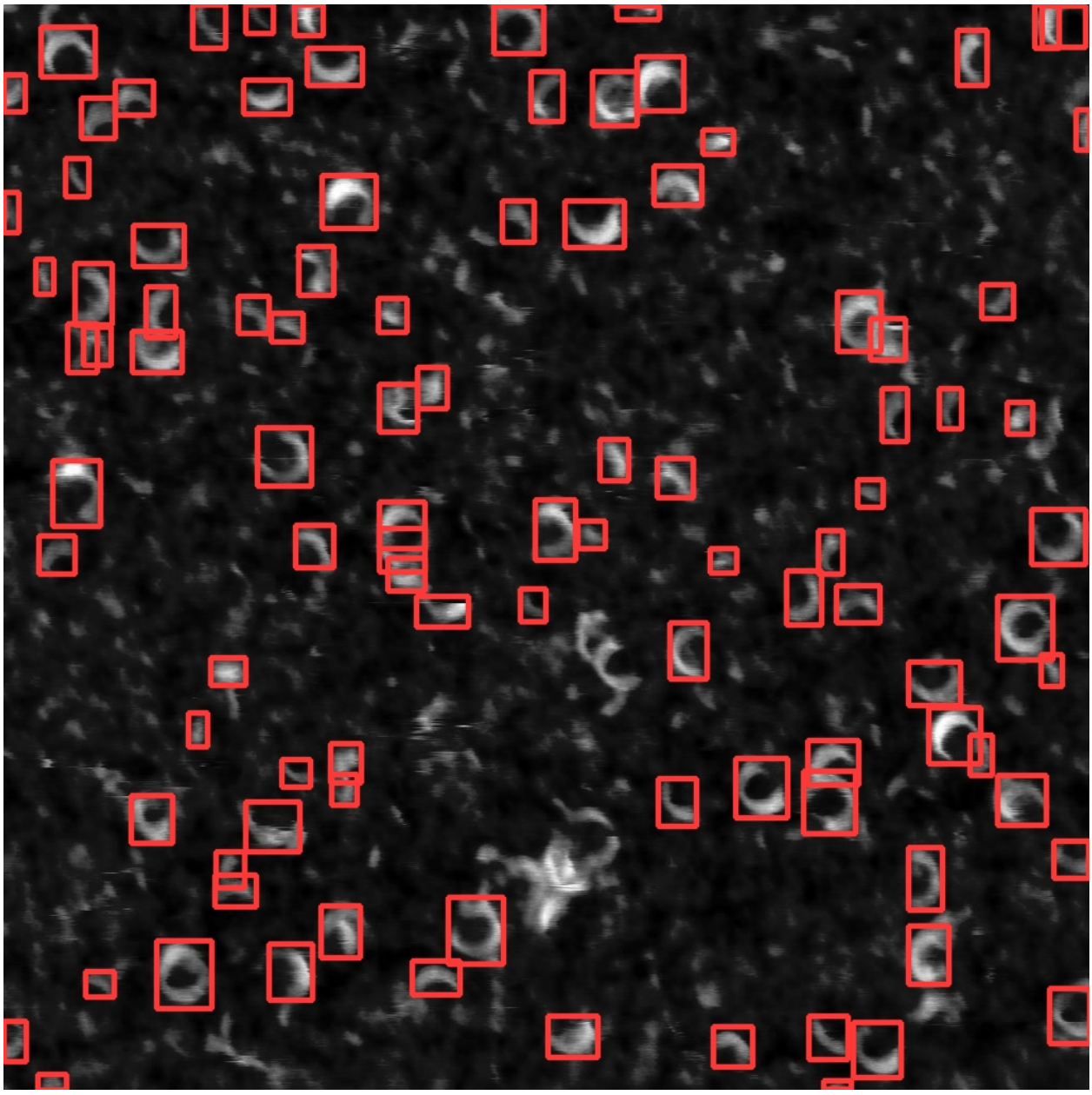


Figure 43. Detections of the model trained using *single_defects_gray* dataset.

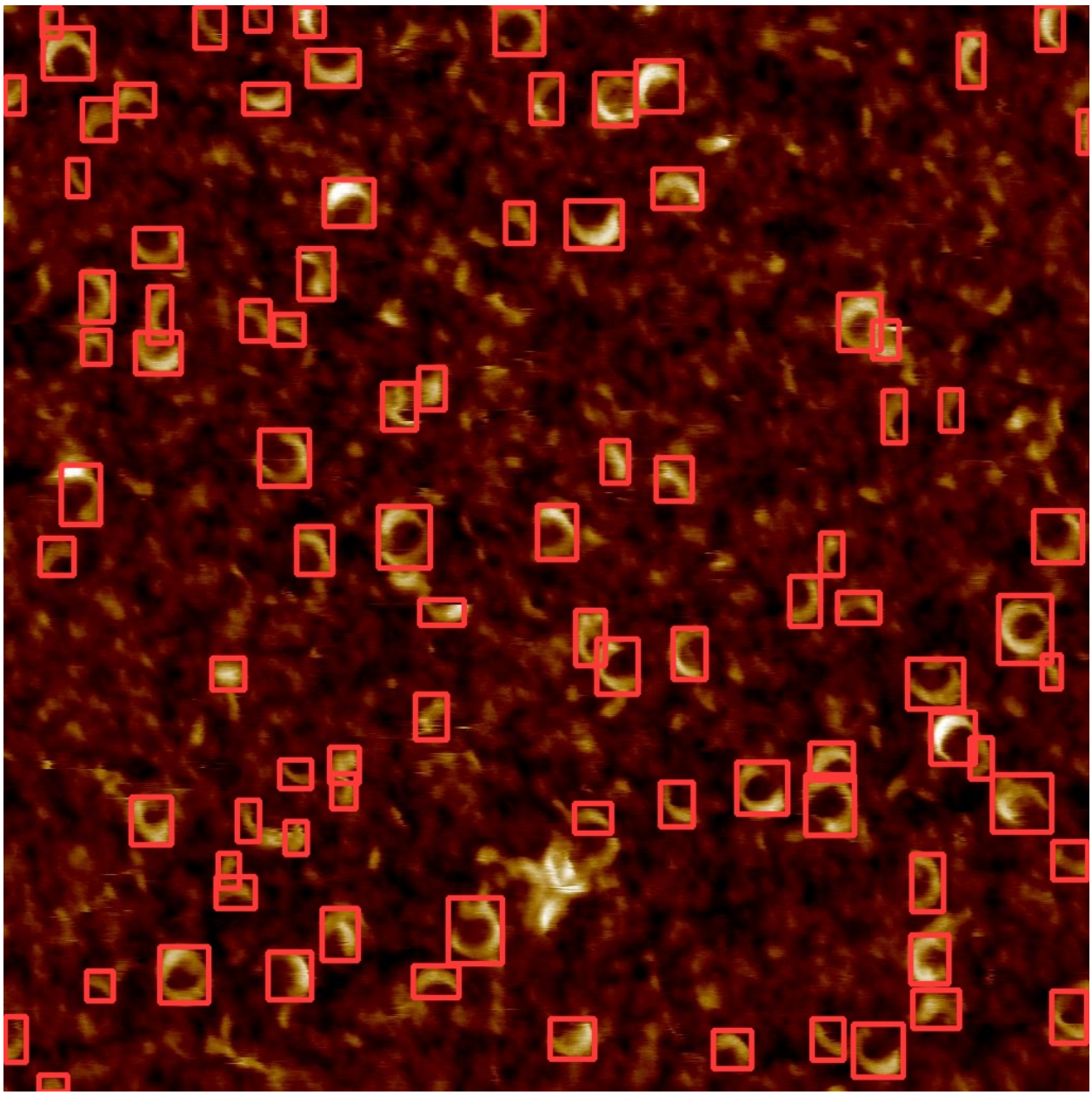


Figure 44. Detections of the model trained using *single_defects_raw* dataset.

