VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ
DEPARTMENT OF BIOMEDICAL ENGINEERING

DETEKCE BIOLOGICKÝCH STRUKTUR VE SNÍMCÍCH Z TEM MIKROSKOPU
DETECTION OF BIOLOGICAL STRUCTURES IN TEM MICROSCOPE IMAGES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Martin Cikánek**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Potočňák**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Biomedicínské a ekologické inženýrství**
Ústav biomedicínského inženýrství

Student: Bc. Martin Cikánek ***ID:*** 170788
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Detekce biologických struktur ve snímcích z TEM mikroskopu

POKYNY PRO VYPRACOVÁNÍ:

1) Proveďte literární rešerši metod rozpoznávaní struktur v obraze, včetně průzkumu možností aplikace shlukové analýzy na nalezené struktury. Prostudujte určující parametry akvizice obrazu pomoci transverzálního elektronového mikroskopu (TEM). 2) Navrhněte teoreticky nejvhodnější postup optimální detekce biologických struktur v TEM obraze a jejich následné děleni do shluků. 3) Navrhněte způsob vyhodnocení výsledků a jejich závislost na akvizičních parametrech elektronového mikroskopu. 4) Proveďte nasnímání obrazů ve spolupráci s firmou Thermo Fisher Scientific. 5) V prostředí Matlab/Python vytvořte vámi zvolený algoritmus, řešící danu problematiku. Otestujte metody na reálných obrazech a výsledky porovnejte s dostupnými metodami. 6) Proveďte diskusi získaných výsledků a zhodnoťte účinnost a využitelnost metod. Diplomová práce je tvořena ve spolupráci s firmou Thermo Fisher Scientific.

DOPORUČENÁ LITERATURA:

[1] KARLIK M., Úvod do transmisní elektronové mirkoskopie. Praha ČVUT, 2011. ISBN 978-80-01-04729-3.

[2] JAN, J. Digital Signal Filtering, Analysis and Restoration. volume 44. volume 44. London: The Institution of Electrical Engineers, 2000. 407 s. ISBN: 0-85296-760- 8.

Termín zadání: 4.2.2019 ***Termín odevzdání:*** 17.5.2019

Vedoucí práce: Ing. Tomáš Potočňák
Konzultant:

prof. Ing. Ivo Provazník, Ph.D.
předseda oborové rady

ABSTRAKT

Cílem první části této diplomové práce je vysvětlit teoretické základy transmisní elektronové mikroskopie and zmínit fundamentální části transmisních elektronových mikroskopů. Další část této práce je zaměřena na možné metody segmentace obrazu, využití neuronových sítí při detekci objektů v obraze a na následné shlukování výsledků. Teoretická část práce je zakončena vysvětlením některých již publikovaných metod automatické detekce biologických struktur v obrazech z mikroskopu a teoretickým návrhem algoritmu, který bude následně vypracován. Na začátku praktické části je vysvětlen postup trénování neuronových sítí za účelem automatické detekce biologických struktur v obraze. Poté následuje zhodnocení výsledků dosažených těmito sítěmi. Následně jsou na tyto výsledky aplikovány metody shlukové analýzy, jejichž výsledky jsou porovnávány mezi sebou a taktéž s výsledky dosaženými již publikovanými metodami.

KLÍČOVÁ SLOVA

Transmisní elektronová mikroskopie, segmentace obrazu, metody shlukování, neuronové sítě

ABSTRACT

The aim of the first part of this thesis is to explain the theoretical basis of transmission electron microscopy and to mention fundamental parts of transmission electron microscopes. The next part of this work is focused on possible methods of image segmentation, the use of neural networks in the detection of objects in an image and the subsequent clustering of results. The theoretical part of the thesis is concluded with an explanation of some already published methods of automatic detection of biological structures in microscopic images and theoretical design of the algorithm, which will be subsequently developed. The process of training neural networks in order to automatically detect biological structures in an image is described at the beginning of the practical part. This is followed by an evaluation of the results achieved by these networks. Subsequently, cluster analysis methods are applied to these results, the products of which are compared with each other and also with the results obtained by already published methods.

KEYWORDS

Transmission electron microscopy, image segmentation, cluster analysis, neural networks

CIKÁNEK, Martin. *Detekce biologických struktur ve snímcích z TEM mikroskopu.* Brno, 2019. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/118376. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav biomedicínského inženýrství. Vedoucí práce Tomáš Potočňák.

PROHLÁŠENÍ

Prohlašuji, že svoji diplomovou práci na téma Detekce biologických struktur ve snímcích z TEM mikroskopu jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....
(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Tomáši Potočňákovi za účinnou metodickou, pedagogickou a odbornou pomoc a cenné rady při zpracování mé diplomové práce.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Electron microscopy is a scientific method using a beam of electrons for studying extremely small details of monitored objects at high resolution. This method of microscopy enabled us to reach a much higher resolution than standard light microscopes. The basic differentiation of methods divides them into two fundamental groups, scanning electron microscopy and transmission electron microscopy.

Transmission electron microscopy (TEM) is a modern method used for characterizing the morphology, crystalline structures or elemental information of the monitored sample. The transmission electron microscope generates a beam of electrons that passes through an ultra-thin sample. The electrons hit the examined sample and interact with it, which may result in a change of their trajectory or even their complete elimination. The electrons that pass through the sample are then caught on a fluorescent screen or a camera with scintillator used as an imaging system. Thanks to this, an image of the sample is created and can be subsequently analysed. Biological samples can also be visualized using transmission electron microscopy. They are kept in a suspension that is mounted on a grid and preserved at very low temperatures using liquid forms of nitrogen or ethylene so they can withstand the high-vacuum environment of the microscope. This method is used during cancer research, virology or nanotechnology.

The focus of this thesis is to design an algorithm that will detect biological structures within the image produced by TEM. These images are heavily affected by noise and can contain a large number of biological structures. This process is vital for further analysis of the image and the biological structures. Further analysis can include single particle analysis or cryo-tomography – methods that are used to produce a 3D representation of biological structures.

The first part of this thesis will contain theoretical research of transmission electron microscopy including possible complications arising from the use of this type of microscopy. This will be followed by a chapter on image analysis; more specifically image segmentation and object detection and the role of convolutional neural networks during these tasks. Next stage of the theoretical part will include the basics of clustering that will also play a part in the proposed algorithm. The First part will be concluded by a short description of the algorithm that is to be designed in the practical part.

The second part of this thesis will the practical that is going to include the description of the process of creating the algorithm. It will present the original set of analysed images and the way to process them to serve as training tools for the developed algorithm. The fine-tuning of the neural network models used for object detection will be included in the next stage of the thesis. The results achieved by the algorithm are then to be described and evaluated. Next phase will be the application of clustering on these results. Results achieved after clustering will be evaluated and compared to those achieved before clustering.

There are several already existing methods that deal with the same problem but working on a different principle. Some of those will be described in the theoretical part and their results will be compared to results produced by the algorithm developed during this thesis at the end of the practical part.

1 TRANSMISSION ELECTRON MICROSCOPY

Transmission electron microscopy is a method that allows us to show the microstructure of an examined material in scale from microns to single atoms. It is also capable to determine the symmetry of crystal structure using electron diffraction and to locally analyse the chemical composition of the examined sample. Transmission electron microscopes are fairly similar to optical microscopes. However, unlike optical microscopes, the sample used in electron microscopes is transparent and they also use radiation on larger frequencies. During the procedure, a very thin sample of examined material is irradiated by a beam of electrons that have energy up to hundreds of keV. Electrons change their trajectory and energy when they interact with the observed material. Thanks to these changes, we are able to reconstruct the image of the examined sample and observe its composition. [1]

1.1 Architecture

The Body of a transmission electron microscope is usually made of several functional parts. These parts include an electron gun as a source of an electron beam, electromagnetic lenses used to focus the electron beam, an imaging system and a vacuum system. [2]

1.1.1 Vacuum system

A vacuum system is needed in transmission electron microscopes for more than one reason. One of the reasons is that the air is not an ideal insulator, which means that there is a risk of ionization of particles present in the air and that would result in electrical discharges between the anode and the cathode of the electron gun. Air also contains atoms of oxygen, nitrogen and carbon dioxide. These elements can contaminate the examined sample and the body of the microscope and that is another reason why the vacuum system must be present. It is also needed to eliminate collisions between electrons and particles in air and subsequent changes in energy and trajectories of the electrons. Under standard conditions, vacuum reached in the microscope should be around 10^{-7} Pa. In order to achieve these values, the microscope must be equipped with adequate pumps. [2],[3]

The rotary pump is the first pump used and it is capable to reach pressure of around 10^{-1} Pa. After the rotary pump has finished its job, the diffusion pump is used. Thanks to evaporation and subsequent condensation of special oil, the diffusion pump is able to get the pressure down to about 10^{-3} Pa. The ion pumps are used afterwards, and they can achieve pressure inside the body of the microscope of 10^{-7} Pa. [3]

The turbomolecular pump is a special type of pump and is used together with the cryo-pump. Even though the pressure inside the microscope body is very low, there are still some steam and carbohydrate molecules present. However, these potential contamination agents can be dealt with using the cold finger, which is a long copper rod extended along the body of the microscope and is cooled down to liquid nitrogen temperatures. The contaminants are then attracted to the rod and they condensate without damaging the body. [3]

1.1.2 Electron gun

An electron gun is composed of a cathode, an anode and the Wehnelt cylinder. The electron gun is supposed to emit electrons from a single point. These electrons also need to have the same phase and energy. The cathode is the source of electrons while the anode attracts the electrons. The Wehnelt cylinder is biased to negative voltage, which means that a cloud of electrons is created around the cathode. Single electrons are then launched towards the anode. After the electrons have passed through the Wehnelt cylinder, they are focused to a single point called crossover. This point can be thought of as the single point from which the electrons are emitted. [1],[2],[4]

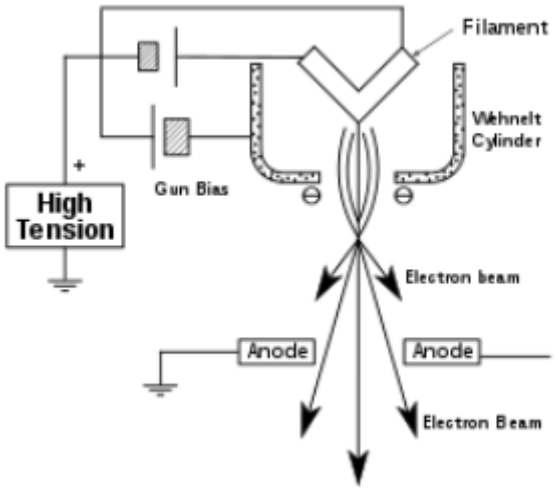


Figure 1: Electron gun schematic. [23]

Function of electron guns can be based on three different principals. First of these principles is secondary emission. If secondary emission is used, the cathode filament is cold and is being bombarded by accelerated ions that give energy to the electrons on the surface of the used material. In case the energy is large enough, electrons leave the material. This principle is not used in modern transmission electron microscopes. [3]

The most popular principle is thermionic emission. The cathode is heated up to very high temperature. Thermic energy passed on to the electrons is high enough for emission. Tungsten is mostly used as the material for the cathode since the electrons on the surface of tungsten need smaller amount of energy to achieve emission, compared to other materials. The tungsten filament is shaped into the letter V and is heated up to circa 3000 K. Another option is to use a crystal of lanthanum hexaboride. Electrons on the surface of this material need much lower energy for emission than tungsten and a beam of higher current density can be produced as well. [1]

Auto emission is the last principle used in electron guns. In this case, an anode with very high voltage is placed opposite to a tip of a cold cathode. A very powerful electric field (up to 10^9 Vm^-1) that has the power to rip out the electrons from the material, is then generated around the tip of the cathode. This type of emission is used in electron guns called FEG – Field emission gun and has the power to generate much more coherent electrons and higher current density beams, compared to thermionic emission. However, these need very high level of vacuum (ultravacuum) that has pressure of 10^{-8} Pa. The cathodes used in the FEG type are made of a tungsten mono-crystal. [3]

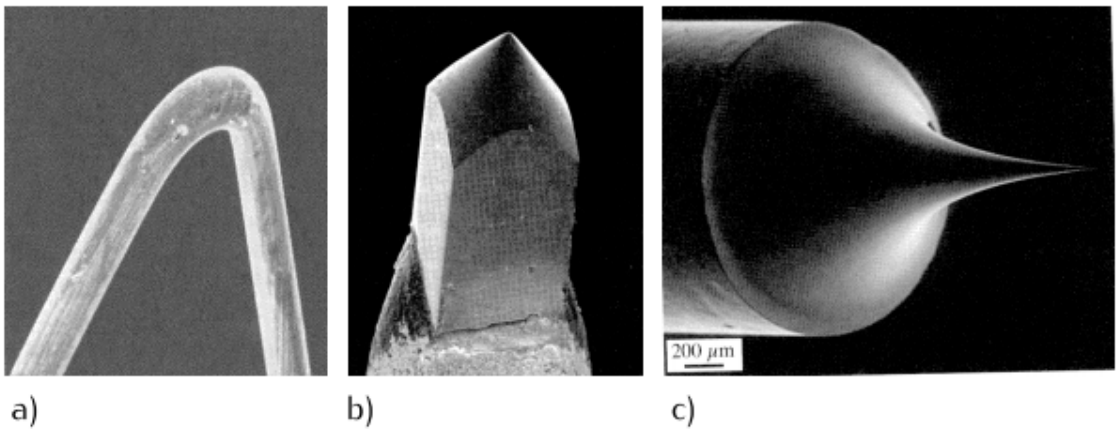


Figure 2: a) tungsten filament, b) LaB6 crystal, c) tip of an auto emitting crystal. [3]

1.1.3 Electromagnetic lenses

As electrons are streaming through the body of the microscope, they must pass through several electromagnetic lenses. These lenses are used to influence the trajectory of those electrons. It is very similar to a situation when glass lenses are used to influence stream of light. The electromagnetic lenses are mostly solenoids. [3]

The first pair of lenses the electron beam passes through are condenser lenses. The first condenser lens is able to create the image of crossover and to change the size of the image by changing its focus. The second condenser lens is used to focus this image to the level at which the specimen is located. Both lenses have apertures and while the aperture size of the first lens is fixed, the aperture size of the second lens can be selected between 100 and 500 µm. These apertures are often contaminated because of bombardments by electrons which leads to deterioration of their function. The condenser lenses and the electron gun are considered as the irradiating part of transmission electron microscope. [3]

Right after the electron beam passes through the specimen, it is focused by another electromagnetic lens – the objective lens. This lens is the most powerful lens of the entire microscope because it can achieve the largest zoom and has the shortest focal length. The objective lens coil is often cooled because it is prone to overheating since large currents pass through it. The objective lens is placed near the examined specimen and the cold finger anticontamination device is located near it as well. Projective lenses and intermediate lenses are used to zoom in the image created by the objective lens. Even though the image is only 100x zoomed, thanks to the projective and intermediate lenses, the transmission electron microscope is able to achieve zoom of 10^6. [2]

1.1.4 Imaging system

The human eye is not able to see the electrons that have passed through the specimen. To be able to see the result of the specimen examination, we need to convert it to the visible light spectra. A fluorescent screen is used for this purpose. This screen is covered in luminophore, which is a substance that can emit light of intensity and wavelength dependant on the amount and energy of electrons hitting the screen. The most popular luminophore is ZnS. Light emitted by this luminophore is usually circa 550 nm. [2]

There are a few ways we can record the image created on the fluorescent screen. One of these ways is the use of special photographic material that must meet certain conditions. First of the conditions is that the material needs to be sensitive to very short waves and the second condition is that it must be able to withstand the extreme vacuum conditions that prevail in the microscope. Because of these restrictions, not all photosensitive materials can be used. The most popular system used as fluorescent screen is a polyester pad covered in small crystals of silver chloride. [2],[3]

Another way to record images created by the microscope is to record using digital technology. In this case, the information created by electrons hitting a scintillator. The scintillator can convert the information into a digital format, which can be visualized using a computer. An ideal scintillator should be able to record image as quickly as possible, with very little distortion and with very high resolution. CCD (charge-coupled device) cameras using YAG (yttrium aluminium garnet) crystal as scintillator are often used in practice. Some of the popular CCD cameras are high-definition Morada and KeenView camera, which is used thanks to its compatibility with nano-samples. [2],[3]

1.2 Image acquisition

As soon as electrons reach the examined material, they interact with it. These interactions can be divided into two basic groups – elastic and inelastic. When an elastic reaction occurs, electrostatic potential of atomic cores is interacting with electrons and so changing their original trajectory and energy. However, because there is a large weight difference between the static atom and the moving electron, the change in energy is very small, near zero. The electrons are more deflected from their original trajectories when they get very close to the nuclei of the atoms than when they are further away. In case of great proximity, the electrons can be deflected back to the source of the electron beam. [1],[3]

An inelastic reaction occurs when the moving electrons interact with the electron shell of the static atoms. During this interaction, the moving electrons pass on a portion of their energy to the electrons in the electron shell. This action makes the static electrons excited, which means that they subsequently radiate with characteristic x-ray radiation. During these inelastic reactions, the energy of moving electrons is reduced, and their wavelength extended. This change can result in an unwanted chromatic aberration. [1]

1.2.1 Acquisition of images of biological material

Transmission electron microscope can be used in more operating modes. Which mode to use depends on what sort of material is examined. When examining biological specimens, a problem arises. The specimen mostly consists of light elements, and these light elements do not interact with the electrons strongly enough to change their energy. For that reason, biological specimens are usually sealed in cases, to which heavier elements are artificially added. These elements are usually lead or osmium and their function is to enhance the contrast of the resulting image. [3]

Diffraction can also be used during the examination of biological specimens. Diffraction of the electron wave occurs on the edges of the specimen. The diffracted beam subsequently interferes with the original non-diffracted waves. Lighter areas are then created on the fluorescent screen in case of interference maxima. Darker areas are created in case of interference minima. Ideal interference is only possible when the waves are

perfectly coherent. Since the coherence of the electron waves is not ideal in real microscopes, only one interference maximum and minimum is created and that is referred to as the Fresnel fringe. These fringes can be then used to create more focused image or to correct astigmatism. [3]

1.2.2 Acquisition of images of crystal material

In the case of a crystalline material specimen, the diffraction of the passing electrons occurs on the crystal planes. This operating mode of the microscope is referred to as a diffractogram. If the electrons interacting with the sample are parallel to one another, the diffracted electrons that have passed through the sample are also parallel. After passing through the objective lens, these electrons are concentrated to the points in the back focal plane. These electrons create the Fraunhofer diffraction image, which depicts the Fourier transform of the electron wave that emerges from the sample. This image can be considered a primary image as it is a source of waves that, after an inverse Fourier transform, create an image of the subject in the image plane. [1],[3]

When operating in the diffractogram mode, or using a diffractive contrast display, a small objective lens aperture is used. This aperture lets through only one beam of electrons, either non-diffracted (T) or diffracted (D). When using a large aperture, the information about the phase interference contrast is obtained. The image A' of point A is the result of the interference of waves ψ (g), ψ (-g), ψ (0) and other waves that have passed through the objective lens aperture. Thanks to this principle, the crystal lattice image can be achieved at atomic resolution. [1],[3]

