Summary

The data used in this work are from researchers using scanning electron microscope images to investigate the properties of oxidation and coating layers on various materials essential to the nuclear industry. Accurately measuring the thickness of these layers is important for material characterization. The current manual measurement process is time-consuming, and automating it can significantly improve efficiency. Many algorithms can assist with automation, varying widely in complexity and input expectations. While many of these algorithms promise powerful and accurate results, real-world scenarios require substantial groundwork before these tools can be effectively deployed and automation established. This work discusses the challenges and solutions encountered, beginning with the collection and assessment of all available data. It then explores experiments with both conventional computer vision and machine learning algorithms, evaluating their performance. The overall parameters of different solutions are examined, and the most suitable approach is selected. This approach is then seamlessly integrated into the researchers' existing workflow.

Souhrn

Data použitá v této práci pocházejí od výzkumníků, kteří využívají snímky ze skenovacího elektronového mikroskopu ke zkoumání vlastností oxidačních a povlakových vrstev na různých materiálech důležitých pro jaderný průmysl. Přesné měření tloušťky těchto vrstev je klíčové pro charakterizaci materiálů. Současný manuální proces měření je časově náročný a jeho automatizace může výrazně zvýšit efektivitu. Existuje mnoho algoritmů, které mohou pomoci s automatizací, přičemž se liší svojí složitostí a požadavky na vstupní data. Ačkoli mnoho z nich nabízí přesné a výkonné výsledky, v realitě si jejich nasazení vyžaduje rozsáhlou přípravu, aby bylo možné efektivně dosáhnout automatizace. Tato práce se zabývá výzvami a řešeními, které byly při vývoji automatizovaného řešení identifikovány. Začíná sběrem a analýzou dostupných dat, následně se věnuje experimentům s konvenčními algoritmy počítačového vidění a strojového učení, přičemž hodnotí jejich výkonnost. Jsou zhodnoceny parametry různých řešení a vybraný nejvhodnější přístup, který je následně integrován do stávajícího pracovního postupu.

Acknowledgements

I would like to express my gratitude to my advisor, **Ing. Petr Čech, Ph.D.**, and my consultant, **Mgr. Jaroslav Knotek**, for their guidance and support throughout this work.

The input data used was created with state support from the Technology Agency of the Czech Republic under the THÉTA Program as part of project No. TK04030082 and utilizing the CICRR infrastructure, which is financially supported by the Ministry of Education, Youth, and Sports – project LM2023041.