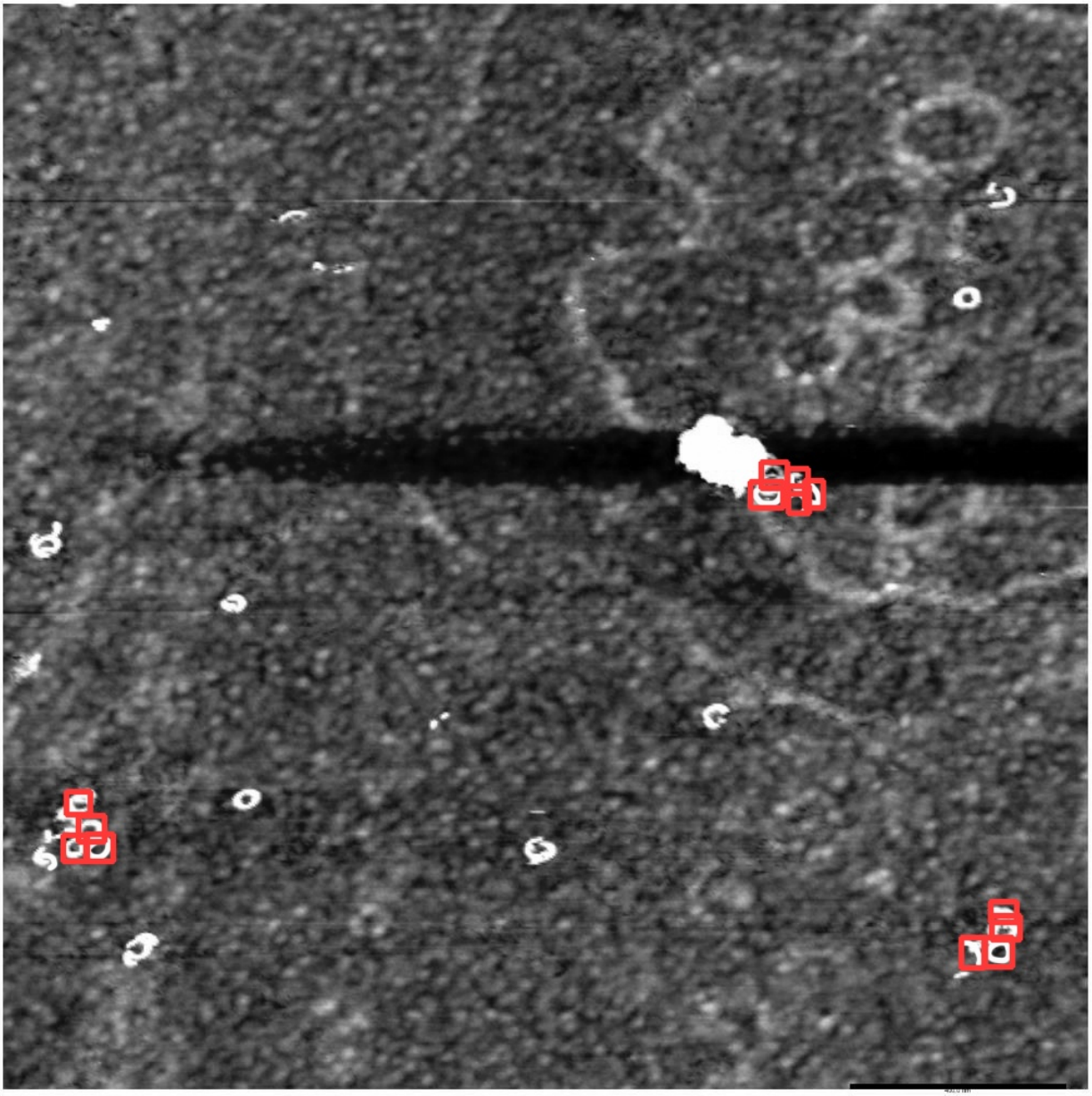


Figure 45. Detections of the model trained using *defects_in_clusters_height_gray_29* dataset. Test 1.

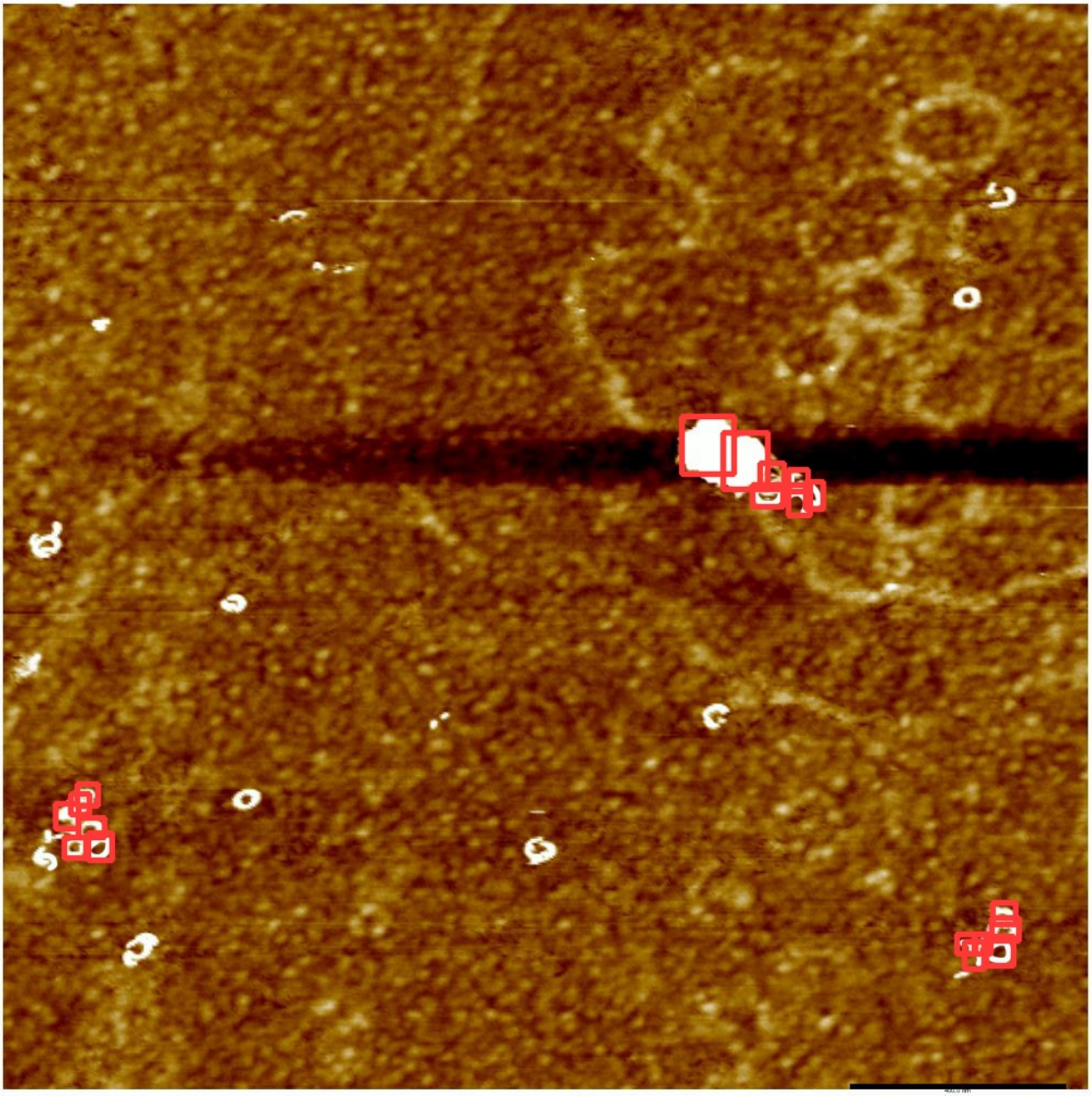


Figure 46. Detections of the model trained using *defects_in_clusters_height_raw_29* dataset. Test 1.

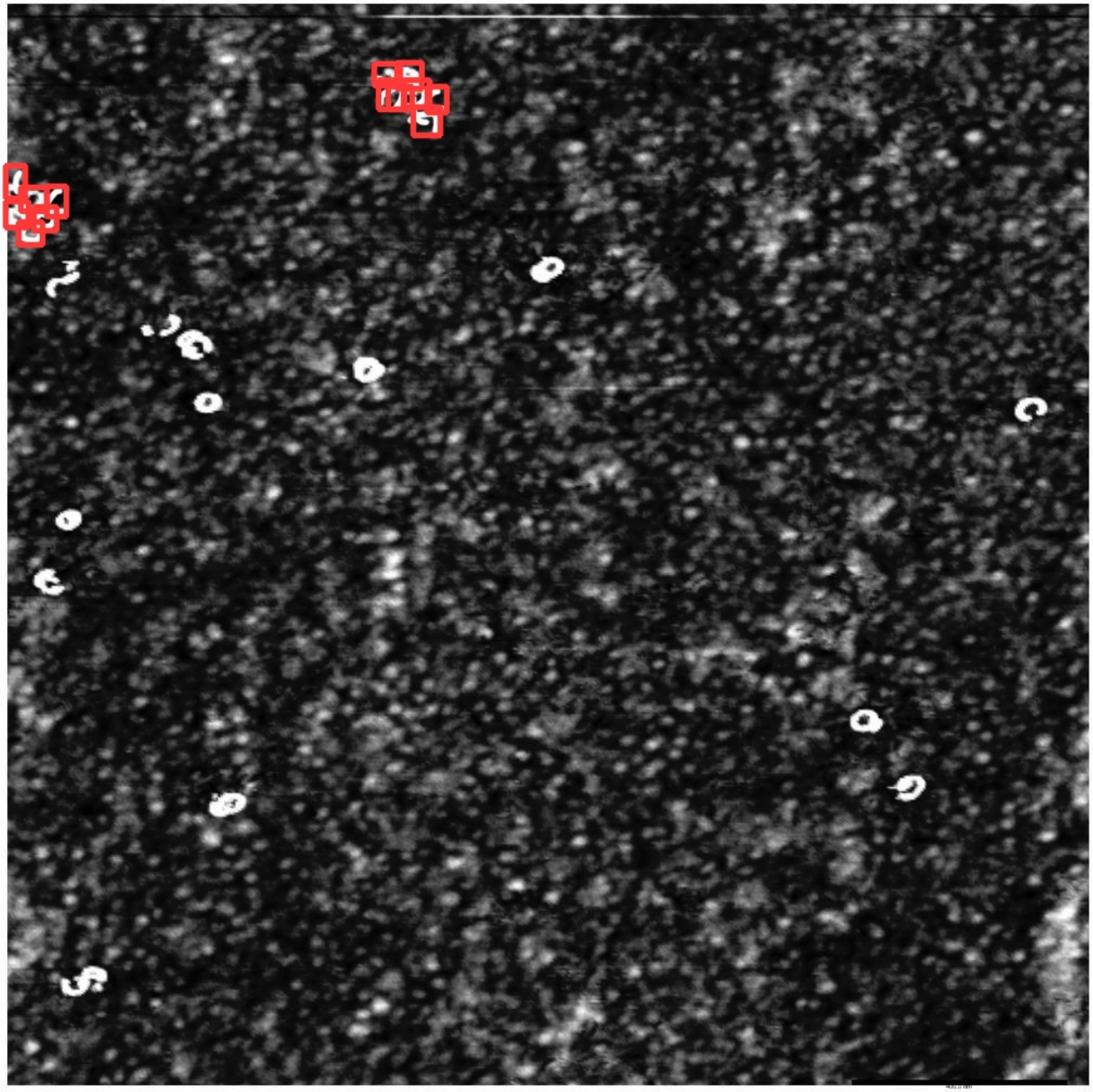


Figure 47. Detections of the model trained using *defects_in_clusters_height_gray_29* dataset. Test 2.

