**Kaunas University of Technology**

Faculty of Mathematics and Natural Sciences

# Application of Deep Learning Methods for Analysis of Titanium Dioxide Nanoparticles

Master‘s Final Degree Project

**Dominykas Babkauskas**

Project author

**Assoc. Prof. Paulius Palevičius**

Supervisor

**Kaunas, 2023**

**Kaunas University of Technology**

Faculty of Mathematics and Natural Sciences

# Application of Deep Learning Methods for Analysis of Titanium Dioxide Nanoparticles

Master's Final Degree Project

Applied Mathematics (6211AX006)

**Dominykas Babkauskas**

Project author

**Assoc. Prof. Paulius Palevičius**

Supervisor

**Assoc. Prof. Tomas Iešmantas**

Reviewer

**Kaunas, 2023**

**Kaunas University of Technology**

Faculty of Mathematics and Natural Sciences

Dominykas Babkauskas

# Application of Deep Learning Methods for Analysis of Titanium Dioxide Nanoparticles

Declaration of Academic Integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (hereinafter – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University;

2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project are plagiarised from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references;

3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law;

4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

Dominykas Babkauskas

*Confirmed electronically*

Babkauskas, Dominykas. Application of deep learning methods for analysis of titanium dioxide nanoparticles. Master's Final Degree Project / supervisor Assoc. prof, Paulius Palevičius; Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.

Study field and area (study field group): Mathematics, Applied mathematics.

Keywords: deep learning, segmentation, titanium dioxide, nanoparticles, statistical analysis, U-Net, Mask R-CNN
Kaunas, 2022. 57 p.

## Summary

In this project, two different deep learning segmentation networks are analysed and evaluated by their capability to segment titanium dioxide particles in scanning electron microscopy images. In total, eight different models are set up, where each segmentation network is set up in four different ways. The models selected are U-Net, a semantic segmentation network, and Mask R-CNN, an instance segmentation network. Different network setups for U-Net consist of two different encoders, either ResNet-18 or ResNet-34, and these encoders are either pre-trained on ImageNet or not. Primary changes in the setup of the Mask R-CNN model consist of changing the limit of how many objects can be detected in a single image. A 396 image dataset is generated from 22 256x256 images by using data augmentation. Data augmentation primarily consists of spatial transformations such as rotations and flips. This generated dataset is split into 380 images for training and 16 images for validating the models. Additionally, 6 images that are not present in either the training or validation sets, are used for testing the model for the final evaluation. The best performing Mask R-CNN model was able to achieve on the validation set a mean intersection over union score of 0.5705, and a test mean intersection over union score of 0.6249. In contrast, the best-performing U-Net model was able to achieve a validation mean intersection over union score of 0.8586 and a test mean intersection over union score of 0.8766, outperforming the Mask R-CNN model. The best-performing model overall, U-Net with a pre-trained ResNet-34 backbone, was used to segment 3 different scanning electron microscopy images of titanium dioxide particles. With these segmented images, statistical analysis was performed. These three different images exhibited vastly different particle counts. It was determined that the particle area values are mainly distributed between 0 and 25 pixel area sizes, with the most spread-out area size distribution being on the image with the least amount of particles present. Further analysis showed that the Feret angles of these particles exhibit a bimodal distribution, which suggested diverse particle orientations. There was a strong correlation found among the variables that describe the compactness, roundness, circularity, area, and solidity of the particles. Welch’s T-test confirmed that the observed particles in each image show a significant difference from random distribution, hinting that the particle patterns are not the result of coincidence. All three images were categorised as having self-avoiding particles. The development of such a model can greatly help experts in the field by automating the segmentation process of the particles and allow them to focus on more prominent tasks

Babkauskas, Dominykas. Giliojo mokymosi metodų taikymas titano dioksido nanodalelių SEM vaizdų analizei. Magistro baigiamasis projektas / vadovas doc. dr. Paulius Palevičius; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Matematikos mokslai, taikomoji matematika.

Reikšminiai žodžiai: gilusis mokymasis, segmentavimas, titano dioksidas, nanodalelės, statistinė analizė, U-Net, Mask R-CNN.

Kaunas, 2022. 57 p.

## Santrauka

Šiame baigiamajame magistro darbe yra tiriami ir vertinami du skirtingi giliojo mokymosi segmentavimo tinklai. Vertinimas priklauso nuo jų galimybių segmentuoti titano dioksido nano daleles, išgautas iš rastrinio elektroninio mikroskopo. Iš viso yra vertinami aštuoni skirtingi modeliai ir kiekvienam tinklui sukuriamos keturios skirtingos pradinės konfigūracijos. Pasirinkti modeliai buvo U-Net, semantinio segmentavimo tinklas, Mask R-CNN, instancijų segmentavimo tinklas. Skirtingos konfigūracijos U-Net tinklui susideda iš dviejų skirtingų užkoduotojų dalių: ResNet - 18 arba ResNet – 34. Šie pateikiami užkoduotojai yra iš anksto apmokyti su *ImageNet* duomenų imtimi arba be jos. Mask R-CNN tinklo pagrindinis pakeitimas konfigūracijai buvo riba kiek objektų gali aptikti per vieną vaizdą. Buvo sukurtas duomenų rinkinys, susidedantis iš 22 vaizdų su rezoliucija 256x256. Iš šių vaizdų buvo sugeneruoti 396 vaizdai pasitelkiant duomenų augmentacija, kuri pagrinde susidėjo iš erdvinių transformacijų, tokių kaip pasukimų ir apvertimų. Iš sugeneruotų 396 vaizdų, 380 vaizdų buvo parinkti mokymosi imčiai, o 16 vaizdų buvo parinkti validacijos imčiai. Taip pat, buvo parinkti 6 skirtingi vaizdai, kurių nėra nei mokymosi, nei validacijos imtyje. Šie 6 vaizdai buvo naudojami testavimo imčiai pagal kurią yra pateikiamas galutinis modelių vertinimas. Geriausias Mask R-CNN modelis, remiantis validacijos imtimi, pasiekė vidutinį susikirtimo santykį 0,5705, o testavimo imčiai pasiektas susikirtimo santykis 0,6249. Tačiau, geriausiai pasirodęs U-Net tinklas su ResNet – 34 užkoduotoju ir iš anksto apmokytais svoriais validacijos imčiai pasiekė vidutinį susikirtimo santykį 0,8586, o testavimo imčiai santykis buvo lygus 0,8766. Lyginant pagal pasiektus vidutinius susikirtimo santykius buvo nuspręsta, kad U-Net tinklas veikia žymiai geriau. Šis U-Net modelis tada buvo naudojamas trijų skirtingų rastrinio elektroninio mikroskopo vaizdams segmentuoti. Su šiais vaizdais atlikta statistinė analizė parodė, kad šie trys vaizdai turi itin skirtingą dalelių skaičių. Taip pat, buvo nustatyta, kad dalelių plotų reikšmės pagrinde yra pasiskirsčiusios tarp 0 ir 25 kvadratinių pikselių ir buvo pastebėta, kad mažiausiai dalelių turintis vaizdas turi labiausiai pasiskirsčiusius plotų dydžius. Atliekant tolimesnę analizę buvo nustatyta, kad dalelių Fereto kampai turi bimodalinį pasiskirstymą, o tai rodo, kad dalėlės turi įvarius orientacijos kampus. Taipogi, buvo nustatyta stipri koreliacija tarp dalelių kompaktiškumą, apvalumą, apskritumą, plotą ir vientisumą aprašančių kintamųjų. Velčo T-testas patvirtino, kad stebimos dalelės kiekviename vaizde reikšmingai skiriasi nuo atsitiktinio pasiskirstymo. Visi trys vaizdai buvo klasifikuoti turintys save vengiančias daleles. Tokio modelio sukūrimas gali padėti srities ekspertams automatiškai segmentuojant vaizdus, taip leidžiant jiems susitelkti ties svarbesnėmis problemomis.

## Table of contents

## List of figures

## List of tables

## List of abbreviations and terms

**Abbreviations:**

Assoc. prof. – associate professor;

SEM – Scanning Electron Microscopy;

TiO2 – titanium dioxide;

RoI – Region of Interest;

GPU – Graphics Processing Unit;

CPU – Central Processing Unit;

KNN – K- Nearest Neighbour;

PSD – Particle Size Distribution;

FPN – Feature Pyramid Network;

RPN – Region Proposal Network;

mAP – mean Average Precision;

IoU – Intersection over Union;

ML – Machine Learning;

CNN – Convolutional Neural Network;

ReLU – Rectified Linear Unit;

NMS – Non-max Suppresion;

NND – Nearest Neighbour Distance;

UEP – Ultimate Eroded Point;

ResNet – Residual Network;

SGD – Stochastic Gradient Descent;

Adam – Adaptive Moment Estimation.

**Terms:**

**Upsampling** – the process of increasing the resolution or size of an image.

**Downsampling** – the process of reducing the resolution or size of an image.

**Machine Learning** – is the field of study that enables computers to learn from data and make predictions about it without explicitly being programmed to do so.

**Feature Map** – a spatial arrangement of learned features extracted from input data in computer vision tasks.

**Data augmentation** – technique of artificially expanding a dataset with the application of various transformations to the original data.

**Artificial Neural Network** – computational model inspired by the structure neurons in the human brain.

**Feed-forward neural network** – a type of artificial neural network where the data flows only in one direction, from the input layer to the output layer.

**Object Detection** – a computer vision technique that's used to localise objects in an image and classify them, marking them with a bounding box.

**Classification** – a machine learning task that predicts a predefined label on the input data.

# Introduction

In 1791, a German chemist named Martin Heinrich Klaproth managed to separate titanium dioxide (TiO2) from the mineral rutile. What he got was an odourless, white, solid powder. Since then, the use of titanium dioxide has become widespread worldwide. According to the European Chemicals Agency, the European Union imports over 1000000 tonnes annually [1]. To put that into perspective, the United States in 2020 alone produced the same amount of TiO2 as the annual EU import and consumed 90% of that [2]. To name a few use cases, TiO2 can be found in coating products, inks, plasters, paints, adhesives, biocides, and food colouring. The compound has recently been scrutinised by the EU as it was suspected to be carcinogenic when ingested, and its use commercially as a food additive was prohibited for a brief period between 2019-2022. Currently, the ban has been lifted, but the Center for Science in the Public Interest is still advising people to avoid products that contain titanium dioxide.

The nature of the compound‘s applications requires analysis of the substance at the microscopic level for it to be used optimally. For example, to make a long-lasting coat of paint, one needs to be sure that when applied, it distributes evenly without any thinning spots or bulging. TiO2‘s use as a disinfectant also requires the same amount of scrutiny to provide an even coating on a given surface. Data collection for this analysis requires a group of researchers to first examine samples using Scanning Electron Microscopes (SEMs) and other types of microscopes. For the data to be processed by particle analysis software, it must first be segmented. In other words, the particles must be separated from their background so that the software knows what particles to account for in statistical measurement. In its most straightforward form, segmentation is usually done by setting a colour threshold value for separating these particles from the background. There are quite a few different algorithms that are available to calculate this threshold value without introducing user bias, but these methods are not always robust enough to fit every particular set of image data.

The aim of this project is to apply deep learning methods for the statistical analysis of TiO2 nanoparticles in SEM images. Deep learning methods could drastically decrease the time experts need to segment microscopic image data and let them focus more on the analysis. To achieve this goal, these are the tasks that must be accomplished:

- Perform literature analysis on segmentation methods for nanoparticle analysis;
- Create a labelled dataset for segmentation of nanoparticles;
- Construct deep convolutional neural networks for titanium dioxide nanoparticle segmentation;
- Evaluate and tune the parameters of convolutional neural networks and choose the best-performing model;
- Perform statistical analysis of identified nanoparticles in the segmented images.

# 1. Literature review

## 1.1. Overview of conventional computer vision methods for Segmentation

Computer vision is an essential field of study for the modern world that focuses on allowing computers to interpret and understand digital images and videos. Although human sight has a head start, computer vision operates similarly to human vision. Humans have a lifetime of context to train themselves to distinguish between objects, estimate distances, observe motion, and identify problems in an image. Image segmentation is a process where pixels are assigned to a certain region or segment of an image. Segmentation can be split into two categories: semantic and instance. In semantic segmentation, every pixel that's not the background is labelled as an object mask. In instance segmentation, every instance of an object is masked separately and treated as a separate object.

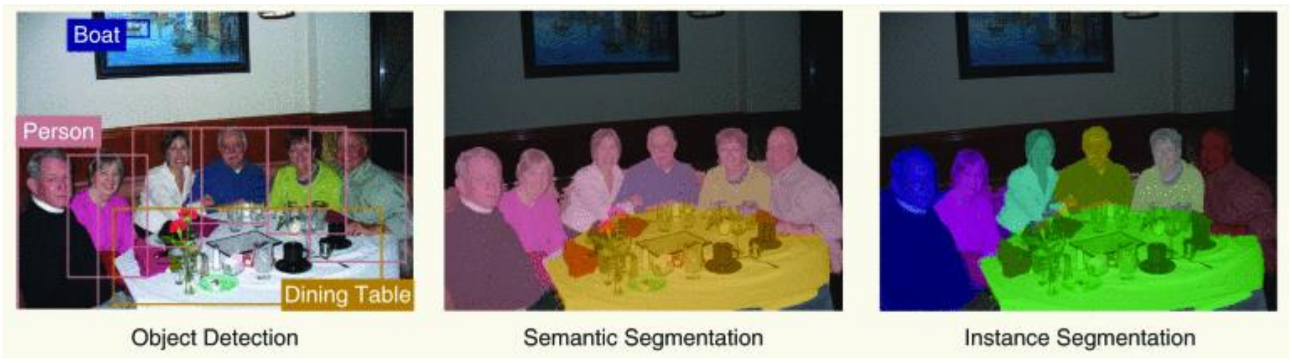

**Fig. 1**. The difference between object detection, semantic segmentation and instance segmentation [3 p. 38].

In segmentation, the pixels are usually grouped by their properties, such as colour, texture, intensity, shape, and edges. One of the most widely used methods is thresholding to obtain a segmented image. The simplest example for this could be a grayscale image where the image gets segmented by pixel intensity. This would be done by selecting some arbitrary value between 0 and 1 that would represent the threshold, and all pixel values that are less than the set threshold get converted to 0 (black) to symbolise the background. All the pixel values above the threshold get converted to 1 (white) representative of the object mask [4]. Albeit, this method might be able to give satisfactory results if the images are similar to each other and have a clear distinction from the background, for example, images of pennies on a black background. This, however, is most often not the case, and most of the time, it is required for many image processing tasks to have additional algorithms that automatically determine the threshold value. A basic approach to this automation would be to fit a polynomial function that represents the image histogram as closely as possible and then identify the threshold value at the minimum turning point of the curve. This approach might not always work, as images can have multiple peaks in their histogram, and as the complexity of a given image increases, it would get harder and harder to fit an appropriate polynomial to the histogram. Another similar approach would be Otsu's method, which sets the threshold by computing the between-class variance for each possible threshold value; the two classes, in this case, would be background and foreground. Even though this method is simplistic and speedy computation-wise, it works best with bimodal image histograms and usually falls short when the foreground and background regions don't have equal variances. Besides this, it is also susceptible to noise, which it doesn't account for and may introduce inaccurate thresholding results [5]. Most of the global thresholding techniques can only be used in simple cases where the foreground and background are easily discernible. A more robust approach would be to use region-oriented techniques, such as the region growing method [6]. In this technique, a set of seed pixels are selected over the given image, and regions are then grown from these seeds by appending neighbouring pixels that meet a similarity criterion. The criterion can be either the intensity level, texture, or colour. This method iterates over all pixel values in the image and creates a segmentation mask. In 2012, a group of researchers from India used a hybrid of this method along with edge-based region growing to segment MRI brain images with decent results. The results showcase that the method is able to discern abstractly shaped edges.

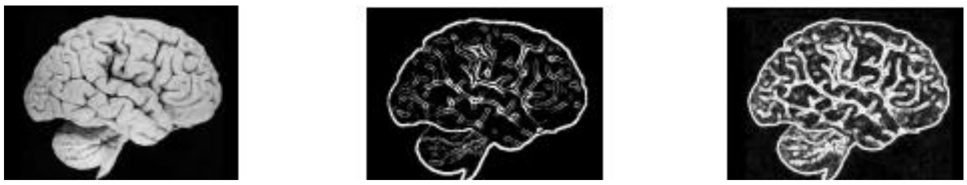

**Fig. 2**. On the left is the original image, middle is the segmented image with a threshold, right is the segmented image with edge-based region growing [7 p. 68].

Even though this method works well, it is not without its disadvantages. It is prone to over segmentation if the input image is noisy or the pixel values in the given image have varying intensities. To add to that, it's also quite time and power-consuming because of the computational demand to iterate over every pixel and perform similarity checks. In particle analysis, another popular approach is to use the watershed algorithm [8]. This method is non-parametric and was originally developed in 1979 for contour extractions in grayscale images. The main advantages of this algorithm stem from the fact that it does not require a predefined threshold value and that, at the time of this method's development, alternative methods required smoothing of histograms, but these methods usually became inoperative when dealing with a greater number of phases. The watershed method works by treating the image as a topographic surface, where the brightness values of each pixel correspond to elevations on said surface. It then identifies the local minima points in the image. From a topographic point of view, these local minima can be interpreted as depressions in the surface.

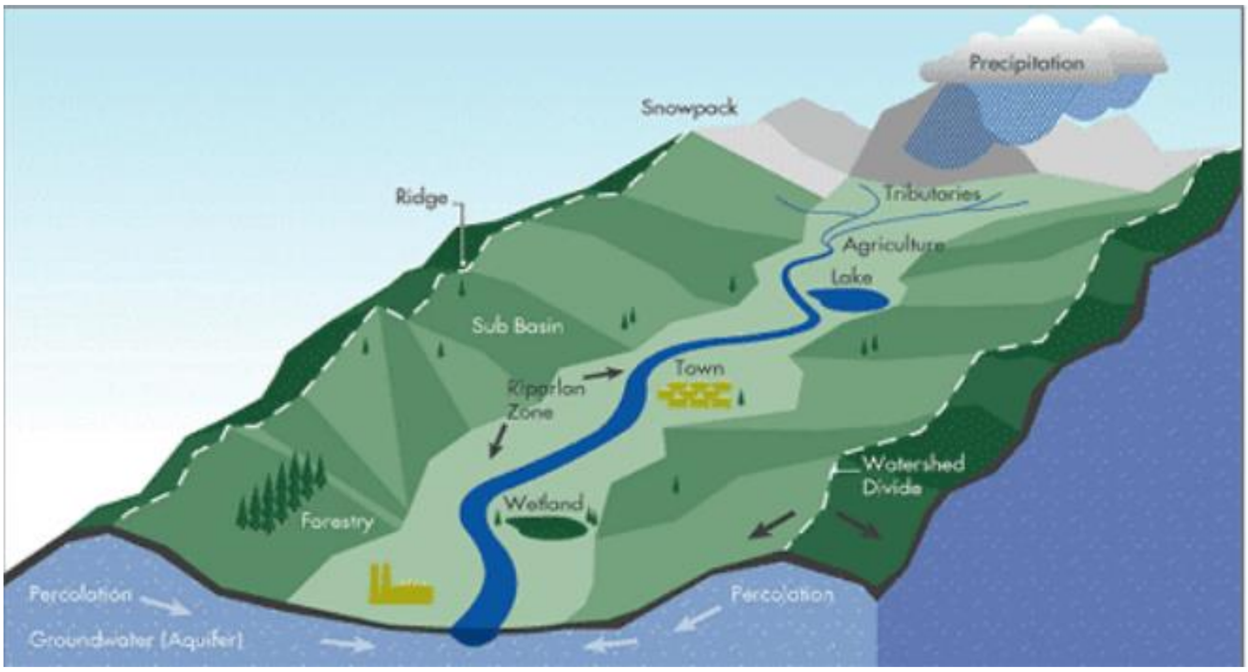

Fig. 3. Watershed example [9].

To better understand the watershed algorithm, one has to know what it is in practical terms: it is based on geological landform areas that channel rainfall and snowmelt to creeks, streams, or rivers that eventually outflow to larger bodies of water such as reservoirs, bays, or the ocean [10]. Everyone lives in a watershed; for example, in Lithuania, the main watershed catchment area is the Nemunas river, and one watershed covers all of Lithuania and part of Belarus too.