1.3 Cryo-TEM

Cryo-TEM is a form of cryogenic electron microscopy where the sample is examined at very low temperatures. These temperatures are usually achieved with the use of liquid nitrogen or liquid ethylene cooling. Unlike X-ray crystallography, in which biological samples are crystallized, samples can be examined in their natural environment and without the need for fixation. [5]

In this process, the biological material is spread on a grid and stored in a frozen state. Fast freezing does not create ice crystals that could disturb the flow of electrons and are therefore suitable for this type of microscope; this process is referred to as vitrification. The samples produced this way are able to withstand the extreme vacuum conditions that prevail in the transmission electron microscope. Most biological samples are extremely sensitive to radiation, so they must be observed using smaller doses of electrons. As a result, the crated images are heavily affected by noise. For some biological images, multiple images can be captured and subsequently averaged, resulting in improved signal-to-noise ratio and resolution. These adjustments and improvements are achieved through the multi-element single-particle analysis method. Three-dimensional reconstruction of protein complexes and viruses has already been achieved with near atomic resolution. [6]

1.3.1 Single-particle analysis

Single-particle analysis is an algorithm that can be applied to images of vitrified samples taken by a TEM. This analysis is able to determine the structure of biological entities such

as proteins of viruses. As mentioned earlier, images of biological specimens are usually heavily affected by noise. When more images of the same specimen are taken, we can then average these images and get rid of the random noise. However, the biological structures on the taken images have often different orientation in each image, so it is necessary to align them. This can be achieved by statistical clustering methods such as k-means but a large number of images of the specimen in question are needed. It is also useful to filter the images before alignment and classification. Low-pass filters get rid of high spatial frequencies and some portion of noise while high-pass filters remove low spatial frequencies such as gradients. [8]

Alternatively, an iterative map-based refinement can be used to align the orientation of the shown samples. The map uses two position parameters and three particle orientation parameters. These parameters are then altered and scored by a scoring criterion. Some problems may occur when using this procedure. For example, when an incorrect starting map does not match the signal in the taken images and proceeds to the end of the refinement algorithm, incorrect results will occur. [8]

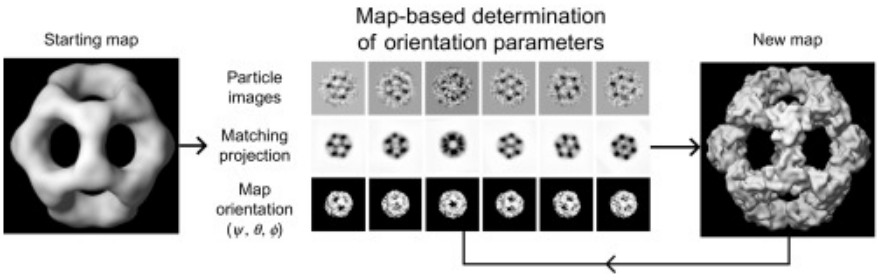


Figure 3: Iterative map of single-particle analysis. [8]

1.3.2 Cryo-Tomography

Using cryo-tomography, researchers are able produce high-resolution 3D models of observed samples. These samples are usually biological, such as macromolecules and cells. The process includes creating 2D images of the observed sample, which is then tilted by about 1 or 2 degrees for every shot between the range of -60° to +60°. After obtaining a series of 2D images, single-particle analysis algorithm can be used to create a 3D model of the observed specimen. The achieved resolution depends also on the thickness of the monitored specimen. Samples less than 500 nm thick need to be used in order to get a macromolecular resolution of 4 nm or smaller. That is the main reason why small organisms like bacteria, viruses or archaea are usually used for this type of imaging. Larger organisms need to be cut into appropriate samples by cryo-sectioning or by focus ion beam milling. [11],[12]

2 COMPLICATIONS WHILE USING TEM

2.1 Lens aberrations

Similar to optical lenses, electromagnetic lenses tend to cause aberrations in the resulting image. Most often this is due to the imperfect homogeneity of the magnetic field within the lens. [3]

2.1.1 Chromatic aberration

The first common type of aberration is chromatic aberration. This is caused by the instability of the exciting current and the accelerating voltage of the lens. This results in uneven energy of the emitted electrons, and thus unequal energy loss and change in momentum after passing through the specimen. Electrons with lower energy are bent in the magnetic field of the coils more strongly than the faster and more energetic electrons and pass through the axis of the coil at a different point. This defect can be corrected by stabilizing the accelerating voltage, thereby making the electron beam more coherent and monochromatic. [3]

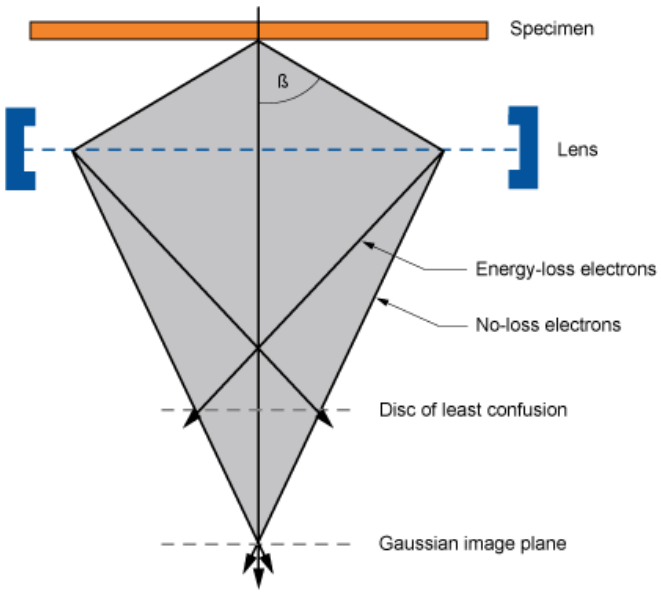


Figure 4: Chromatic aberration illustration. [24]

2.1.2 Spherical aberration

The second common form of aberration is spherical aberration. This aberration occurs when a wide monochromatic beam of electrons enters the lens. The electrons that pass through the peripheral points of the lens do not converge at the same point as the electrons that pass through the electron-optic axis of the lens. This defect can be alleviated by adjusting the aperture diameter. [3]

In the figure below, an illustration of the spherical aberration is shown. C_s is a spherical aberration coefficient which is constant for every lens. β represents the

maximum semi-angle of the lens aperture collection. Disc of the least confusion is a compromise of sort that shows the plane where the aberration is least visible. [3]

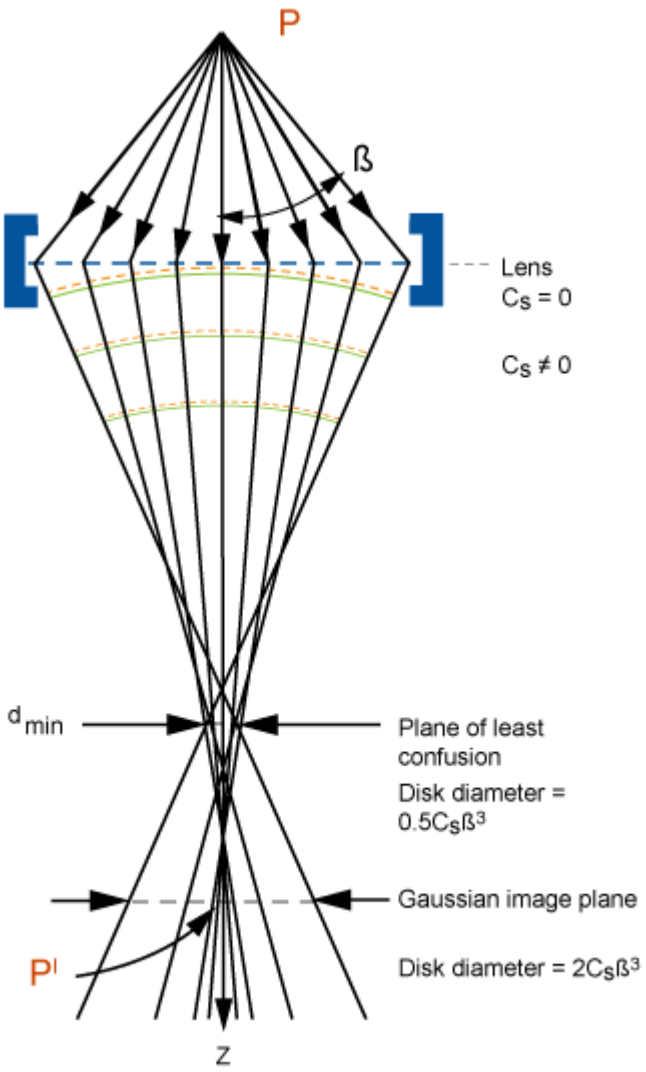


Figure 5: Spherical aberration illustration. [24]

2.1.3 Astigmatism

The astigmatism aberration is the last form of common aberrations. It is created when the beam of electrons passes through an inhomogeneous magnetic field inside the lens. This leads to non-uniform focus of the beam. This aberration can be easily solved by stigmators. These small components can compensate the magnetic field and fix the inhomogeneities. [3]

2.2 Other complications

Other complications caused by using a transmission electron microscope are usually linked to biological sample preparation for a non-cryo-TEM. All water would immediately vaporize inside the body of the microscope, so the sample needs to be dehydrated. The samples must also be chemically stabilised and fixed. This step must be

done as soon as possible after removing the tissue out of its natural environment because some parts of the tissue like mitochondria of micelles start changing their appearance right after the removal. If the chemical procedure is not carried out quickly enough, reshaped mitochondria and micelles need to be regarded as artefacts. [13]

The next possible source of problems is cutting the samples. They need to be cut into very thin (100 nm or less) layers or they cannot be used in the microscope. As mentioned in previous chapters, biological samples are very radiosensitive, so they must be irradiated by low dosage of electrons. Even though the dosage of electrons is low, temperatures caused by beam hitting the sample can be up to 150 °C in non-cryo-TEMs, which can be a cause of other problems as well. [13]

3 SEGMENTATION AND OBJECT DETECTION

Segmentation is a procedure in which interest objects can be defined in the image. These objects can be segmented in the image using automatic or interactive methods. Thanks to segmentation, it is possible to separate objects by defined parameters and to determine what is an object and what is background. Object detection is a process that is capable of recognition and classification of the objects. [7]

3.1 Segmentation methods

There are many principles that can be used during segmentation. There are algorithms using some parameters of the image (such as intensity), region-based segmentation or edge-based segmentation. Other techniques use explicit or implicit models. [7]

Techniques using explicit or implicit models may have certain parameters of the segmented object predefined and further specified by additional information. The models are based on three basic algorithms. These algorithms include a geometric algorithm, statistically active algorithm and active contour algorithm. However, these algorithms are time consuming, computationally demanding and require input from the user, either in the form of manual initialization or in the form of a training set. [7]

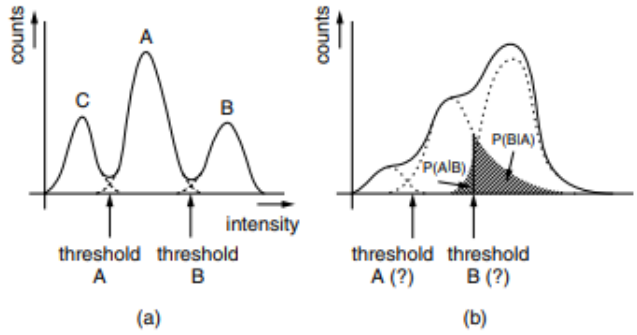


Figure 6: Histograms used during segmentation. (a) a histogram with well-defined classes, (b) a histogram with poorly defined classes. [10]

3.1.1 Parametric segmentation

This type of segmentation is based on the idea, that object segmented in an image should be homogenous regarding a defined parameter. Often, the parameter chosen is brightness or grayscale intensity of the pixels contained in the object. However, intensity is not the only parameter that can be defined. Other defined parameters may be a scalar or a vector, but the idea is pretty much the same. Scalar parameters within the segmented object must have values in some pre-defined range and similarly, vector parameters inside the object also need to belong to some spatial area. This sort of approach logically has some limitations. For example, non-uniform illumination can disrupt segmentation using intensity as a parameter. [10]

When using intensity as the defined parameter, the simplest way to choose the range of values is just to define and interval using the largest and smallest value of intensity that the object can obtain. This approach is used for some medical images such

as a CT image where the different tissues have different grayscale values. When ranges of intensity values are set, every range depicts a different class of segmented objects. The ranges of these classes should be disjunctive – no two classes should have the same intensity values. However, this is not as easy as it sounds. The ranges of values can be set interactively by the user or automatically by an algorithm. When the limits between classes are set correctly without any overlaps, the segmentation is carried out without any major errors. When overlaps occur, high error rate should be expected. [10]

The most basic case of this type of segmentation occurs when only one threshold is set for a grayscale image. When this happens, only two classes are created, and the original grayscale image is transformed into a binary image. All pixels that possess value higher than the mentioned threshold are white and the other pixels are black. It can be of course done the other way around, depending on the user's preference. When more than two classes are set, pseudo-colouring can be used to show the final segmentation. [10]

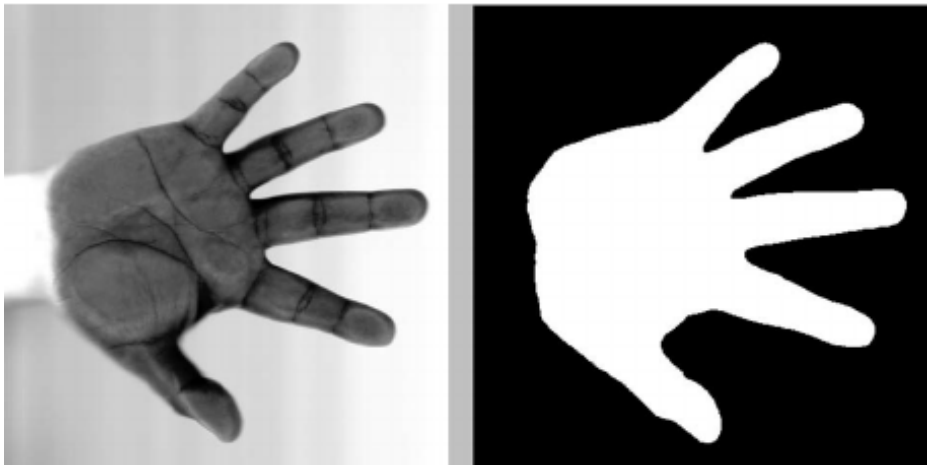


Figure 7: Two-class segmentation on a grayscale picture. Darker parts are chosen as the object, lighter parts are chosen as the background. [25]

Grayscale images are not the only ones that can be segmented using parametric methods. Fused multimodal or colour images can also be processed. Pixels of these images are not scalar values but multidimensional vectors. However, a segmentation histogram created for multidimensional vectors is quite similar to the one created for grayscale images. In case of bimodal images that contain two-dimensional values in pixels, the created histogram contains two-dimensional intensity scale. Classes are then created and are represented as intensity clusters in the histogram. During the segmentation, both components must fit into a certain class. The pixels of certain intensity values are therefore excluded, when one of the components does not fit in. [10]

3.1.2 Region-Based Segmentation

This type of segmentation techniques also works with the idea of homogeneity. However, it is usually applied more locally, so it can be more precise than parametric segmentation techniques that are mostly applied on the whole image. [10]

- **Region growing**

Region growing is a region-based segmentation procedure, during which a pixel in a potential region is interactively of stochastically determined. This pixel is referred to as a

seed. A parameter *p* that can represent intensity value, local mean, local variance etc. is then chosen as well. Pixels surrounding the seed are then put to the test of homogeneity:

$$|p_s - p_j| \leq T \qquad \text{Equation 1}$$

Where p_s is the seed value of the chosen parameter, p_j is the value of the tested pixel and *T* is the used threshold. If the calculated value is smaller than or equal to *T*, the pixel does fit into the same region as the seed. This algorithm logically applies not only to direct neighbours of the seed pixel but also to neighbours of the pixels that were deemed good enough to join the region. The procedure stops where no surrounding pixels can be added to the region. This not only means that they do not meet the condition presented in Equation 1 but also that they may belong to a previously segmented region. Because of this, in case that parameter values for neighbouring regions overlap, pixels are added to the region that was processed first. [10]

This approach can be modified when the parameter value of a new candidate pixel is not compared to the value of the seed but to the value of a pixel neighbouring the candidate pixel but already belonging to the region:

$$|p_i - p_j| \leq T \qquad \text{Equation 2}$$

Where p_i is the parameter value of the already belonging pixel, p_j is the parameter value of the candidate pixel and *T* is the threshold. Using this modification, slow changes in the p_i value can be observed in case the pixels in the region are not the same as the seed. When a very swift change of the p_i value occurs, the algorithm should stop since the calculated difference will be higher than the threshold value. This usually occurs when a border edge is met. However, this can be influenced by the order, in which the pixels are processed. The value growth may not seem as large when compared to a different pixel of the region. [10]

- **Region merging**

Region merging segmentation algorithm starts very similarly to the previously mentioned region growing. Image is separated into primary homogeneous regions. These regions are usually very small, they also might be only single pixels. Two regions located next to each other can be merged if some given homogeneity condition is met. This condition may be defined statically, which means that a precise range of a selected parameter is set and all pixels in the given region must have values within this range. Another region merging procedure includes more dynamic definition of the homogeneity condition. During this dynamical type of region merging, mean values of chosen parameter in both regions are compared and then it is assessed whether they merge or not. The mean values change during the region growth and that makes it possible to create bigger and more relevant regions compared to the static condition. [10]

Even more dynamic approach to region merging can be achieved by considering strength of the borders between created regions. Parameter values of two adjacent pixels from two different regions are compared. So-called crack in the boundary is created, when the difference between two pixels is smaller than a defined threshold. This comparison is calculated for all pairs of pixels along the border. Strength $S_{i,j}$ of the common border along *N* pixels is then considered to be:

$$S_{i,j} = \frac{N_s}{N} \qquad \text{Equation 3}$$

where N_s is number of cracks in the border. When the border strength $S_{i,j}$ is smaller than a defined threshold, the border is than removed and the adjacent regions merge. [10]

- **Region splitting and merging**

This segmentation procedure starts when image is divided into quarters. These quarters are then subsequently divided into smaller square regions until each region is homogeneous by a given criterion. This however usually is not the end of the segmentation process. Some neighbouring regions may be homogeneous but since they were not in the same parental region, they are not merged. This means that after splitting process is finished, merging process commences and goes on until no region can be merged. The merging steps can be also included to the splitting process. [10]

- **Watershed segmentation**

The watershed method is fairly different from the other region-based segmentation methods. During the watershed process, local minima are found inside the image. These local minima can be in this case taken as the deepest parts of a topographical map of the image. The next step of this metaphorical approach is that it starts to rain upon this topographical area. The local minima and surrounding catchment basins (regions) are then flooded by water and borders between these regions act like dams that do not let the water through. [10]