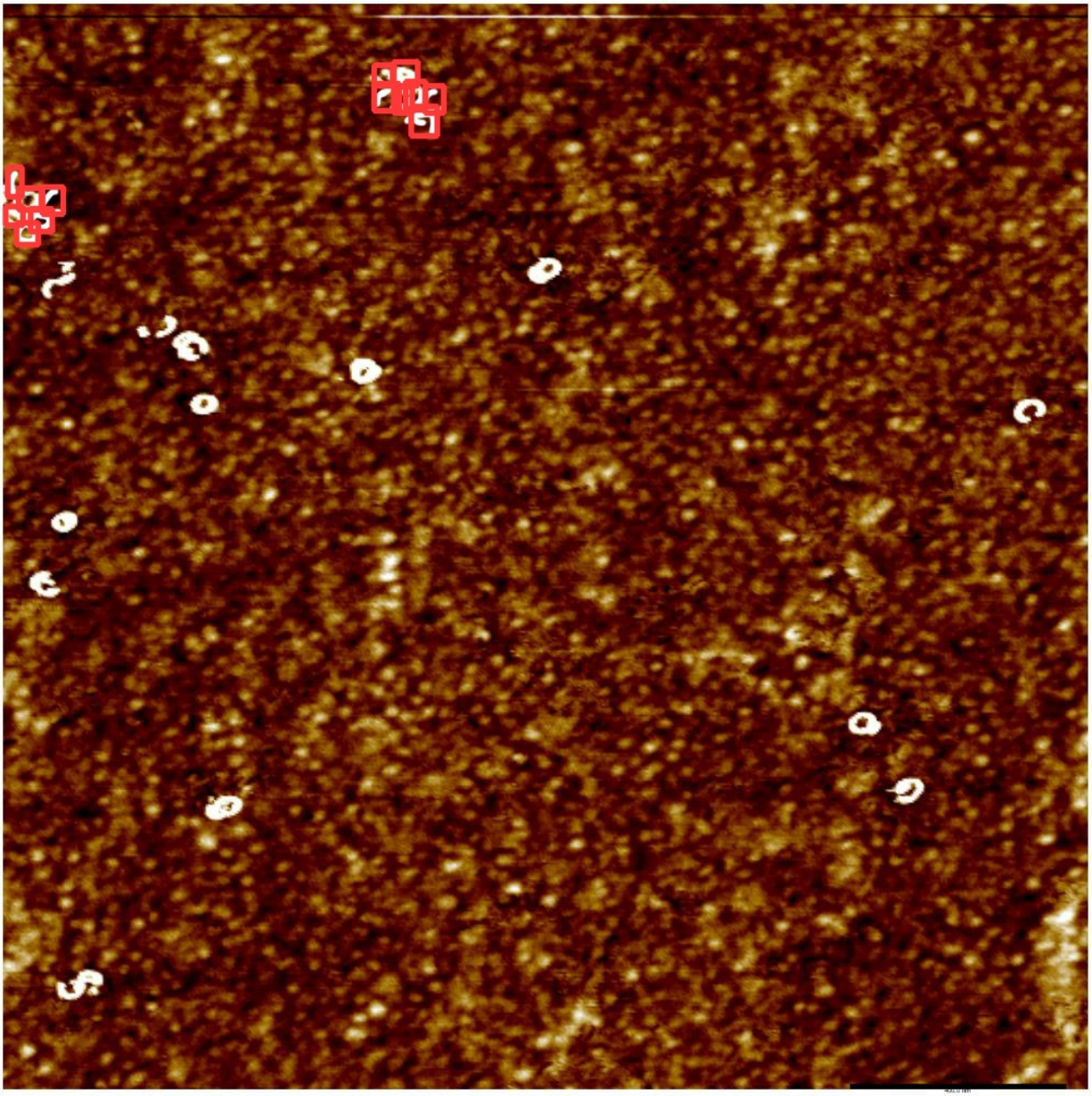


Figure 48. Detections of the model trained using *defects_in_clusters_height_raw_29* dataset. Test 2.

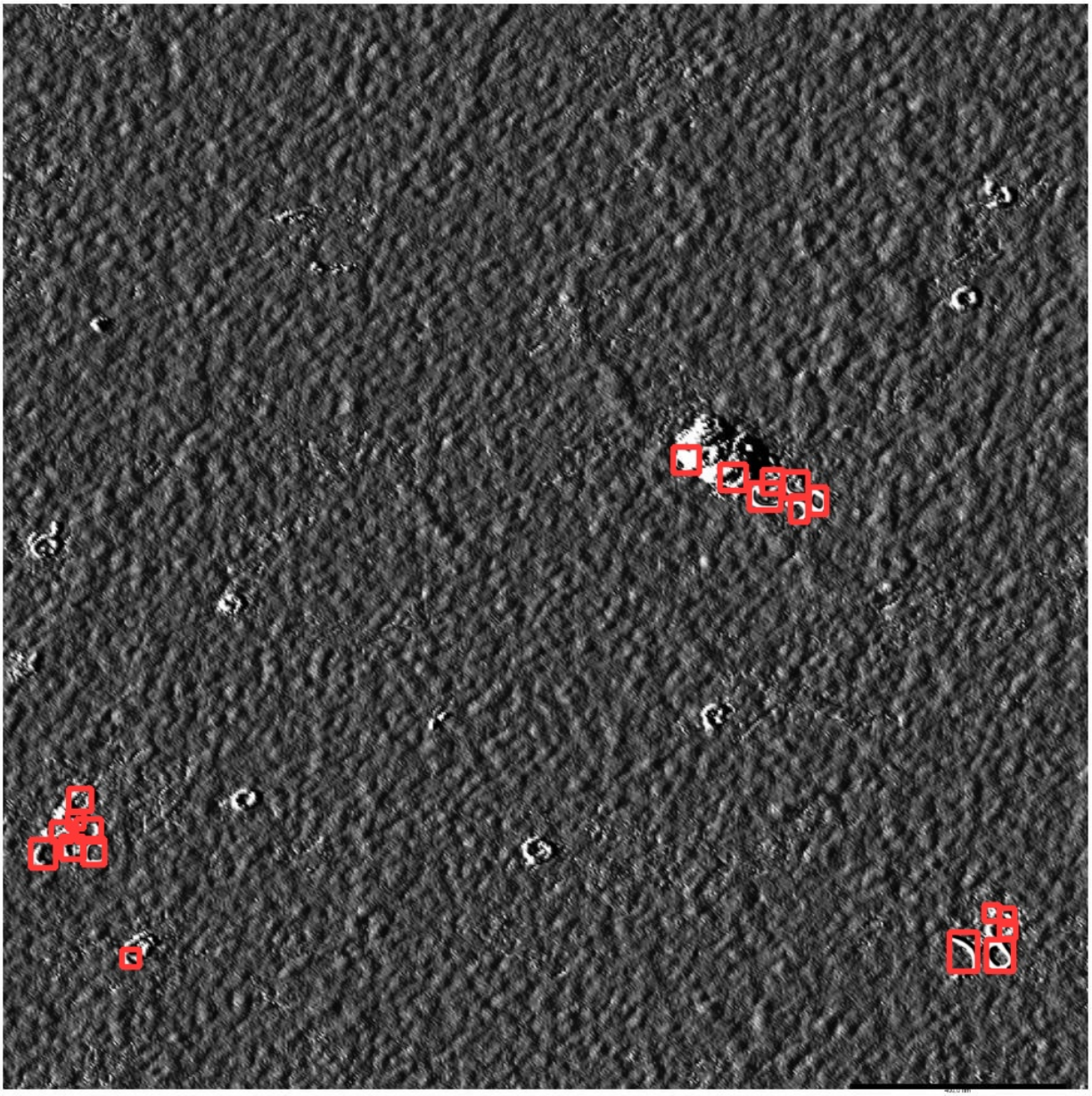


Figure 49. Detections of the model trained using *defects_in_clusters_amplitude_gray_29* dataset. Test 1.