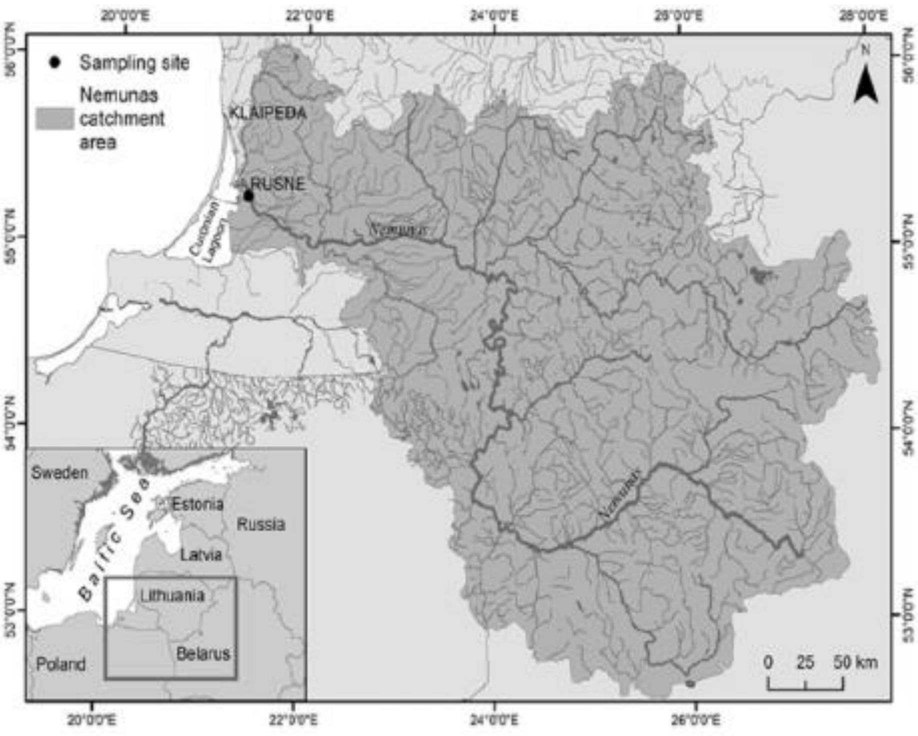

**Fig. 4**. Nemunas river watershed coverage [11 p. 3].

For this algorithm to segment images, it does so by “filling up” each basin with water until it reaches a “dam” or some sort of ridge that separates it from adjacent basins. The whole process creates a watershed that separates each object in the image from its neighbours based on the differences in brightness values. A watershed with a gradient approach was utilised to segment coal particles for size distribution analysis in 2021 by a group of researchers. The main reason for not using conventional methods, such as simple thresholding, stemmed from the fact that images of coal particles have a narrow colour range and uneven grey levels due to their reflective characteristics. Edge detection is also a problem because of the specular reflection that coal particles exhibit; these reflections can cause false detection of edges. This led the researchers to conclude that a single image segmentation method cannot achieve good results for this particular type of image. This was shown by another similar work in 2013, where the researchers utilised a combination of the Hessian matrix edge intensity along with a watershed segmentation method [12] to achieve good results, although the combination struggled to correctly split the gaps between the particles, creating masks that are larger than the actual particles. The 2021 research introduced another combination for image segmentation using the aforementioned watershed method with gradient along with a k-nearest neighbour (KNN) region merging algorithm. All image data was pre-processed using a median filter. After this, all images had a gradient transformation applied to them; in particular, the Sobel operator was chosen due to the fact that it includes a distance weight factor, which helps in detecting edges.

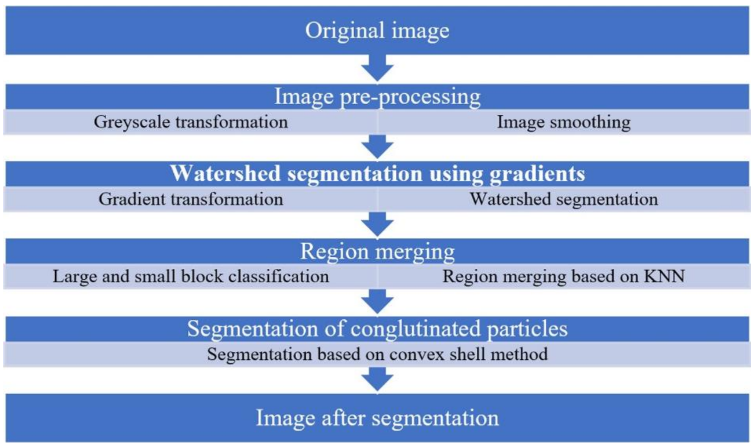
Original image
Image pre-processing
Greyscale transformation
Image smoothing
Watershed segmentation using gradients
Gradient transformation
Watershed segmentation
Region merging
Large and small block classification
Region merging based on KNN
Segmentation of conglutinated particles
Segmentation based on convex shell method
Image after segmentation

**Fig. 5**. The image segmentation steps [13 p. 164].

After the pre-processing phase, the further steps can be seen in Fig. 5. KNN was used here to help segment the smaller particles around the larger particles highlighted by watershed segmentation. Finally, segmentation is done by applying the convex shell method, which further helps separate particles by their edges.

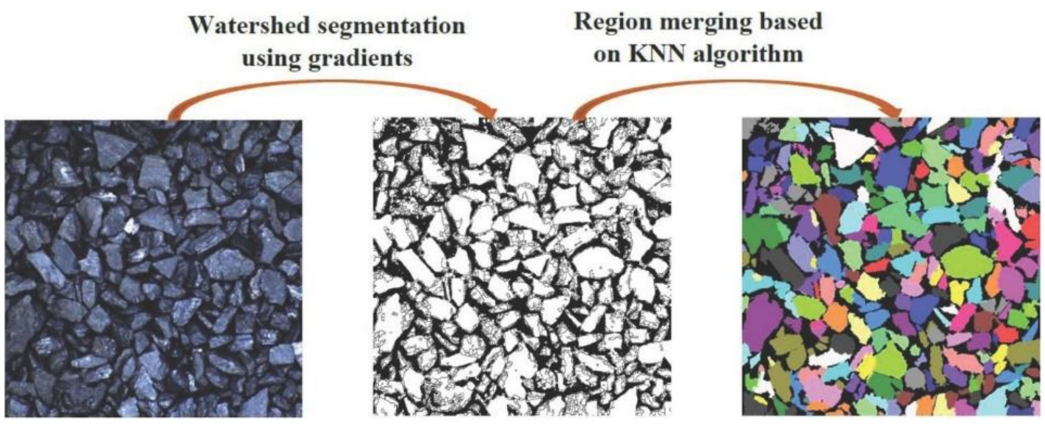
Watershed segmentation using gradients
Region merging based on KNN algorithm

**Fig. 6**. The segmentation process shown in three stages [13].

The method was tested on 4 samples. Although automatic segmentation was mostly outmatched by manual segmentation when the particle sizes were smaller than 1.25-1.50 mm, it performed reasonably well. Especially if we take into account that automatic segmentation takes less than 1 minute in this case. Out of the 4 samples tested, all showed standard deviations less than 3%, which shows that this type of method combination is capable of reaching good segmentation performances [13].

## 1.2. Overview of deep learning based methods for particle analysis

Deep learning is a subfield of machine learning that involves training artificial neural networks with multiple layers. It is defined as deep in regards to the depth of a neural network, meaning the number of layers it has. In recent years, following the trend of better hardware, specifically graphics processing units (GPUs), it has become increasingly popular. GPUs were primarily developed for rendering complex three dimensional worlds as seen in animated movies or video games, but *Nvidia* released a framework called *CUDA* in 2007 allowing users to utilise most *Nvidia* GPUs for complex matrix calculations. Before *CUDA*, there was no way to utilise these GPUs for deep learning tasks, and using a central processing unit (CPU) for these calculations was not feasible due to the amount of time that it takes to calculate them. The main uses of deep learning revolve around computer vision, speech recognition and natural language processing. Because of the large number of layers, deep learning methods have the capability of recognising complex patterns. Basically, by using deep learning, we are capable of teaching a computer to do what comes naturally to humans.

Deep learning-based computer vision trains machines to accomplish the tasks of image classification, segmentation and object detection, but with digital images, data, and algorithms instead of retinas, optic nerves, and visual cortexes. Even though humans have all these advantages over computers in regards to, extracting contextual clues from images, a machine learning (ML) image classification algorithm is able to surpass humans in accuracy and time. In fact, a 2020 study that evaluated machine learning versus human performance based on the ability to classify 247 different abstracts, where each abstract is one of 19 classes, showed results that the machine surpassed human performance. The test subjects were 63 undergraduates and 26 Ph.D. students from Nanyang Technologic University [14].

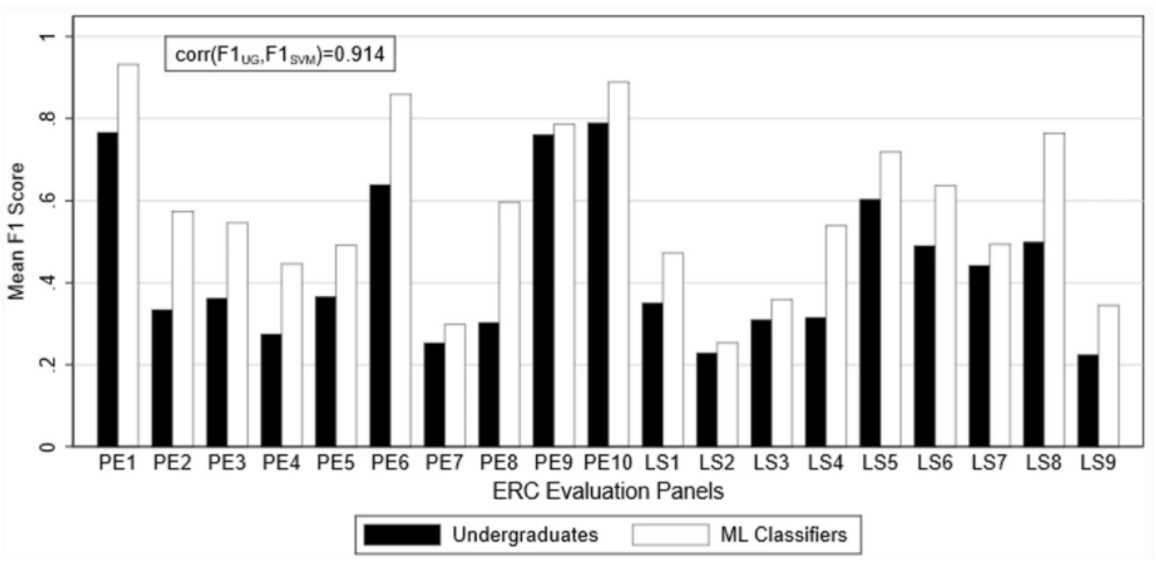

**Fig. 7**. Comparison of undergraduate results versus ML classification performances [14 p. 11].

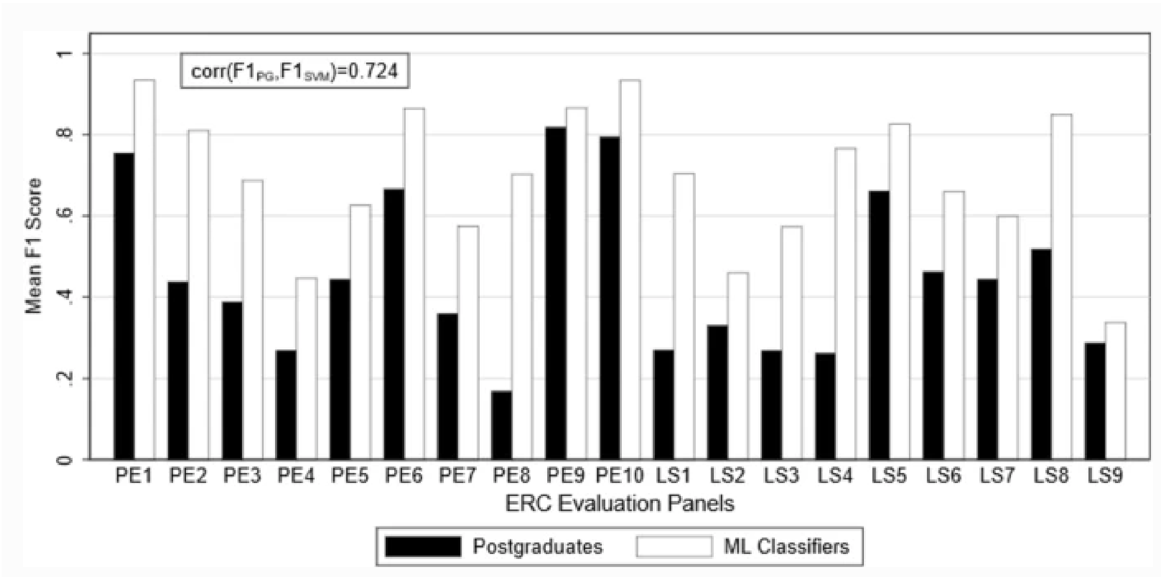

**Fig. 8**. Comparison of Postgraduate results versus ML classification performances [14 p. 11].

Not only did the classification algorithm surpass the test subjects, it was able to classify 247 abstracts in a mere 5 seconds, whereas the quickest human classifiers took over 2 hours to do the same task and also required morning training.

In most cases, deep learning-based segmentation outperforms traditional machine learning methods due to the fact that it is able to learn from raw data without the need for much pre-processing. A 2022 study done by a group of researchers from China highlights the capabilities of a deep learning based instance segmentation network called the path aggregation network (PANet). In this particular study, the researchers were interested in fly ash, a glass-like waste that is recovered from coal-fired gas streams. The use of fly ash in cement binders helps the concrete become more durable while also being a cheaper alternative to industry standard binder additives. Particle size distribution (PSD) is of utmost importance here due to the fact that the finer the size distribution, the fewer voids are present in the concrete, resulting in a more durable structure. Conventional methods to analyse PSD of such particles require heavy and high-cost SEM devices and experts to operate them. The application of deep learning in this study allows onsite inspection of fly ash particle features using a simple optical microscope, greatly cutting down on the time and resources required otherwise. Images used for the training dataset contain microspheres of different sizes and also contain various debris that is of no concern to the subject of study. To overcome the difficulty of a network recognising different-sized fly ash microspheres, the researchers introduced a feature pyramid network (FPN) to the backbone. Backbones in neural networks such as these are used as feature extractors, meaning they extract meaningful feature maps that are derived from the original input image. FPNs are capable of capturing multi-scale information from an input image by constructing a pyramid of multiple levels. Essentially, as an input image is processed through the backbone, features at different depths of the network are extracted using a convolution operation. These features are then enhanced with higher-level, upsampled feature maps. It has been shown that the use of a FPN as a feature extractor in region proposal and object detection tasks improves the results significantly [15]. In PANet, these extracted multi-scale features are fed into adaptive feature pooling, which, in short, pools different scaled features and passes them along to the masking head or the bounding box prediction head. The model was trained on 1800 256x256 images captured by a polarised light microscope. During training, the model iterated 10000 times over the data, and the optimal model was selected at 5000 iterations.

**Table 1**. Fly ash microsphere detection performance of the PANet model [16].

| **Dataset** | **AP** | **AP 0.5 IoU** | **AP 0.75 IoU** | **AP small** | **AP medium** | **AP large** |
|---|---|---|---|---|---|---|
| Training | 0.523 | 0.713 | 0.612 | 0.488 | 0.694 | - |
| Validation | 0.495 | 0.705 | 0.621 | 0.459 | 0.654 | - |

The researchers managed to reach an AP of 0.705 at IoU 0.5, and an AP of 0.621 at IoU 0.75 for the validation set. Furthermore, the masking performance was tested on 4 different samples, and all of the IoU scores were above 0.8, showing that the model is able to produce acceptable prediction results [16].

Indeed, FPNs proved to be of paramount importance in developing state-of-the art instance segmentation models [17, 18]. The ability to obtain multi-scale features allows these models to be as versatile as they are. Models that were developed and tested with great results on general imagery, such as the Microsoft Common Objects in Context (MS COCO) dataset, where the images are of people, vehicles, pets, and others common objects, are versatile enough to be used in particle analysis.

To use the FPN optimally, a different operation for extracting small feature maps from each region of interest (ROI) was needed. A standard operation for this used to be RoIPool, an operation that divides a ROI into a grid of bins and then applies max pooling to each bin. The main problem with this method was that the coordinates of these bins are rounded, which in turn introduces discrepancies between the RoIs and the features. These discrepancies are of little consequence when it comes to classification, as this method was developed for an object detection model, Fast R-CNN [19], which only had to classify an object in a given box. The novel proposed operation is called RoIAlign, which circumvents the coarse quantization problem by not rounding the proposed coordinates and additionally applies bilinear interpolation, allowing to compute exact values of the input features from 4 sample points in a bin. RoIAlign shows better results and outperforms the RoIPool operation in instance segmentation tasks [17].

**Table 2**. Mask and bounding box average precision comparison of RoIPool and RoIAlign [17 p. 6].

| | $AP$ | $AP_{50}$ | $AP_{75}$ | $AP^{bb}$ | $AP^{bb}_{50}$ | $AP^{bb}_{75}$ |
|---|---|---|---|---|---|---|
| *RoIPool* | 23.6 | 46.5 | 21.6 | 28.2 | 52.7 | 26.9 |
| *RoIAlign* | **30.9** | **51.8** | **32.1** | **34.0** | **55.3** | **36.4** |
| | +7.3 | +5.3 | +10.5 | +5.8 | +2.6 | +9.5 |

For deep learning approaches to semantic segmentation, one of the more popular neural network architectures to use is U-Net. U-Net is a well-established segmentation network that was originally developed in 2015 for medical image segmentation, and its main advantage over other architectures at the time was that it was able to achieve good segmentation results with less training data. This architecture consists of an encoder and decoder part, and the main thing that helps achieve such good results is the utilisation of skip connections between adjacent neural network stages between these two parts. Skip connections allow the network to combine low-level features from the encoder part with high-level features in the decoder part. Even though this network was developed for medical image segmentation, it can be utilised with great results in various different segmentation tasks [20]. In 2022, a study was carried out by Japanese researchers on U-Net based semantic segmentation of microscopic images of colourants. The training data set was comprised of 2592 images, which were generated from 60 images using data augmentation. For this study, the researchers modified the conventional U-Net architecture into two different variations. U-Net #1 retained the same architecture, but the number of channels was drastically reduced. These changes were made because, according to their findings, by analysing the feature maps of the deep layers, they noticed that the number of channels was excessive for their target images. U-Net #2 was similar to U-Net #1, but the two bottom skip connections that connect two of the deepest layers were removed. By removing these skip connections, by their assumptions, allowed the model to more accurately segment particles that have dark interiors and less visible contours. During testing, the results showed that U-Net #1 was better at segmenting images with a large amount of particles present and with a small amount of particles present. Reaching an AUC score of 0.9066 and 0.9187, respectively [21]. Another study highlighting the capabilities of the U-Net architecture was done in 2021 by a group of researchers from Kaunas University of Technology. The main focus of the researchers was to find out how various technological parameters and properties of gold and TiO2 thin films influence the formation of gold nanostructures. To perform particle analysis, the researchers utilised the U-Net architecture to localise and automatically segment the experimental images. The training images were captured using SEM, and more training data was generated by employing data augmentation. Data augmentation consisted of shear deformations, rotations, zooms, and width and height shifts. A small tweak was done to the U-Net architecture: padded convolutions were used instead of the default unpadded convolutions; this ensures that the output images are not smaller than the input. To measure the accuracy of the model, Intersection over Union (IoU) was used. The values of this metric range from 0 to 1, where 1 signifies perfect segmentation accuracy. After training, the model was able to achieve a 0.9497 IoU score on the training set and a 0.9601 IoU score on the validation set [22].

## 1.3. Related work with Mask R-CNN implementation

Mask R-CNN is an instance segmentation network that is heavily reliant on initial parameters when adapting it to segment particles. This is due to the fact that it was initially created to only segment up to 100 objects in a given image, and the types of these objects typically cover a large area. The below-analysed studies provide valuable insight on how a Mask R-CNN model should be set up and trained, for instance segmentation of small particle-like objects.