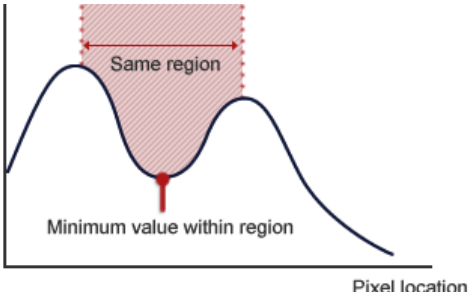


Figure 8: Graphical representation of watershed segmentation method. [27]

3.1.3 Edge-based segmentation

The edge-based segmentation does not investigate the homogeneity of segmented regions, but it rather tries to find borders between them. The procedure finds areas in the image, where rapid changes of the defined parameter happen. As said earlier, the homogeneity of segmented regions is not considered, it is only known that the changes within the regions happen more slowly than at the edges. In order to have a correctly segmented image, these borders need to be closed curves and must clearly define the regions. This, however, is not always the case. [10]

The edges are very often too thick, disconnected and many of the edges do not represent the real borders between the regions. Some modifications thus need to take place in order to achieve a correct result. Every border should only be one pixel thick and no blind spaces can be present. These modifications are usually done via morphological operators. Hough transformation can also be used to perform an edge-based segmentation. [10]

Figure 9: An example of the edge-based segmentation (used on an image of coins). [26]

3.1.4 Active contour segmentation

Classical segmentation methods do not cope well with images heavily affected by noise. The obtained regions and borders between them may not correspond to reality at all. Active contour segmentation attempts not to be affected by noise by allowing the segmentation curves to be deformed a little and by not considering small details and errors created by the noise. [10]

3.2 Segmentation using neural networks

Neural networks usually use more of the previously mentioned methods combined. It is possible to divide the function of segmentation using neural networks into two larger groups – segmentation based on pixel data and segmentation based on features. [14]

3.2.1 Based on pixel data

These methods generate segmented result directly from the image that was presented to them in pixel form. Several types of neural networks can be designed and created for this type of segmentation such as: acyclic neural networks, self-organizing maps, cellular networks, Hopfield networks and others. These networks can have hierarchic organization, where lower-level networks are specialized on finding certain features in the data and provide only partially segmented image. Higher-level networks connect these results provided by lower-level networks and perform a complete segmentation. Usually, these networks are trained for segmentation based on differences in texture or in texture and shapes. [14]

3.2.2 Based on features

This type of segmentation does not concentrate on attributes of only one pixel like the previous method, but also on its surroundings. Same types of neural networks may be used for feature-based as for pixel-based segmentation. Hierarchic networks used to classify optic features and to determine distance in the image may also be included in this group. Feature-based networks can use more features than just texture and shape differences to perform segmentation but also histogram thresholding, region growth, region merging and edge connecting. They also have the advantage that unlike pixel-based procedures, they work regardless image rotation and scale. [14]

3.3 Usage of convolutional neural networks

Convolutional neural networks can be taught to extract features from pixels of given images and subsequently recognize patterns, which is very useful during image segmentation and object detection. Convolutional neural networks are built from layers of perceptrons that consist of one or more planes. Because of this, every layer has three major attributes: height, width and depth. These layers also can have different purpose, such as convolutional layers (CONV), rectifiers (RELU), pooling layers (POOL), or fully connected layers (FC). The layers are then connected to create a convolutional neural network. CONV and FC layers carry out transformations of the input data which depend on weights and biases of present neurons, while POOL and RELU layers perform a fixed function such as subsampling. Thanks to these layers and their function, the network is able to transform an input image from original pixels into a feature vector, which can be subsequently classified. [15]

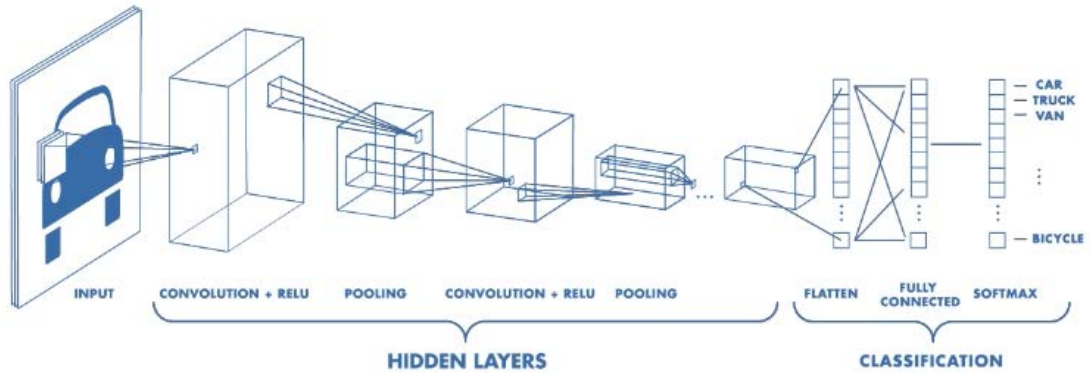


Figure 10: Architecture of a convolutional neural network used for pattern recognition. [28]

3.4 Object detection

Object detection is a process that is able to recognise multiple different objects within an image. This helps us understand, what the image is containing and can also be applied in real life uses such as face recognition, security surveillance or driving assistance. [33],[34]

First part of the object detection process is training a model. Models are predefined structures that are able to extract features from an image and then classify those features. Several models will be presented in the next chapter. In order for the model to detect images suitable for our purposes, it has to be trained using our set of images. The general rule is that more training images can help create better detecting models. Around 15% of all used images should be so-called test images, on which the training process calculates error and determine when the model is trained ideally. When the error is converging for some time, the graph of the trained model is exported and can be used as a detector. [33],[34]

As mentioned before, features must be extracted from an image before the detection itself happens. The detection is separated into several blocks within the model. Each block reduces feature maps, which helps concentrate on useful features and then passes the reformed data onto next block, until the final block is reached. Every detection block has three major tasks. First task is to resize the boxes, in which the model is predicting an object. Second task is to predict confidence scores for these boxes and the third is to move the boxes as well as it can with the provided information and then pass these calculations onto the next block. [33],[34]

When the object detection is over, the algorithm shows the image with detected objects and their confidence score. The confidence score threshold, when an object is classified and marked in the image can be manually determined. Usually the threshold is set at around 50% confidence score. [33],[34]

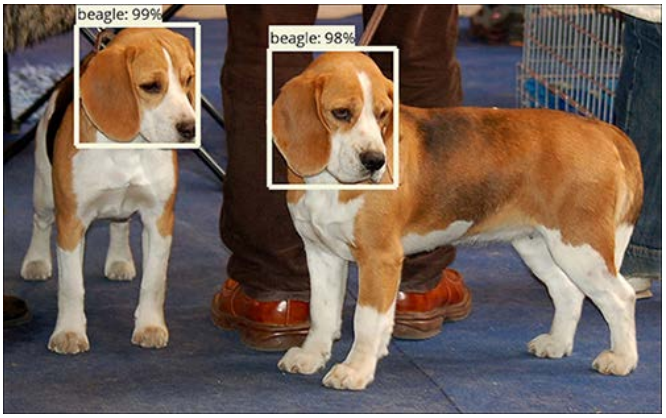


Figure 11: Example of object detection showing detected objects inside bounding boxes with class and confidence scores [35]

3.4.1 Basics of TensorFlow

TensorFlow is and open source library that can design and train models used for object detection. First version of TensorFlow was created by Google in 2015 and it has been updated and used ever since. It is an open-source software library that is mainly used for machine learning operations. The name ‘TensorFlow’ is derived from its key functional part - tensors. A tensor is an object that can be a matrix or a vector of multiple dimensions that can represent different type of data with a specified shape. This shape can be described as the dimensionality of the data stored within the tensor. The tensor can be specified by the input data or it can also be calculated. Computations inside TensorFlow are done with data flow graphs. Edges of these graphs describe the data while nodes of these graphs represent mathematical operations carried out among the data. [31], [32]

For our purpose, the input data of the TensorFlow graphs will be feature vectors of images, which are vectors extracted by a model from an image and fed into the input tensors. After mathematical operations are performed within the nodes of the graph, a new tensor is created, which can be also a starting point for another node operation and this process can be repeated as many times as necessary. [31],[32]

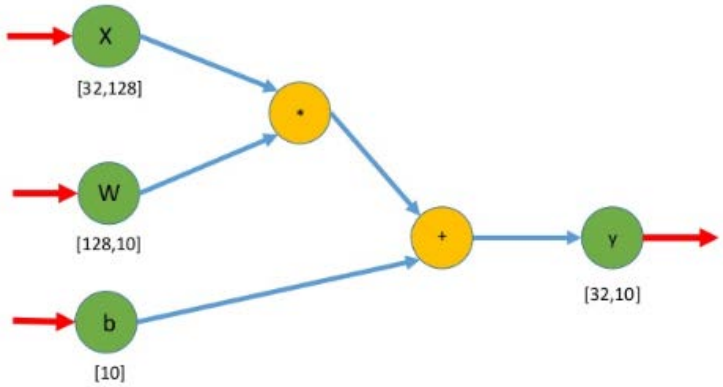


Figure 12: Example of a TensorFlow graph with data on the edges and nodes representing the mathematical operations [30]

3.4.2 Faster RCNN

Faster RCNN (Regional Convolutional Neural Network) together with MobileNet SSD (described in the following chapter) are one of the most used object detection models and can also serve as a basis for neural networks designed for segmentation of even 3D object detection. Faster RCNN is comprised of three fundamental parts – Feature Network, Region Proposal Network and Detection Network. [36]

The Feature Network is mostly some kind of a pre-trained image classification network such as VGG or Resnet that was slightly changed to fit into the RCNN scheme. This part of the RCNN provides us with features describing basic shapes and structures from the given image. Region Proposal Network (RPN) consists of 3 convolutional layers. First of them is a common layer that distributes data into two subsequent layers, one designed for classification and the other for bounding box regression. RPN creates bounding boxes that are called Regions of Interest (ROI), in which there are higher chances of containing a desired object. Output provided by the RPN is a list of coordinates of bounding boxes and values that determine whether the box can be ignored or should be considered for further investigation. The last stage of the RCNN – the Detection Network then takes outputs from both previous parts and generates final position and size of bounding boxes and respective classes. To make this process easier, the features are cropped according to the bounding boxes. Both the Detection Network and Regional Proposal Network need to be trained in order to work correctly. [36], [37]

Training stage of RPN part of the model begins with generating bounding boxes. During the training, boxes that overlap with other boxes with higher scores are deleted. To generate labels for the RPN classification stage, the newly created bounding boxes are compared to the ‘ground truth’ bounding boxes. These ‘ground truth’ boxes are set in place and labelled by the user, so the training algorithm assumes that they are correct. These similarity calculations are then used to label 256 regions of interests as foreground, background, and ignored. Training of Detection Network is very similar, regions of interest are also generated and assessed set as foreground, background or ignored based on the similarity to the ‘ground truth’ boxes. The difference is that the regions of interest assessed as foreground are then classified into more classes rather than staying just as foreground. [36], [37]

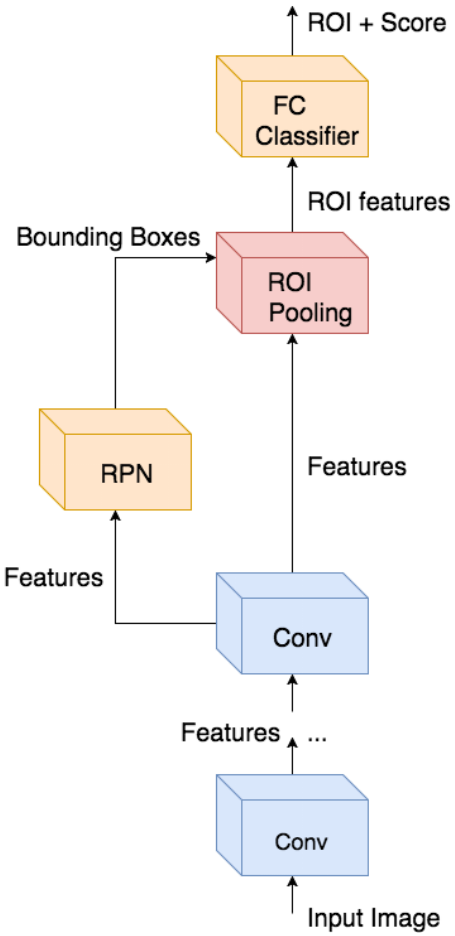


Figure 13: Simplified architecture of the Faster RCNN [44]

3.4.3 MobileNet + SSD

MobileNet V1 is a neural network architecture, which is designed to run on mobile devices and that means it is less computationally demanding than previously mentioned Faster RCNN. Convolutional layers that are present in classical CNNs are replaced by separable convolutions. First stage is the depth wise convolution that filters the input and that is followed by point wise convolution that works with the filtered values and creates new data that will be sent forwards. These two blocks do pretty much the same work as a conventional convolution layer, but they do it in shorter time. [41]

The regular MobileNet V1 scheme is consisted of a 3x3 convolutional layer followed by the separable convolutions block, which is repeated 13 times. There are no pooling layers, but the depth wise layers are able to reduce the size of feature map of the input, which can be subsequently analysed. However, in my work, I used MobileNet V2, which is slightly different than the first version. [41]

MobileNet V2 still keeps the two parts of separable convolution – depth and point wise. However, the point wise convolution works a bit differently. Instead of keeping the number of channels (dimensions) the same or doubling them, this time, it converts the data into a tensor with lower number of dimensions. It is known called the projection layer. Another change is that a new layer is added in front of the two original layers. It is known as the expansion layer. This layer expands the number of tensor dimensions before they reach depth wise convolution layer and does pretty much the opposite job of the projection layer that was described earlier. Thanks to these changes, the model uses low-dimensional tensors that decrease the computational demands. The expansions and projections are biased by learnable parameters, so the model knows how to change the data in the best way possible. [41]

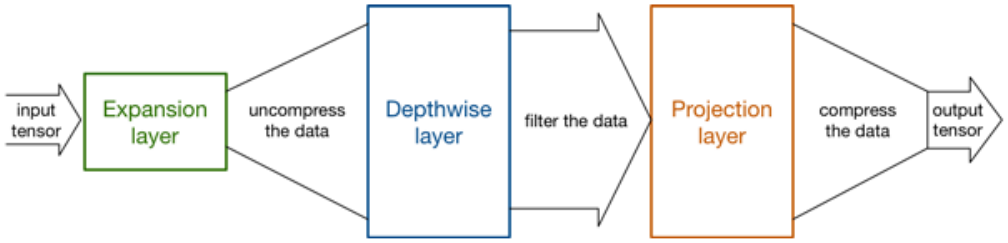


Figure 14: Scheme of one MobileNet block [41]

MobileNet can be retrained to work as an object detector or a classifier. In my work however, it is used as a feature extractor for SSD – Single Shot Detector. SSD is a neural network model that is designed for object detection and when connected on top of the MobileNet used for feature extraction, it uses a process called SSDLite. This process allows the use of depth wise separable layers during object detection and helps the SSD classifier to work a lot faster. [41]

3.4.4 CUDA

Training and object detector using TensorFlow needs more than just its libraries. It also needs CUDA. CUDA is a parallel computing platform and programming model. It was developed by Nvidia and it is used for computing using GPUs (graphics processing units – graphic cards). Thanks to CUDA developers can make compute-intensive applications faster by harnessing the power of GPUs. Function of CUDA is described on Figure 21.[42]

In the figure 15, you can see how CUDA works. The data is copied from main memory to GPU memory, CPU gives an order to start the computation using the GPU, CUDA makes sure that computations are carried out parallelly in order to make the process much faster and then the results are then moved back to the main memory. [42]

CuDNN is another vital part of training using TensorFlow. It is a library of functions that are necessary for training of deep neural networks. Both CUDA and cuDNN used for training must be compatible with the version of TensorFlow that is used for training. For example, I used TensorFlow 1.13.1, which means that CUDA must be version V10 and cuDNN needs to be version 7.4. [43]

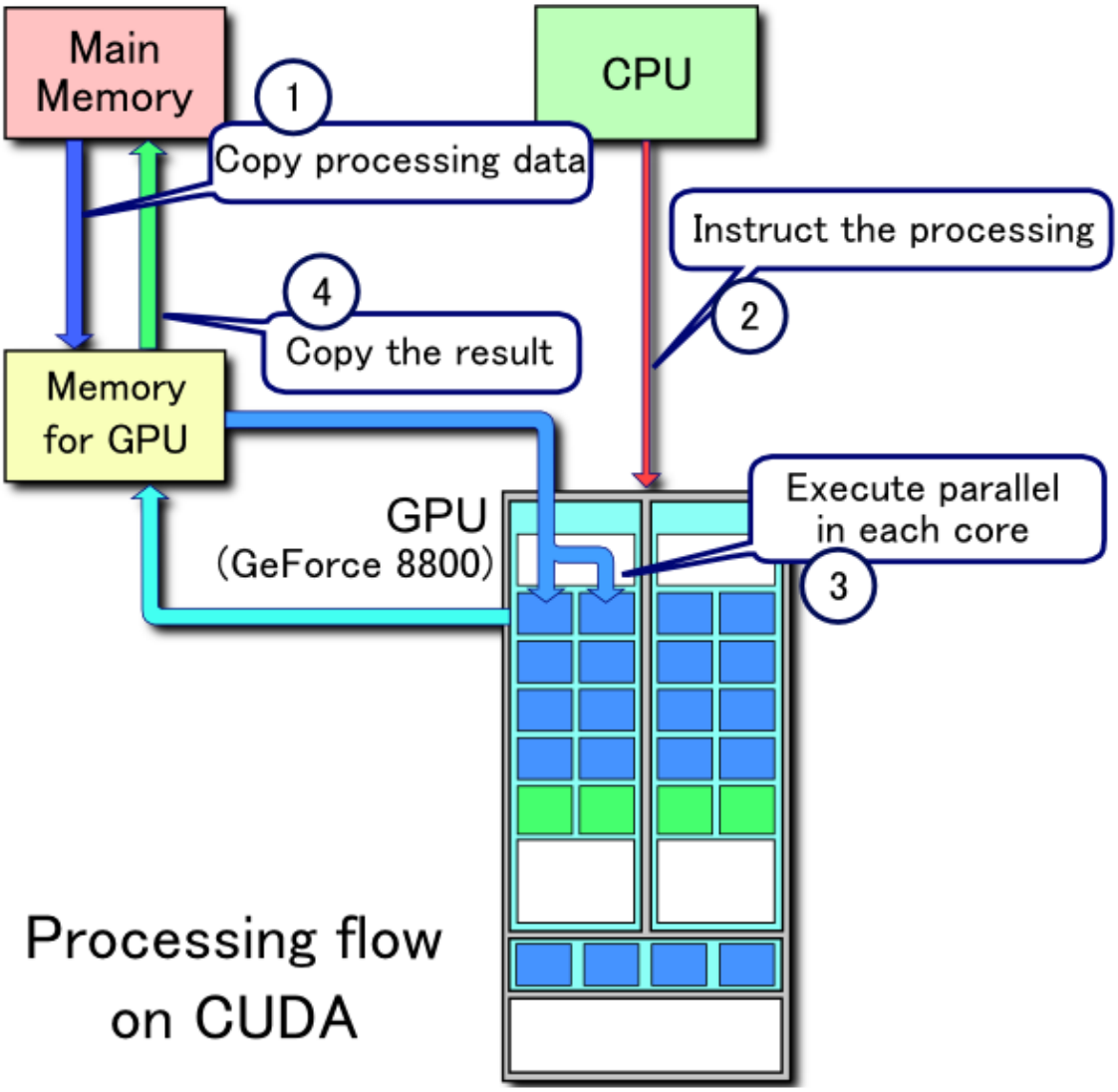


Figure 15: Explanation of the function of CUDA. [42]