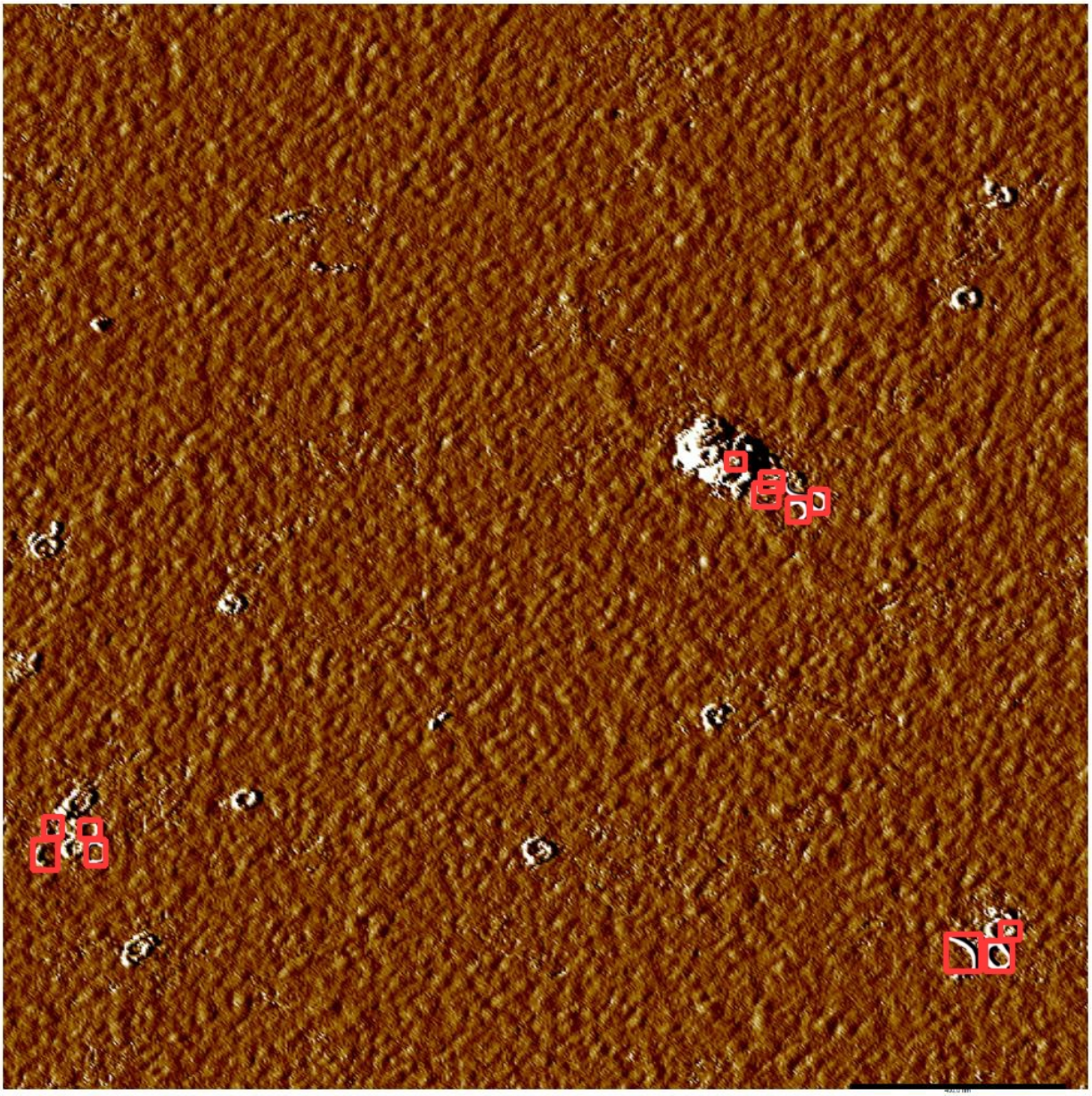


Figure 50. Detections of the model trained using *defects_in_clusters_amplitude_raw_29* dataset. Test 1.

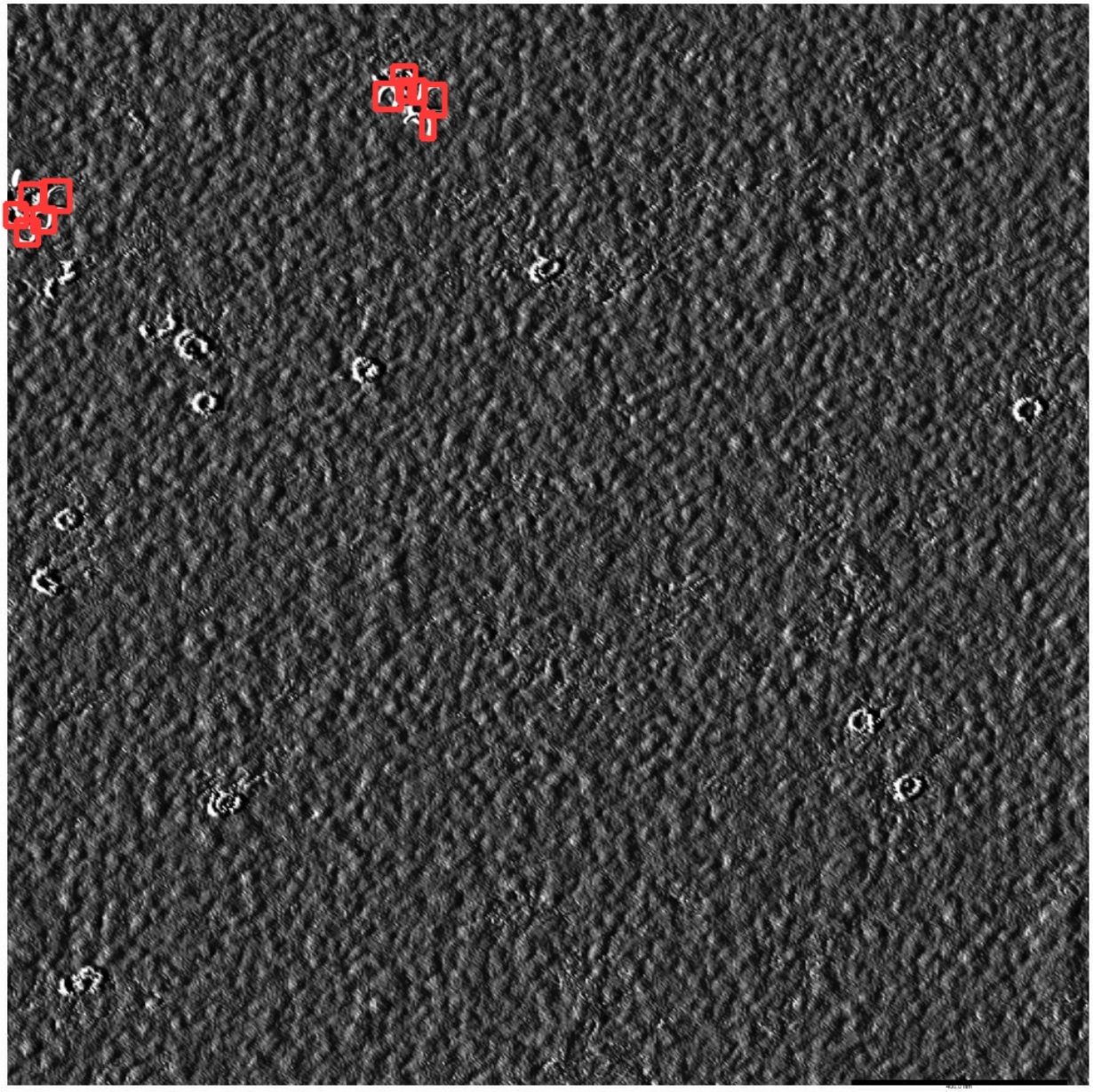


Figure 51. Detections of the model trained using *defects_in_clusters_amplitude_gray_29* dataset. Test 2.

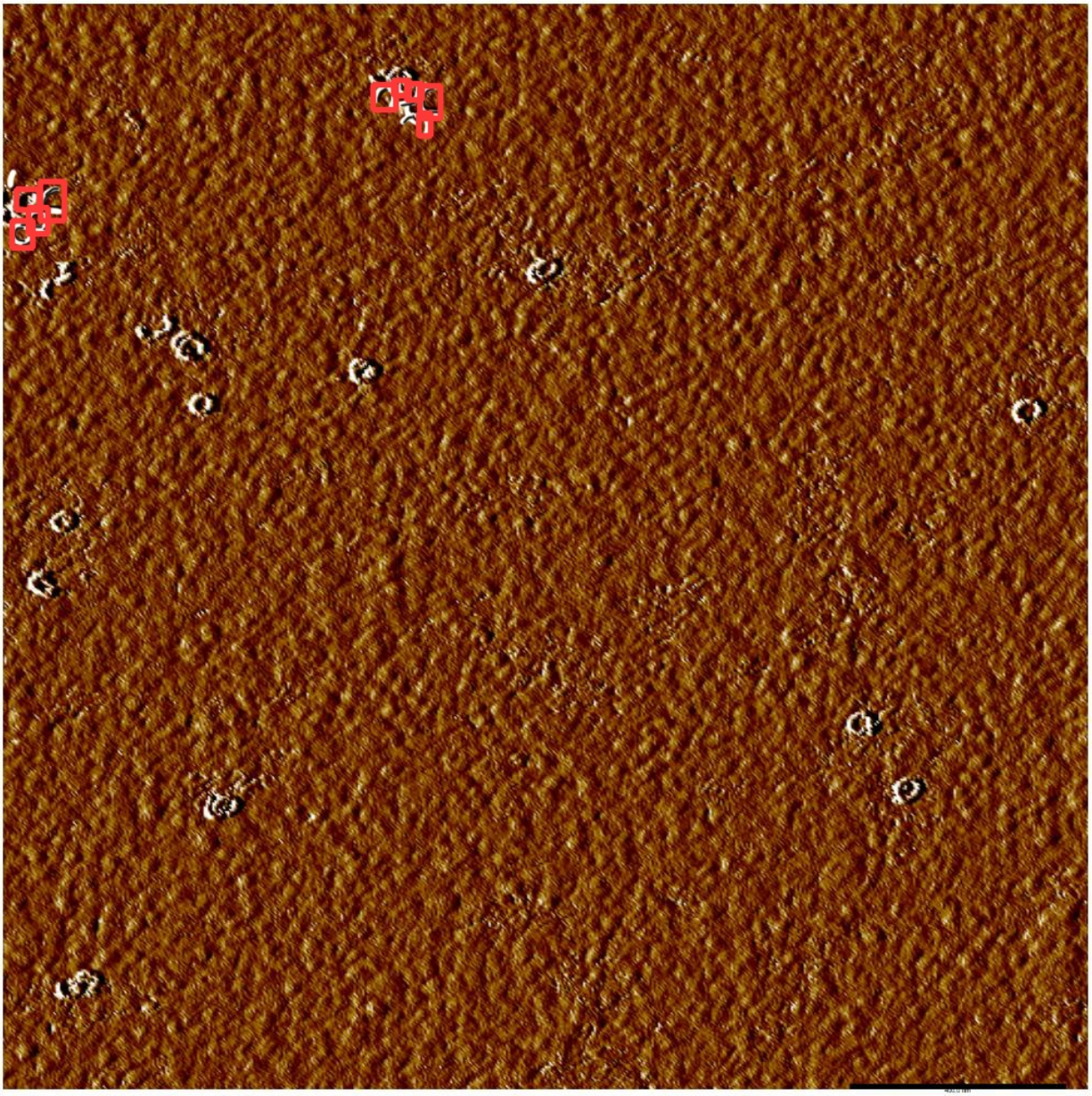


Figure 52. Detections of the model trained using *defects_in_clusters_amplitude_raw_29* dataset. Test 2.