In 2021, a group of researchers from Belgium released an article about their application of a deep learning method called Mask R-CNN for counting bacteria colonies in Petri dishes. They explain that counting and differentiating bacterial colonies is a non-avoidable step in vaccine development. This is an error-prone and time-consuming task, which prevents biologists from focusing on more meaningful work. There are numerous different computer vision algorithms that attempt to threshold these Petri dish images into binary masks of the background and the colony-forming units (CFUs). These methods, however, suffer from limitations. It’s been observed that these approaches exhibit poor performance when CFUs overlap, which leads to miscounting of the agglomerated colonies. In addition, the methods require careful calibration and near perfect exposure conditions to properly exclude the background. The researchers suggest that a deep learning approach to this problem might be a more effective solution.

For this task, the researchers had 101 laboratory images of Petri dishes that contained two different types of bacterial colonies. They split this dataset into training, validation, and test sets. The Mask R-CNN algorithm combines the Faster R-CNN object detection algorithm with some extra convolution layers to classify the pixels inside the bounding boxes predicted by the Faster R-CNN model. This pixel classification part is called a mask head. To adapt the Mask R-CNN algorithm to their type of data, they changed parameters in the Anchor Generator part of the RPN. The RPN receives different-sized squares, called anchors, and then chooses the best-fitting anchor for the object. The researchers changed the anchor sizes according to the size of objects in their image; this is a crucial step for the model to be able to effectively segment CFUs.

The training process was done iteratively. First, they used a ResNet-50 backbone for the Mask R-CNN model, trained this model for 399 epochs on COCO pre-trained images, and followed this up with another 50 epochs with their original images as input, passing 64 images per epoch. After this, they trained the same model for 50 epochs using augmented images, this time passing 500 images per epoch. Finally, they changed the backbone to ResNet-101 and trained all layers with 500 images per epoch for 46 epochs. The augmentations include stochastic rotation of images between -180 and 180 degrees, scalings, translations, and additive and multiplicative noise. By using these augmentations, they managed to generate 500 images out of the original 101. The use of a pretrained ResNet-50 backbone in the first iterations of training resulted in decent performance on the training data, but the performance lacked on the validation set. The final steps with aggressive augmentation helped the model generalise the data better and led to an increase in performance on the validation set.

At the end of the project, they managed to reach a 97.1 mean average precision (mAP) at the intersection over union (IoU) threshold of 0.5 on the training data, 97.6 mAP on the validation set, and a 94.1 mAP on the test set [23].

Another study was carried out in 2023 by a three-person group of researchers about using Mask R CNN to segment mine dump particles from images. During mining operations, a huge amount of debris piles up, accumulating into millions of cubic metres over the course of a year. For the mine to function properly, it is crucial to maintain and monitor the stability of these dumps. Usually, the particle size distribution is determined by using sieve analysis, but recent research is more focused on estimating particle size distributions from dump images. However, a new challenge had arisen, as identifying particles in the images is a difficult task due to various natural conditions that affect the image. Most prominently, it is affected by daylight intensity, moisture content, and occlusion. Previous attempts to utilise computer vision for these calculations fell short, as most of them are based on colour contrast and are not effective when the given particles are multi-scale and multi-shape. For these reasons, the authors decided to use the Mask R-CNN model.

The authors gathered image data from an overburden dump at the JSPL iron ore mine in India. In total, they captured 500 different images, from which 31505 particles were extracted. The image data was annotated using the VGG Image Annotator, a popular platform to prepare data for instance segmentation that allows the user to export the annotations in COCO format, a widely used format type for instance segmentation.

The model‘s parameters were tweaked so that a maximum of 2000 proposed bounding boxes were selected. The IoU threshold for non-max suppression (NMS) was set to 0.5. This was done in order to suppress RPN proposals that overlap heavily. The anchor generator was tweaked to have 6 anchors of different sizes: 16, 32, 64, 128, 256, and 512, along with aspect ratios of 0.5, 1, and 2. The model had a ResNet50 backbone with a FPN.

The researchers split the gathered data into two different datasets; one dataset had 16448 total particles across all images, and the other had 31505 particles. At the end of it, they managed to reach 94.3% accuracy on the 16448 particle dataset after 7000 iteration steps over the data. The formula used to calculate accuracy was:

$$z_i = \begin{cases} 1, & if\ (\widehat{y_i} - y_i) = 0 \\ 0, & if\ (\widehat{y_i} - y_i) \neq 0 \end{cases} \tag{1}$$

$$accuracy = \frac{1}{n_{particles}} \sum_{i=0}^{n_{particles}-1} (z_i) \tag{2}$$

Here $\widehat{y_i}$ is the predicted value of $i^{th}$ sample and $y_i$ is the ground truth value, $n_{particles}$ is the number of total particles in the dataset. The predicted value and the ground truth value can only be either 0 or 1. On the dataset containing 31505 particles they managed to achieve 97.2% training accuracy. The model was evaluated on 10 unseen images, it was able to identify 73.58% particles on average in a given image [24].

In 2021, a group of researchers from the National Laboratory of Metrology and Testing in France released an article about applying deep learning-based instance segmentation to TiO2 particles. The main problem outlined is the complexity of the dimensional properties of the TiO2 particles due to their non-spherical shape and proneness to agglomeration. These attributes tend to impede the effectiveness of conventional methods that use various transformations and post-processing techniques to characterise this type of content. In other words, the conventional approaches lack robustness. The problem can be easily circumvented by having nanometrology experts characterise the particles, but this process is extremely time-consuming. Hence, the researchers opted to try using the deep learning instance segmentation method Mask R-CNN.

For this analysis, they composed 77 images that were manually segmented by experts. The image dimensions were 2048x1536x1, where 2048 is the width, 1536 is the length, and 1 is the number of channels. In this case, it is 1 because SEM measurements return grayscale images. The original images were augmented, and new data was generated from them. The data augmentation consists of these transformations:

- Random flipping;
- Random rotations;
- Randomly positioning agglomerate clusters on different empty SEM backgrounds;
- Random gaussian blur;
- Contrast normalization;
- Additive Gaussian noise;
- Pixel value multiplication.

Furthermore, the model and its architecture required changes for it to be able to work on the given task. The most important and primary change to the algorithm was made to the RPN. The sizes of the anchors were modified to reflect the sizes of the particles in an image from the minimum to the maximum size as follows: 8, 16, 32, 64, and 96. These sizes are in pixels. Also, the number of trained anchors was modified to be 1024, the maximum number of particles in an image. Then the researchers modified the mask head of the model by adding an extra transposed convolutional layer; this change allowed the model to slightly improve its segmentation performance.

Another crucial part of training a model for such a task is utilising transfer learning. This helped the researchers avoid training the model from scratch and utilise already pre-trained weights. The pre-trained weights for this model were trained on the MS COCO database [25]. The researchers first trained the model’s heads only for 38 epochs using a learning rate of 0.001, then another 4 epochs with a learning rate of 0.0001. After these epochs, they trained the whole network for another 28 epochs with a learning rate of 0.001, and the final 7 epochs after this were trained with a learning rate of 0.0001. A stochastic gradient descent optimizer was used with a momentum parameter of 0.9 and a gradient norm clipping value of 5.0; along with this, L2 regularisation was also utilised with a weight decay of 0.0001.

Finally, the model was evaluated on a test set consisting of 19 images. The whole segmentation task was performed in 110 seconds. For comparison, manual segmentation takes about 15 to 30 seconds for one particle. With this in mind, if a human expert were to try and segment the entire test set by hand, it would take about 15,6 hours. With this model, they achieved a mAP score of 60.6 and a mean dice coefficient score of 0.936 [26].

## 1.4. Research topic and task relevance

Albeit conventional methods for image segmentation are effective for simple tasks, they soon start lacking as more complex data is introduced, such as microscopy images. Their susceptibility to noise and dependence on variance of the image makes them lack robustness. By analysing the literature provided on deep learning applications for particle analysis, one can expect that the versatility of these methods can help overcome these hurdles. The application of deep learning for such tasks can help reduce the time needed to perform particle analysis and let the experts focus on more prominent tasks.

TiO2 is a widely used substance in everyday consumer products, the production of solar cells, and as a photocatalyst in the treatment of wastewater. Recent discoveries about this substance have shown that it may pose a potential health and environmental risk, particularly in its nanoparticle form, due to the small size and its ability to penetrate biological membranes. Therefore, the understanding and analysis of these particles is crucial in ensuring that it could be used safely by minimizing the impact on human health and the environment while also helping to create more durable products.

## 2. Data and methodology

### 2.1. Data

Data was gathered by depositing TiO2 thin films onto room-temperature silicon dioxide and fused quartz substrates using reactive magnetron sputtering. The sputtering method involves using ions to dislodge atoms from the targets and deposit them onto the substrate. Argon and oxygen gases were used to create a high-vacuum environment to ensure a clean deposition surface. The sputtering power and gas flow rates were optimised to achieve a growth rate of 0.027 nm/s and a uniform coating.

The deposited thin films had an amorphous structure, which was less desirable for some applications than a crystalline structure. Therefore, the films were annealed at high temperatures and allowed to cool slowly to promote the formation of a crystalline structure. Annealing is a process that involves heating, holding at a temperature, and cooling at controlled rates. After this, the resulting thin films had a crystalline structure [22].

The images were then captured at different annealing stages using SEM at a resolution of 1280x960x1. Here, 1 signifies that the image is grayscale. Mainly the images that were captured at the annealing stage at 500°C with a 5x zoom level and a 10 nm resolution scale were used.

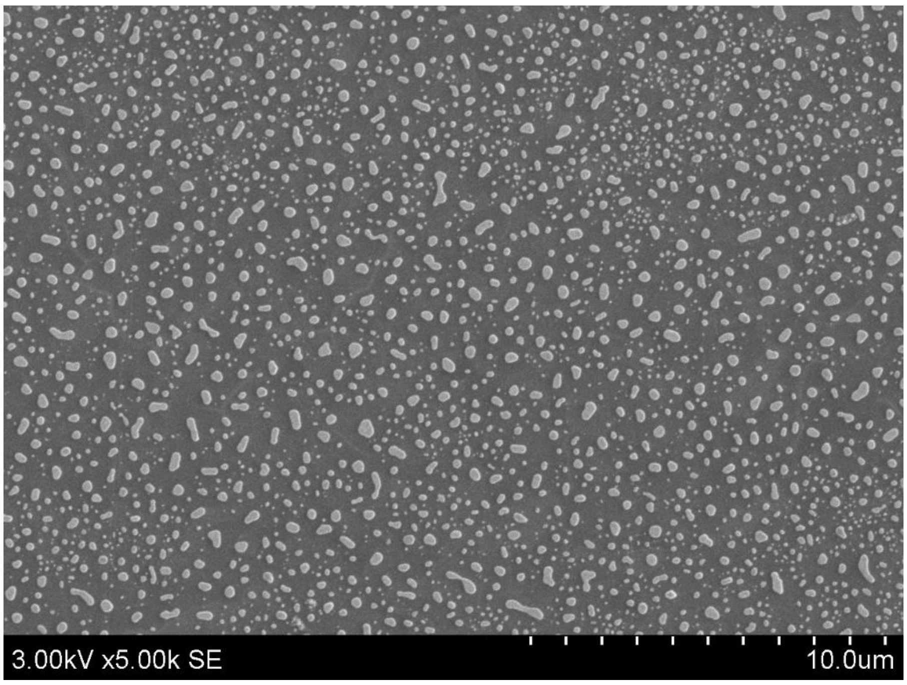

**Fig. 9**. Example of TiO2 nanoparticles captured at 5x zoom level and at 10nm resolution scale.

### 2.2. Methodology

#### 2.2.1. Convolutional Neural Network

A convolutional neural network (CNN) is a type of artificial neural network that behaves like a feed-forward network. CNNs are classified as deep learning methods due to the high number of layers that they have. Instead of single neurons, a CNN uses filters and pooling layers to efficiently process and analyse data that has a grid-like structure, such as image data. The filters, also known as kernels, are small matrices that are usually 3x3 in size. These kernels slide over the input data and perform element-wise multiplication and addition to produce feature maps. The weights in the kernel matrix are trainable, meaning that as soon as the kernel passes through the whole batch of images its weights are updated using backpropagation. The pooling layers down-sample the feature maps by taking the maximum, average, or other function of a small rectangular neighbourhood of values. This reduces the spatial dimensionality of the data while preserving the important features. The output of the last convolutional and pooling layers is flattened and fed into a fully connected layer or layers, which perform classification or regression tasks [27].

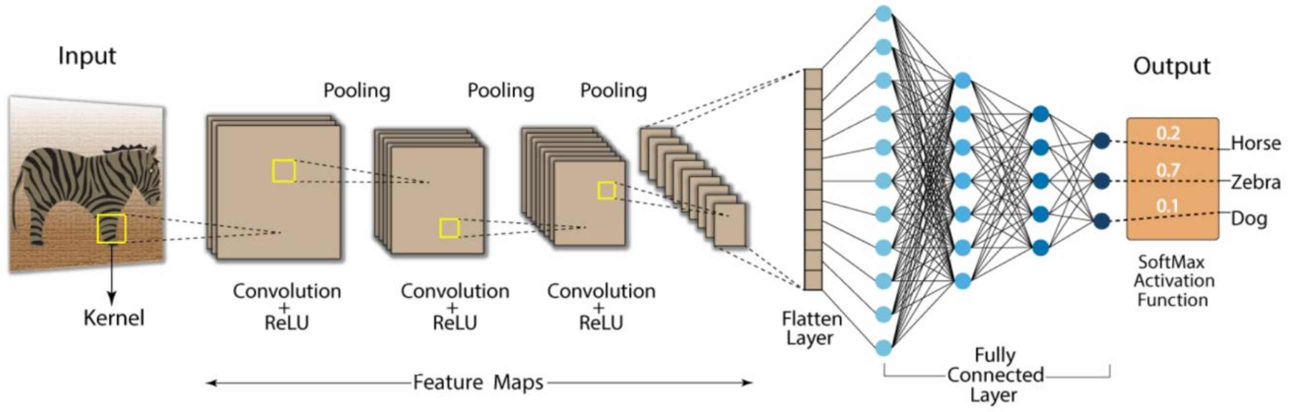

**Fig. 10**. Example of a convolutional neural network architecture [28].

#### 2.2.1.1. Convolution

Kernels in CNNs are used to perform the convolution operation. Where the input is a single-channel image, a kernel can be described as:

$$O(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i+k-1, j+l-1) K(k,l) \tag{3}$$

O(i,j) is the value of a convolution operation at the given coordinates i and j. Here m and n represent the kernel’s row and column counts respectively; the I operator is the input image; and K is the kernel. Kernels pass over the given input image when calculating the convolution operation,;the amount of pixels that a kernel moves to the side is defined by the stride parameter. So, for example, if we have a 3x3 kernel with a stride parameter of 1, it would calculate the convolution value at the starting point of the image, and then after this, it would move one pixel over to the right and do the same again. This process is repeated until the kernel passes through the whole image.

Another important parameter in kernels is padding. Padding allows kernels to calculate values at the edges of an image. Where a kernel, or part of it, would be out of bounds, padding pads the image with some value, usually 0, and allows the calculations to proceed.

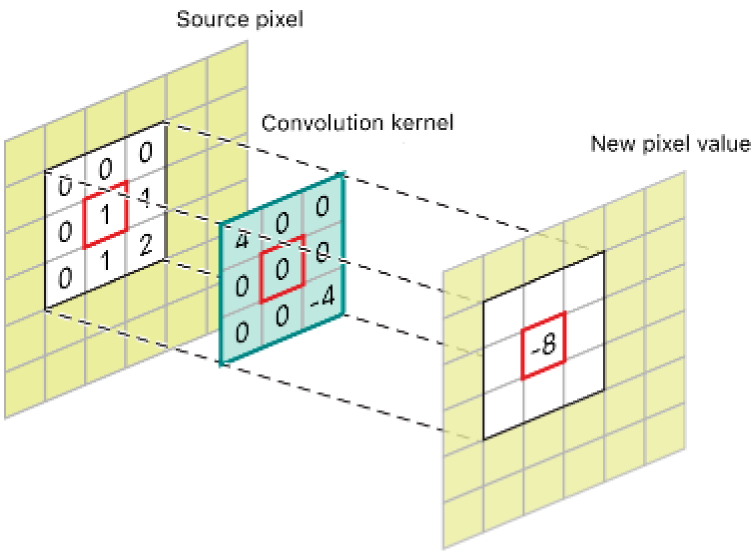

**Fig. 11**. Visualization of a kernel calculation.

#### 2.2.1.2. Pooling layers

Pooling layers in neural networks helps reduce the spatial dimensions of the input image while also retaining important information. Pooling is accomplished by summarising groups of adjacent values into a single representative value. This can help the CNN to be more invariant to small translations or rotations of the input image [48].

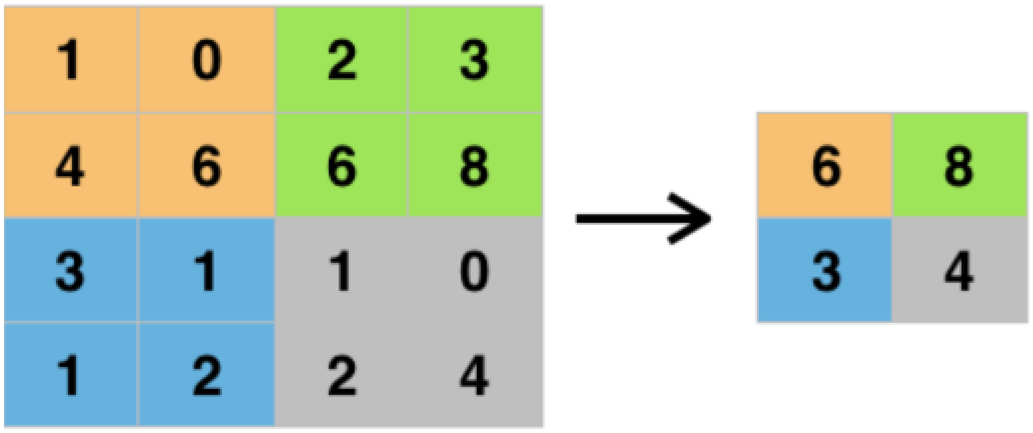

**Fig. 12**. Example of a max pooling operation [29].

Fig. 12 shows how a max pooling operation works on input values that are shown on the left of the image. In this example, the max pooling operation has a kernel size of 2x2 and a stride of 2; this means that the kernel takes the maximum value from a 2x2 grid and then moves further along the input to the right by 2 pixels and does the same again. Most pooling operations work identically in regards to traversing the input; only the operations may differ. There are different pooling operations, such as average pooling, where the averages of 2x2 grids are taken instead of maximum values.

#### 2.2.1.3. Loss functions

A loss function, also known as a cost function, is typically defined as a function of the predicted outputs and the ground truth values. During training, a CNN receives an input image and makes a prediction on it. This prediction is then compared to the ground truth value by utilising a loss function. A commonly used loss function in binary image classification is the binary cross-entropy loss:

$$L(y_i, \hat{y}_i) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i) \tag{4}$$

Here N is the total number of observations, $y_i$ is the ground truth value, either 1 or 0, $\hat{y}_i$ is the predicted probability that the observation belongs to class 1, $1 - \hat{y}_i$ is the predicted probability that the observation belongs to class 0. Loss functions can be used as a way to evaluate a model's performance, low loss means that a model is performing well on the given data. But most importantly, they are used in backpropagation, it serves as a cost value that the network has to minimise.

#### 2.2.1.4. Backpropagation

Backpropagation is an algorithm used for training artificial neural networks. It is an iterative process that computes the gradients of the model's parameters, i.e., the weights, with respect to the loss function. The gradients are then used to update the parameters of the network by using gradient descent, typically by utilising an optimisation algorithm. Simple example of a weight update:

$$W_{new} = W_{old} - \alpha\frac{dL}{dW} \tag{5}$$

Here $W_{new}$ is the new weight value, $W_{old}$ is the old weight value, $\alpha$ is the set learning rate, $\frac{dL}{dW}$ is the gradient computed for this particular weight with respect to the loss function. Gradients mathematically point to the local maximum of a function, by making this value negative it intuitively points to the local minimum. In optimization algorithms the learning rate is a hyperparameter of some small value generally ranging from $10^{-2}$ to $10^{-6}$. It dictates how much a model's weights are updated. Choosing an appropriate learning rate is crucial to balance the trade-off between convergence speed and accuracy. Setting it too low might risk a slow convergence speed; setting it too high risks an inability to converge [47].