4 CLUSTER ANALYSIS

The cluster analysis deals with methods that organize the data obtained into meaningful structures, creating taxonomies. Clustering is a data analysis tool that sorts objects into clusters so that the similarity of two objects belonging to one group is as high as possible, while the similarity to objects outside this cluster is minimal. By clustering, it is possible to find relations between objects without their further explanation or interpretation. In other words, cluster analysis finds a structure between objects without explaining why they exist. Clustering algorithms are widely used for example in computer science, biology, bioinformatics or market research. Cluster analysis algorithms can be divided into hierarchical and non-hierarchical. [16]

4.1 Hierarchical clustering methods

Hierarchical methods use previously found clusters to create new clusters. The intersection of every two subsets in hierarchical clustering is either an empty set or one of the original ones. Clustering with hierarchical methods can be represented by a binary tree - dendrogram. Horizontal tree levels are the degree of cluster decomposition. The vertical direction is the distance between the clusters. The disadvantage of hierarchical clustering is that it strives to achieve only the best local solution at every step and does not consider the next process. Hierarchical algorithms are further divided into divisive and agglomerative. In agglomerative clustering, clusters that are already clustered together cannot be divided, and vice versa, in divisive clusters, they cannot be reunited in the next steps and thus improve the decomposition properties. [17]

4.1.1 Agglomerative methods

An agglomerative approach takes every element of the processed set as a cluster, and then combines it until a cluster containing all the elements of that set is created. The disadvantage of these methods is that ambiguities may arise at the beginning of the clustering algorithm, which will only be shown later in large clusters. The previous step cannot be changed. [17]

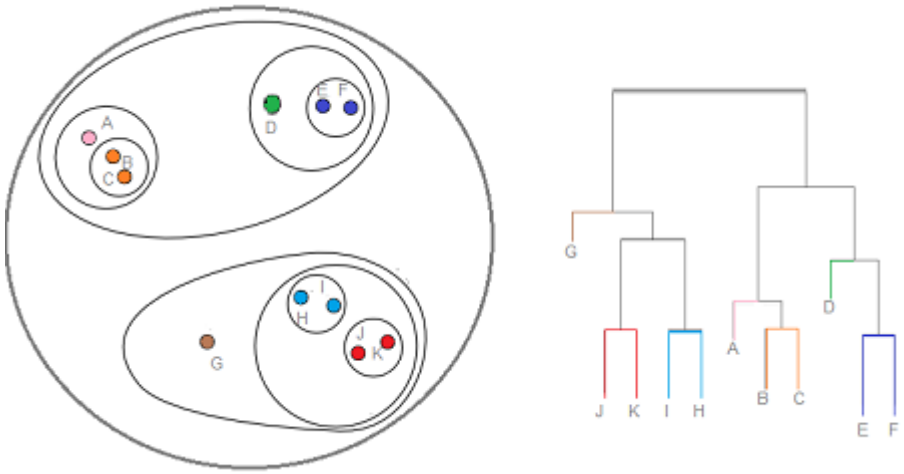


Figure 16: An example of a dendrogram created by hierarchical clustering. [29]

- **Simple linkage**

This is an agglomerative method that creates a cluster or clusters of objects that have the smallest distance between them compared to other sorted objects or clusters. The distance between the clusters is calculated by taking the smallest distance from each two objects from two different clusters. The resulting clusters are not spherical in nature. Complications may arise when two different objects are located in the same distance from existing clusters. There may also be premature merging of two well-distinguishable clusters whose boundaries will come closer at some point. Non-ecliptic clusters can also be sorted using this method. [17]

- **Complete linkage**

This method puts together objects or clusters that are furthest from each other within a set of data. This means that the greatest possible distance between two objects in two clusters is taken as the determining distance. From these calculated distances, it selects the shortest and associates the corresponding objects. It creates tight clusters of approximately the same size. Sorted clusters tend to form spherical clumps. [17]

- **Average linkage**

The similarity of two clusters is calculated as the average of the distances between each two objects belonging to the two clusters. The most similar are clusters with the smallest average distance. The resulting dendrogram is similar to the tree created using the method of the closest neighbour, with the difference that the connection is made at greater distances. This method is not too sensitive to statistical variations in data. [17]

4.1.2 Divisive methods

Unlike the agglomerative methods, the divisional algorithms take the input set of objects as one cluster, and then divide them. At each step, the cluster divides into two new ones that best meet a given division criterion. This procedure is computationally demanding and is feasible for a small number of input objects. [17]

- **MacNaughton-Smith method**

This method is applicable to larger sets of objects with fewer computation time requirements. Compared to agglomerative approaches, however, it is still slower. An object within a cluster is selected to create a new cluster. Based on differences in the mean distance of objects from the original, and objects from the new cluster, other objects located in the original cluster are assigned to the new cluster or they remain in the original one. An advantage over the agglomerative approach is that the results are clearer for larger clusters. [17]

4.2 Non-hierarchical clustering methods

These methods do not create a hierarchical structure, they break the given set into subsets according to a predetermined criterion. The first decomposition to subsets is not further divided. It is edited to optimize the distance between clusters and to distribute the objects equally in them. Finding the best solution by testing all possible clustering arrangements is usually impossible. The disadvantage is that methods usually end only with locally optimized decomposition. [17]

4.2.1 K-means

At the beginning of the K-means algorithm, it is necessary to select the initial sample points. Points can be selected randomly from the input set of objects. Subsequently, the individual objects are taken and assigned to the nearest sample points. After this assignment, the centre of gravity of each newly formed cluster is calculated. Some of the data is then assigned to new clusters according to the closest centre of gravity. The new centre of gravity is then calculated again, possibly reassigning the data to new clusters. The process is finished, when not changes regarding the assignment of data to the clusters are made. [17]

4.3 K-means used for clustering of images

K-means will be used in the second part of this thesis in order to attempt to perfect the results of object detection. The function of K-means will be based on clustering the features of detected objects and then, based on the cluster analysis, some images that will be part of a chosen cluster shall be eliminated from the process. The main reason for this procedure will be to get rid of falsely positive occurrences. In order to obtain the features of detected objects, feature extractors must be used. [17]

4.3.1 Feature extraction using Resnet and VGG

The basic architectural blueprints of convolutional neural networks are described in the chapter 3.3. Both Resnet50 and VGG16 networks comply with the basics while having some modifications that slightly alter their function and capabilities. [38]

VGG16 network is a bit older than Resnet but is still used by most of the users trying to extract and possibly classify image features. It consists of 16 convolutional layers, where each layer performs 3x3 convolution. The input image is resized to 224x224x3 and then convoluted and pooled to reach a size of 1x1x1000 that can be classified if needed. [38]

Resnet50 also known as residual network has a slightly different function. Resnet neural network can use much larger number of layers than previously mentioned VGG – up to 152. With a regular network, when large number of layers is used, the network accuracy gets saturated and because of this, it cannot be trained as well. This problem can be solved using so called “shortcut connections”. These connections can skip one or more layers and their output can help with managing training through back-propagation. [38], [39]

5 PROPOSED EVALUATION OF RESULTS

The results of particle picking done by an algorithm that will be constructed during the second part of this thesis will be evaluated using RELION program provided by Thermo Fisher Scientific. This program was developed by MRC Laboratory of Molecular Biology and its main purpose are 3D reconstructions or 2D class averages. However, before any advanced techniques may be applied on the images acquired by transmission electron microscope, structures in the image must be detected – picked, which is the main focus of this master's thesis. That is the reason why RELION may be used as an evaluating tool. [19]

The algorithm is template-based, and all the templates are CTF-corrected, which means that the Contrast Transfer Function of the micrograph is applied to the template images. Thanks to this, it has a good potential of finding particles even in noisy data. Since the approach is semi-automatic, it needs a little input from the user. The user needs to manually pick particles from a small amount of already processed micrographs. 2D class averaging is then used to determine average images of these particles and then the RELION program automatically picks particles thanks to template-based approach on all micrographs. [19]

After the particles are picked, a sorting algorithm comes into play. It can determine, which particles are picked incorrectly. An associated template and its orientation are subtracted from every extracted particle. The difference is then used to determine several parameters such as mean, standard deviation or skewness of the image. If the particle was picked correctly, the difference image will contain only background noise. If anything else than noise is found in the difference image, the particle was not picked correctly. [19]

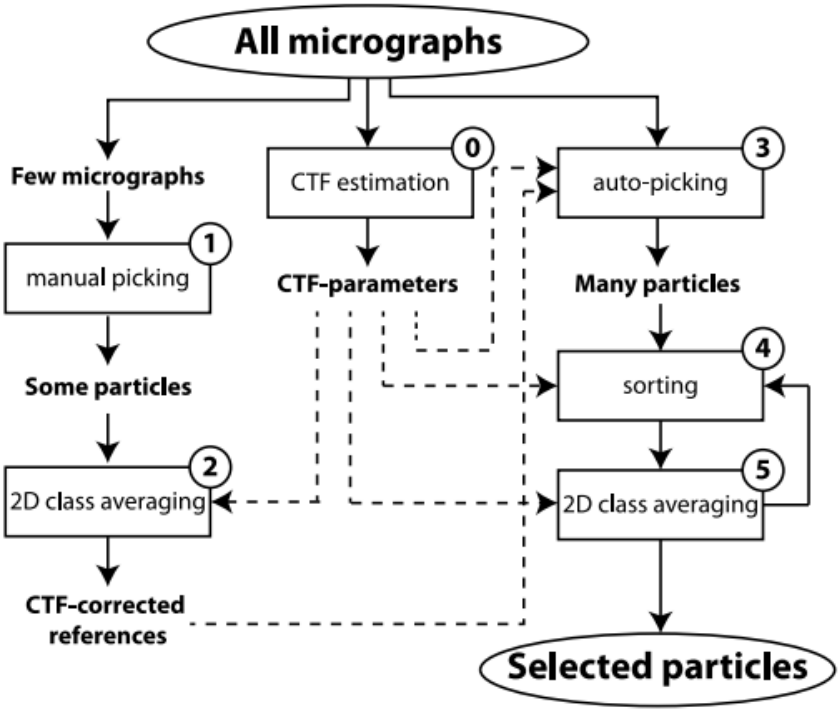


Figure 17: Workflow of the particle-picking stage of the RELION program algorithm. [19]

6 DESING OF THE ALGORITHM

In this chapter, I will present a theoretical design of the algorithm I will be using in the next part of my master's thesis.

6.1 Existing algorithms and their comparison

Several algorithms with similar goals have already been published by other researchers. This sub-chapter will present their basic idea and achieved results.

6.1.1 SLEUTH

SLEUTH is a program used to automatically detect particles in images acquired by electron microscopes. The basic premise of this method is to introduce reference images to this program and to select areas with already specified regions of interest. The program then calculates few simple criteria such as variance or density sum. These values are then compared to the values from a moving window on the examined image – micrograph. When values of the chosen criteria are in a specified range, the area on the micrograph is labelled as a candidate. Areas not labelled as candidates are immediately excluded from the process to minimize occurrence of false positives. [20]

The user needs to provide images with selected structures in boxes that are ideally affected with very little noise. These structures need to be located separately in the image, meaning that they should not be connected in any way. The structures should be in the centre of corresponding boxes so that the criteria parameters are calculated as precisely as possible. After this preparation is complete, the examined micrograph is then presented to the program, which automatically gets rid of unwanted areas such as labels or areas with unexposed film. All remaining areas are then tested regarding the chosen criteria and if they fail one of these test values, they are excluded from the selection. [20]

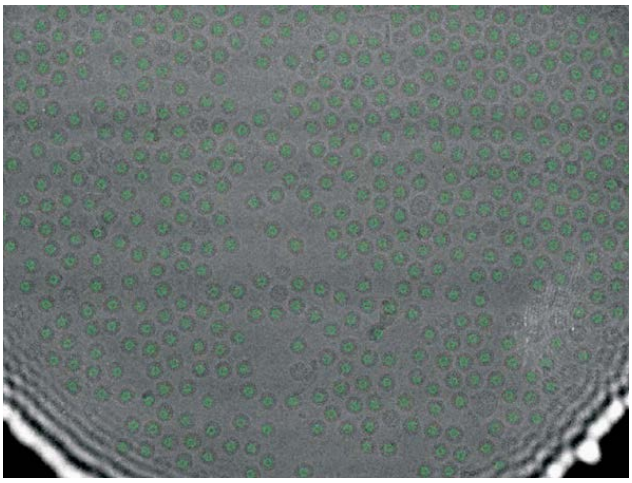


Figure 18: An image of spherical hepatitis B virus cores processed by SLEUTH algorithm. Detected particles are marked by green circles. [20]

When applied to real images, the SLEUTH algorithm can be useful. In average, it achieved 7% of false positive rate and 9% of false negative rate. Usual false positive

occurrences happened due to damaged or non-isolated particles. Processing time is dependent on image size, possible particles and computational power of the machine. [20]

6.1.2 EMAN2

EMAN2 is an image processing package which is able to perform single particle analysis on images acquired by transmission electron microscopes. Unlike previously mentioned SLEUTH algorithm, its primary focus is 3D reconstruction of detected biological structures using single particle tomography (SPT) and related tasks. It can also supply monitoring of structures within observed cells. The algorithm can use different approaches for each step of its functions, which makes it very versatile. [21]

6.1.3 FindEM

FindEM is a particle picking algorithm that can use two different approaches to solve the problem. One of these approaches is attempting to match the new undetected structures in the examined image with structures already detected in older images. This however requires a large number of these previously detected structures to ensure the correct function of the algorithm. It uses Fast Local Correlation Function, which is an algorithm that calculates correlation of provided template structures with areas of the new image. It then creates a local correlation map, where peaks represent potential particle positions. The second approach is to use multivariate statistical analysis (MSA), which is able to filter out potential structures. This approach only needs one template image. [22]

When comparing these two approaches, following results were achieved. The first method using multiple end- and side-view templates achieved 10.6% FPR and 22.8% FNR, while second method using MSA and one side-view template with low correlation threshold achieved 18.3% FPR and 23.1% FNR. [22]

6.1.4 Comparison

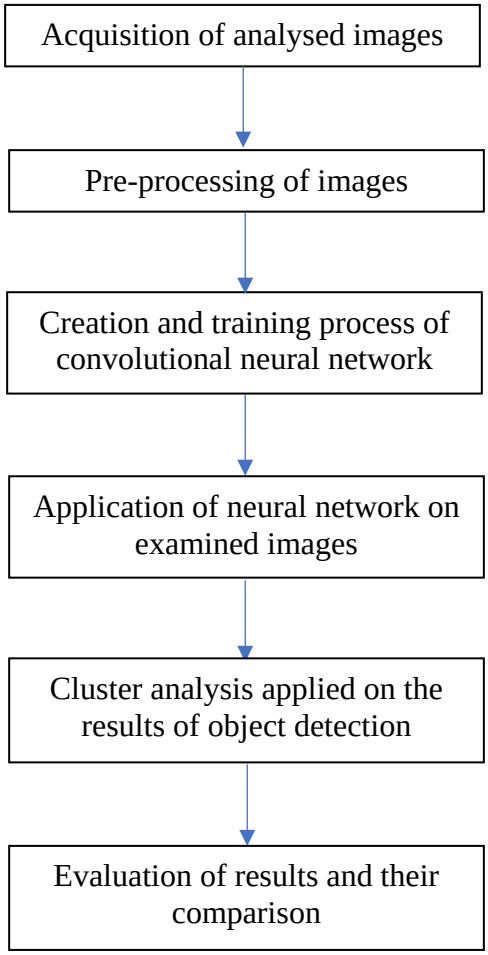
The table below shows false positive and false negative rates of described algorithms. The EMAN2 algorithm is not compared here, since it has fairly different goal than the other two. [20], [21], [22]

Table 1: Comparison of existing methods of particle picking

Algorithm	**SLEUTH**	**FindEM (Method 1)**	**FindEM (Method 2)**
FPR [%]	7	10.6	18.3
FNR [%]	9	22.8	23.1

6.2 Theoretical design of the proposed algorithm

Basic steps of the proposed algorithm, I would like to create in the second part of this thesis, are located in the flowchart below.



The algorithm that will be created during the practical part of this thesis will use neural network models used for object detection and subsequent clustering of their results. Images used during this part will be large-scale images acquired by a transmission electron microscope. The first batch of images will serve as training set for the neural network. These will be already analysed images with proteins that will have been already picked. The training images will be subsequently sliced and resized to serve as a solid training tools for the neural network.

After the neural network is trained, it will be applied on testing set of micrographs. The resulting objects created by this stage are then to be sliced into small images. These small images will then go through feature extraction using a neural network designed for that purpose. The features produced by this stage will subsequently be clustered using k-means algorithm. Clusters containing the least number of images will be deemed as falsely positive occurrences and filtered out of the object detection process. Coordinates of particles produced by object detection both before and after clustering will be then compared to the coordinates of particles detected by the RELION software.