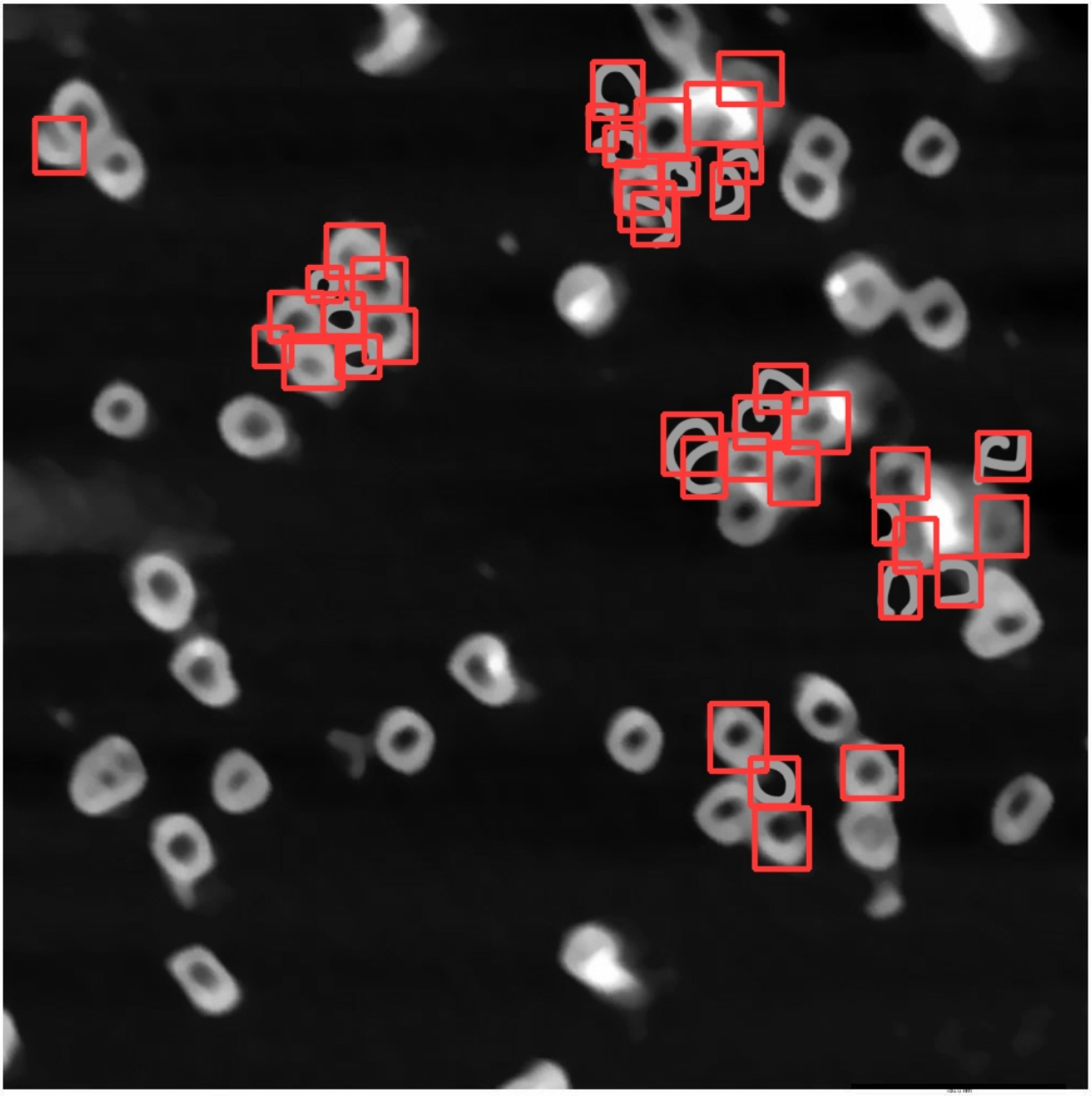


Figure 53. Detections of the model trained on *defects_in_clusters_height_gray* dataset. Test 1.

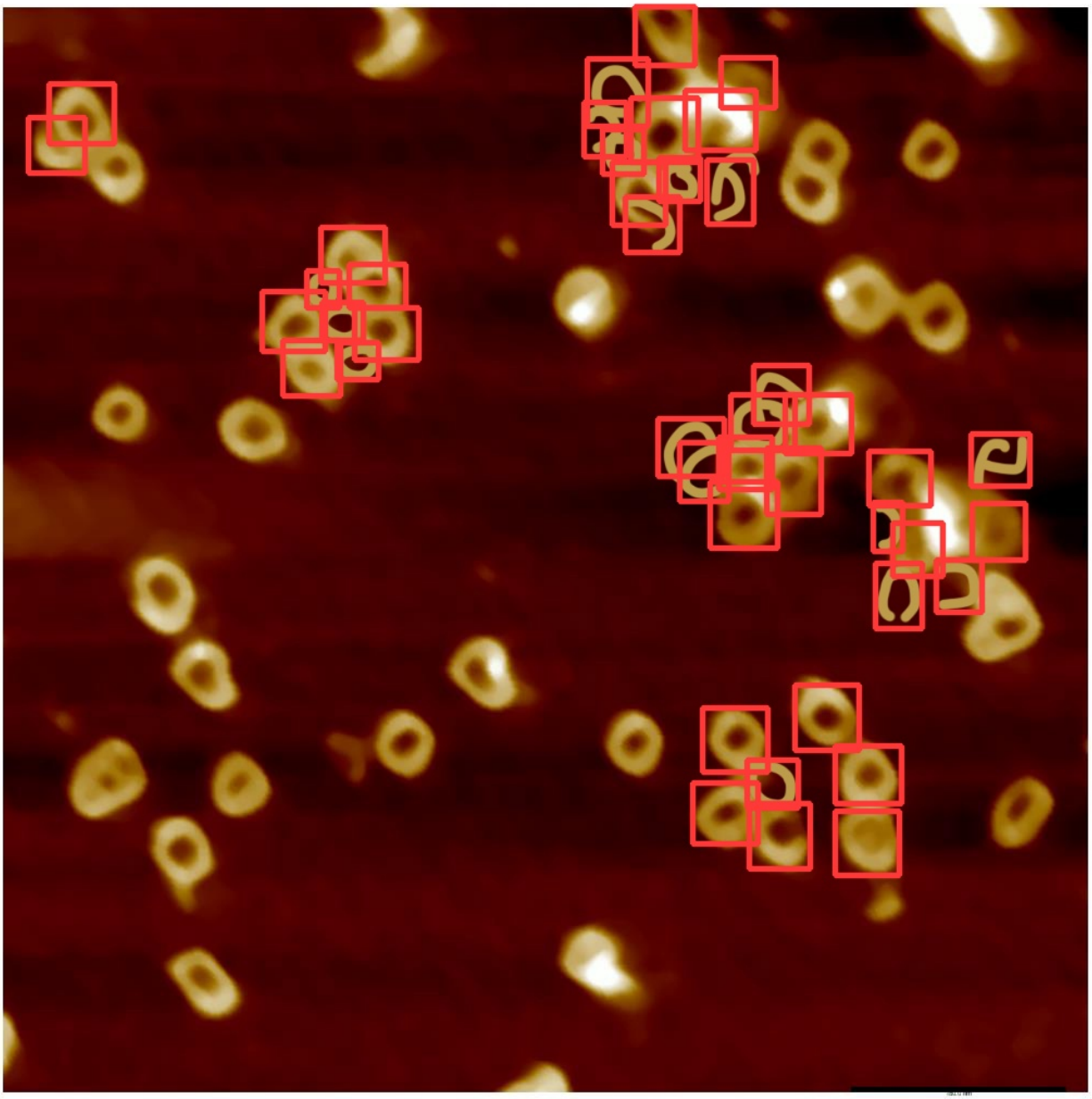


Figure 54. Detections of the model trained on *defects_in_clusters_height_raw* dataset. Test 1.