#### 2.2.1.5. Activation function

Some activation functions are applied to the outputs of the hidden layers to introduce non-linearity and are capable of creating complex relationships between the input and the output of the layer. Most commonly, a Rectified Linear Unit (ReLU) activation function is used in between hidden layers. ReLU is computationally efficient to evaluate and also helps to avoid the vanishing gradient problem, that occurs in other activation functions, such as sigmoid or tanh, which tend to converge to small values when the inputs are large. The ReLU function is defined as follows:

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{6}$$

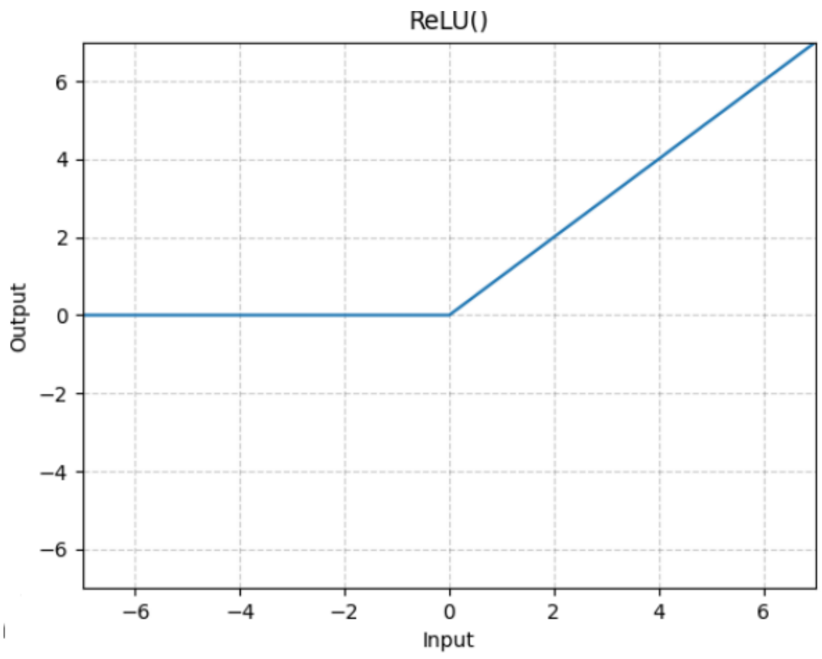

Fig. 13. ReLU function [30].

The way ReLU introduces non-linearity is that it essentially blocks a signal from a given neuron if the value is less than zero.

Other activation functions are usually used in the output layer of a network. For binary image classification, a popular function is sigmoid:

$$S(x) = \frac{e^x}{e^x + 1} \tag{7}$$

All values that pass through a sigmoid activation function get mapped between 0 and 1, allowing to interpret this value as the probability of a prediction.

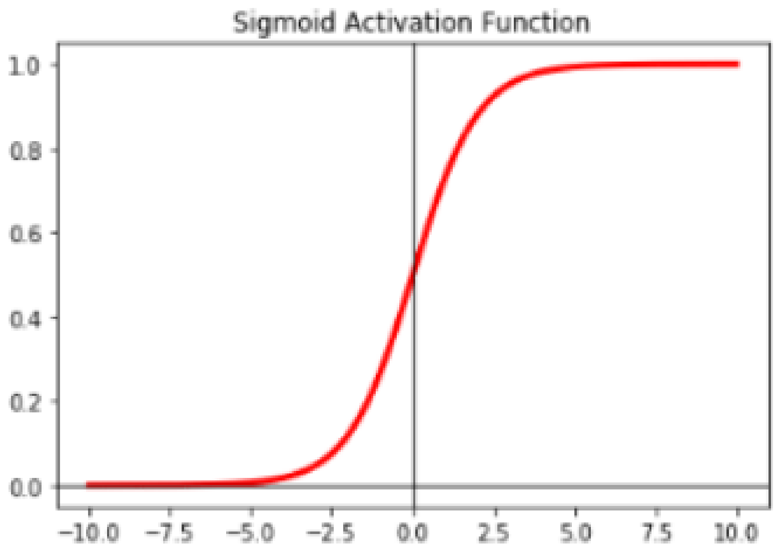

Fig. 14. Sigmoid function [31].

### 2.2.2. Models

#### 2.2.2.1. U-Net

U-Net is a semantic segmentation CNN that was first proposed in 2015 by O. Ronneberger, P. Fischer, and T. Brox. Since then, this network has become a popular choice for numerous applications in image segmentation tasks.

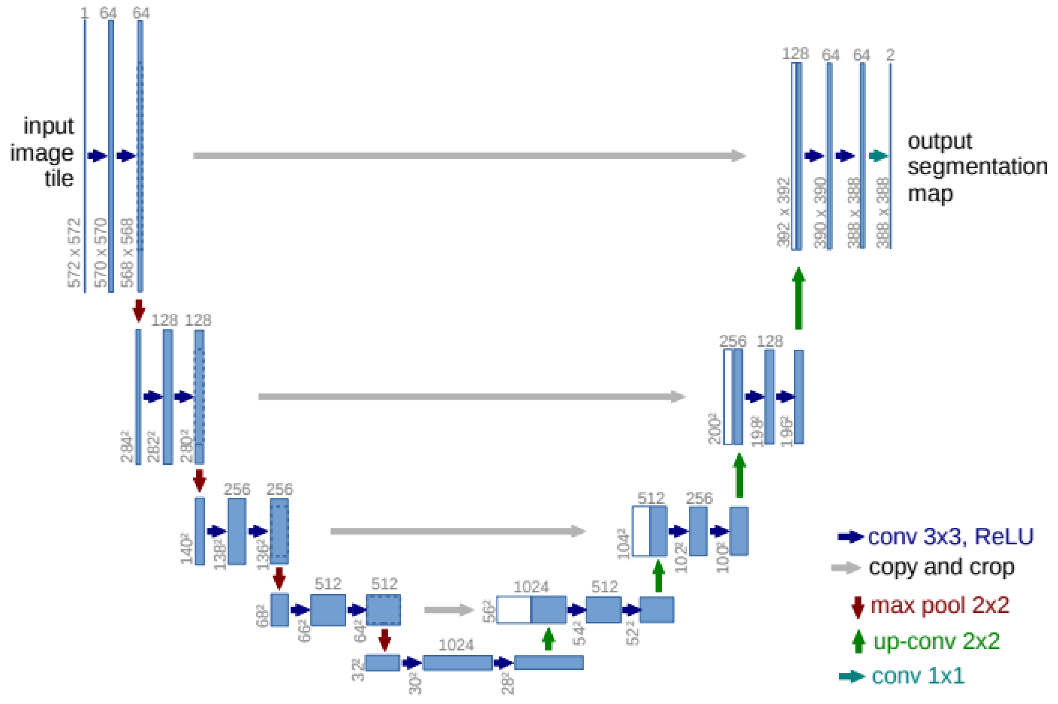

Fig. 15. U-Net architecture [20].

As seen in Fig. 15, the architecture consists of an encoder and decoder part with skip connections in between these parts. The encoder repeatedly applies two 3x3 unpadded convolutions to the input, followed by an activation using ReLU and a 2x2 max pooling operation with a stride of 2 for further downsampling. Every time the downsampling operation is done, the number of channels is doubled. The decoder part at each stage is concatenated with an appropriately cropped feature map that is adjacent to the encoder part of the network. Feature maps then go through two 3x3 convolution operations and also have a ReLU activation function applied to them. The feature maps are then upsampled, or deconvoluted, and the number of feature channels is halved [20]. In this case, deconvolution can be seen as the inverse operation of convolution; instead of convolving a group of features into one feature map, it upsamples the feature map to a higher resolution. In this architecture, the deconvolution layer is a learnable filter. Deconvolution can be defined as follows:

$$Y_{i,j,k'} = \sum_{u=0}^{kh-1} \sum_{v=0}^{kw-1} \sum_{c=0}^{C-1} X'_{i+su,j+sv,c} \times K'_{u,v,k',c} \tag{8}$$

Let‘s say that we have an input feature map X of size H x W x C, where H is the height, W is the width, and C is the number of channels. We also denote a learnable kernel K with size kh x kw x C x C`, where kh and kw are the height and width of the kernel, C is the number of channels in the feature map, and C` is the number of output channels. For deconvolution to happen, the input feature map has to be padded to increase its spatial resolution. If we denote p as pixels on each side, the resulting padded feature map X` has a size of (H+2p) x (W+2p) x C. The padded feature map is then convolved with the transposed filter K, that we can denote as K`; its new dimensions are kh x kw x C` x C. Stride is denoted by the letter s. The output $Y_{i,j,k'}$ is a 3D matrix with a size of H x W x C`, where H and W are the height and width of the output feature map, respectively, and C` is the number of output channels. Each element $Y_{i,j,k'}$ represents the activation of the k`-th output channel at position (i,j) of the output feature map.

Modifications were made to the network; residual networks, also known as ResNets [32], were used as encoders instead of the one that the authors described in their paper. Specifically, the ResNet-18 and ResNet-34 networks were chosen. These networks are pretrained on *ImageNet*, a large-scale dataset consisting of around 1400000 labelled images from 1000 object categories. *ImageNet* has become a staple contributor to many of the main advances in deep learning based - computer vision tasks. The creators of this dataset host an annual competition called the ImageNet Large Scale Visual Recognition Challenge; which brings in researchers from all around the world to compete in the development of the best-performing computer vision models [33]. The use of a pre-trained network as an encoder is called transfer learning. It's a technique in which a model trained on one task is adapted to a different but related task. In computer vision, this is especially useful if one has limited labelled data, as the features learned from the previous task can be used to identify features on the current task.

ResNet was introduced in 2015 and has since become an immensely popular architecture for deep learning and computer vision tasks. This network addresses the vanishing gradient problem that mainly arises in deep networks. As gradients are propagated through many layers of the network, they become increasingly smaller, making it difficult for the network to learn any features. This problem is overcome by utilising residual connections that allow the gradients to flow directly from one layer to another by bypassing the intermediate layers.

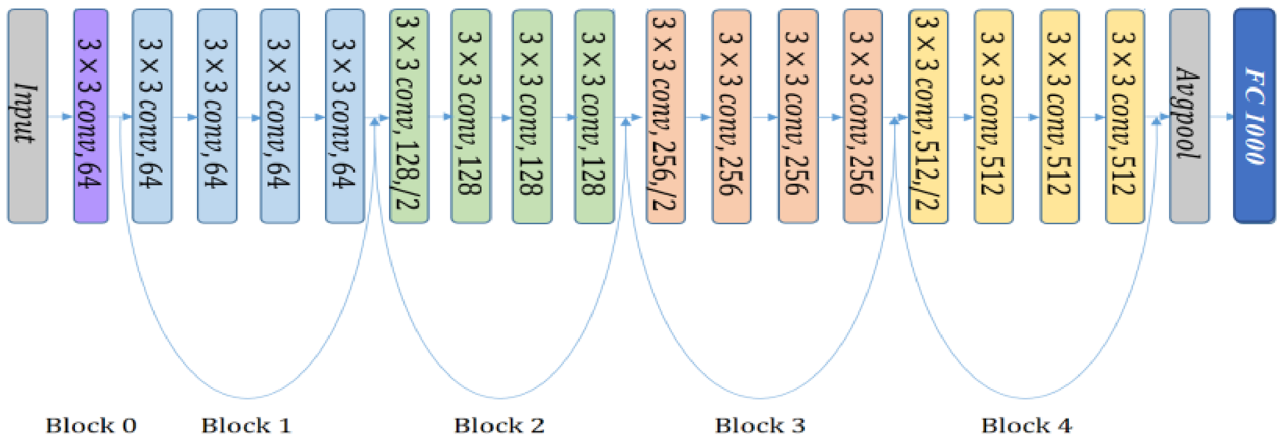

**Fig. 16**. Architecture of ResNet-18 [34 p. 1587].

The blocks in Fig. 16 are called residual blocks, skip connections are visualised by the blue under-arching arrow in between the blocks. ResNet-34 has a similar structure, also consisting of 5 blocks, but each block has twice as many convolution operations. The numbers in the name refer to the number of layers present in the network. These residual blocks replace the encoder convolution stages in the U-Net architecture.

For the training of this network, the adaptive moment estimation, also known as Adam, optimizer was used. Adam's weight update can be defined with the following formula:

$$w_{t+1} = w_t + \Delta w_t \tag{9}$$

Here $\Delta w_t$ is defined as:

$$\Delta w_t = -\alpha \frac{v_t}{\sqrt{s_t + \epsilon}} \cdot g_t \tag{10}$$

$$v_t = \beta_1 \cdot v_{t-1} - (1 - \beta_1) \cdot g_t \tag{11}$$

$$s_t = \beta_2 \cdot s_{t-1} - (1 - \beta_2) \cdot g_t^2 \tag{12}$$

In this case α is the initial learning rate; $v_t$ is the exponential average of gradients along $w_j$; $g_t$ is the gradient at iteration t along $w_j$; $s_t$ is the exponential average of squares of gradients along $w_j$; $\beta_1 \beta_2$ are the hyperparameters usually set to a constant value of 0.9 and 0.999, respectively.

To evaluate the loss of the model Dice Loss was used which is defined as:

$$DiceLoss = 1 - \frac{2|X \cap Y|}{|X| + |Y|} \tag{13}$$

Here, |X| and |Y| represent the number of elements in each set. X is the ground truth value, and Y is the predicted value. Dice loss values range from 0 to 1.

To measure accuracy the IoU metric was used, defined as follows:

$$IoU = \frac{X \cap Y}{X \cup Y} \tag{14}$$

X and Y are the same values as defined in (13); simply put. the intersection between the ground-truth value and predicted value is divided by the union of the two values. IoU values range from 0 to 1.

Implementation of these networks was done using the *segmentation models pytorch* library.

#### 2.2.2.2. Mask Region – Based Convolutional Network

Mask R-CNN is an instance segmentation network. It's heavily influenced by the architecture of Faster R-CNN [35], an object detection network. The main difference between these two architectures is that Mask R-CNN has an extra head, called the mask head, that's responsible for predicting the mask of an object. Another change is the use of RoIAlign instead of the RoIPool suggested in the Faster R-CNN network RoIAlign, as already discussed in the literature review section, allows for pooling features at different scales while retaining valuable spatial information.

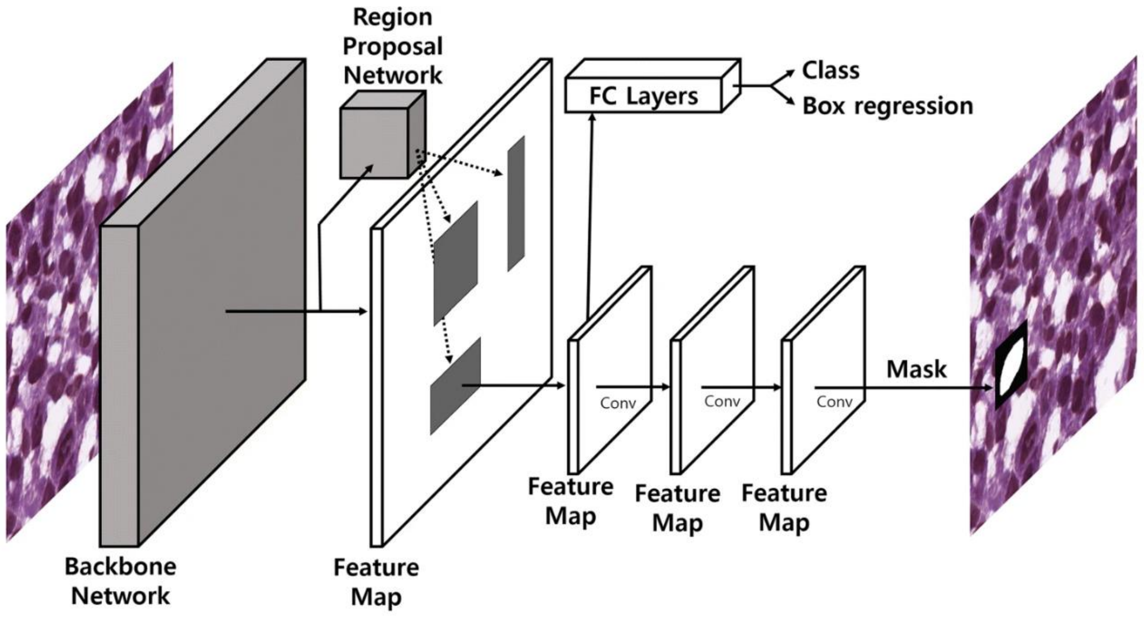

**Fig. 17**. Mask R-CNN architecture [36].

This network passes the received input through a ResNet-50 backbone pre-trained on *ImageNet* with FPN applied to it. Features are extracted at different levels of the FPN and passed along to the RPN. The RPN proposes regions in the extracted feature maps; the sizes of these regions are determined by anchors that are created using an anchor generator. These proposed regions are then passed along to the head layers. The proposals are flattened in the fully connected layers, and predictions are made on the class of the object in the proposed region along with a prediction of the bounding box of the object. To predict the masks, these same regions are passed to the mask head, where they are processed by two convolutional operations. To evaluate the total loss of the model, 5 different losses are summed [17]:

$$L = L_{cls} + L_{boxreg} + L_{mask} + L_{obj} + L_{rpn} \tag{15}$$

$L_{cls}$ is the loss associated with the classification of the object in an image; in this project, an object can have only two classes: background and nanoparticle. In this case, binary cross-entropy loss is used, which is defined in formula (4).

$L_{boxreg}$ is used to calculate the bounding box regression loss and is defined as follows:

$$L_{boxreg} = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}\left(t_i^u - v_i\right) \tag{16}$$

Here $t_i^u$ is a predicted tuple of 4 values that define the bounding box for class u, $v_i$ is the ground truth value for the bounding box, $smooth_{L_1}$is the L1 loss, that's defined with:

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2, & if\ |x| < 1 \\ |x| - 0.5, & otherwise \end{cases} \tag{17}$$

$L_{mask}$ is defined as the average binary cross-entropy loss. This loss is only accounted for RoIs that are positive that there's an object in them. Only those ground-truth pixels are accounted for that are in the RoI, meaning that the mask loss is not penalised if the proposed RoI does not cover the whole object. The mask loss for a single RoI is defined as follows:

$$L_{mask} = -\frac{1}{N_{mask}} \sum_{i=1}^{N_{mask}} m_i \log(\hat{m}_i) + (1 - m_i) \log(1 - \hat{m}_i) \tag{18}$$

Here $N_{mask}$ is the total number of ground-truth pixels in the RoI, $m_i$ is the binary ground-truth mask value for the pixel i, $\hat{m}_i$ is the predicted probability for the pixel i belonging to the instance.

$L_{obj}$ is the loss used for calculating the RPN network's capability of identifying the presence or absence of an object in a proposed RoI. It's calculated using average binary cross-entropy loss that's already defined in (4), with the only difference being that the value is averaged over the sum of positive anchor boxes and negative anchor boxes.

$L_{rpn}$ is the region box regression loss; the loss is evaluated with (10). Only in this case, the ground truth objects are the ones that have the highest IoU value out of all the proposed regions with the given object.

The model used in this project has an anchor generator with anchor sizes of 10, 24, 32, 48, and 64 with aspect ratios of 0.5, 1.0, and 2.0. Anchor sizes were chosen in accordance with the particle sizes in the data. Also, the model was modified so it could take grayscale images as input. Another essential change was made to the RoIAlign operation. The formula that assigns a RoI to a level $P_k$ of the feature pyramid was changed to:

$$k = \lfloor 3 + \log_2(\sqrt{wh}/128) \rfloor \tag{19}$$

Here 3 is the canonical target level of the feature pyramid on which a RoI with a width of w and a height of h should be mapped to, 128 is the canonical scale of the feature maps. The default values were 4 and 224. These changes were done because the authors in [15] pointed out that if the scale of a RoI becomes smaller, it should be mapped to a finer resolution, meaning to a lower level of the feature pyramid. The canonical scale was adjusted due to the fact that most of the particles in a given image were extremely small and should have appropriately small feature maps.