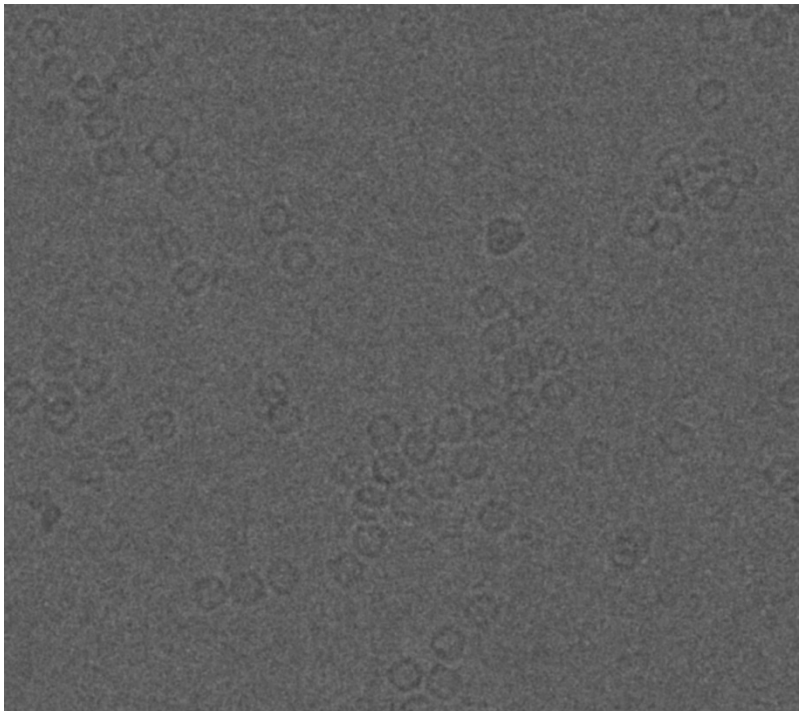


Figure 20: An image with well visible Keyhole Limpet Hemocyanin protein units

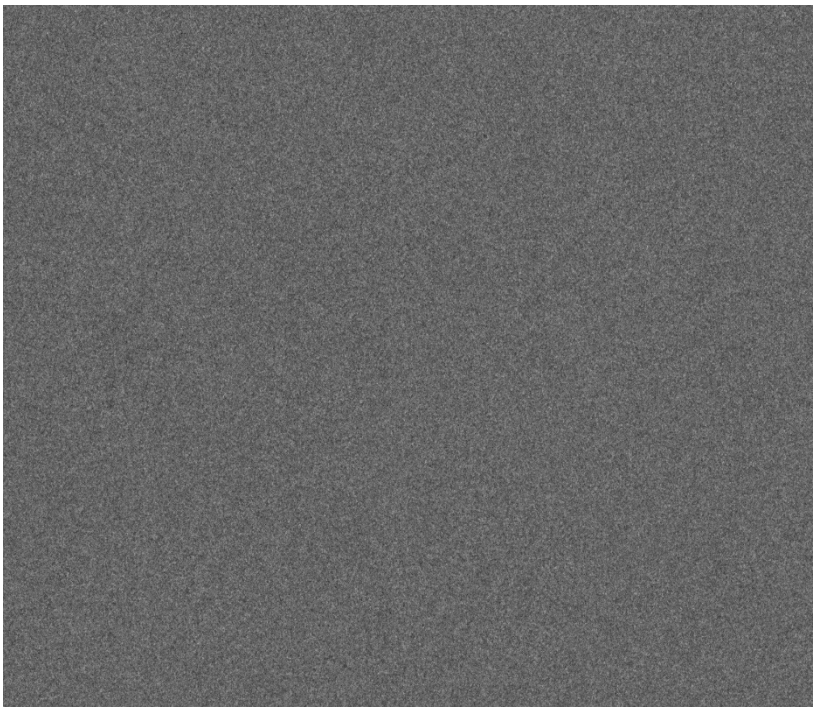


Figure 19: An image of Keyhole Limpet Hemocyanin protein units but with different defocus than Figure 14, which makes the protein units invisible by human eye

7 PRACTICAL PART

This part of the document will include description of tasks that were done in the practical part of the thesis. It was roughly described in the chapter 'Theoretical design of the proposed algorithm' but some changes had to be made. For the first section of the practical part, object detection models using TensorFlow will be used as means for detecting proteins in the images. The second section will include application of cluster analysis to the results of the object detection as an attempt to filter out unwanted outputs of the object detection.

7.1 Creation of training set and test set

In the first stage of the practical part of the thesis, I received 10 4K images of proteins shot by a transmission electron microscope. From these 10 images, 8 of them served as a source for training data and the last 2 of them were used for testing of the trained or fine-tuned models. All the received images had gone through the first part of the protein detection process before I got them. This means, that I had received not only the images themselves but also **.star* files containing the positions of proteins that were detected during the first phase. **.star* is a format that allows to store coordinates of detected structures together with information regarding the acquisition of micrograph such as defocus, spherical aberration or phase shift.

The first stage of the original detection however does not contain any kind of cluster analysis or any similar process and because the first phase of protein detection is very sensitive, some number of false positives were present in each **.star* file. This format also needs specialized software to read correctly but for my purpose, few simple commands using Excel were enough to get a **.txt* file containing X and Y coordinates of every detected object in the image.

The next step was to split the received images into smaller ones, since their original resolution was 4096x4096 pixels, which is too large for the training of an object detection model. Several sizes of images were chosen for the training stage. The first attempts of training an object detection model were done using the Faster RCNN Inception v2 COCO and they were rather ineffective. The training data set consisted of only 256x256 px images that contained at max 2 or 3 proteins per image. This dataset was approximately 500 images large and these images contained no more than 1000 boxes with proteins. All these proteins were of original size – a circles of circa 120 px in diameter and were captured in a box 150x150 px in case they were entirely located in the image. Upon testing the model, it was found out that it is only capable of working with images of the same size as its training dataset (256x256 px). It was therefore decided to enlarge the dataset with larger images with more proteins in them. Sizes of additional images were set on 512x512 and 1024x1024, since it is not recommended to train the models with larger images.

The newly created dataset contained 1763 images of proteins in less than 800 training images. The results after training with this dataset were much more successful. It was able to work with all three sizes of images, on which it was trained on. The largest images (1024x1024 px) with original size of proteins usually contain between 15 and 20 proteins. This number of proteins is much more suitable for statistical analysis and clustering than the number of proteins contained in the smaller images. However, it was

still not able to process larger images such as 2048x2048 px or 4096x4096 px. These images contain much larger number of proteins and would therefore be more statistically significant. Thus, even though it is not recommended, the next dataset used for further training contained roughly 100 2048x2048 px images in hope that it would help the model to be able to detect proteins in larger images.

That did not turn out to be true. The model could spot some structures in the larger images, but there were very little of them and they most certainly were not proteins. However, it was still desirable for the model to be able to detect more proteins than 15-20 it gets from the 1024x1024 px images. Since it seemed impossible for the model to detect objects on anything larger than that, resizing of the original image was introduced. 2 levels of resizing were used, 50% and 25%. All three previously sizes mentioned were used for training images resized to 50% and images resized to 25% were used in versions of 256x256 px and 512x512 px. Altogether, the final dataset was comprised of 1038 images and since some of them contained more than 60 proteins, the final count of proteins used for training stage was 11830.

Images were created thanks to a script in MATLAB. This script was first told, how large the images should be – one of the previously mentioned sizes. Then it searched through the image and assessed whether the current window contains a detected protein. If it did, the image was saved, and the coordinates of the protein or proteins were re-calculated in relation to the position of the window in the processed image. Since the original coordinates pointed to the middle of the protein, a 150x150 px box centred around that coordinates had to be created to satisfy the needs of the training algorithm.

These recalculated boxes were then fit into a **.csv* files that is used as an input to the training process. These files are called *test_labels.csv* and *train_labels.csv*. Each file contains names of the files in one column and width and height of the used images in the next two columns. The fourth column contains a description of the class of the object(s) identified in the image. Since the model was supposed to be only trained to identify protein, there is only one class used throughout both documents. The last four columns describe the coordinates of the object found in the image. There may be more than one object and if that is the case, the image is listed in the **.csv* file once per every object found within.

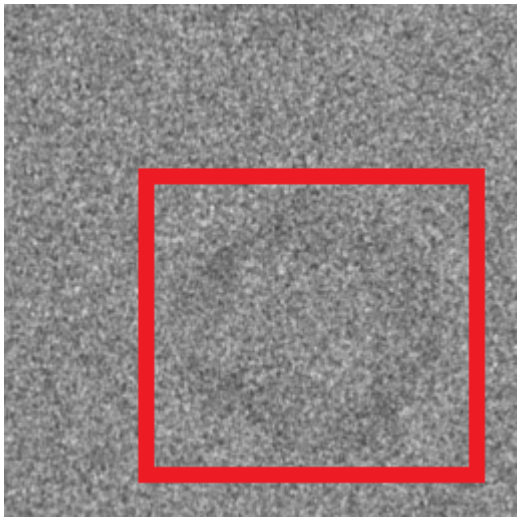


Figure 21: Example of a 256x256 image with a well visible protein used for the training stage

7.2 Setting up the training

Before training an object detection model using TensorFlow can be commenced, Object Detection Repository that contains all the necessary utility functions and model configuration files has to be downloaded. Installation manual is described online, and its shortened version will be present in the attachment of this thesis.

As described earlier in chapter Creation of training set and test set, **.csv* files containing sizes and names of images and coordinates of proteins within the images must be developed. However, TensorFlow cannot work with these **.csv* files and they need to be transformed to **.record* files using a script *generate_tfrecord.py* that is supplied by the creators of TensorFlow. This script needs to be changed in order to suit the needs of the assignment. Names and corresponding id of classes must be set. Using this script, two **.record* files are created – *test.record* and *train. record.*

After the **.record* files are set up, it is necessary to choose a model that will be trained or more precisely fine-tuned to suit our needs. Most of the models used for object detection are either based on SSD MobileNet and Faster RCNN models, which are both described in previous chapters. These models were trained on datasets containing hundreds of thousands of labelled images in order to be able to extract suitable features for object detections. Theirs variations can differ regarding how fast the training proceeds and on what dataset were they originally trained.

When the model is downloaded and properly placed, *labelmap.pbtxt* that contains id and names of the classes, which are to be detected, needs to be configured – very similar task to when the *generate_tfrecord.py* was altered. Then the **.config* file belonging to the chosen model has to be edited to contain paths to folders containing training images, **.record* files and label map. There are several other parameters that can be manipulated with in the in the **.config* file. These parameters include learning rate, which is a parameter that determines how fast or slow should the parameters of the neural network be updated during the training. It can be changed during the training, e.g. after a number of steps. Another parameter is second stage IoU (Intersection over Union) threshold that determines, whether the proposed bounding boxes of potentially detected objects can or cannot be passed into further analysis and classification. The amount of proposals that are considered for further classification can also be changed, just as the activation functions used by the classifier. Choosing the activation function is in most cases a choice between two variants, a simpler sigmoid function and a more complicated softmax function.

The TensorFlow *train.py* script is used to start the training after all necessary steps are completed. When launching this script, the directory containing label map and **.config* file must be stated. The progress of the training can be monitored using TensorBoard, which is an interface that can plot different loss functions and their evolution in time together with other values that are used during the training.

When the training is over, *export_inference_graph.py* is used to export the last inference graph that was saved. This is done because the inference graph of the model is changed during the fine-tuning and when the result of the loss function is low enough, the training can be stopped since it is presumed that the model is ready to be used. The value of the loss function low enough for ending the training is set for 0.05 in case of the Faster RCNN model and its derivatives and between 1 and 2 for the MobileNet networks. The altered inference graph is saved every 10 minutes and is described using the step of the training during which it was saved. This parameter is also needed when launching the script for graph export.

7.3 A way of evaluating results of the object detection

As described earlier, this stage of the practical part does not include the cluster analysis of the results. However, it contains an evaluation of the application of object detection using different models and settings. This application and evaluation were done using the script.

This script needs a user input regarding the placement of files such as the exported inference graph and the location of images, on which the model is to be tested. These images come from the same dataset as the ones, on which the models were fine-tuned but they were not used for the process. When all the necessary pre-requisites are taken care of, the script launches a TensorFlow session. During this session, the inference graph of the chosen model is used for object detection and calculates position of the potential proteins and their confidence score. Parameters of the boxes around the proteins are then used to calculate coordinates of the newly detected proteins. The script then reads the coordinates of the originally detected proteins from a **.txt* file that was created using a script, which was very similar to the one that was used to create the training dataset. Centres of both the new and original coordinates are then calculated and compared.

When a centre of new coordinates is found in the original ones with a tolerance of 20x20 pixels in the original image size, it is assessed as a true positive. When a new coordinate is not found in the original set, it is labelled as a false positive. False negatives are calculated as the difference between the number of original coordinates and true positive ones from the newly detected set. However, it is not possible to count the number of true negatives, since there is no original set of objects marked as truly negative occurrences. Therefore, to gain some sort of an idea about how effective the model is, values of sensitivity and precision (also known as positive predictive value) are calculated.

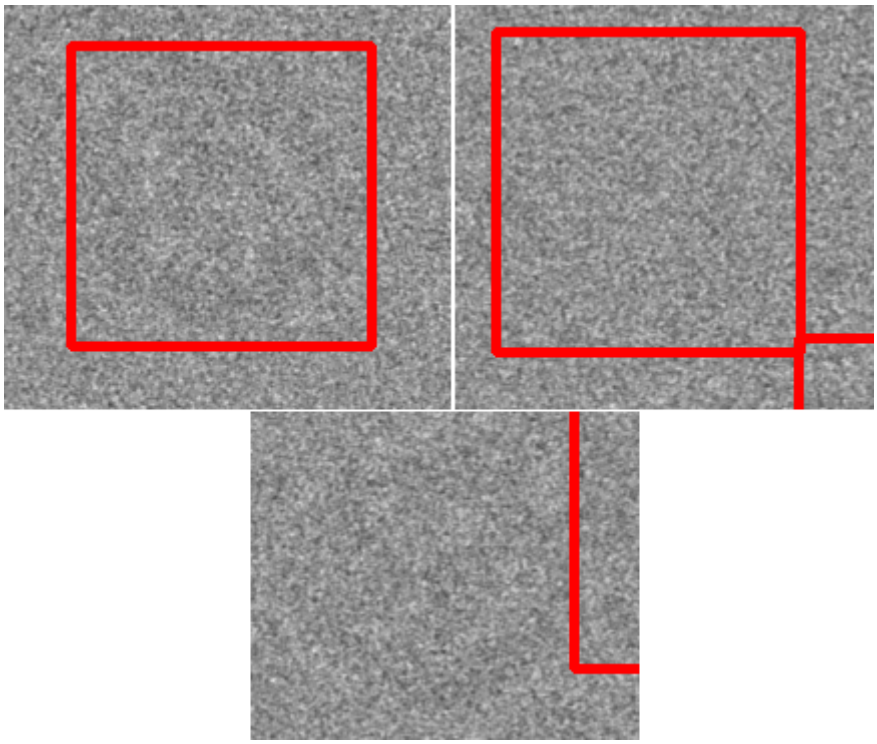


Figure 22: Top left = true positive, top right = false positive, bottom = false negative

$$sensitivity = \frac{TP}{TP+FN}$$ Equation 4

$$precision = \frac{TP}{TP+FP}$$ Equation 5

Figure 22 shows examples of true positive, false positive and false negative occurrences. True negative occurrences are not shown in this figure, since they are not calculated with.

7.4 Results after using versions of Faster RCNN

In this chapter, I will describe the results after using different forms of Faster RCNN with a short description of the model and results of the detection performed on images that were not part of the training process – those are derivations of images 1952.png and 1955.png. The dataset used for this detection is described in the previous.

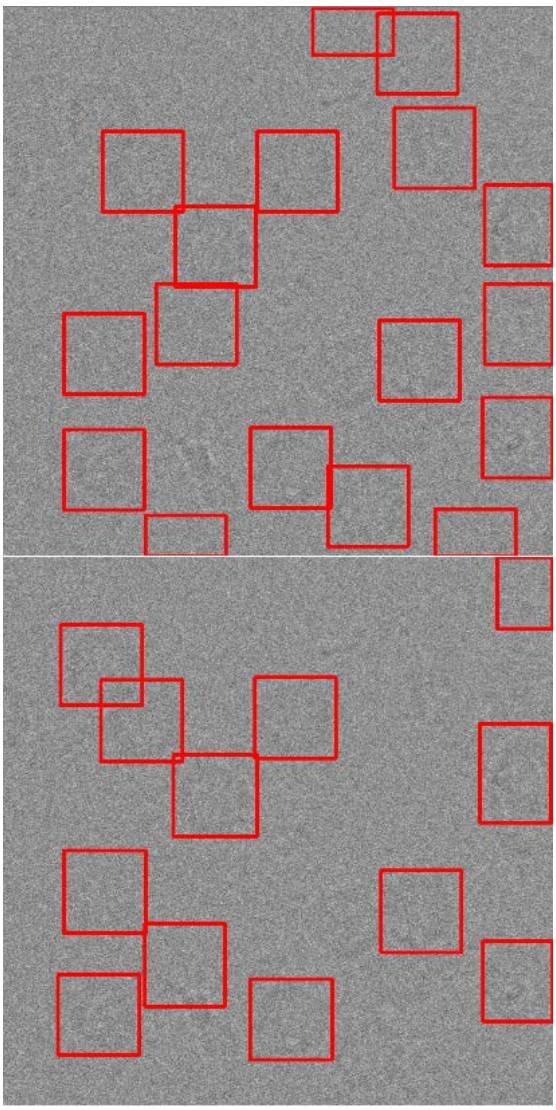


Figure 23: Top image = originally detected proteins, bottom image = newly detected proteins in 1952_1.png. Original size of the image is 1024x1024 px.

7.4.1 Faster RCNN Resnet50

Faster RCNN Resnet50 is a version of Faster RCNN model trained on COCO dataset. This object detection network is described in chapter 3.4.2. This variation uses Resnet network described in chapter 4.3.1 as a feature extractor. In this case, Resnet50 is used for, which has 50 convolutional layers. These convolutional layers create feature map that is fed into region proposal network. Training parameters of this network were set for a maximum of 300 proposals. These proposals are then subsequently classified as foreground or background, with the threshold value being 0.6, and the foreground objects are given confidence score. Learning rate was set to 0.0003 for first 90000 steps, 0.00003 for next 30000 and the rest of the steps had a learning rate of 0.000003. A more complicated softmax function was chosen as an activation function for the classifier. This model was being fine-tuned for 12.5 hours until it reached the recommended 200 000 steps.

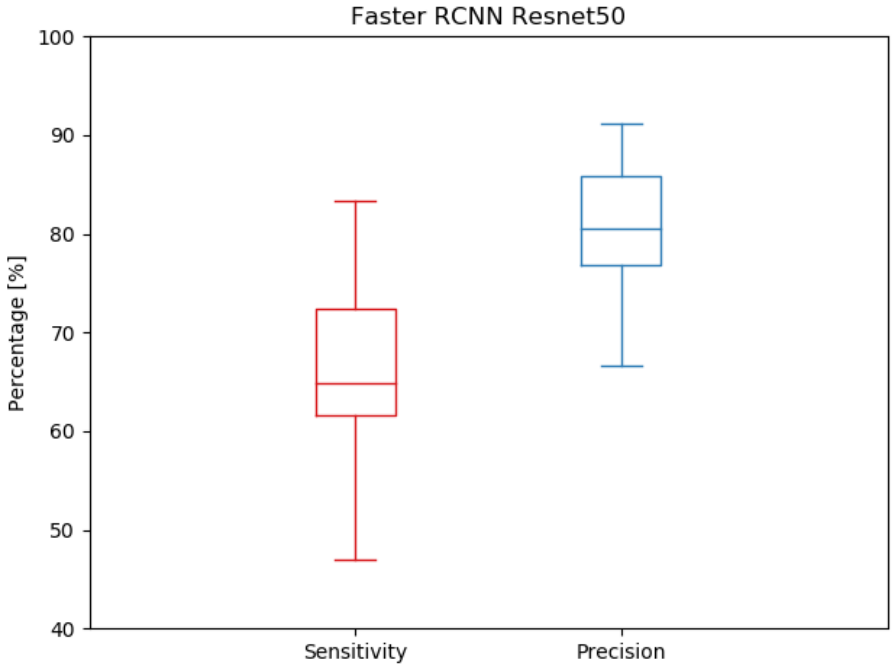


Figure 24: Sensitivity and precision boxplot graph of Faster RCNN Resnet 50 results

As you can see in the Figure 23, roughly two thirds of the originally detected proteins were also recognized by the fine-tuned Faster RCNN Resnet50. Below, you can see a table of results of the detection on other images. The confidence score threshold was set on 50%.