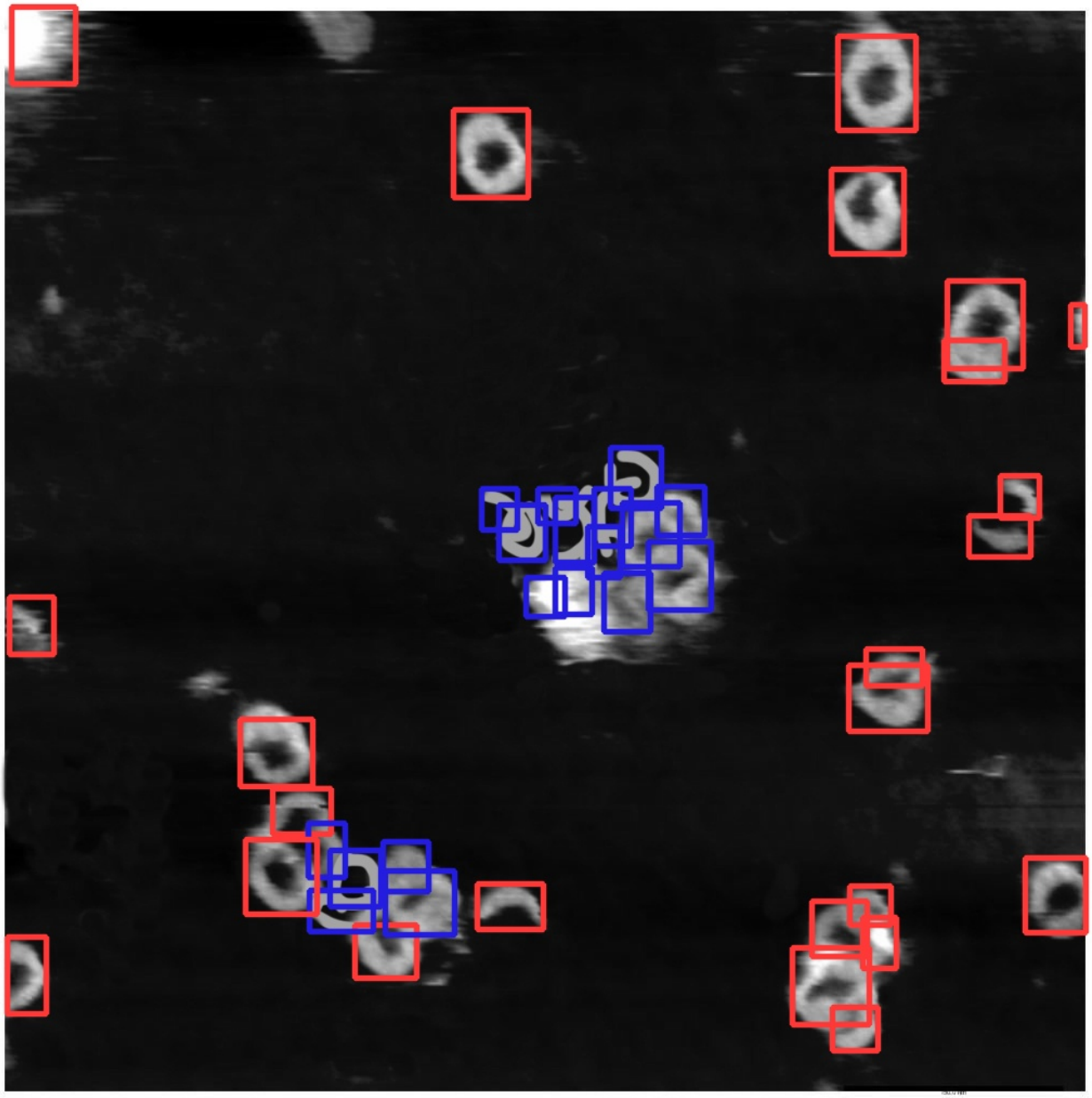


Figure 55. Detections of the model trained using *single_and_inside_clusters_gray* dataset. Test 1.

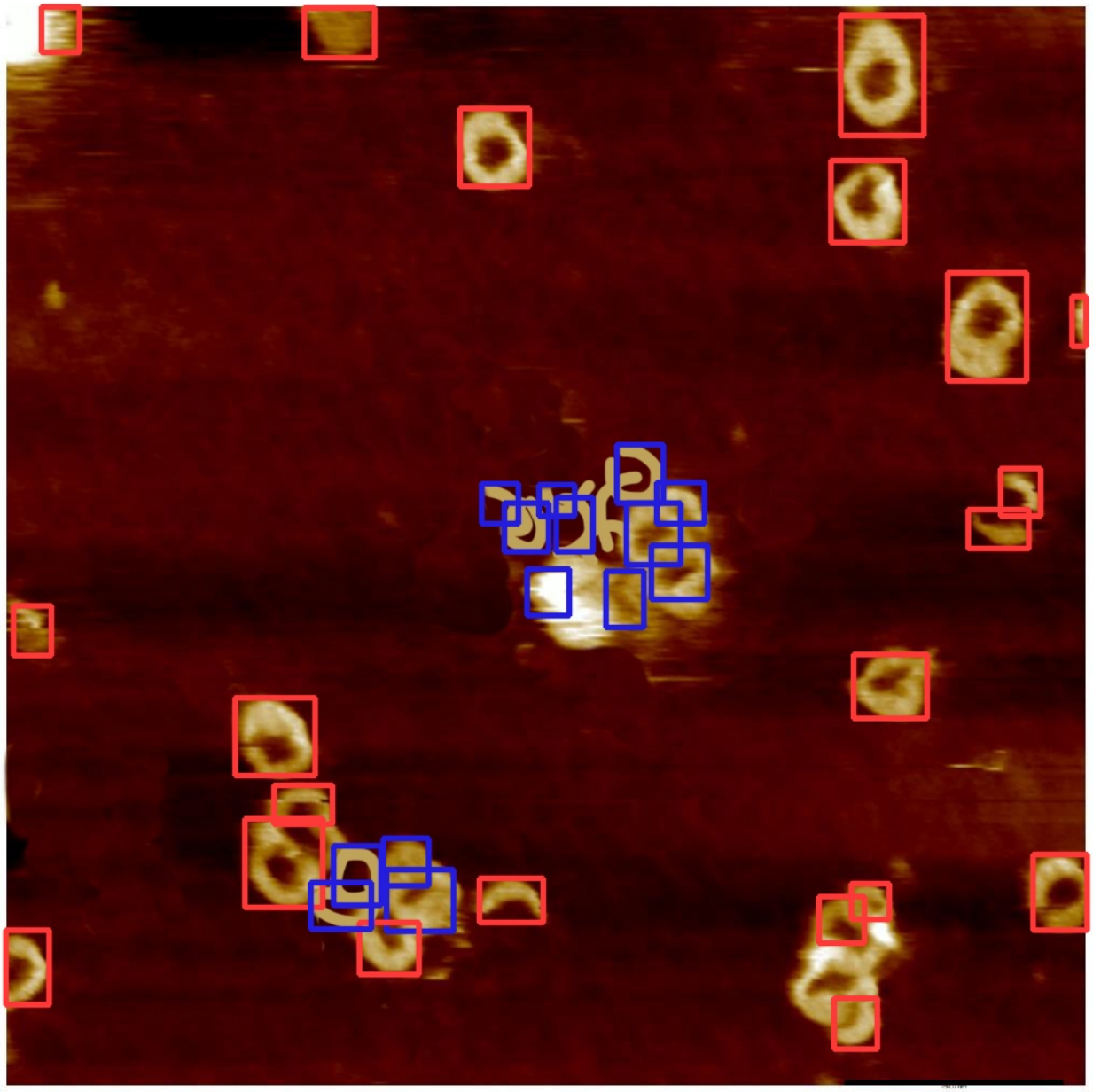


Figure 56. Detections of the model trained using *single_and_inside_clusters_raw* dataset. Test 1.

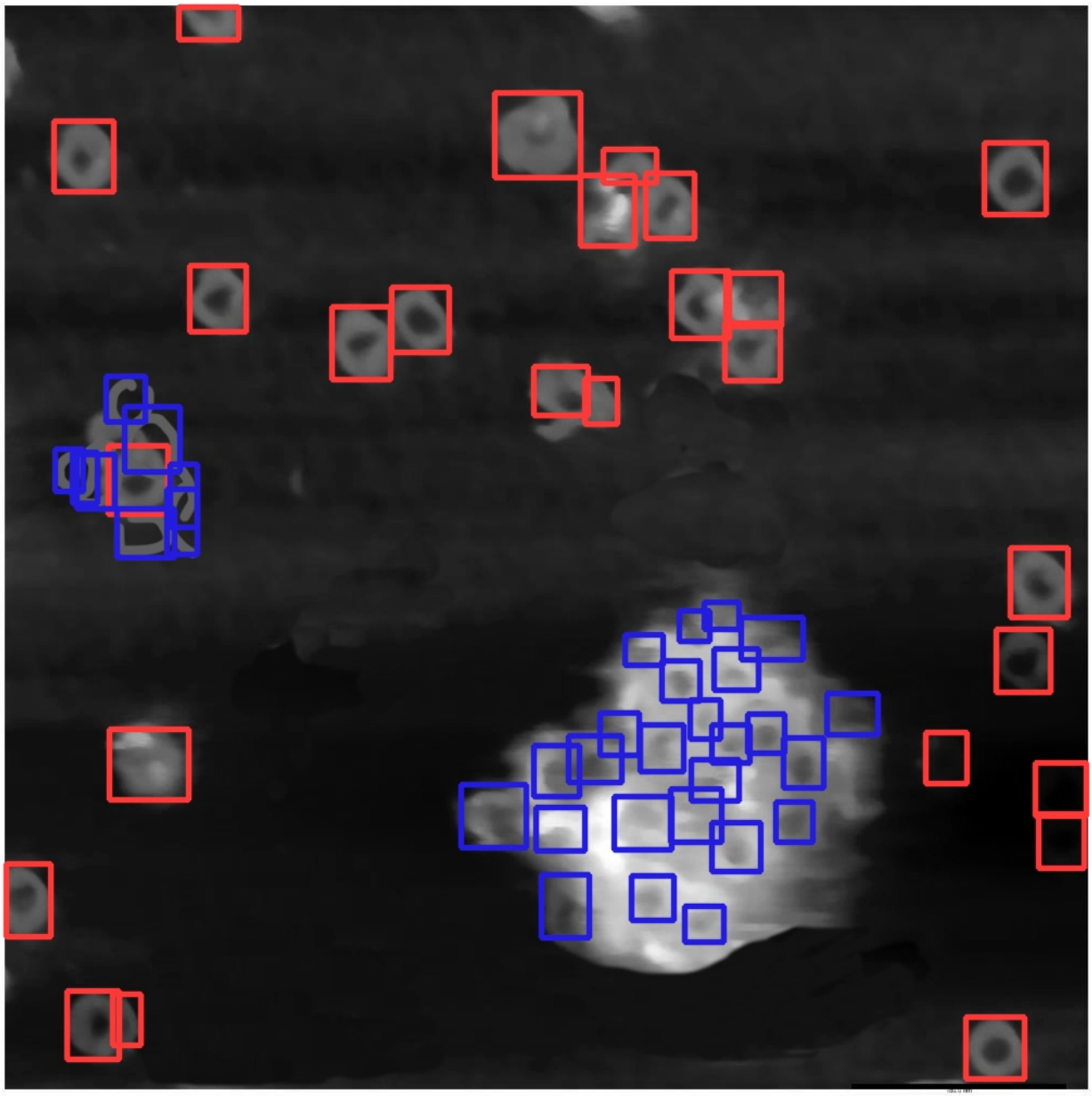


Figure 57. Detections of the model trained using *single_and_inside_clusters_gray* dataset. Test 2.