To evaluate the model's accuracy, the IoU metric was used (14). Another metric used is the *PASCAL VOC* Average Precision (AP) at the IoU threshold of 0.5. The IoU threshold of 0.5 means that if the IoU score between the prediction and ground-truth value exceeds this value, it is considered as a positive prediction. The AP metric helps evaluate the model's capabilities for localising an object and predicting its mask. To measure this metric, these formulas are used:

$$Precision = \frac{TP}{TP + FP} \tag{20}$$

Here TP is the number of true positive predictions, and FP is the number of false positive predictions. In the context of evaluating masking capabilities, TPs are those mask predictions where they coincide with the ground-truth mask, and FPs are those mask predictions where there shouldn't be a mask according to the ground-truth mask. Precision measures how often the model is capable of correctly predicting positive instances relative to all instances predicted as positive.

$$Recall = \frac{TP}{TP + FN} \tag{21}$$

Here FN is the amount of false negative predictions; false negative predictions are the predictions where a mask pixel is not predicted even though it should have been predicted according to the ground-truth mask [37]. The recall metric is used to measure how often the model can correctly predict positive instances out of all positive instances.

Using Precision and Recall, AP is calculated using the following formula [38]:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \dots, 1\}} p_{interp}(r) \tag{22}$$

Where $p_{interp}(r)$ is defined as:

$$p_{interp}(r) = \max_{\breve{r}:\breve{r} \geq r} p(\breve{r}) \tag{23}$$

Here $\max_{\breve{r}:\breve{r} \geq r} p(\breve{r})$ is the maximum precision in each recall interval [39].

AP is the mean precision at a set of 11 equally spaced recall levels between 0 and 1. The intuition of recall levels comes from the precision-recall curve, where the x-axis represents recall, which ranges from 0 to 1, and the y-axis which represents precision, which ranges from 0 to 1 also.

The stochastic gradient descent (SGD) algorithm with momentum was used as an optimizer in training this model. SGD's weight update can be defined with the following formula:

$$W_{t+1} = W_t - V_t \tag{24}$$

Where $V_t$ is defined as:

$$V_t = \beta \cdot V_{t-1} + \alpha \Delta W_t \tag{25}$$

Here, $\beta$ is the momentum parameter, $\alpha$ is the learning rate.

Another important method used in this model is the non-max suppression (NMS) operation. NMS is a widely used computer vision and object detection algorithm that's used to filter out duplicate detections of a single object. It works by sorting the detected bounding boxes based on their confidence score and selecting the highest score. This highest-scoring box is then compared to any other remaining boxes and discards, or suppresses, any overlapping bounding boxes with a lower confidence score. This allows to discard redundant detections and helps get more accurate detection results. In this model, NMS is used in the RPN to filter out excess RoIs.

Other key parameters in the setup of the model:

– *min_size*, defines the minimum size of the image to be rescaled before passing it through the whole network, set at a constant 256;

– *max_size*, defines the maximum size of the image to be rescaled before passing it through the whole network, set at a constant 256;

– *trainable_backbone_layers*, determines how many of the backbone layers are trainable starting from the final block of the backbone;

- *box_detections_per_img*, determines the maximum number of detections per image;
- *weights_backbone*, determines the pre-trained weights for the backbone, set to *ImageNet* pre-trained weights for the ResNet-50 backbone;
- *box_batch_size_per_image*, determines the number of proposals that are used during training of the box classification head, in other words how many samples are used to calculate the loss;
- *rpn_batch_size_per_image*, determines how many anchors are sampled during training of the RPN;
- *rpn_pre_nms_top_n_train*, determines how many anchor proposals are kept before applying NMS for training;
- *rpn_post_nms_top_n_train*, determines how many anchor proposals are kept after applying NMS during training;
- *rpn_pre_nms_top_n_test*, determines how many anchor proposals to keep before applying NMS during inference;
- *rpn_post_nms_top_n_test*, determines how many anchor proposals are kept after applying NMS during inference;
- *rpn_nms_thresh*, threshold value that determines what IoU overlap between proposed RoI warrants the use of NMS, set at a constant 0.2.

The *PyTorch* library in *Python* was used to implement this model.

## 2.3. Particle analysis methods

To perform any kind of particle analysis the particle properties must be known beforehand.

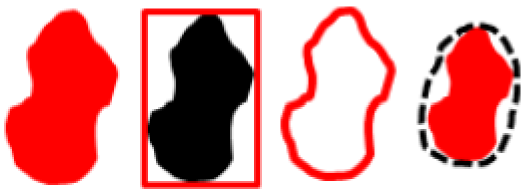

**Fig. 18**. From left to right – the area of a particle, the extent of a particle (red bounding box), the perimeter of a particle, convex area (the area marked with a dashed ellipsoid) [40].

Circularity is a dimensionless measure that determines how similar the particle is to a perfect circle; the value ranges from 0 to 1, where 1 is considered a perfect circle. It is defined with the formula:

$$f_{circularity} = 4\pi \cdot \frac{A}{P^2} \tag{26}$$

Here A is the area of the particle, P is the perimeter of the particle.

Roundness is a dimensionless metric that considers the smoothness of the edges and corners of a particle; any angular protrusions that arise from the particle itself cause a decrease in the roundness value [41]. Roundness can be calculated with the formula:

$$f_{roundness} = \frac{4 \cdot A}{\pi \cdot max_d^2} \quad (27)$$

Here $max_d$ is the diameter of the maximum inscribed circle, the largest possible circle enclosed by the particle's profile.

Solidity is the ratio of the area and the convex area; it measures the density of an object. Values range from 0 to 1, where 1 is a completely solid object, and any value less than 1 can mean that the particle has an irregular boundary or is perforated. It is defined with the following formula:

$$f_{solidity} = \frac{A}{A_{convex}} \quad (28)$$

Compactness of a particle can be calculated with the following formula:

$$f_{compactness} = \frac{\sqrt{\frac{4 \cdot A}{\pi}}}{max_d} \quad (29)$$

The Feret diameter is the longest possible distance between two points in a particle that's restricted by two tangent lines at these two points. Using the Feret diameter, the Feret angle can be measured as the angle between the x-axis parallel to the particle and the angle of the Feret diameter.

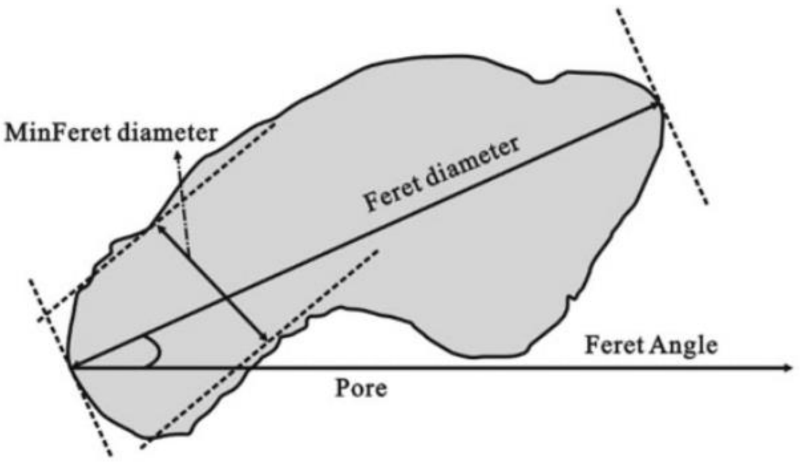

**Fig. 19**. Illustration of the Feret diameter and Feret angle on a particle [42 p. 4].

To perform particle distribution analysis the processes described below are used.

Ultimate Eroded Points (UEP) are calculated for each particle. A UEP is a morphological concept that's used to determine the last remaining pixel of a particle after erosion. Using UEPs, we are able to tell if the particles are randomly distributed, self-avoiding, or tend to cluster. The theoretical random nearest neighbour distance (NND) is calculated by:

$$NND_{rand} = 0.5 \cdot \sqrt{\frac{A}{n}} \quad (30)$$

Here, n is the total particle count in the given area. The measured theoretical NND is then compared to the measured value of the mean NND using a statistical test. Euclidean distance is used as the distance measure, which can be described with the formula:

$$d_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{31}$$

Here, $d_{i,j}$ is the Euclidean distance between particle's i UEP and particle's j UEP, $x_i$, $x_j$ and $y_i$, $y_j$ are the x and y coordinates of each particle's UEP. Mean NND is then calculated with the formula:

$$\overline{NND} = \frac{1}{N} \cdot \sum_{i=1}^{N} d_i \tag{32}$$

Where N is the total particle count, $d_i$ is the measured NND for each particle.

It's also assumed that if the mean and median are equal, then the particles are normally distributed. The first test performed is the Fisher's F-test to test the homogeneity of two variances. The null hypothesis states that the two variances are homogenous:

$$H_0: \sigma_1^2 = \sigma_2^2 \tag{33}$$

The alternate hypothesis states that:

$$\begin{aligned} &H_\alpha: \sigma_1^2 < \sigma_2^2 \quad \text{For a lower one-tailed test} \\ &H_\alpha: \sigma_1^2 > \sigma_2^2 \quad \text{For an upper one-tailed test} \end{aligned} \tag{34}$$

The F statistic is calculated as follows:

$$F = \frac{\sigma_1^2}{\sigma_2^2} \tag{35}$$

The null hypothesis is rejected if

$$\begin{aligned} &F > F_{\alpha, N_1-1, N_2-1} \\ &F < F_{1-\alpha, N_1-1, N_2-1} \end{aligned} \tag{36}$$

Here, $F_{\alpha, N_1-1, N_2-1}$ is the critical value with $N_1 - 1$ and $N_2 - 1$ degrees of freedom, where $\alpha$ is the significance level [43].

If the Fisher's F-test's null hypothesis isn't rejected, meaning that the two variances are homogenous, the Student's t-test is used for the final statistical evaluation to test the hypothesis if the means of the two sample groups are equal. The t-test hypotheses are as follows:

$$\begin{aligned} &H_0: \mu_1 = \mu_2 \\ &H_\alpha: \mu_1 \neq \mu_2 \end{aligned} \tag{37}$$

The t-test is calculated with the formula [44]:

$$t = \frac{m_A - m_B}{\sqrt{\frac{S^2}{n_A} + \frac{S^2}{n_B}}} \tag{38}$$

Here $m_A$ and $m_B$ represent the mean value of group A and B accordingly; $n_A$ and $n_B$ represent the sample sizes of the groups respectively; $S^2$ is an estimate of the pooled variance of the two groups and is calculated as follows:

$$S^2 = \frac{\sum(x - m_A)^2 + \sum(x - m_B)^2}{n_A + n_B - 2} \tag{39}$$

The degrees of freedom in this case is equal to the denominator in (33).

If Fisher's F-test's null hypothesis is rejected, it implies heteroscedasticity, and Welch's t-test is then used to compare the means of the two groups. The Welch's t-statistic is calculated with the formula [44]:

$$t = \frac{m_A - m_B}{\sqrt{\frac{S_A^2}{n_A} + \frac{S_B^2}{n_B}}} \tag{40}$$

Here $S_A^2$ and $S_B^2$ are the variances of each respective group. The degrees of freedom are estimated as follows:

$$df = \frac{(\frac{S_A^2}{n_A} + \frac{S_B^2}{n_B})^2}{\frac{S_A^4}{n_A^2(n_A - 1)} + \frac{S_B^4}{n_B^2(n_B - 1)}} \tag{41}$$

The methods described for particle analysis are implemented using *ImageJ* and the plugin *BioVoxxel*.

## 2.4. Essential libraries

### 2.4.1. PyTorch

PyTorch is an open-source machine learning library for Python that provides a wide range of tools and functionalities for building and training deep learning models. The library was developed by Facebook's AI Research team and is one of the most widely used libraries in academic research and industry applications. PyTorch provides several high-level abstractions for building complex deep learning models, including modules, layers, and optimizers. These abstractions make it easier to write and configure large and complex models. Additionally, PyTorch supports popular neural network architectures, such as ResNet, Mask R-CNN, Faster R-CNN, and many more that are beyond the scope of this project. There is also extensive support for GPU acceleration, which allows deep learning models to be trained much faster on compatible hardware [45].

### 2.4.2. PyCOCOtools

PyCOCOtools is an open-source Python library. This library provides the necessary tools for working with the MS COCO dataset. It provides a simple interface for working with the COCO dataset, including tools for loading, manipulating,, and visualizing the dataset, and also provides a way to evaluate the accuracy of object detection and or segmentation.

A MS COCO dataset is an annotation file in JavaScript Object Notation (JSON) format. In this JSON file, each image has a unique ID attached to it. Tied to this ID are the bounding box coordinates for each instance of an object in the given image, along with the mask polygons for each instance. There are more parameters attached to each image, but they are beyond the scope of this project as they do not factor into the calculations [46].

## 2.5. Hardware

The calculations were done on a JupyterLab server provided by the Artificial Intelligence Centre of Kaunas University of Technology. The CPU specifications are unknown but the most important piece of hardware is the GPU which is an Nvidia A100 40 GB.

## 3. Research results

The following chapters will detail the data preparation and training processes of 2 deep learning architectures and assess the performance of 8 distinct models on a test set in order to find the most optimal one. Using the optimal model, masks will be generated from 3 different full-sized SEM images of TiO2 nanoparticles. From these masks, particle analysis will be done using *ImageJ*, an image processing programme designed for particle analysis.

### 3.1. Data preparation

To produce the dataset, 22 images with a resolution of 256x256 were cropped from different original images. Out of these 22 images, binary masks were created by using thresholding in the image manipulation software *GIMP*. To generate more data, data augmentation was employed. Each of the 22 images and their binary masks were:

- Vertically flipped;
- Horizontally flipped;
- Vertically and horizontally flipped;
- Each flipped image variant was randomly rotated with the angles 90, 120, 180, 225, 270.

At the end, 396 images and 396 binary masks were generated. 380 images were used for training and 16 for validation. Additionally, 6 images were used for the test set; these images were not generated by the original 22 images.

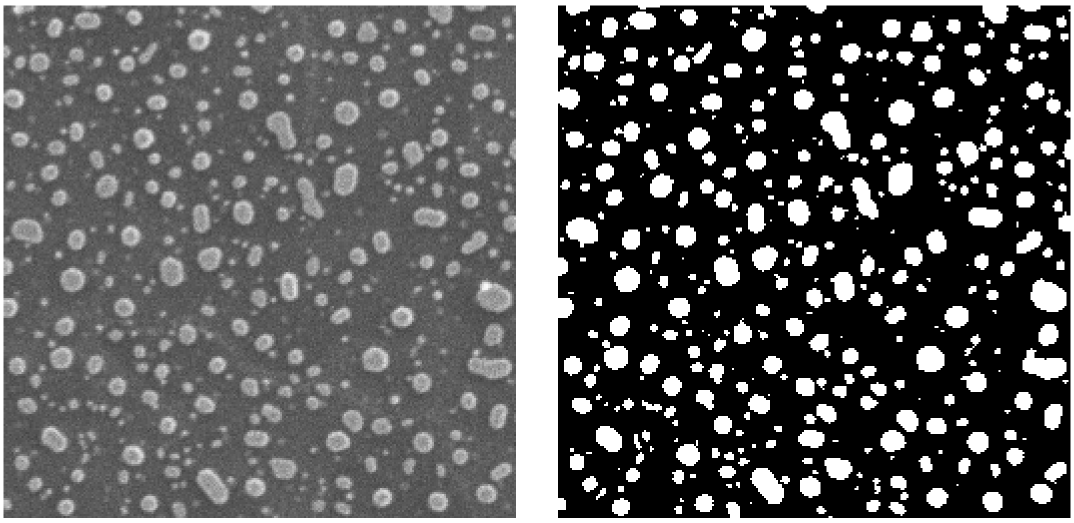

**Fig. 20**. Image (left) and the image‘s binary mask (right).

The particle amount in each image varies greatly; some images have just 400 particles in them while others surpass 4000 particles in a given image. The particles themselves also vary in size; there are quite a few that are extremely small-scale, being barely two pixels wide, and there are others that can be 40 - 50 pixels wide. In addition, the particles can have different shapes, most of the particles are elliptical in form, but there are particles that fork out or have holes inside of them.

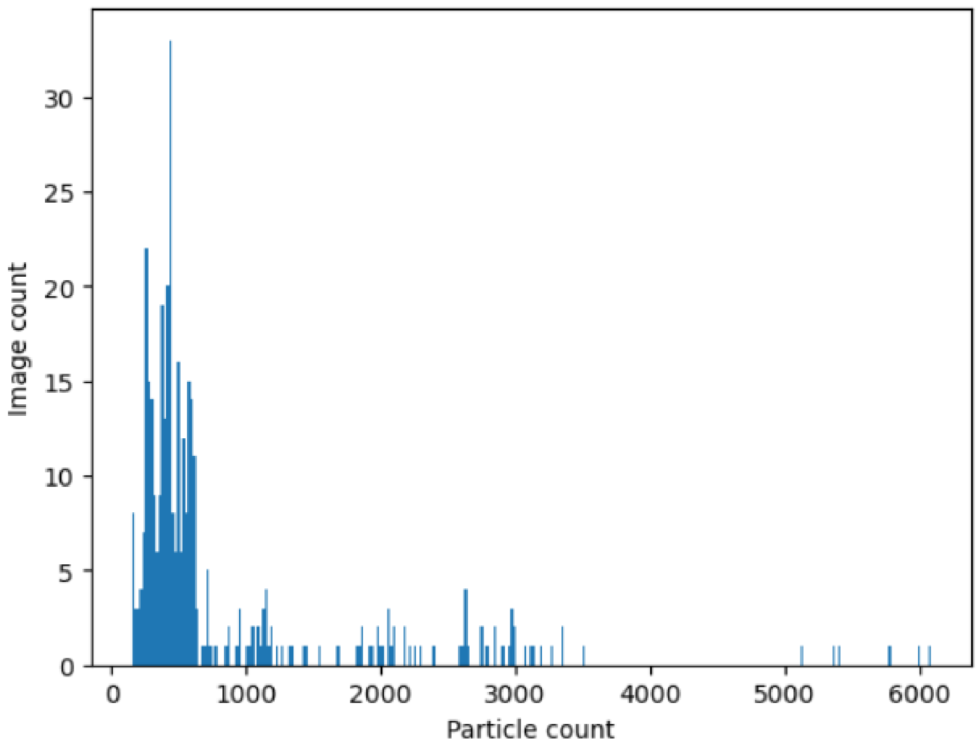

**Fig. 21**. A histogram showing the image count along with the particle count.

For instance segmentation, additional steps had to be taken to convert it to the COCO format. From each binary mask every single particle had to be saved as a different image. For example, if a binary mask has 400 particles in it, 400 images are saved separately, where in each image there’s only one particle mask.