On the figure 24, you can see boxplot representation of results achieved by Faster RCNN Resnet50 model. It averaged sensitivity of 65.98% and precision of 80.98%. The values of precision are also more condensed. This means that the model missed roughly a third of all objects to be detected, but it was quite successful in regards that more than 80% of all detected proteins were actually there – truly positive.

Table 2: Results of protein detection using a fine-tuned Faster RCNN Resnet50 model. Images are in a **.png* format. Since the images are square, sizes are given by the length of one of the 4 equal edges. No. orig. is the number of the originally detected proteins, No. new is the number of proteins detected by the fine-tuned model. Sensitivity and precision values are shown in percentage [%].

Name	Size	Resize	No. orig.	No. new	Sensitivity	Precision
1952_1	1024	1	17	12	47.06	66.67
1952_2	1024	1	19	17	68.42	76.47
1952_3	1024	1	19	14	63.16	85.71
1952_4	1024	1	18	18	83.33	83.33
1952_5	1024	1	19	13	53.63	76.92
1955_1	1024	1	16	11	56.25	81.81
1955_2	1024	1	9	7	55.56	71.43
1955_3	1024	1	17	15	76.47	86.67
1955_4	1024	1	15	13	60	69.23
1955_5	1024	1	17	16	70.59	75
1952_05_1	1024	0.5	68	45	60.29	91.11
1952_05_2	1024	0.5	60	48	71.67	89.59
1952_05_3	1024	0.5	68	54	61.76	77.78
1952_05_4	1024	0.5	66	45	62.12	91.11
1952_05_5	1024	0.5	63	52	69.84	84.62
1955_05_1	1024	0.5	52	47	75	82.98
1955_05_2	1024	0.5	74	63	77.03	90.48
1955_05_3	1024	0.5	71	53	59.15	79.25
1955_05_4	1024	0.5	62	51	74.19	90.2
1955_05_5	1024	0.5	73	64	75.34	85.94
1952_025_1	512	0.25	64	47	64.06	87.23
1952_025_2	512	0.25	62	52	66.13	78.85
1952_025_3	512	0.25	62	58	72.58	77.59
1952_025_4	512	0.25	63	56	63.49	71.43
1952_025_5	512	0.25	70	56	67.14	83.93
1955_025_1	512	0.25	64	56	65.63	75
1955_025_2	512	0.25	73	62	72.6	85.48
1955_025_3	512	0.25	73	57	61.64	78.95
1955_025_4	512	0.25	74	59	62.16	77.97
1955_025_5	512	0.25	68	56	63.24	76.79

7.4.2 Faster RCNN Inception v2

This model is also a faster RCNN based model that uses an Inception modification in its feature extraction process. This model should be slightly faster than the previously mentioned Faster RCNN Resnet50 model. But this increased speed is supposed to be accompanied by a decrease in accuracy. The IoU threshold was set to 0.6 and maximum number of proposals to 300. The learning rate was set to 0.0002 for first 90000 steps, 0.00002 for next 30000 steps and 0.000002 for the rest of the training. Activation function for classifier was set to softmax. The model was being fine-tuned for little less than 8 hours until it reached 200 000 steps. The same testing images and algorithms were used to evaluate the functionality of this model and the results can be seen in the boxplot on figure 25.

The sensitivity and precision values were recorded in the same way as with the Faster RCNN Resnet50 fine-tuned model. This time, precision averaged 71.37%, while sensitivity only reached an average of 58.39%.

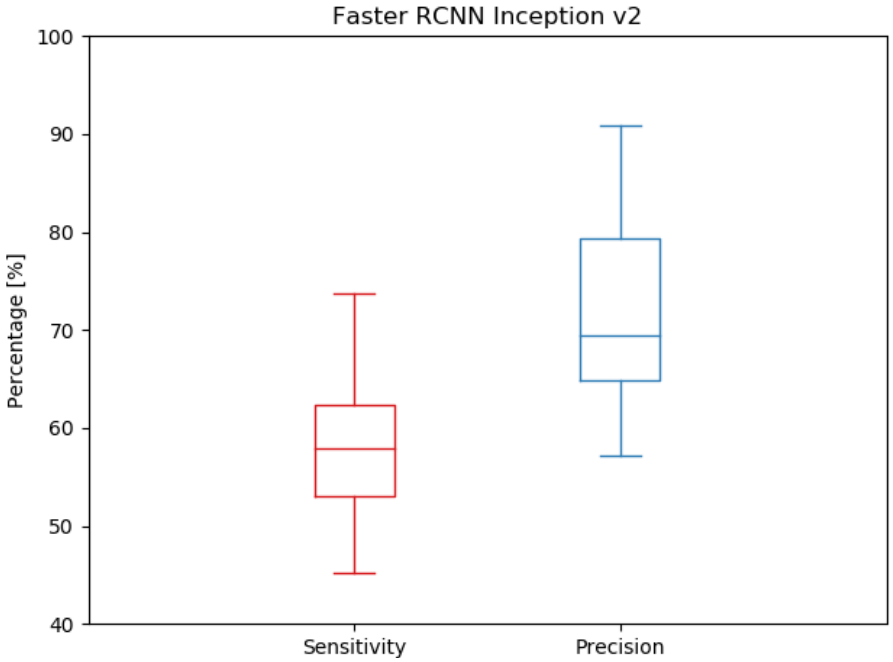


Figure 25: Sensitivity and precision boxplot graph of Faster RCNN Inception v2 results

7.5 Results after using RFCN model

Unlike Faster RCNN, which derives the regions of interest directly from feature map and thus applying the computation on every region of interest separately, the RFCN model applies the computation on the entire image. It uses a fully convolutional network independent of the regions of interest. The architecture of feature extractor for this model is based on Resnet101. It uses 100 convolutional layers to compute the feature maps. The last block before generating score maps is size reduction convolutional layer. Feature extraction is followed by region proposal network, which calculates the regions of interest and is fully convolutional on its own. Regions of interest are then pooled, scored and differentiated between background and foreground and subsequently classified. IoU threshold set for this training was 0.6, maximal number of proposals was 300. Learning

rate was the same as with the Faster RCNN Inception v2 model, 0.0003 for first 90000 steps, 0.00003 for next 30000 steps and 0.000003 for the rest of the training. Softmax was chosen as the activation function of the classifier. This model was fine-tuned for more than 12 hours to reach a sufficiently low error margin. [40]

On the figure 26, there is a boxplot graph summarizing the results of testing of the fine-tuned RFCN model. The model achieved fairly high values of both sensitivity and precision with small variance. Sensitivity averaged at 71.79% and precision reached an average of 82.33%. On the figure 27, you can see the actual results of detection by the fine-tuned RFCN model. The image was resized to a quarter of the original size, resulting in a size of 512x512 pixels and containing more than 62 originally detected and 58 newly detected proteins.

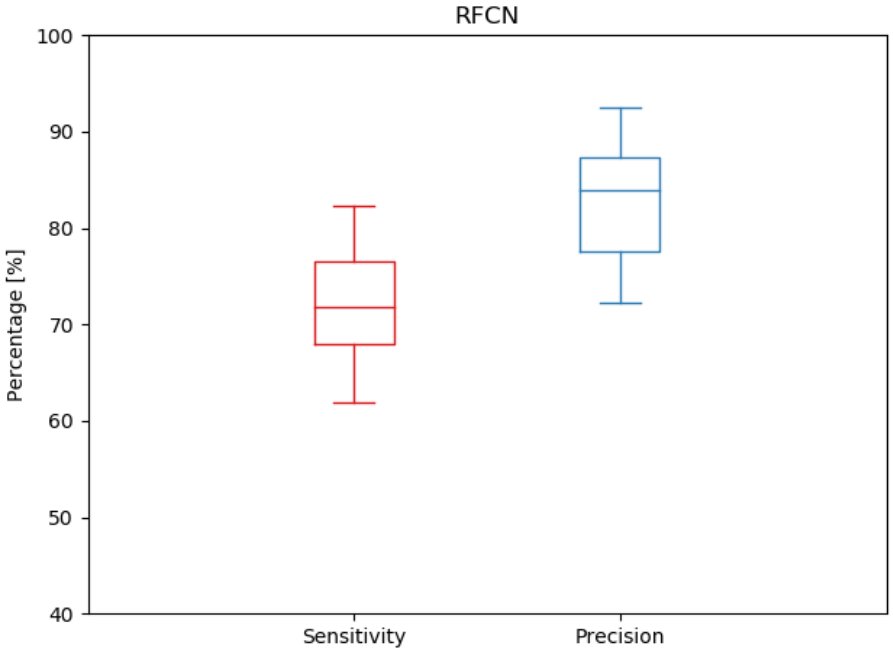


Figure 26: Sensitivity and precision boxplot graph of RFCN results

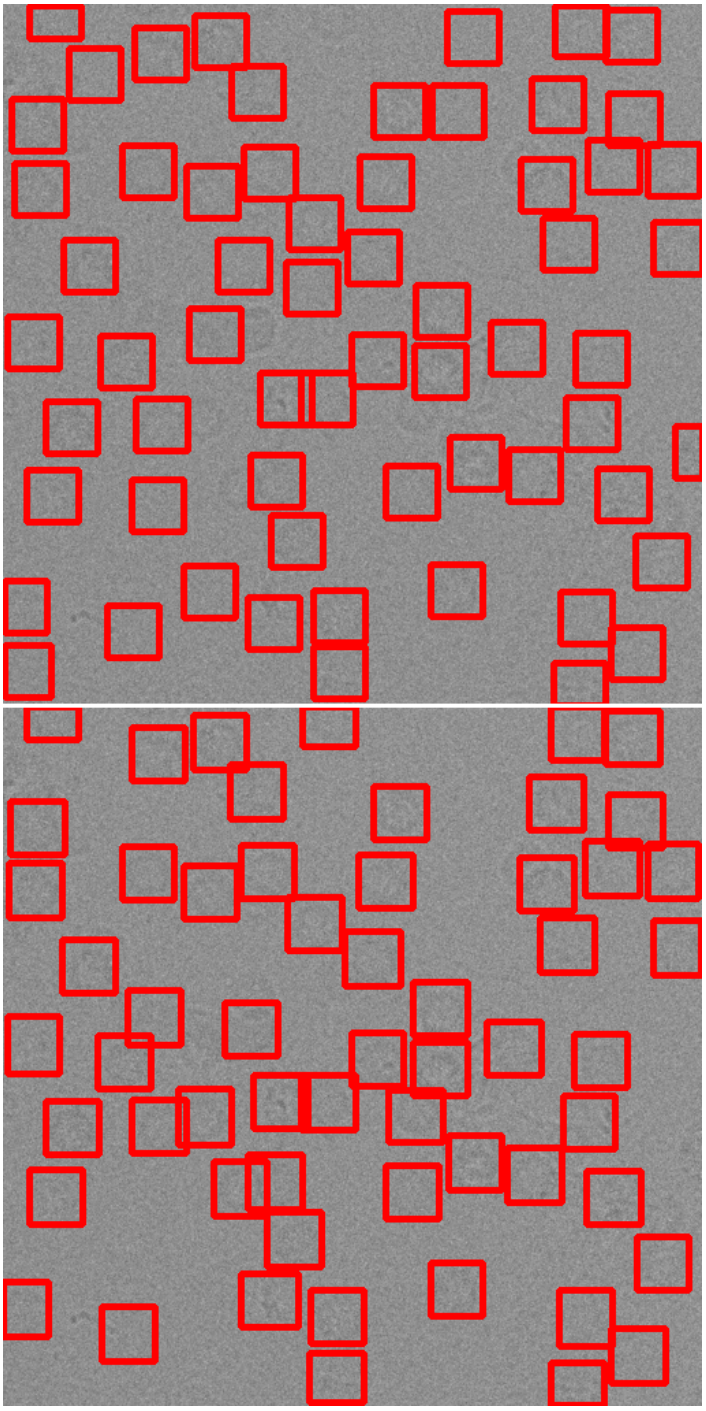


Figure 27: Top image = originally detected proteins, bottom image = newly detected proteins in 1952_025_3.png. Original size of the image is 512x512 px. Resize: 0.25

7.6 Results after using MobileNet SSD v2

The last model that was fine-tuned was the MobileNet SSD v2, which is more closely described in the theoretical part of this thesis. The model was originally trained on the same COCO dataset as the rest of these models. Parameters of the training were set to a

maximum of 100 proposals, IoU threshold to 0.6 and learning rate to 0.004, which did not change for the duration of training. The activation function of the classifier was defined as sigmoid. Fine-tuning by the dataset created for the purposes of this thesis took more than 14 hours. However, the loss margin did not decrease to the intended value of 1 but stayed fairly high at around 2.5. significant. This model is comprised of 17 basic MobileNet building blocks (one of these blocks is shown on figure 14), followed by 1x1 standard convolutional layer, pooling layer and a classifier.

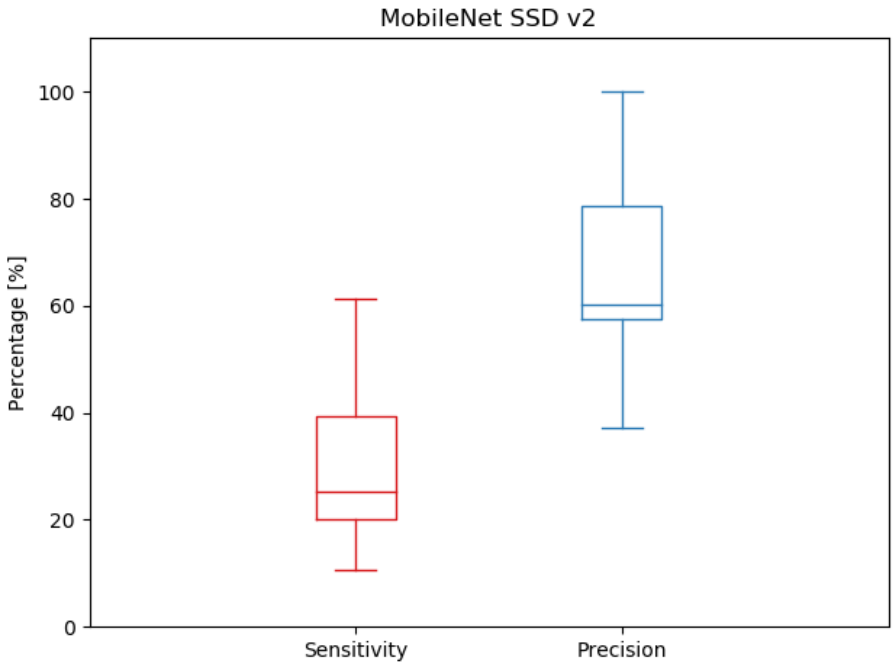


Figure 28: Sensitivity and precision boxplot graph of MobileNet SSD v2 results

Results of testing of MobileNet SSD v2 can be seen on Figure 28. It is apparent that both sensitivity and precision values have large variance and are in average much smaller than in the cases of previously mentioned models. The average of sensitivity is only 30.87% and the average of precision reached 67.98%. The model is able to detect objects in an image much faster than the rest of the tested models but there is a decrease in performance.

7.7 Comparison of methods

In this chapter, the comparison of performance of models mentioned in previous chapters is shown using the table 3 and a boxplot graph on figure 29.

Table 3: Comparison of average results achieved by object detection models mentioned in a few previous chapters.

Model name	Average sensitivity [%]	Average precision [%]
Faster RCNN Resnet50	65.98	80.98
Faster RCNN Inception v2	58.39	71.37
RFCN	71.79	82.33
MobileNet SSD v2	30.87	67.98

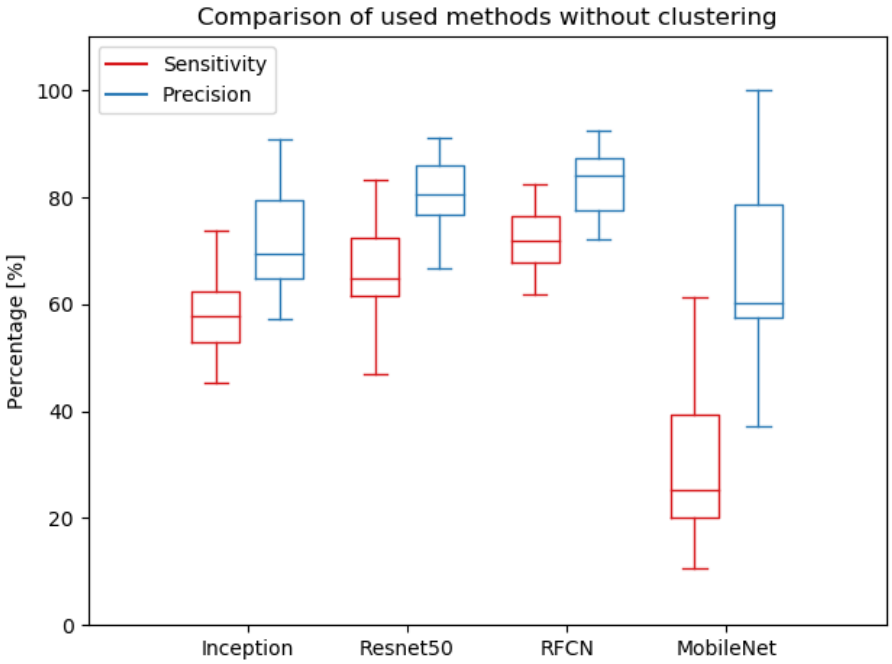


Figure 29: Sensitivity and precision boxplot graph of all used models

Both Table 3 and Figure 29 present the comparison of results achieved by the fine-tuned models. It is apparent that MobileNet SSD v2 model achieved the worst precision and sensitivity and for that reason is by far the worst in regards of its performance. The speed of the model is however much greater than of any other used model. The remaining three models are somewhat similar performance-wise. Nevertheless, the best performance was achieved by the fine-tuned RFCN model, since it reached the highest average sensitivity, precision. Its results are also much more consistent, which means that its high sensitivity and precision values are no accident.

It is necessary to mention that the order, in which the models take place regarding their performance is very consistent with how large the models are and how long did they take to train effectively. The RFCN model is by far the largest, almost twice as large as Faster RCNN Resnet50 in terms of size in megabytes. On the other hand, MobileNet SSD v2 is quite small and simpler model and that is perhaps the reason its performance is so weak and ineffective.

All of the models used in this part were analysed by their creators, stating their speed during object detection and also their performance using mean average precision score (mAP). This score is calculated from the relation between precision and sensitivity (also known as recall). The general rule says that the higher the sensitivity, the lower the precision. This is caused by the increase of false positive occurrences in the cases with higher sensitivity. During the calculation of AP, the sensitivity value is changed several times. These alternations in sensitivity are achieved by changing the threshold that differentiates regions of interest between foreground and background. The models were originally trained on real life objects such as people, cars or dogs, creating multiple classes for object detection. The AP score is then calculated for each class and since these models were trained on COCO dataset, a validation set created from these images was used for these calculations. The mAP score is then calculated as the mean value of AP values for each class. [45]

The formula for calculation of average precision score is shown below.

$$AP = \frac{1}{n}\sum_{Sensitivity_i} Precision(Sensitivity_i) \qquad \text{Equation 6}$$

The variable *n* stands for number of different sensitivity values that were used during the calculation. Usually, there are 5-10 of theses different values. Another vital parameter of the used models is their speed. Their authors measured the object detection speed of models using 600x600 px images. The speed values are shown in milliseconds and will be different for every hardware configuration of the computer, on which they run. The relative changes in between them should however stay the same. The table below shows the speed and mAP scores of used models: [45], [46]

Table 4: Speed and mAP scores of models used for object detection

Model name	**Speed [ms]**	**mAP score [%]**
Faster RCNN Inception v2	58	28
Faster RCNN Resnet50	89	30
RFCN	92	30
MobileNet SSD v2	31	22

During the usage of these models on problem discussed in this thesis. The relative speed of the models stayed pretty much the same, with MobileNet SSD v2 being the fastest by a fairly large margin. They are slight differences between values of performance given in the table above and values measured during the experiment contained in this thesis. The table suggests that RFCN and Faster RCNN Resnet50 model should have the same performance but that does not comply with the results achieved in this thesis, where RFCN is clearly the more powerful model. However, this can be explained by the fact that images and objects analysed in this thesis are very different to the ones used during the calculations stated in the table above.