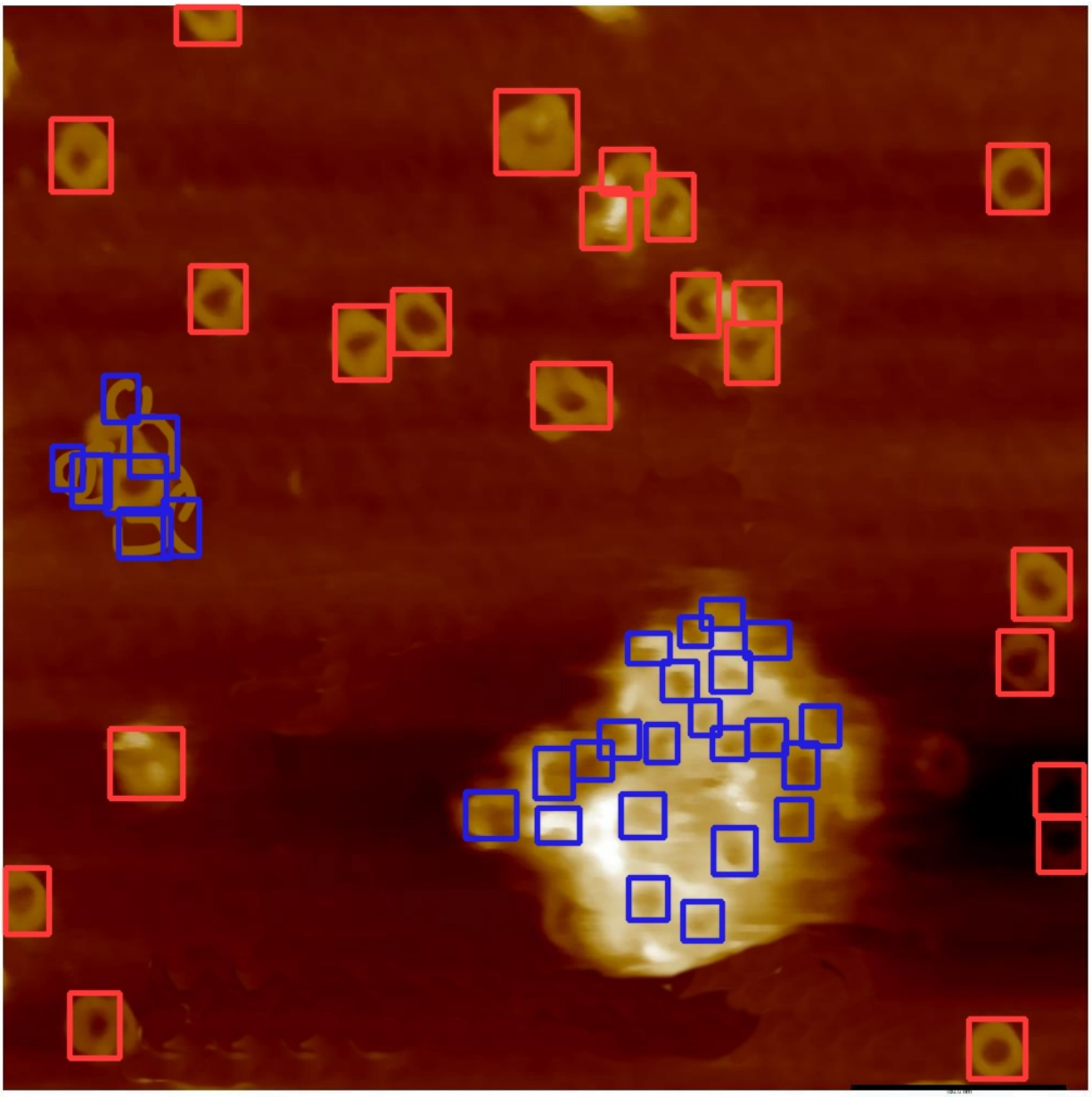


Figure 58. Detections of the model trained using *single_and_inside_clusters_raw* dataset. Test 2.

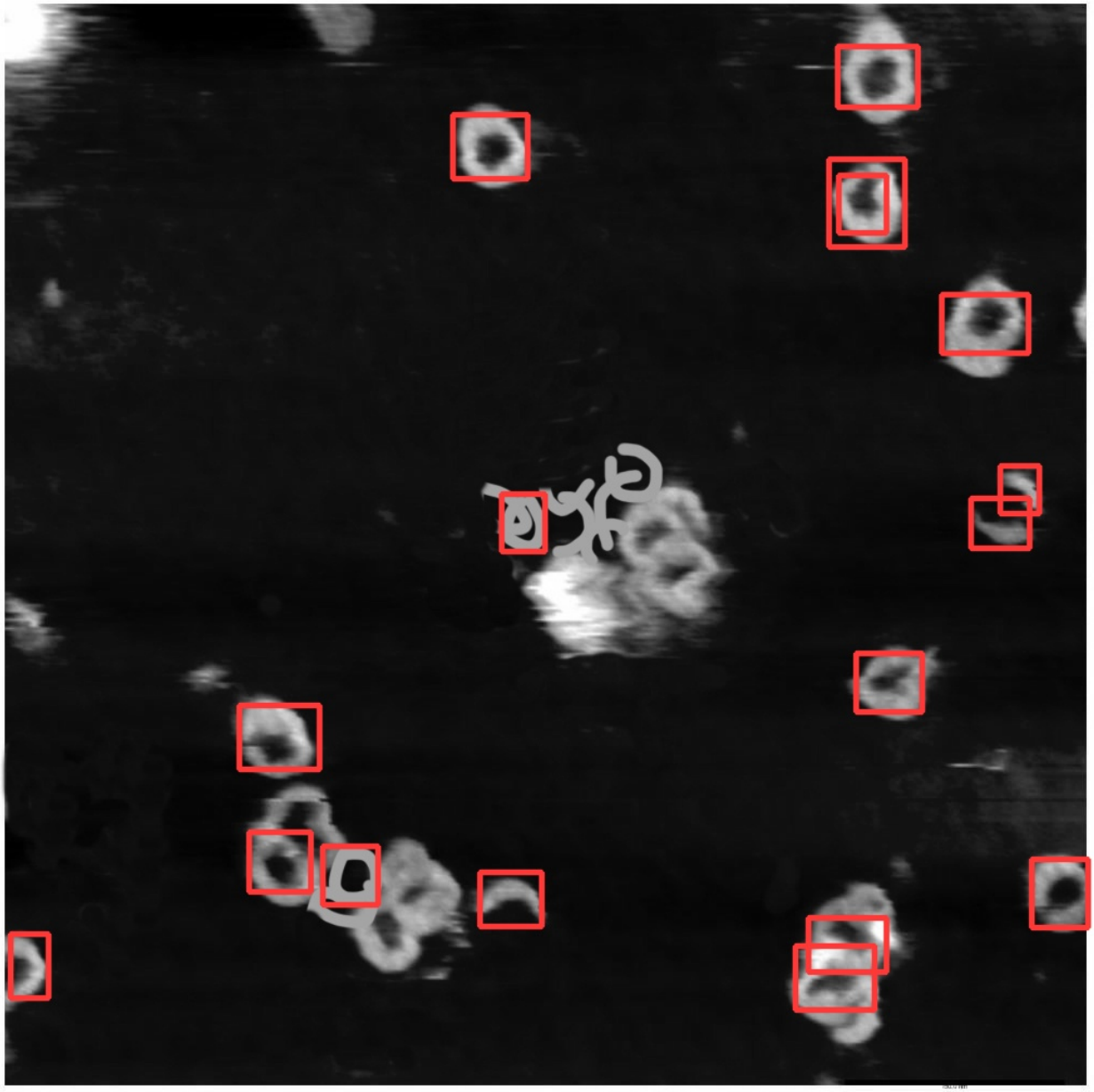


Figure 59. Detections of the model trained during SRW. Test 1.