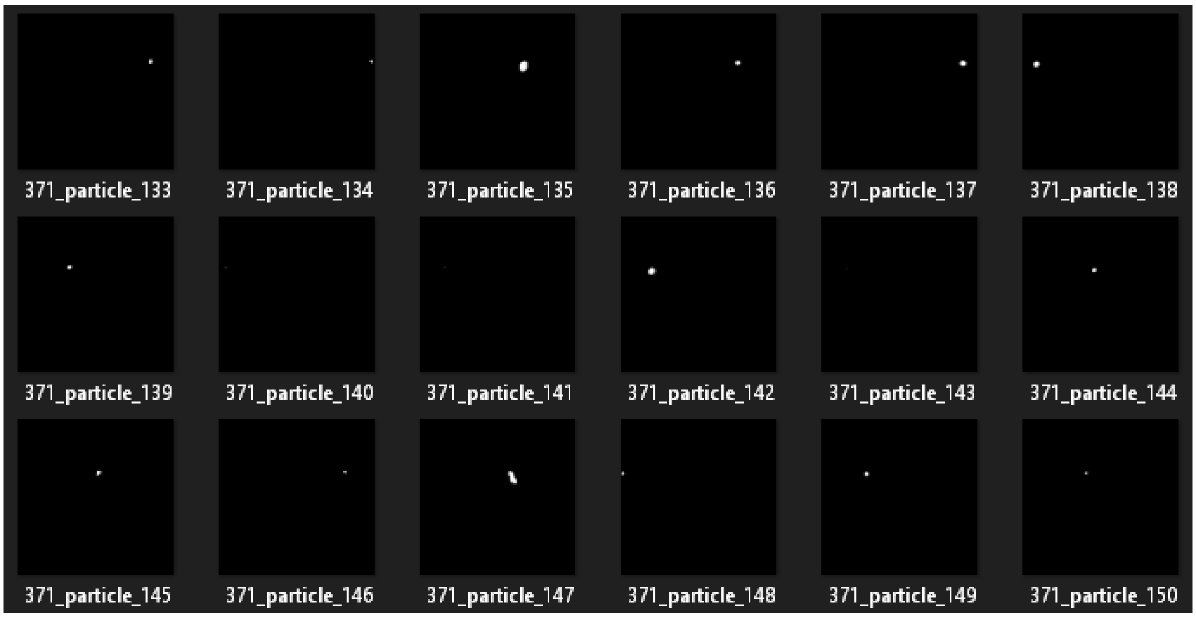

**Fig. 22**. Image showing how the particle masks should be saved separately.

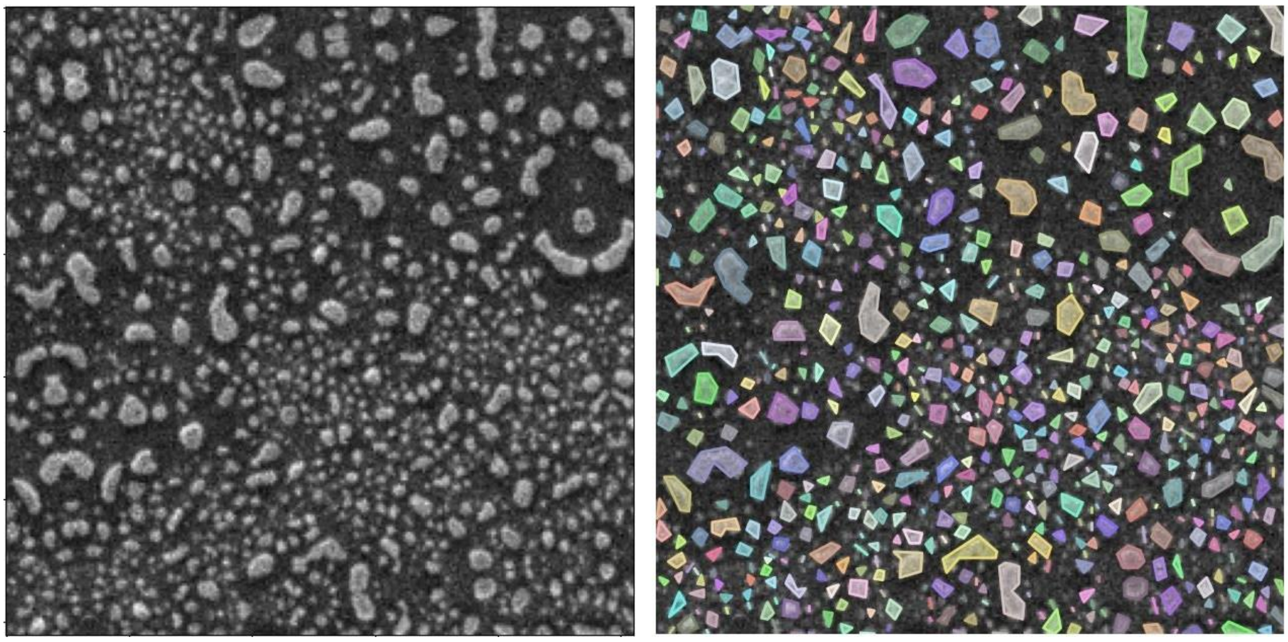
**Fig. 23**. Original image (left) and image with annotated instances (right).

The data is additionally visually augmented with a 0.4 probability to apply two transformations:

- Colour jitter with random fluctuation in brightness between 0.9 and 1.32, here 1.0 represents original brightness.
- Gaussian blur with a sigma value of 1 and kernel size of 3.

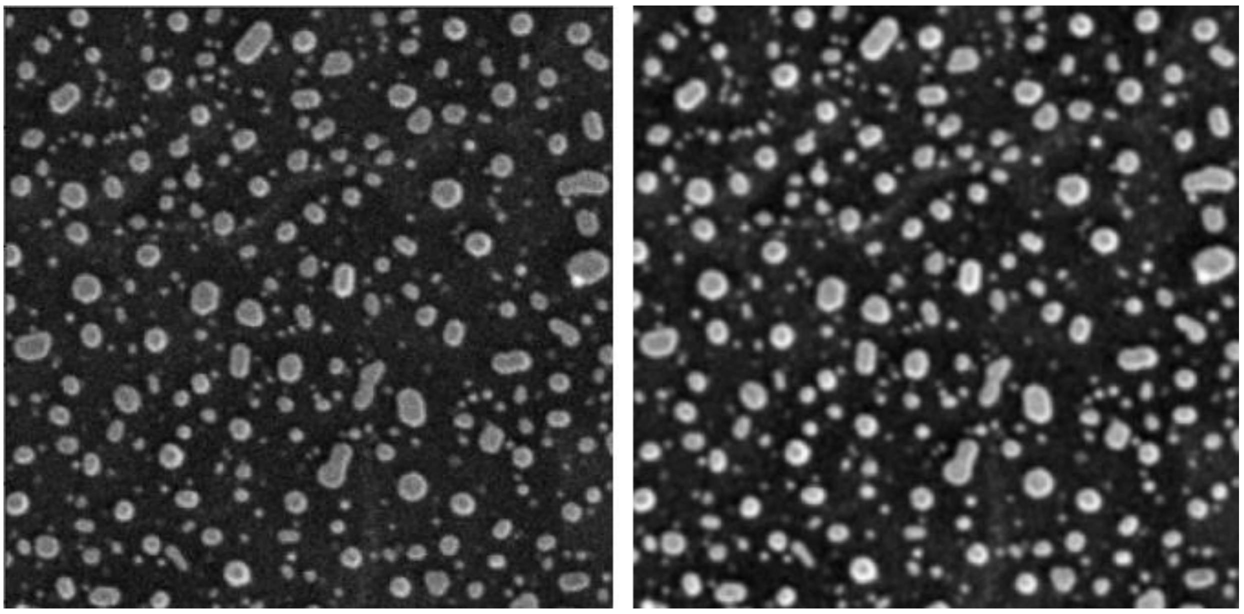
**Fig. 24**. Image without transformations (left), image with transformations applied (right).

## 3.2. Models

### 3.2.1. Mask Region-Based Convolutional Neural Network

The Mask R-CNN model was configured in 4 different ways and all the different configurations were trained on the same dataset.

**Table 3**. Different parameter setup of the Mask R-CNN model.

| **Model No.** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| **box_detections_ per_img** | 1500 | 2000 | 2500 | 3000 |
| **weights_backbone** | 'ResNet50_Weights.IMAGENET1K_V2’ | | | |
| **box_batch_size_ per_image** | 2000 | 2500 | 3000 | 3500 |
| **rpn_batch_size_ per_image** | 2000 | 2500 | 3000 | 3500 |
| **rpn_pre_nms_ top_n_train** | 3500 | 6500 | 6500 | 5500 |
| **rpn_post_nms_ top_n_train** | 2500 | 3000 | 3000 | 3000 |
| **rpn_pre_nms_ top_n_test** | 3500 | 5000 | 5000 | 4000 |
| **rpn_post_nms_ top_n_test** | 2500 | 3000 | 3000 | 3000 |
| **rpn_nms_thresh** | 0.2 | | | |

A similar training approach was used as suggested by [26], the heads were trained separately, and only then was the full network trained on the visually augmented data that was described in Section 3.1. The network heads were trained for 30 epochs using the SGD optimizer with a momentum value of 0.9 and a learning rate of 0.01 for the first 15 epochs, which was later adjusted to 0.00001. After training the heads, the whole network is trained for 50 epochs with the same learning rate adjustment and optimizer parameters as the heads. During the training of the whole network, the model is evaluated on the validation set after each epoch and is checkpointed if the IoU score on the validation set exceeds the previously set maximum.

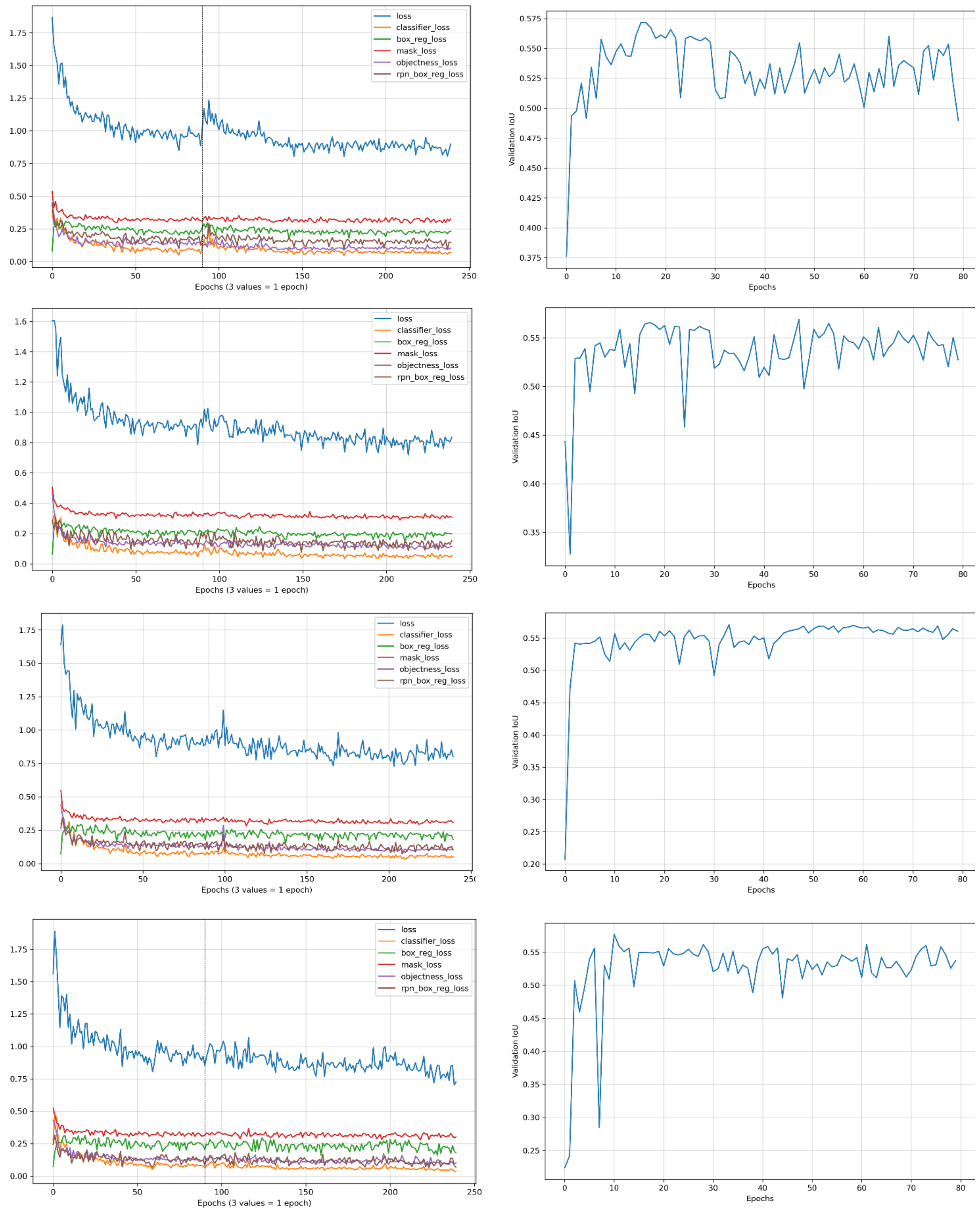

**Fig. 25**. Each row of graphs show a model's training loss (left) and the model's IoU score on the validation set (right), the dashed line in the training graph marks where the head training ends, from top to bottom the models are as listed: Model #1, Model #2, Model #3, Model #4.

Judging by Fig. 25, the models seem to benefit from the separate training of the heads and the network, as the overall loss decreases after unlocking the backbone layers after 30 epochs. The mask and box regression losses tend to fluctuate around the same values and are not affected by unlocking the backbone layers. Whereas the classifier, RPN box regression, and objectness losses show a slightly more positive response to the unlocking of the backbone layers as they continue to decrease ever so slightly more. The IoU scores on the validation set fluctuate around the 0.55 mark. Model #3 seems to experience the least amount of fluctuations as it can be seen that around 40 epochs it starts to even out.

**Table 4**. Performance of the models on the validation set.

| **Model** | **Validation AP@IoU=0.5** | **Validation IoU** | **Best epoch** |
|---|---|---|---|
| Model #1 | 0.291 | 0.5601 | 66 |
| Model #2 | 0.296 | 0.5689 | 48 |
| Model #3 | 0.267 | 0.5705 | 34 |
| Model #4 | 0.292 | 0.5619 | 61 |

Even though all of the models reached higher AP scores than Model #3 (see Table 4); it managed to get the highest IoU score on the validation set with the least number of epochs – 34. The models are further evaluated on the test set.

**Table 5**. The models and their reported test IoU.

| **Model** | **Test IoU** |
|---|---|
| Model #1 | 0.6019 |
| Model #2 | 0.6170 |
| Model #3 | 0.6249 |
| Model #4 | 0.6072 |

Against all odds, the model with the smallest AP on the validation set and with the least amount of epochs trained scored the highest on the test set (see Table 5). Model #3 managed to reach a test IoU score of 0.6249. Another interesting thing to note is that the longest trained models, model #1 and model #4, scored the lowest on both the validation and test sets.

This means that the parameters chosen for Model #3 (see Table 3) are best suited for this type of data. Overall, the models, based on their AP scores on the validation set, exhibit poor object localization and masking capabilities. The best-performing Mask R-CNN model was chosen to be Model #3.

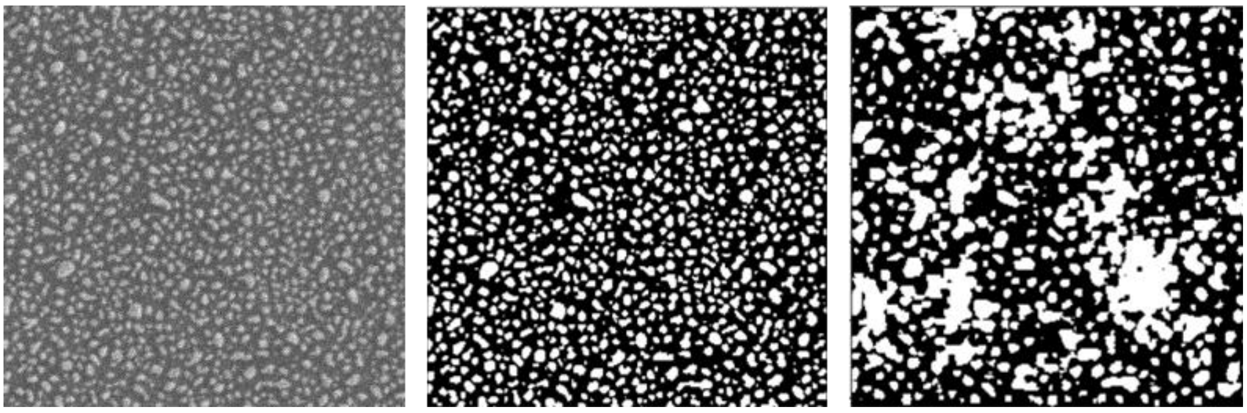
**Fig. 26**. Original image (left), ground-truth mask (middle), Model #3 prediction as a binary mask (right).

The capabilities of the model are depicted in Fig. 26; this is the most difficult image out of the whole test set, having over 1000 particle objects. Judging by the prediction, the model seems to have difficulties masking smaller particles and is prone to predicting overlapping masks when particles agglomerate in one area. In this agglomerate of particles, even the larger particles with more prominent features tend to get lost. Such behaviour from the model can be expected from the low AP score that it exhibited on the test set.

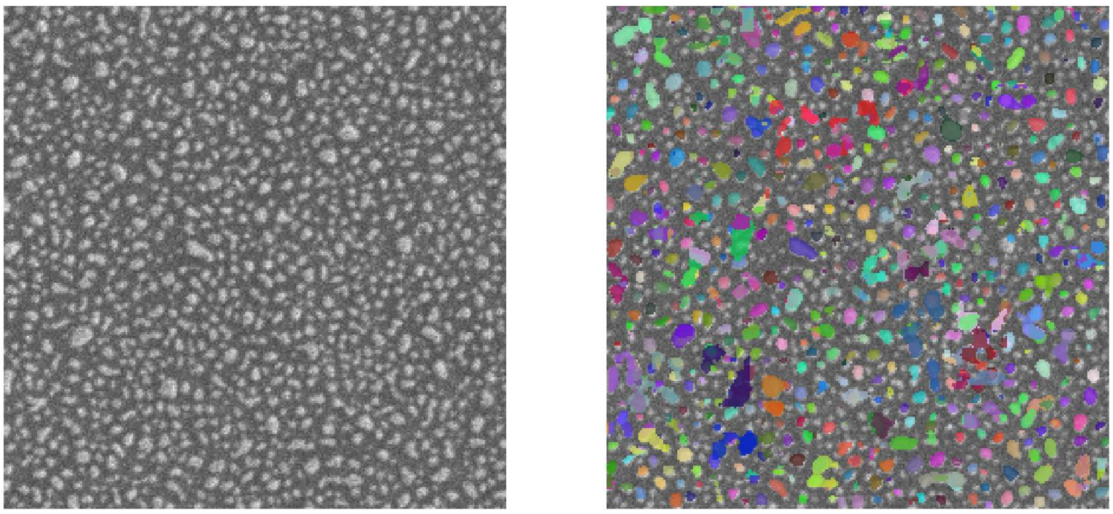

**Fig. 27**. Same image that‘s shown Fig. 26 showcasing instance segmentation, instead of a binary mask.

### 3.2.2. U-Net

The U-Net architecture was setup in 4 different ways: the encoder part is either ResNet – 18 or ResNet – 34 and either is pretrained on *ImageNet* or not. In total, 4 different models are used. Models were trained for 50 epochs on augmented data, and based on their validation IoU score, they were checkpointed mid-training. An Adam optimizer with a learning rate of 0.0001 was used for training, with a batch size of 1 image per batch.