8 APPLICATION OF CLUSTER ANALYSIS

In this part of the thesis, I will write about the reason, application and results of applying cluster analysis on the recorded data, which are described in the previous chapter. Some information about this procedure are described in the theoretical part of this thesis, more specifically in chapter 4.3 K-means used for clustering of images. The whole purpose of the application of cluster analysis is to exclude incorrectly detected objects – false positives that will ideally end up in different clusters than correctly identified objects – true positives.

The application of cluster analysis is in this case performed by the script *OD_and_Slice.py* that calls upon functions from scripts *Cluster_RN.py* and *Cluster_VGG.py*. The first script works very similarly to the script used for evaluation of results of object detection carried out by the fine-tuned models. It also loads up the inference graph of the model, the label map and the test image. Then it carries out the object detection and saves the coordinates of newly detected proteins. However, unlike the script mentioned in the previous chapter, it does not directly evaluate the results of the detection. Instead, it makes small images of all newly detected boxes that are to be used for clustering and saves then into a folder defined by the user.

After the images are created, functions from clustering scripts come into play. These scripts firstly load up the small images representing detected boxes. Then a feature extractor – either Resnet50 or VGG16 – is used to extract features of all the images. These features are then clustered by the K-means algorithm into 5 clusters. Out of these clusters, the rarest is found and coordinates belonging to the images that are located in this cluster are then deleted. The newly created coordinates after clustering are then subjected to the same evaluation as the coordinates described in the previous chapter – sensitivity and precision values are calculated.

The reason for selecting the rarest cluster, in other words the cluster containing the least images, is that the precision achieved by the models is fairly high. This means that it would not be wise to get rid of some large clusters, because that would almost definitely delete some correctly detected objects. As mentioned in the beginning of this chapter, the cluster analysis is applied in order to exclude false positives. It cannot work as a tool with the ability to detect new objects, sensitivity there cannot be increased. The only parameter that can be made better is the rate between false positives and true positives – the precision.

8.1 Cluster analysis using Resnet50 network as feature extractor

This chapter will describe a cluster analysis of detected objects using Resnet50 as the feature extractor. K-means will create 5 clusters and the cluster containing the least images will be excluded. If there are two or more clusters containing the same and smallest number of images, the cluster to be excluded will be chosen randomly. In the Table 5, you can see the calculated values of precision and sensitivity achieved by the RFCN model both before and after clustering of the results. A new parameter added to evaluate the success of the clustering will be the F1 score – harmonic mean of precision and sensitivity. The F1 score is calculated using the formula below.

$$F1 = 2 * \frac{(Sensitivity * Precision)}{(Sensitivity + Precision)}$$ Equation 7

Table 5: Results of protein detection using a fine-tuned RFCN model before and after clustering. Images are in a **.png* format. Since the images are square, sizes are given by the length of one of the 4 equal edges. Sensitivity, precision and F1 score values are shown in percentage [%].

	Before Clustering			**After Clustering**		
Image name	**Sensitivity**	**Precision**	**F1**	**Sensitivity**	**Precision**	**F1**
1952_1	52.94	60	56.25	52.94	64.29	58.07
1952_2	78.85	75	76.88	73.68	77.78	75.67
1952_3	68.42	76.47	72.22	63.16	80	70.59
1952_4	77.78	77.78	77.78	77.78	87.5	82.35
1952_5	68.42	72.22	70.27	63.16	70.59	66.67
1955_1	81.25	81.25	81.25	81.25	86.67	83.87
1955_2	66.67	75	70.59	66.67	85.71	75.00
1955_3	82.35	77.78	80.00	76.47	76.47	76.47
1955_4	73.33	73.33	73.33	66.67	71.43	68.97
1955_5	76.47	86.67	81.25	70.59	85.71	77.42
1952_05_1	70.59	85.71	77.42	69.18	87.04	77.09
1952_05_2	78.33	90.38	83.92	71.67	89.58	79.63
1952_05_3	72.06	87.5	79.03	69.18	87.04	77.09
1952_05_4	74.24	92.45	82.35	72.73	94.18	82.08
1952_05_5	74.6	88.68	81.03	73.01	88.46	80.00
1955_05_1	67.31	77.78	72.17	63.46	78.57	70.21
1955_05_2	71.62	85.48	77.94	70.27	88.14	78.20
1955_05_3	61.97	77.19	68.75	60.56	79.63	68.80
1955_05_4	67.74	91.3	77.77	67.74	93.33	78.50
1955_05_5	76.71	90.32	82.96	69.86	89.47	78.46
1952_025_1	76.56	90.74	83.05	76.56	92.45	83.76
1952_025_2	72.58	83.33	77.58	72.58	84.91	78.26
1952_025_3	80.64	86.21	83.33	79.03	87.5	83.05
1952_025_4	71.43	77.59	74.38	68.25	78.18	72.88
1952_025_5	71.42	83.33	76.92	71.43	84.75	77.52
1955_025_1	68.75	84.62	75.86	68.75	86.27	76.52
1955_025_2	75.34	90.16	82.09	73.97	91.53	81.82
1955_025_3	65.75	85.71	74.41	63.01	85.19	72.44
1955_025_4	63.51	87.04	73.44	62.16	86.79	72.44
1955_025_5	66.18	78.95	72.00	66.18	80.36	72.58
Average	**71.79**	**82.33**	**76.54**	**69.4**	**83.98**	**75.88**

Figure 31: Sensitivity, precision and F1 score of RFCN model before and after clustering with Resnet50 as feature extractor

In the table 5 and figure 31, you can see that precision improved by clustering and sensitivity worsened, just as expected. However, there was a slight decrease in the F1 score, which means that clustering features produced by Resnet50 network did not improve overall results of the object detection carried out by the RFCN model.

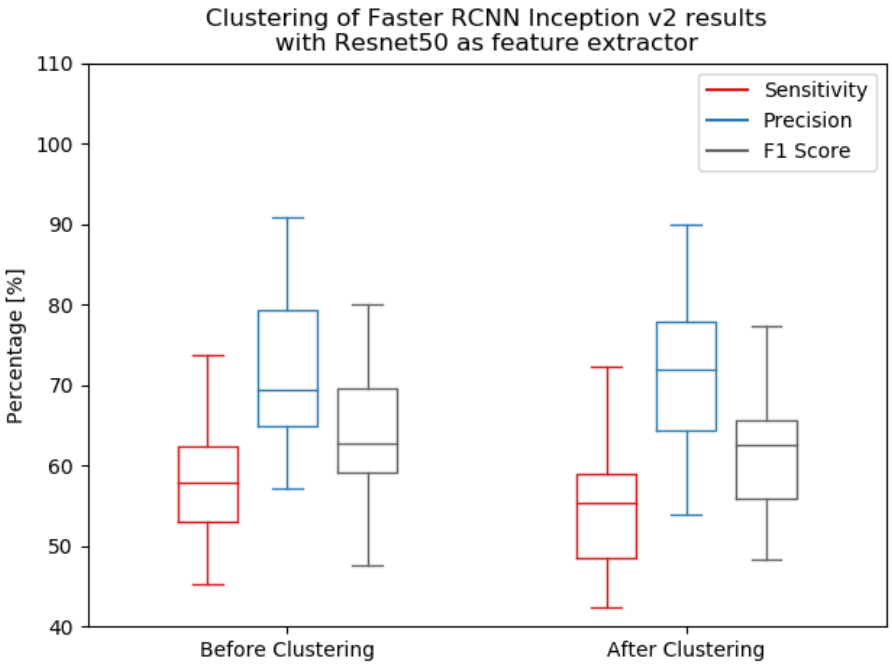


Figure 30: Sensitivity, precision and F1 score of Faster RCNN Inception v2 model before and after clustering with Resnet50 as feature extractor

Unfortunately, clustering using Resnet50 as feature extractor did not improve results of the Faster RCNN Inception v2 either. There was only a slight improvement

regarding precision but larger decrease in sensitivity. F1 score before clustering reached 63.87% and 62.3% after clustering, which is worse by more than 1.5%.

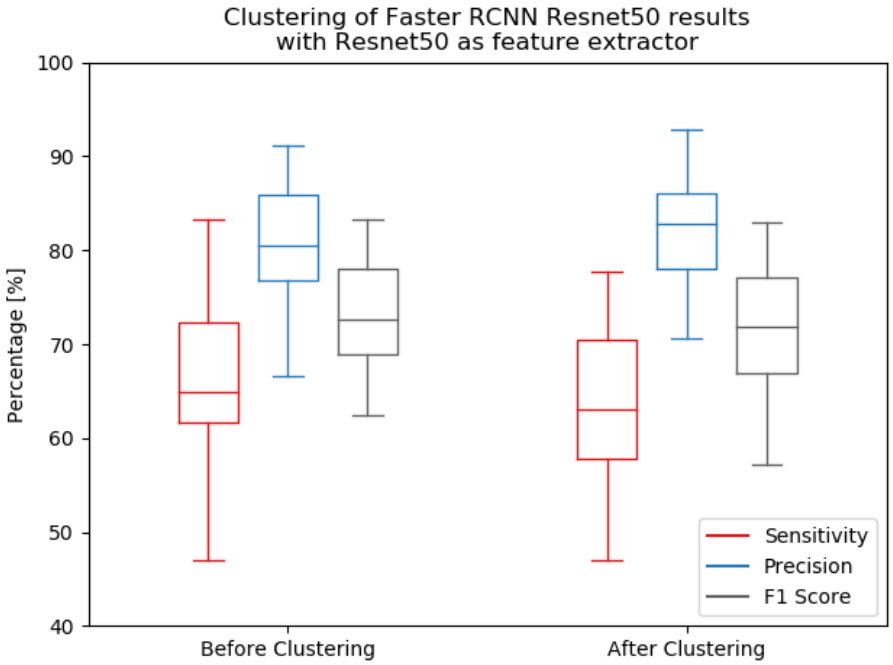


Figure 32: Sensitivity, precision and F1 score of Faster RCNN Resnet50 model before and after clustering with Resnet50 as feature extractor

Figure 32 shows a boxplot representation of object detection results of Faster RCNN Resnet50 model before and after K-means clustering with Resnet50 as feature extractor. Once again, the clustering did not help the overall performance of the object detection. The average F1 score reached 72.53% before the clustering and only 71.42% after it.

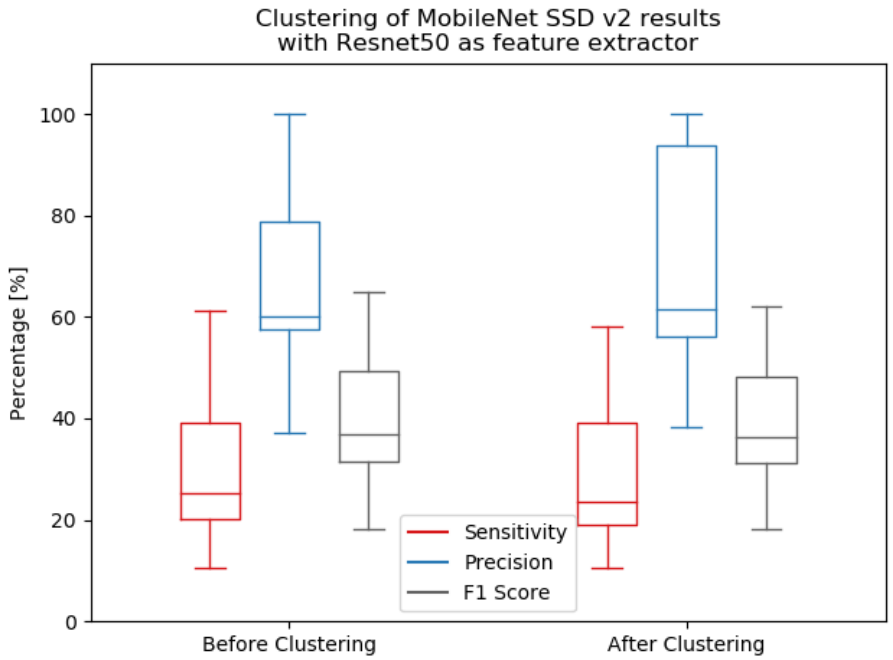


Figure 33: Sensitivity, precision and F1 score of MobileNet SSD v2 model before and after clustering with Resnet50 as feature extractor

As you can see on the Figure 33 above, MobileNet SSD v2 model did not do very well during object detection, and clustering did not improve the results either. Before and after clustering F1 scores are 40.19% and 39.32% respectively. In some cases, the object detection by MobileNet SSD v2 was so ineffective, it did not produce enough objects from an image to be clustered (5 or more). Whenever that happened, the clustering step was not performed for that particular image.

8.2 Cluster analysis using VGG16 network as feature extractor

This chapter will be very similar to the previous one. The only difference is that the k-means clustering was applied on features produced by VGG16 feature extractor rather than Resnet50. The results were collected in the same way.

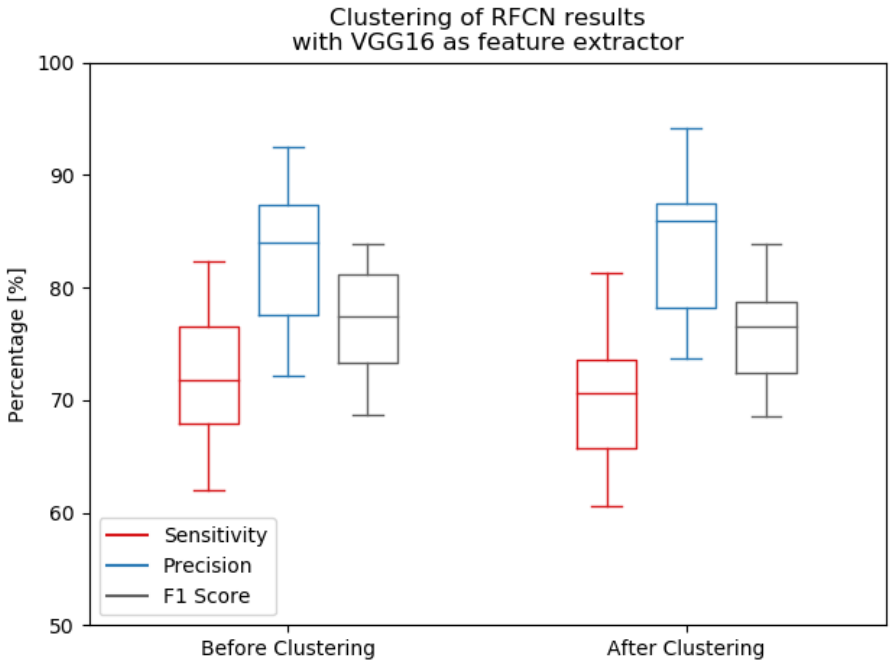


Figure 34: Sensitivity, precision and F1 score of RFCN model before and after clustering with VGG16 as feature extractor

Figure 34 shows the results of clustering of RFCN results using VGG16 as feature extractor. The outcome is fairly similar to the usage of Resnet50 as feature extractor. The precision values are higher in average and also higher extremes were achieved. Sensitivity slightly worsened as expected and F1 score seems to be a little bit more consistent but once again worse than before clustering, averaging 75,93% after clustering compared to 76.54% before.

As you can see on Figure 36, thanks to clustering using VGG16 as feature extractor, Faster RCNN Inception v2 object detection has improved its precision values, while having more consistent numbers as well. F1 score reached almost the same value as before clustering, with the difference of only 0.57%.

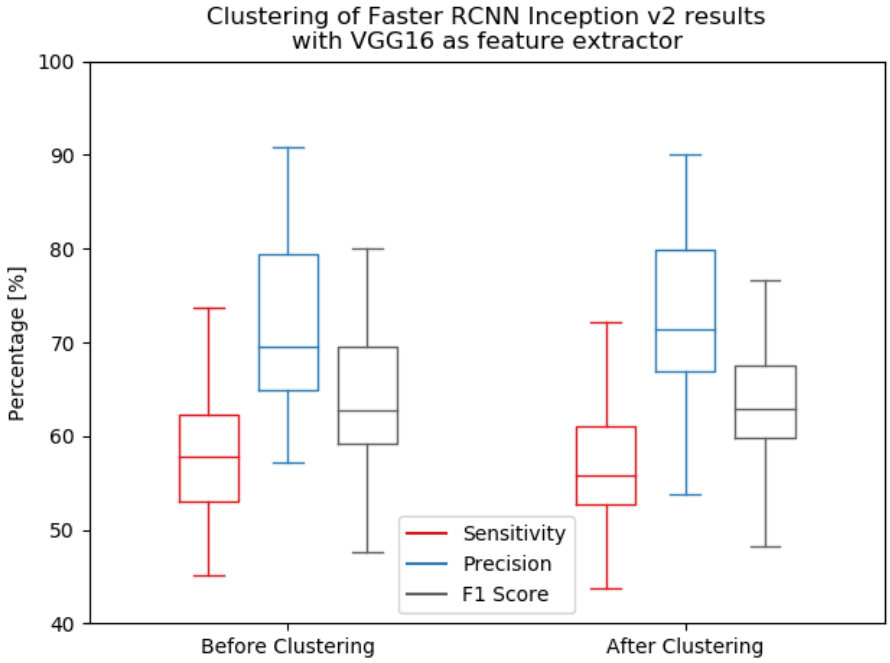


Figure 36: Sensitivity, precision and F1 score of Faster RCNN Inception v2 model before and after clustering with VGG16 as feature extractor

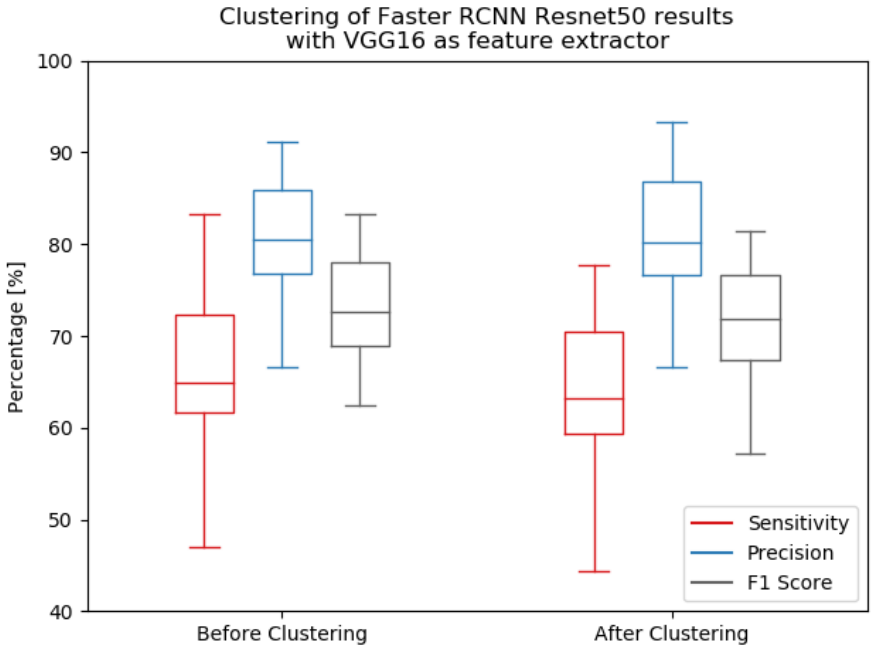


Figure 35: Sensitivity, precision and F1 score of Faster RCNN Resnet50 model before and after clustering with VGG16 as feature extractor

In case of Resnet50, clustering using VGG16 did not improve precision of the object detection model as much as in the rest of the cases. However, sensitivity decreased by almost 3%, which means that the F1 score went down quite significantly.