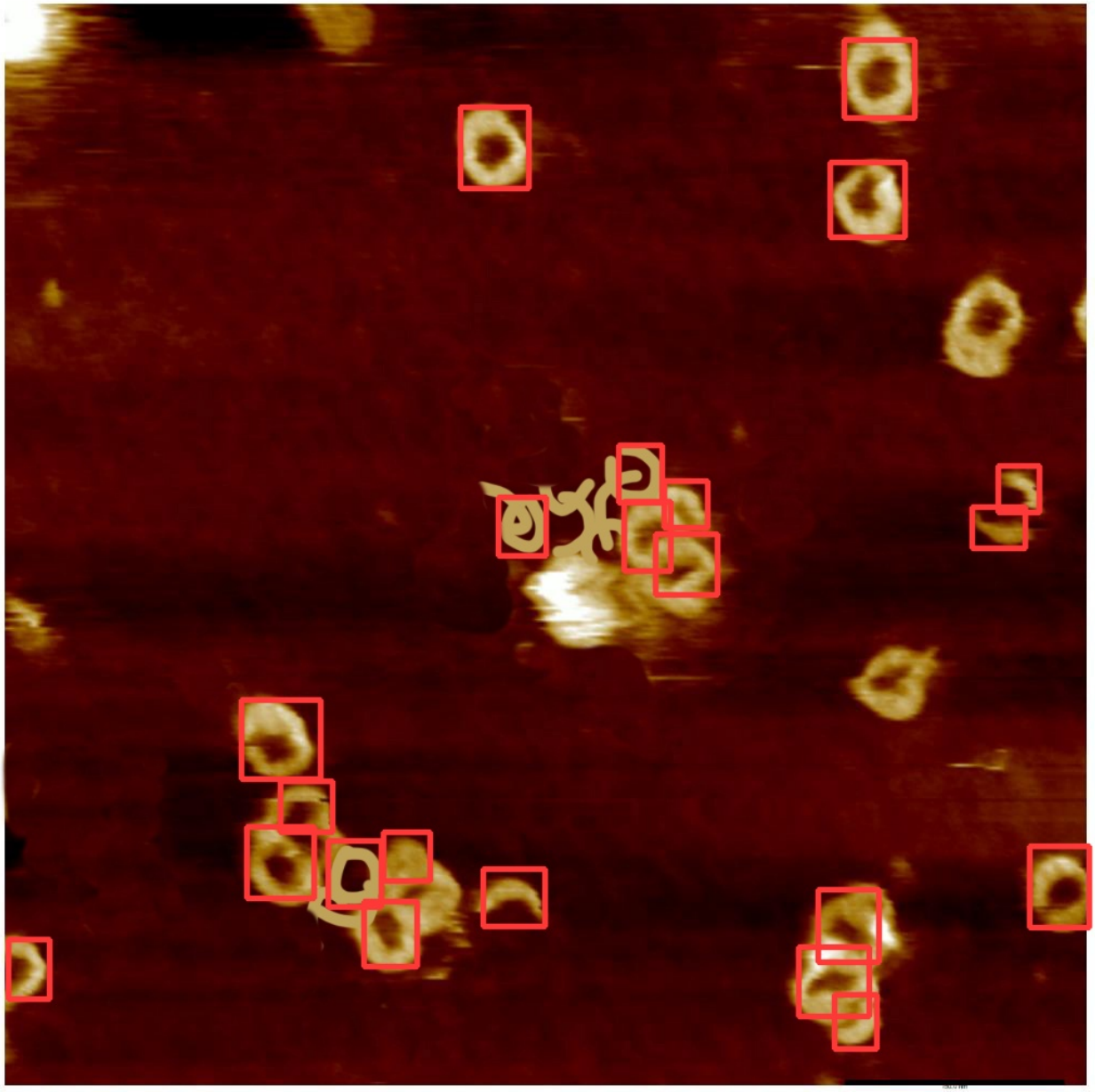


Figure 60. Detections of the model trained during SRW. Test 1.

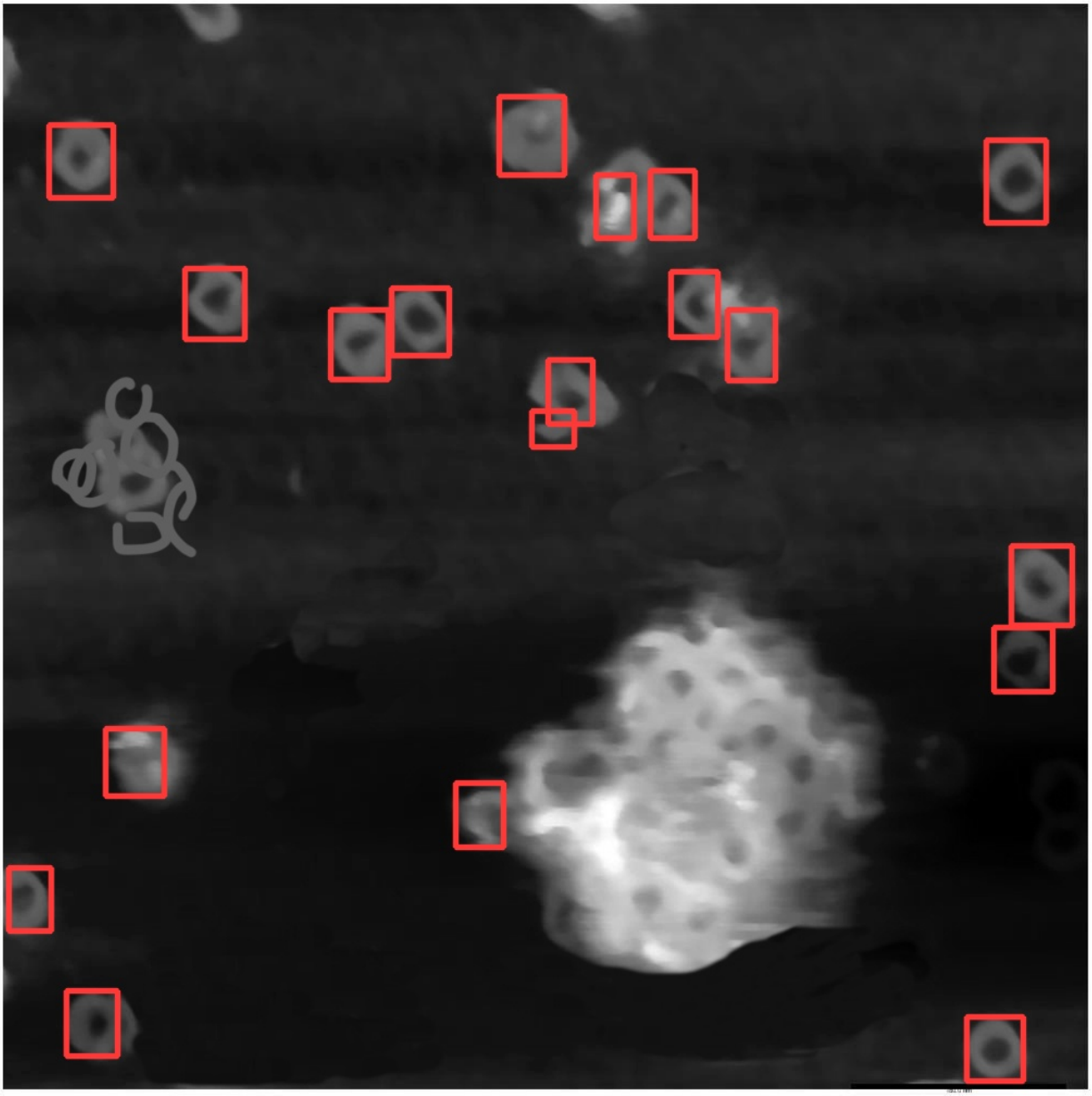


Figure 61. Detections of the model trained during SRW. Test 2.

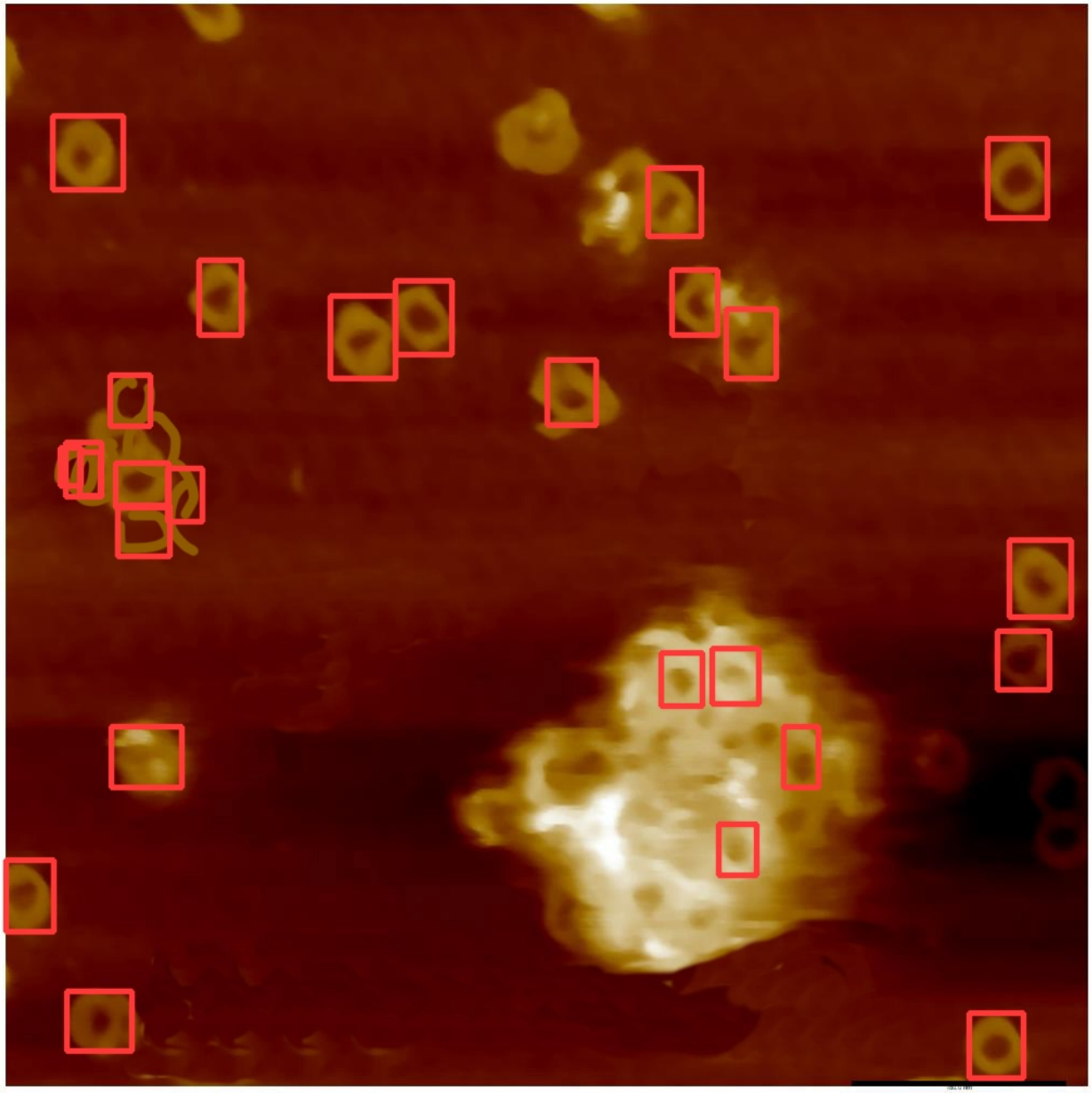


Figure 62. Detections of the model trained during SRW. Test 2.