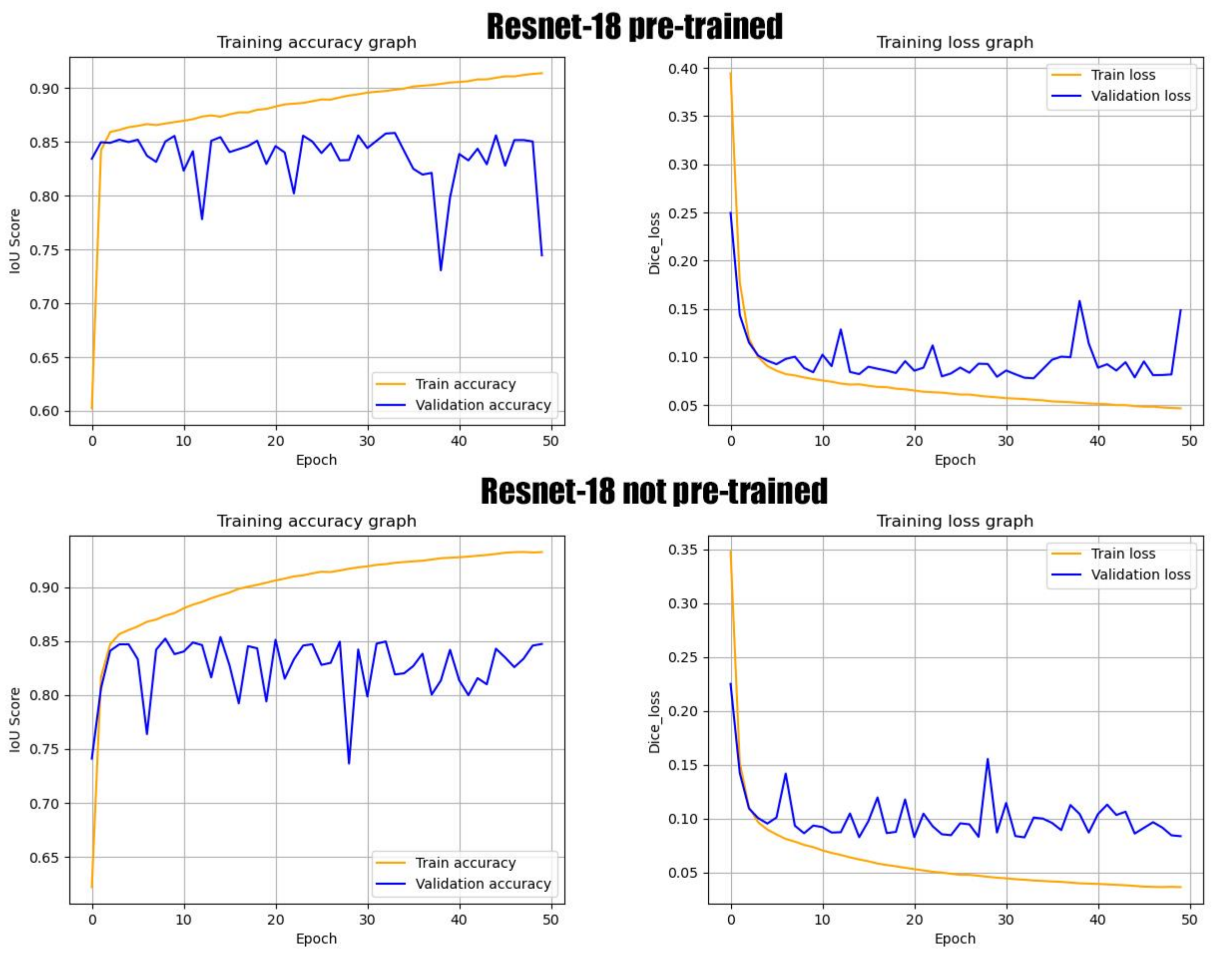

**Fig. 28**. The training process of U-Net with ResNet – 18 backbone with pre-trained weights (top) and U-Net with a ResNet – 18 backbone with not pre-trained weights (bottom).

Judging by Fig. 28, the training process for U-Net with a pre-trained ResNet-18 backbone shows better validation results than the alternative with a not pre-trained backbone. The validation accuracy with the pre-trained backbone often climbs over the 0.85 IoU score, whereas the other model mostly stays under the 0.85 value. Also, the loss curve on the pre-trained backbone tends to experience fewer fluctuations and more often stays under the 0.1 dice loss value.

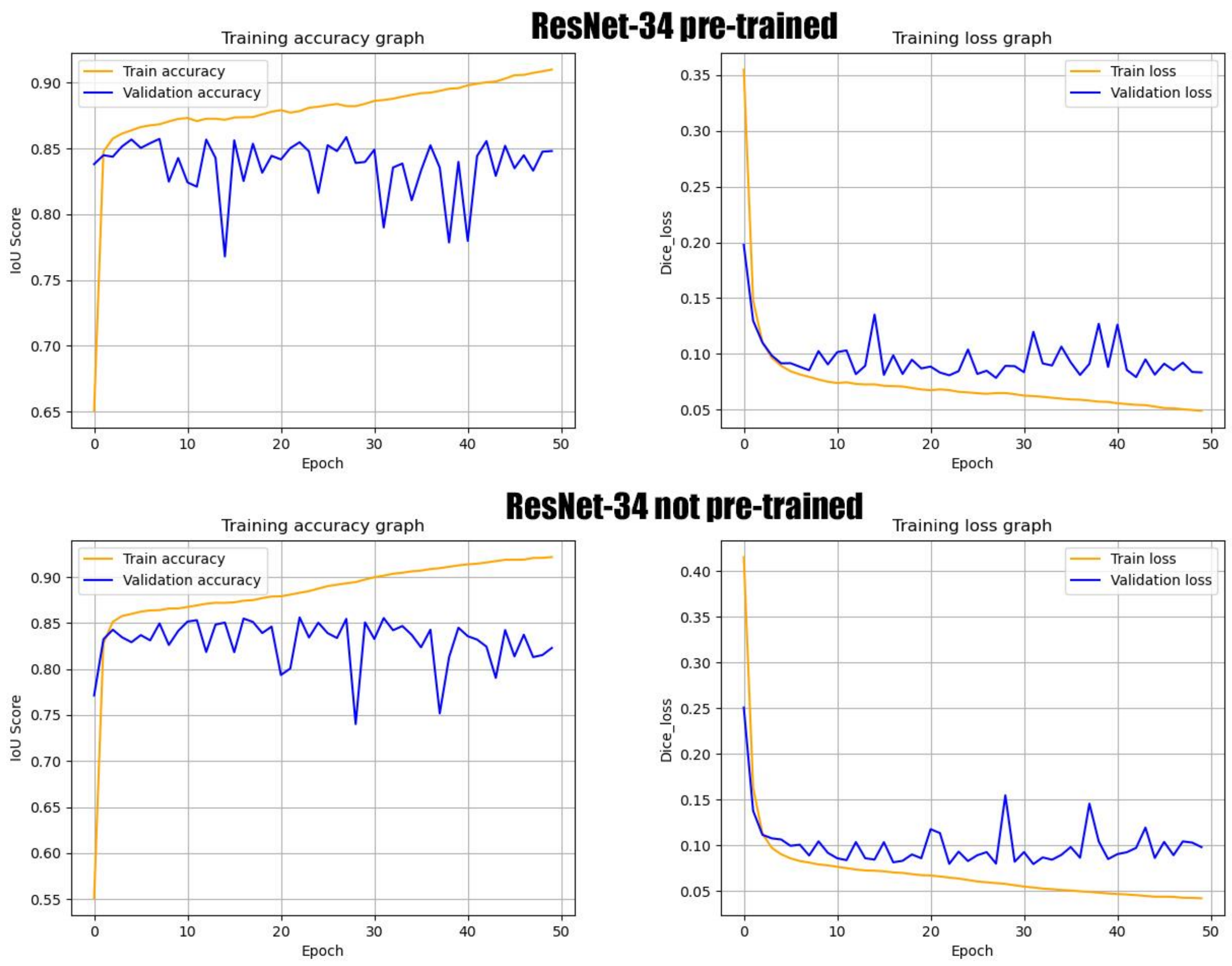

**Fig. 29**. The training process of U-Net with ResNet – 34 backbone with pre-trained weights (top) and U-Net with a ResNet – 34 backbone with not pre-trained weights (bottom).

U-Net with a ResNet-34 encoder was trained using the same training parameters as the ResNet-18 encoder. The training patterns in Fig. 29 do not differ much from the ones observed in Fig. 28, except that in this case the model is overfitting more slowly.

**Table 6**. Validation IoU and best epoch on which the model was checkpointed of all 4 models.

| **Model** | **Validation IoU** | **Best epoch** |
|---|---|---|
| **U-Net ResNet-18 pre-trained** | 0.8582 | 34 |
| **U-Net ResNet-18 not pre-trained** | 0.8537 | 15 |
| **U-Net ResNet-34 pre-trained** | 0.8586 | 28 |
| **U-Net ResNet-34 not pre-trained** | 0.8560 | 23 |

With the information provided in Table 6, U-Net with a ResNet-34 pre-trained encoder manages to achieve the highest IoU score of 0.8586 out of all 4 models at the 28th epoch. U-Net with a pre-trained ResNet-18 encoder seems to be not far off from the best, reaching a validation IoU of 0.8582 at the 34th epoch. For further assessment of the best U-Net model, they're tested on the test set.

**Table 7**. IoU score of 4 different U-Net models on the test set.

| **Model** | **Test IoU** |
|---|---|
| **U-Net ResNet-18 pre-trained** | 0.8549 |

| | |
|---|---|
| **U-Net ResNet-18 not pre-trained** | 0.8466 |
| **U-Net ResNet-34 pre-trained** | 0.8766 |
| **U-Net ResNet-34 not pre-trained** | 0.8512 |

Evaluation results on the test set shown in Table 7 show that the best performing model overall is U-Net with a pre-trained ResNet-34 backbone, achieving an IoU score of 0.8766, surpassing the second-best model by 0.0217. It's worthy to note that the non-pre-trained models performed the worst on both the validation and test sets.

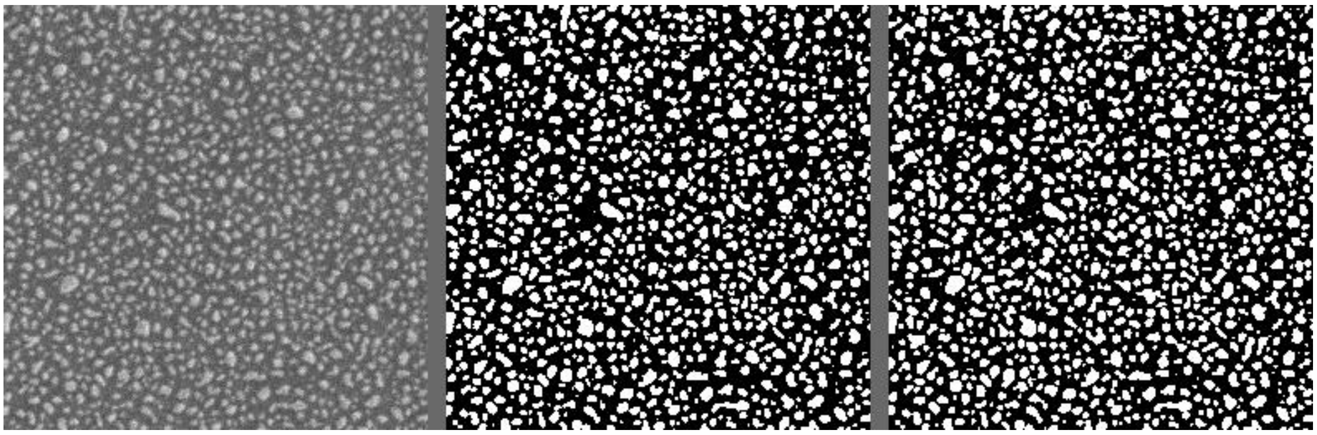

**Fig. 30**. Original image (left), ground truth mask (middle), the best performing U-Net model's predicted mask for the original image (right).

Fig. 30 showcases the capabilities of the best-performing U-Net model; it managed to detect 1042 out of 1059 particles in the given image. Even though the test IoU score isn't perfect, it can at least detect a proper number of particles in the given image without missing too much information. Although it can be observed that this model slightly struggles to predict separate masks for agglomerated small objects, therefore, there's some variability in how many particles are actually in the image because overlapping masks are counted as one, as seen in Fig. 31.

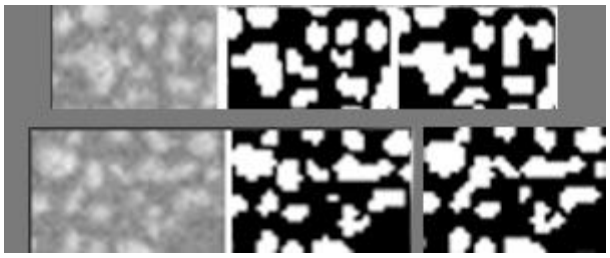

**Fig. 31**. Original image area (left), ground-truth mask (middle), predicted mask by U-Net (right).

### 3.2.3. Best model comparison

The best Mask R-CNN model, model #3, and the best U-Net model, U-Net with a pre-trained ResNet-34 backbone, are compared based on their performance on the test set.

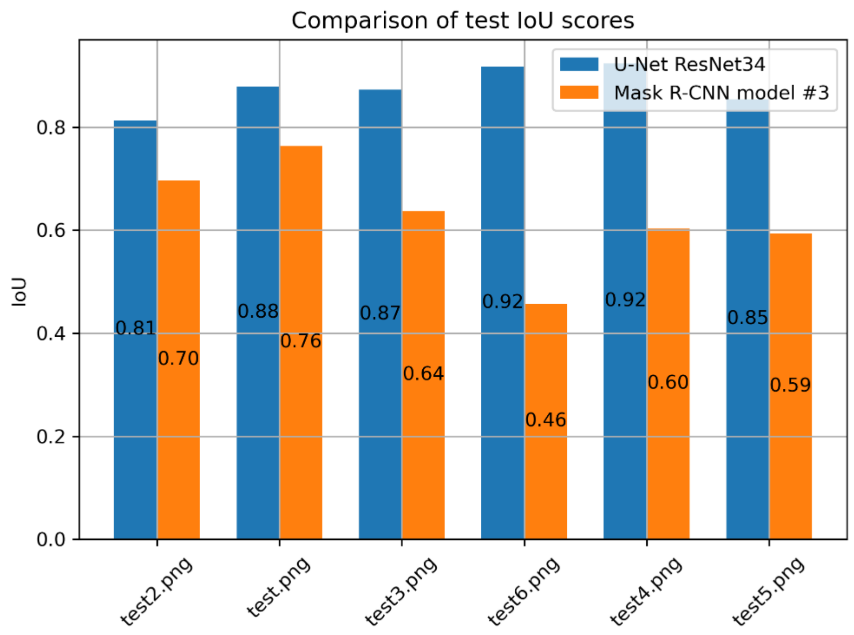

**Fig. 32**. The comparison of the two models‘ performance on the test set.

Fig. 32 demonstrates the stark difference between the two models' capabilities. U-Net greatly outperforms the Mask R-CNN model on the given data. This assumption can be further bolstered by comparing Fig. 30 and Fig. 26, in which the masking performance is visualised. The image that the performance was visualised on is *test6.png*, and as one can see in Fig. 32, U-Net reached a 0.92 IoU score on this image, whereas the Mask R-CNN model reached only a 0.46 IoU score. On the whole test set, model #3 achieved a mean IoU score of 0.6249, while the best U-Net model reached a mean IoU score of 0.8766. Having evaluated the performance of these two models, it is safe to say that the overall best model is the U-Net model.

### 3.3. Particle analysis

To perform particle analysis 3 SEM TiO2 images were chosen with a resolution of 1280x960. The images are then divided into patches with a size of 256x256 and fed through the U-Net model to predict their masks. Patches that were not able to fit within the height boundary of the image were mirrored until they reached the size of 256x256. Predictions were made on 20 patches per image. The predicted mask patches are then stitched back together, and the mirrored part is cropped out.

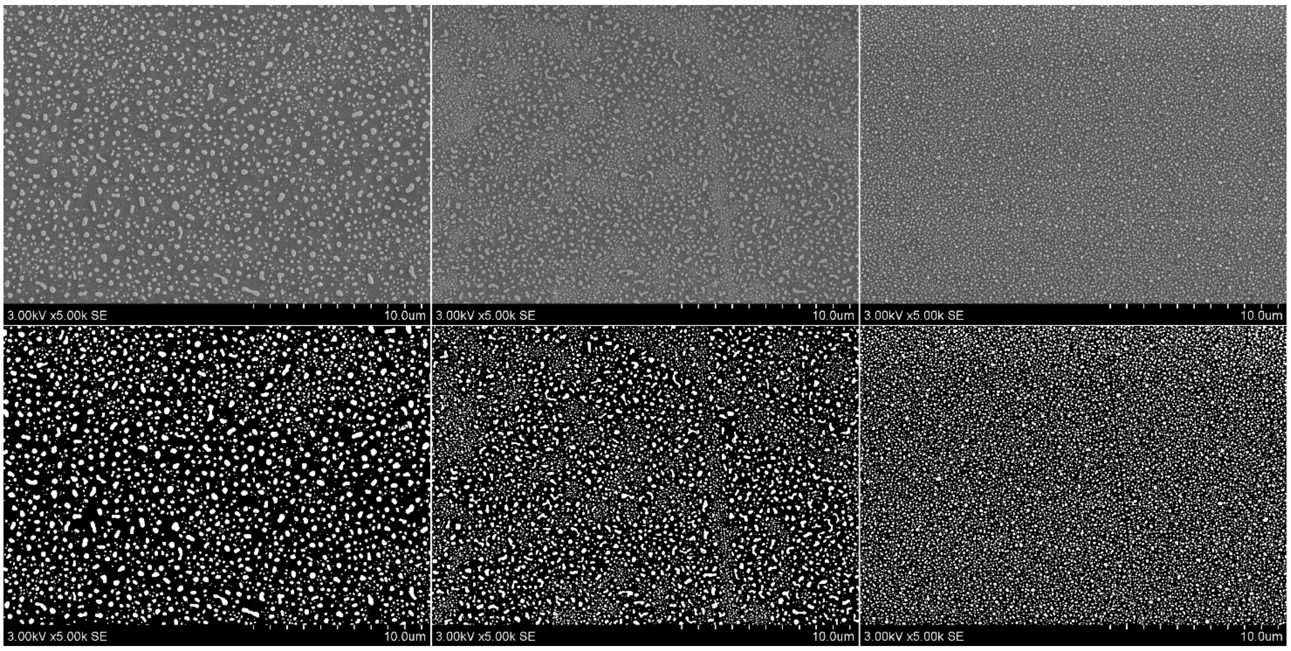

**Fig. 33**. Image #1 and its respective mask below (left), Image #2 and its mask below (middle), Image #3 and its mask below (right).

Segmentation of each image took around 1.4 s.

Image #1 was measured to have 3472 particles, Image #2 had 8179 particles, and Image #3 had 14733 particles.

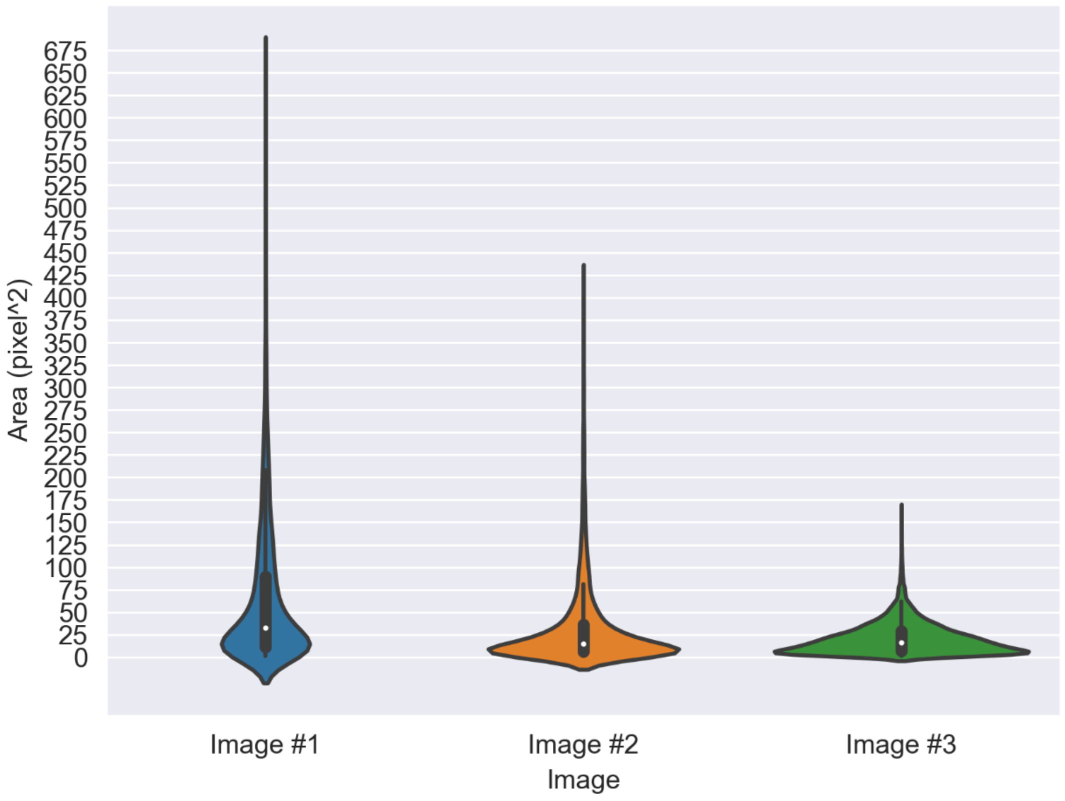

**Fig. 34**. Violin plots of each image's particle area.

Judging by Fig. 34, we can see that the images have the highest densities of particle area sizes between 0 and 25. Image #1, having the least number of particles, tends to have more spread-out area values;

this is also shown by the size of the interquartile range. Whereas images #2 and #3 have much tighter distributions of particle area sizes.