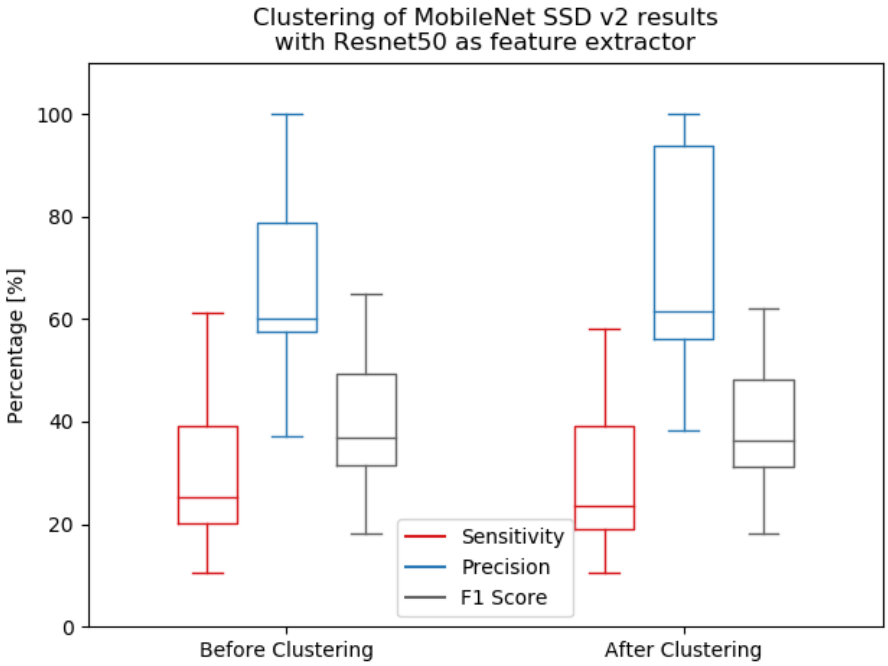


Figure 37: Sensitivity, precision and F1 score of MobileNet SSD v2 model before and after clustering with VGG16 as feature extractor

As you can see in the boxplot graph on figure 37, precision of the MobileNet SSD v2 object detection model became less consistent after clustering, having a large range of values in the main 25-75 percentile box. Unfortunately, sensitivity and F1 score decreased compared to the values before clustering, meaning that the object detection is less efficient than before.

9 EVALUATION OF THE RESULTS

In this final chapter, I will try to compare the results of the two clustering methods. Another comparison will be made, taking into an account results of object detection using other already published methods.

9.1 Comparison of the two clustering methods

Previous chapter shown graphs containing the results of object detection before and after clustering. The table below shows average values of all variations that were carried out during this thesis.

Table 6: Comparison of sensitivity, precision and F1 scores of all used models before clustering, after clustering using Resnet50 as feature extractor and after clustering using VGG16 as feature extractor.

Model	Sensitivity [%]	Precision [%]	F1 score [%]
RFCN			
Without clustering	71.79	82.33	76.54
After clustering using Resnet50	69.4	83.98	75.88
After clustering using VGG16	69.73	83.68	75.93
Faster RCNN Inception v2			
Without clustering	58.39	71.37	63.87
After clustering using Resnet50	55.37	72.16	62.3
After clustering using VGG16	56.73	72.55	63.3
Faster RCNN Resnet50			
Without clustering	65.98	80.98	72.53
After clustering using Resnet50	63.39	82.35	71.42
After clustering using VGG16	63.21	81.81	71.09
MobileNet SSD v2			
Without clustering	30.87	67.98	40.19
After clustering using Resnet50	29.58	69.88	39.32
After clustering using VGG16	29.98	69.84	39.62

It is apparent from the table 6 above that neither one of the two used clustering methods was helpful regarding the final F1 score of the object detection. The decrease in sensitivity was always way too significant for the improvement in precision to even it out. However, there are still some slight differences between the two methods that can be mentioned. For example, the clustering method using VGG16 network achieved often better F1 scores than the method using Resnet50, although the differences are rather small. This slight advantage is caused by the fact that in most cases, the sensitivity of the detection was not harmed as much as when using Resnet50. The only time, this does not

apply, is the case of object detection model Faster RCNN Resnet50, which uses the same feature extractor. The subsequent process is different obviously, but there can be some sort of bias that makes the clustering method using the same Resnet50 feature extractor more effective.

9.2 Comparison with published methods

A few already published methods are described in chapter – Existing algorithms and their comparison. The chapter also includes a table with their results. Two parameters are described in the table FPR – False positive rate and FNR – false negative rate. FPR is calculated using this formula:

$$FPR = 1 - sensitivity \qquad \text{Equation 8}$$

FNR is calculated in a similar way, instead of sensitivity, specificity is used. However, as mentioned earlier, it is not possible to calculated specificity using this data set, since the number of true negatives is unknown. And that is why, the comparison can be made using only FPR as shown in the Table 7 located below. In this case, the lower the number, the better the result

Table 7: A comparison of methods used in this thesis and previously published methods using values of FPR:

Model	False positive rate [%]
RFCN	
Without Clustering	28.21
After clustering using Resnet50	30.6
After clustering using VGG16	30.27
Faster RCNN Inception V2	
Without Clustering	41.61
After clustering using Resnet50	44.63
After clustering using VGG16	43.27
Faster RCNN Resnet50	
Without Clustering	34.02
After clustering using Resnet50	36.61
After clustering using VGG16	36.79
MobileNetSSD	
Without Clustering	69.13
After clustering using Resnet50	70.42
After clustering using VGG16	70.02
FindEM (Method 1)	10.6
FindEM (Method 2)	18.3
SLEUTH	7

It is apparent from the table 7 that the methods developed in this thesis are not as effective as methods published previously, which are working using different principles. The best method that was developed here, the fine-tuned RFCN model, reached FPR of 28.21% but the best method outside this thesis, the SLEUTH method, reached FPR of 7%, which is far better. The FindEM methods are slightly less efficient than SLEUTH but still better than methods developed during this thesis.

However, it is not easy to compare these methods, since they were not tested on the same dataset. Our dataset contained images that were heavily affected by noise and many proteins were invisible to the naked eye. It is hard to judge how much affected by noise were the images used by other authors, but it is possible that those images were different, perhaps even less noisy. Algorithms work differently on different datasets, so they often need to be adapted to the problem at hand. Results achieved by algorithm developed in this thesis could be perhaps improved if the algorithm was made more suitable for the images that were dealt with.

CONCLUSION

This thesis on the topic detection of biological structures in TEM microscope images contains a theoretical analysis of the topic in the first part of the thesis. This is followed by the practical part, where the created algorithm is described, and the results of the detection are analysed and compared.

From the results, it is obvious that the detection is not perfect at all. Some of the models such as the RFCN or Faster RCNN Resnet50 reached fairly good results but they are generally less efficient than already used methods. There are several possible reasons to cause this imperfection. For example, usage of more resize rates, different lighting and images affected by different levels of noise could be used to further improve the content of dataset and subsequently make the detection more efficient. Another possible improvement could be done by using a different model to be fine-tuned. In the last chapters of the thesis, it is mentioned that more complex and larger models achieve better results, so in order to improve the detection, it would be wise to go through the largest and most complex models.

When it comes to the clustering part, it is obvious and also explained that the sensitivity of the detection cannot get any better after clustering. In the best-case scenario, clustering excludes all the false positive occurrences, raises the precision value to 100% and does not change sensitivity. Even though clustering usually excluded several false positives, it excluded true positives as well, decreasing sensitivity to the point when it was deemed ineffective to use. This problem could be theoretically solved by using a different configuration of clustering and feature extraction. Several other configurations were attempted but neither of them showed better results than using five clusters and excluding the rarest one of them. There are also several other feature extractors that could, in theory, improve the results of cluster analysis.

Another way to improve precision but not sensitivity would be to use a higher confidence score threshold during the primary object detection. Fairly low threshold of 0.5 that increased the detection of false positive occurrences was used for two reasons. The first one is that some correctly detected objects did actually have low confidence score and it would be inefficient not to detect them. The second reason was that the false positive occurrences and more specifically their exclusion were meant to help show the function of cluster analysis.

LITERATURE

[1] KARLÍK M: Transmisní elektronová mikroskopie: pohled do nitra materiálů [online]. [cit. 2018-10-14]. Katedra materiálů, FJFI ČVUT. Available at: https://nanoed.tul.cz/pluginfile.php/603/mod_resource/content/1/TEM_05_Karlik.pdf

[2] Transmisní elektronová mikroskopie [online]. [cit. 2018-10-14]. Available at: http://atmilab.upol.cz/texty/TEM-teorie.pdf

[3] KUBÍNEK R, ŠAFÁŘOVÁ K, VŮJTEK M: Elektronová mikroskopie [online]. [cit. 2018-10-14]. Katedra experimentální fyziky a centrum výzkumu nanomateriálů, Univerzita Palackého v Olomouci. Available at: https://fyzika.upol.cz/cs/system/files/download/vujtek/granty/elmikro.pdf

[4] KARLÍK M: Úvod do transmisní elektronové mikroskopie. Praha ČVUT, 2011. [cit. 2018-10-14]. ISBN 978-80-01-04729-3.

[5] EWEN C: Cryo-EM enters a new era. eLife. 3:e03678 [online]. [cit. 2018-10-21]. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4131193/

[6] BROADWITH P: Explainer: What is cryo-electron microscopy [online]. [cit. 2018-10-21]. Available at: https://www.chemistryworld.com/news/explainer-what-is-cryo-electron-microscopy/3008091.article

[7] KREČMEROVÁ P: Segmentace obrazů z optické mikroskopie pro podporu laboratorní diagnostiky vybraných chorob krve [online]. Brno, 2011 [cit. 2018-10-21]. Centrum pro výzkum toxických látek v prostředí, Masarykova univerzita. Available at: https://is.muni.cz/th/qgsfn/dp_211618.pdf

[8] ROSENTHAL PB: Testing the Validity of Single-Particle Maps at Low and High Resolution. Methods in Enzymology, Volume 579. [cit. 2018-11-06]. doi: 10.1016/bs.mie.2016.06.004

[9] Preparing samples for the electron microscope [online]. [cit. 2018-11-06]. Available at: https://www.sciencelearn.org.nz/resources/500-preparing-samples-for-the-electron-microscope

[10] JAN J: Digital Signal Filtering, Analysis and Restoration. volume 44. volume 44. London: The Institution of Electrical Engineers, 2000. 407 s. [cit. 2018-11-06] ISBN: 0-85296-760- 8.

[11] LUCIC V, RIGORT A, BAUMEISTER W: Cryo-electron tomography: the challenge of doing structural biology in situ. The Journal of Cell Biology, 202 (3): 407 - 419. [cit. 2018-11-24]. doi: 10.1083/jcb.201304193.

[12] Cryo-Tomography [online]. [cit. 2018-11-24]. Thermo Fischer Scientific. Available at: https://www.fei.com/life-sciences/cryo-tomography/

[13] Preparing samples for the electron microscope [online]. [cit. 2018-11-24]. Science Learning Hub. Available at: https://www.sciencelearn.org.nz/resources/500-preparing-samples-for-the-electron-microscope

[14] EGMONT-PETERSEN M, DE RIDDER D, HANDLES H: Image processing with neural network – a review. [cit. 2018-11-24]. Pattern Recognition 35/10. doi:10.1016/S0031-3203(01)00178-9

[15] Convolution neural networks [online]. [cit. 2018-11-24]. CS231n Convolutional Neural Networks for Visual Recognition. Available at: http://cs231n.github.io/convolutional-networks/

[16] GHUMAN SS: Clustering Techniques – A Review [online]. [cit. 2018-11-24]. International Journal of Computer Science and Mobile Computing, Vol. 5, Issue

5, May 2016, 524 – 530. Available at: https://www.ijcsmc.com/docs/papers/May2016/V5I5201699a5.pdf
[17] KUČERA J: Shluková analýza [online]. [cit. 2018-11-24]. MUNI. Available at: https://is.muni.cz/th/w8lgz/5739129/web/web/main.html
[18] DHANACHANDRA N, MANGLEM K, CHANU YJ: Image Segmentation using K-means Clustering Algorithm and Subtractive Clustering Algorithm. [cit. 2018-11-24]. Procedia Computer Science 54 (2015), 764 – 771. doi: 10.1016/j.procs.2015.06.090.
[19] SCHERES SHW: Semi-automated selection of cryo-EM particles in RELION-1.3. [cit. 2018-12-25]. Journal of Structural Biology 189 (2015) 114-122. doi: 10.1016/j.jsb.2014.11.010.
[20] SHORT MJ: SLEUTH – a fast computer program for automatically detecting particles in electron microscope images. [cit. 2018-12-25]. MRC Laboratory of Molecular Biology. PMID: 15065678.
[21] TANG G, PENG L, BALDWIN PR, MANN DS, JIANG W, REES I, LUDTKE SJ: EMAN2: An extensible image processing suite for electron microscopy. [cit. 2018-12-25]. Journal of Structural Biology 157 (2007) 38-46. doi: 10.1016/j.jsb.2006.05.009.
[22] ROSEMAN AM: FindEM – a fast, efficient program for automatic selection of particles from electron micrographs [online]. [cit. 2018-12-25]. Journal of Structural Biology 145 (2004) 91-99. PMID: 15065677.
[23] Transmission electron microscopy clipart collection [online]. [cit. 2018-12-26]. Available at: http://diysolarpanelsv.com/transmission-electron-microscope.html
[24] Problems with lenses of TEMs [online]. [cit. 2018-12-26]. Available at: https://myscope.training/legacy/tem/background/concepts/problems/
[25] RAMIRÉZ-CORTÉS J, GÓMEZ-GIL P: Shape-based hand recognition approach using the morphological pattern spectrum. [cit. 2018-12-26]. doi: 10.1117/1.3099712
[26] Edge Detection [online]. [cit. 2018-12-26]. Available at: https://www.mathworks.com/discovery/edge-detection.html
[27] Watershed (Region segmentation) [online]. [cit. 2018-12-26]. Available at: https://www.visco-tech.com/english/technology/gauging/
[28] WHITE D: Inception Network Overview [online]. [cit. 2018-12-26]. Available at: https://www.cs.colostate.edu/~dwhite54/InceptionNetworkOverview.pdf
[29] SIYUAN J, QI Q: Regression, classification and clustering: The advantages and disadvantages of machine learning algorithms in three major directions [online]. [cit. 2018-12-26]. Available at: https://weiwenku.net/d/100416195
[30] TONIONI A: TensorFlow – 101 [online]. [cit. 2019-03-31]. Available at: https://www.slideshare.net/alessiotonioni/tensorflow-intro-2017
[31] What is TensorFlow? [online]. [cit. 2019-04-01]. Available at: https://www.guru99.com/what-is-tensorflow.html
[32] WILLEMS K: TensorFlow Tutorial for Beginners [online]. [cit. 2019-04-01]. Available at: https://www.datacamp.com/community/tutorials/tensorflow-tutorial
[33] KURT: Object Detection Tutorial in TensorFlow: Real-Time Object Detection [online]. [cit. 2019-04-07]. Available at: https://www.edureka.co/blog/tensorflow-object-detection-tutorial/

[34] LYSUKHIN D: TensorFlow Object Detection API: Basics of Detection [online]. [cit. 2019-04-07]. Available at: https://becominghuman.ai/tensorflow-object-detection-api-basics-of-detection-7b134d689c75
[35] CHOW D: Training an object detector using Cloud Machine Learning Engine [online]. [cit. 2019-04-07]. Available at: https://cloud.google.com/blog/products/gcp/training-an-object-detector-using-cloud-machine-learning-engine
[36] GOSWAMI S: A deeper look at how Faster-RCNN works [online]. [cit. 2019-04-07]. Available at: https://medium.com/@whatdhack/a-deeper-look-at-how-faster-rcnn-works-84081284e1cd
[37] GIRSHICK R: Fast R-CNN. [cit. 2019-04-07]. arXiv:1504.08083v2
[38] DAS S: CNN Architectures [online]. [cit. 2019-05-02]. Available at: https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5
[39] HE K, XIANGYU Z, SHAOQING R, JIAN S: Deep Residual Learning for Image Recognition. [cit. 2019-05-02]. arXiv:1512.03385
[40] DAI J, LI Y, HE K, SUN J: R-FCN: Object Detection via Region-based Fully Convolutional Networks. [cit. 2019-05-09]. arXiv:1605.06409
[41] HOLLEMANS M: MobileNet Version 2. [online]. [cit. 2019-05-13]. Available at: https://machinethink.net/blog/mobilenet-v2/
[42] SHUKLA A: Simulation on programmable graphics hardware (GPUs). [online]. [cit. 2019-05-13]. Available at: http://run.usc.edu/cs599-s10/scribeNotes/Akshay_Shukla_scribe_notes.pdf
[43] cuDNN Developer Guide. [online]. [cit. 2019-05-13]. Available at: https://docs.nvidia.com/deeplearning/sdk/cudnn-developer-guide/index.html
[44] GRANGER E: Faster Regional-CNN (R-CNN) architecture. [online]. [cit. 2019-05-13]. Available at: https://www.researchgate.net/figure/Faster-Regional-CNN-R-CNN-architecture_fig2_320719820
[45] ARLEN TC: Understanding the mAP Evaluation Metric for Object Detection. [online]. [cit. 2019-05-14]. Available at: https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3
[46] TensorFlow detection model zoo. [online]. [cit. 2019-05-14]. Available at: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

LIST OF ATTACHEMENTS

1. Electronic version of this thesis
2. Scripts used for image preparation, object detection and cluster analysis
3. Original images used to for generation of training and testing sets of images
4. Test images used for evaluation of the algorithm
5. An inference graph of the Faster RCNN Resnet50 fine-tuned model and the associated label map
6. Manual with instructions regarding the use of the algorithm