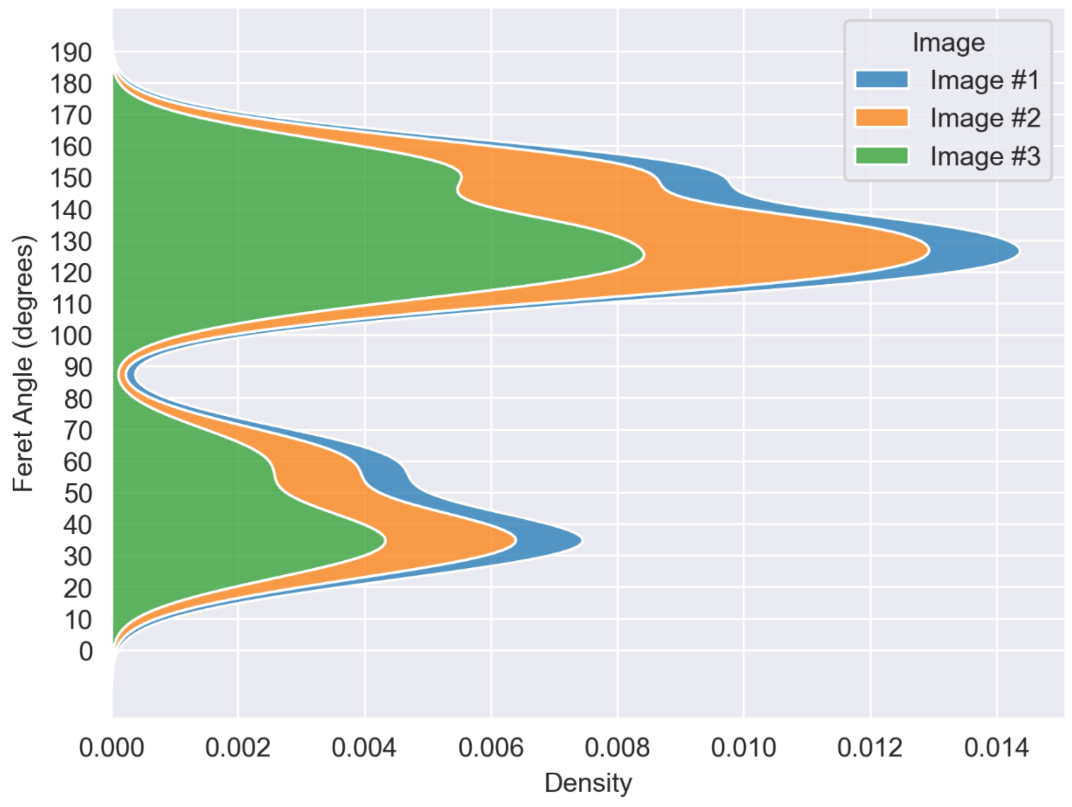

**Fig. 35**. Kernel Density plot of Feret Angle values for each image.

The graph shown in Fig. 35 is bimodal, with the highest density regions being around 40° and 130°. There is also the presence of a steep valley at the 90° angle, indicating that observations at such an angle are sparse. Overall, the particles have diverse orientations.

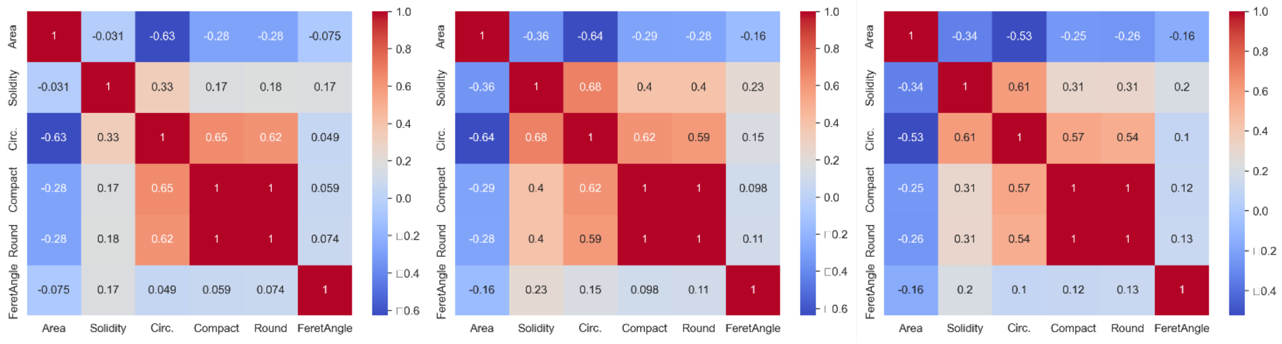

**Fig. 36**. Correlation matrices for the three images between the variables Area, Solidity, Circularity, Compactness, Roundness, Feret Angles. Image‘s #1 correlation matrix is on the left, image‘s #2 correlation matrix is in the middle, and image‘s #3 correlation matrix on the right.

By observing the presented correlation matrices in Fig. 36, it is apparent that the variables compactness and roundness are entirely dependent on each other. Other observed variables, such as circularity, exhibit strong correlations with area, solidity, compactness, and roundness. If one were to develop a regression model with a dependent variable, it would be advisable to remove these independent variables due to the fact that multicollinearity negatively impacts the performance of such predictors.

**Table 8**. Averaged results for each image.

| **Image** | **Particle count** | **% Area coverage** | **Circularity** | **Solidity** | **Feret Angle** |
|---|---|---|---|---|---|
| #1 | 3472 | 19.237 | 0.948 | 0.910 | 97.939° |
| #2 | 8179 | 22.144 | 0.924 | 0.893 | 106.823° |
| #3 | 14733 | 26.446 | 0.956 | 0.897 | 105.275° |

**Table 9**. Result output from *ImageJ* for testing if the image samples are significantly different from random distribution along with the classification of the particles.

| **Image** | **Theoretical random NND** | **Measured median NND** | **Sample size** | **Test** | **Critical T value** | **T value** | **Classification** | **Confidence Interval** |
|---|---|---|---|---|---|---|---|---|
| #1 | 9.087 | 11.636 | 3472 | Welch's t-test | 1.645 | 29.403 | Self-avoiding particles | 95% |
| #2 | 5.920 | 7.480 | 8179 | | | 36.654 | | |
| #3 | 4.411 | 6.569 | 14733 | | | 104.354 | | |

In reference to Table 9, all three images were categorised as containing self-avoiding particles, which indicates that these particles avoid clustering and tend to stay separate. The distribution of particles in all images was significantly different from random distribution with a significance level of 0.05, indicating that the observed patterns of particles in all the given images were unlikely to have occurred by chance.

## Conclusions

1. The literature review showed the capabilities and shortcomings of conventional segmentation methods and highlighted the abilities of deep learning approaches for particle analysis. Further research showed that convolutional neural networks perform exceptionally well on such tasks and are capable of discerning features from irregularly lit or shaped particles.

2. To train the convolutional neural networks, a dataset was created from 22 256x256 images. From these 22 images, 396 images were generated by applying random rotations, horizontal and vertical flips. 380 images were chosen to be used for training the convolutional neural networks, and 16 images were chosen to represent the validation set. Additionally, 6 256x256 images that were not present in either the training or validation sets were chosen to serve as the test set. The training set was further augmented visually by applying a random brightness scaling factor between 0.9 and 1.32, and a gaussian filter to induce slight blur to the images.

3. To implement the deep convolutional neural networks, the *Python* library *PyTorch* was used. From this library the instance segmentation Mask Region-based convolutional neural network was used with 4 different parameter setups, the primary parameter change being the limit of how many objects the network can detect in a given image. The other convolutional neural network chosen was U-Net, 4 different implementations of it were used. Encoders were set to either ResNet-18 or ResNet-34, and they were either pre-trained on *ImageNet* or not.

4. The different convolutional neural networks were trained and evaluated on the test and validation sets, it was assumed that the Mask Region-based convolutional neural network would perform better than U-Net due to the latter's tendency to miss the edges of particles and thus segment them as one. This, however, proved to be false as the best-performing Mask Region-based convolutional neural network, trained for 34 epochs, with a 2500 object detection limit, reached a validation mean intersection over union score of 0.5705, and a test mean intersection over union score of 0.6249. Whereas, the best performing U-Net model with a pre-trained ResNet-34 backbone, trained for 28 epochs, was able achieve a validation mean intersection over union of 0.8586 and test mean intersection over union score of 0.8766, greatly outperforming the best-performing Mask Region-based convolutional neural network.

5. Statistical analysis was performed on three images that were segmented by U-Net. The segmented images showed varying particle counts: Image #1 had 3472 particles, Image #2 had 8179 particles, and Image #3 had 14733 particles. Using violin plots, the distribution densities of the particle area sizes were examined, showing that Image #1 has a more spread-out pattern compared to Image #2 and #3, which showed significantly tighter distributions between 0 and 25 area sizes. The kernel density plot of Feret angles exhibited a bimodal distribution, suggesting diverse particle orientations. A Strong correlation was observed between the variables compactness, roundness, circularity, area, and solidity. Further statistical testing by utilising Welch's T-Test confirmed that the observed particle patterns in the images were significantly different from random distribution and were categorised as self-avoiding particles.

## List of references

1. **European Chemicals Agency.** Substance Information. *ECHA.* [Online] An agency of the European Union, 21 April 2023. [Cited: 15 May 2023.] https://echa.europa.eu/lt/substance-information/-/substanceinfo/100.033.327.

2. **U.S. Geological Survey.** USGS Publications Warehouse. *Titanium and Titanium Dioxide.* [Online] January 2021. [Cited: 15 May 2023.] https://pubs.usgs.gov/periodicals/mcs2021/mcs2021-titanium.pdf.

3. *Conditional Random Fields Meet Deep Neural Networks for Semantic Segmentation: Combining Probabilistic Graphical Models with Deep Learning for Structured Prediction.* **Anurag Arnab, Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Mans Larsson, Alexander Kirilov, Bogdan Savchynskyy, Carsten Rother, Fredrik Kahl, Philip Torr.** 1, s.l. : IEEE Signal Processing Magazine, 2018, Vol. 35. 1558-0792.

4. *Image Segmentation by Using Thershod Tehcniques.* **Salem Saleh Al-amri, N.V. Kalyankar, Khamitkar S.D.** 5, s.l. : Journal of Computing, 2010, Vol. 2. 2151-9617.

5. **Chris Solomon, Toby Breckon.** *Fundamentals of Digital Image Processing A Practical Approach with Examples in Matlab.* Chichester : Wiley-Blackwell, 2011. 978-0-470-84473-1.

6. *Gray-scale and colour image segmentation via region growing and region merging.* **N. Ikonomakis, K.N. Plataniotis, A.N. Venetsanopoulos.** 1-2, Toronto : Canadian Journal of Electrical and Computer Engineering, 1998, Vol. 23. 0840-8688.

7. *A hybrid region growing algorithm for medical image segmentation.* **D. Muhammad Noorul Mubarak, M. Mohamed Sathik, S.Zulaikha Beevi, K. Revathy.** 3, Trivandrum : International Journal of Computer Science & Information Technology, 2012, Vol. 4.

8. *Use of watersheds in contour detection.* **S. Beucher, C. Lantuejoul.** Rennes : International workshop on image processing, real-time edge and motion detection, 1979.

9. **Council, Long Tom Watershed.** Long Tom Watershed Council. *What is a Watershed?* [Online] [Cited: 15 May 2023.] https://www.longtom.org/about-ltwc/watershed-diagram/.

10. **NOAA.** National Ocean Service. *What is a watershed?* [Online] 20 January 2023. [Cited: 15 May 2023.] https://oceanservice.noaa.gov/facts/watershed.html.

11. *Drivers of Cyanobacterial Blooms in a Hypertrophic lagoon.* **Mindaugas Zilius, Diana Vaiciute, Marco Bartoli, Mariano Bresciani.** 434, s.l. : Frontiers in Marine Science, 2018, Vol. 5.

12. *Multi-scale image segmentation of coal piles on a belt based on the Hessian matrix.* **Zhang Zelin, JianguoYang, Su Xiaolan, Ding Lihua, Wang Yuling.** 5, s.l. : Particuology, 2013, Vol. 11. 1674-2001.

13. *Image segmentation method for coal particle size distribution analysis.* **Bai Feiyan, Fan Minqiang, Yang Hongli, Dong Lianping.** s.l. : Elsevier, 2021, Vol. 56. 1674-2001.

14. *Evaluating human versus machine learning perfromance in classifying research abstracts.* **Yeow Chong Goh, Xin Qing Cai, Walter Theseira, Giovanni Ko, Khiam Aik Khor.** s.l. : Scientometrics, 2020, Vol. 125. 1197-1212.

15. *Feature Pyramid Networks for Object Detection.* **Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie.** s.l. : IEEE Conference on Computer Vision and Pattern Recognition, 2017.

16. *Deep learning enabled particle analysis for quality assurance of construction materials.* **Zeng Ziyue, Wei Yongqi, Wei Zhenhua, Yao Wu, Wang Changying, Huang Bin, Gong Mingzi, Yang Jiansen.** s.l. : Automation in construction, 2022, Vol. 140. 0926-5805.

17. *Mask R-CNN.* **Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick.** Venice : IEEE, 2017. 2380-7504.

18. *ISTR: End-to-end instance segmentation with transformers.* **Hu Jie, Cao Liujuan, Lu Yao, Zhang ShengChuan, Wang Yan, Li Ke, Huang Feiyue, Shao Ling, Ji Rongrong.** s.l. : arXiv, 2021.

19. *Fast R-CNN.* **Girshick, Ross.** Santiago : IEEE, 2015. 2380-7504.

20. **Olaf Ronneberger, Philipp Fischer, Thomas Brox.** U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI.* s.l. : Springer, 2015. Vol. 9351. 978-3-319-24574-4.

21. *U-Net-Based Segmentation of Microscopic Images of Colorants and Simplification of Labeling in the Learning Process.* **Ikumi Hirose, Mari Tsunomura, Masami Shishikura, Toru Ishii, Yuichiro Yoshimura, Keiko Ogawa-Ochiai, Norimichi Tsumura.** 7, s.l. : MDPI, 2022, Vol. 8.

22. *Formation of Au nanostructures on the surfaces of annealed TiO2 thin films.* **Mantas Sriubas, Vytautas Kavaliūnas, Kristina Bočkutė, Paulius Palevičius, Marius Kaminskas, Žilvinas Rinkevičius, Minvydas Ragulskis, Giedrius Laukaitis.** s.l. : Surfaces and Interfaces, 2021, Vol. 25. 2468-0230.

23. **Tanguy Naets, Maarten Huijsmans, Paul Smyth, Laurent Sorber, Gael de Lannoy.** *A Mask R-CNN Approach to Counting Bacterial Colony Forming Units in Pahrmaceutical Development.* s.l. : arXiv, 2021.

24. *Segmentation of mine overburden dump particles from images using Mask R CNN.* **Shubham Shrivastava, Sudipta Bhattacharjee, Debasis Deb.** 2046, s.l. : Scientific Reports, 2023, Vol. 13. 2045-2322.

25. **Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hayes, Pietro Perona, Deva Ramanan, Lawrence Zitnick, Piotr Dollar.** Microsoft COCO: Common Objects in Context. s.l. : arXiv, 2015. 1405-0312.

26. *Deep Learning Based Instance Segmentation of Titanium Dioxide Particles in the Form of Agglomerates in Scanning Electron Microscopy.* **Paul Monchot, Loic Coquelin, Khaled Guerroudj, Nicolas Feltin, Alexandra Delvalee, Loic Crouzier, Nicolas Fischer.** 4, s.l. : Nanomaterials, 2021, Vol. 11. 1104-0968.

27. **IBM.** IBM. *Convolutional Neural Networks.* [Online] IBM. [Cited: 15 May 2023.] https://www.ibm.com/topics/convolutional-neural-networks.

28. **E, Swapna K.** Developers Breach. *What is a CNN?* [Online] [Cited: 15 May 2023.] https://developersbreach.com/convolution-neural-network-deep-learning/.

29. **DeepAI.** *Max Pooling.* [Online] DeepAI. [Cited: 15 May 2023.] https://deepai.org/machine-learning-glossary-and-terms/max-pooling.

30. **The PyTorch Foundation.** PyTorch. *RELU.* [Online] [Cited: 15 May 2023.] https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html.

31. *Transfer Learning with Convolutional Neural Networks for Diabetic Retinopathy Image Classification. A Review.* **Ibrahem Kandel, Mauro Castelli.** 6, s.l. : Applied Sciences, 2021, Vol. 10. 1006-2021.

32. *Deep Residual Learning for Image Recognition.* **Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun.** Las Vegas : IEEE Conference on Computer Vision and Pattern Recognition, 2016. 1063-6919.

33. **Stanford Vision Lab.** ImageNet. [Online] Stanford Vision Lab, 11 March 2021. [Cited: 15 May 2023.] https://www.image-net.org/.

34. *Multi-Class Weather Classification Using ResNet-18.* **Qasem Abu Al-Haija, Mahmoud Smadi, Saleh Zein-Sabatto.** Las Vegas : IEEE, 2020. 978-1-7281-7624-6.

35. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* **Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun.** Montreal : MIT Press, 2015. 978-1-7281-7624-6.

36. *An automatic nuclei segmentation method based on deep convolutional neural networks for histopathology images***. Hwejin Jung, Bilal Lodhi, Jaewoo Kang.** 24, s.l. : BMC Biomedical Engineering, 2019, Vol. 1. 2524-4426.

37. **Brownlee, Jason.** Machine Learning Mastery. *How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification.* [Online] Guding Tech Media, 3 January 2020. [Cited: 15 May 2023.] https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/.

38. **Mark Everingham, Luc Van Gool, Christopher Williams, John Winn, Andrew Zisserman.** *The PASCAL Visual Object Classes (VOC) Challenge.* s.l. : International Journal of Computer Vision.

39. **NeuralCeption.** NeuralCeption. *Evaluation Metrics.* [Online] [Cited: 15 May 2023.] https://www.neuralception.com/objectdetection-evaluationmetrics.

40. **Brocher, Jan.** ImageJ. *BioVoxxel Toolbox.* [Online] 6 January 2022. [Cited: 15 May 2023.] https://imagej.net/plugins/biovoxxel-toolbox#extended-particle-analyzer.

41. *Sphericity and roundness computation for particles using the extreme vertices model.* **Irving Cruz-Matias, Dolors Ayala, Daniel Hiller, Sebastian Gutsch, Margit Zacharias, Sonia Estrade, Francesca Peiro.** s.l. : Journal of Computational Science, 2019, Vol. 30. 1877-7503.

42. *Pore Microstructure and Multifractal Characterization of Lacustrine Oil-Prone Shale Using High-Resolution SEM: A Case Sample from Natural Qingshankou Shale.* **Shansi Tian, Yuanling Guo, Zhentao Dong, Zhaolong Li.** 11, s.l. : Fractal and Fractional, 2022, Vol. 6.

43. **Engineering Statistics Handbook. NIST.** *F-Test for Equality of Two Variances.* [Online] [Cited: 15 May 2023.] https://www.itl.nist.gov/div898/handbook/eda/section3/eda359.htm.

44. **Kassambara, Alboukadel.** Data Novia. *T-TEST ESSENTIALS: DEFINITION, FORMULA AND CALCULATION.* [Online] Data Novia. [Cited: 15 May 2023.] https://www.datanovia.com/en/lessons/t-test-formula/.

45. **Facebook AI Research lab.** PyTorch. [Online] [Cited: 15 May 2023.] https://pytorch.org/.

46. **Contributors, Various.** PyPi. *pycocotools 2.0.6.* [Online] 4 November 2022. [Cited: 15 May 2023.] https://pypi.org/project/pycocotools/.

47. **John McGonagle, George Shalkouski, Christophers Williams.** Brilliant. *Backpropagation.* [Online] [Cited: 15 May 2023.] https://brilliant.org/wiki/backpropagation/.

48. **Brownlee, Jason.** Machine Learning Mastery. *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks.* [Online] 5 July 2019. [Cited: 20 May 2023.] https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/.