Contents

1 Introduction **1**

2 Practical Part **4**

2.1 Data 4

2.1.1 Manual Data Processing 5

2.2 K-means Algorithm 6

2.3 Segment Anything Model 7

2.4 Convolutional Neural Networks 8

2.4.1 Label Reconstruction Strategies 9

2.4.2 Fiji Adjustments 9

2.4.3 Types of Labels 10

2.4.4 Dataset Splitting Strategy 12

2.4.5 U-Net Architecture 13

2.4.6 Data Preprocessing – Augmentation 15

2.4.7 Technical Details 16

2.4.8 Learning Rate and Schedulers 18

2.4.9 Loss Function 19

2.4.10 Training 20

2.4.11 Optimization Strategy 21

2.4.12 Optimization results 21

2.4.13 Evaluation on Test Data 25

2.5 Integration of the Model 29

2.6 Results - Time Experiment 30

2.6.1 Time per one set 31

2.7 Results - Precision 32

3 Conclusion **34**

4 Appendix **36**

5 Disclaimer **37**

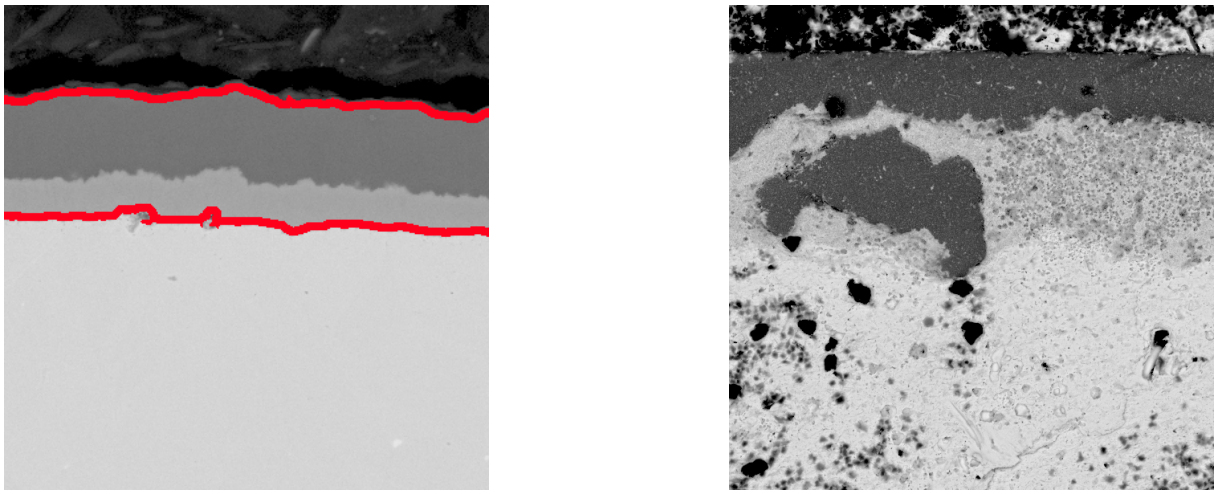
6 List of Abbreviations **42**

1. Introduction

The Research Institute Řež conducts numerous projects within the nuclear industry, many of which involve electron microscopy and the analysis of materials exposed to high temperatures and aggressive environments, such as those found in nuclear reactors. A significant portion of these projects requires extensive manual image analysis, making automation a key challenge. Currently, image analysis is performed using various processing tools, including Fiji [13] by ImageJ. This thesis aims to automate one of these manual processes.

The images analyzed in this thesis are part of a project that includes two main image categories: material samples with protective coating layers and samples with extensive oxidation and no coating. This work focuses exclusively on the coated samples, which typically exhibit more uniform and structured features compared to the irregular and heterogeneous appearance of the oxidized samples. Examples of both image types are shown in Figure 1.1. The coated images originate from a study investigating the thickness and integrity of cladding layers applied to various materials, both before and after exposure to high temperatures and aggressive environments, with the aim of evaluating their protective performance. This thesis focuses on the automated measurement of the coating layers. Although the current work is limited to coated samples, the methods developed here may later be adapted for use with the more complex oxidized samples.

To address the segmentation task, three computer vision approaches are explored, ranging from traditional clustering methods to advanced deep learning models. The first method investigated is K-means clustering, an unsupervised algorithm that does not require labeled data. Although K-means offers a lightweight and accessible approach, its performance is constrained by the complex and non-homogeneous nature of the coating layers. Variability in layer structure, color, and visual features requires image-specific parameter tuning, which limits its general usability. A more detailed evaluation of this method is presented in Section 2.2.



(a) Cropped image showing a protective coating layer highlighted in red.

(b) Cropped image of a sample exhibiting oxidation without coating.

Figure 1.1: Examples of coating and oxidation layers in material samples.

The second method involves the Segment Anything Model (SAM) [7], a state-of-the-art segmentation approach based on transformers. SAM is designed for general-purpose segmentation and demonstrates considerable potential due to its flexibility and robust architecture. However, it requires significantly higher computational resources and, in practice, was found to lack the specificity and precision needed for accurate segmentation of the coating layer. These limitations are further analyzed in Section 2.3.

The third approach utilizes Convolutional Neural Networks (CNNs), which demand labeled training data but are capable of achieving high segmentation accuracy once trained. In this work, three types of labels were developed for training and evaluation. Initially, no ground truth annotations were available. To address this, the Fiji image processing software, commonly used by scientists for manual analysis, was modified to support manual annotation while capturing precise measurement data. This enhancement enabled the creation of the first type of labels: polygon-based annotations.

During the manual annotation process, the results from a previous experiment using K-means clustering were collected to generate a second dataset. However, K-means struggled to reliably distinguish between coating and oxidation, especially in more complex cases. To address this, a third label type, called manually refined labels, was introduced. These labels were manually adjusted versions of the K-means labels, corrected and improved using image editing tools by researchers. This final set of labels represents the most

accurate annotations and serves as the ground truth for evaluating the CNN models. A more detailed discussion on the CNN approach and label development is provided in Section 2.4.

Following the creation of the label sets, a CNN model was developed and trained using the manually refined label annotations. The model underwent hyperparameter tuning to optimize its performance. For comparison, additional models were trained on the first and second label types to estimate their expected performance in future applications.

The final model was evaluated not only on a dedicated test dataset but also in a practical setting by researchers. This evaluation considered both segmentation accuracy and the time required for image analysis, as discussed in Sections 2.6 and 2.7. Given the limited size of the initial dataset, the long-term goal is to establish a feedback loop: by automating the labeling process, more samples can be efficiently annotated. This expanded dataset would, in turn, allow the model to further improve its performance, continuously enhancing both the speed and accuracy of the image analysis workflow.

This thesis is structured as follows: Sections 2.2 and 2.3 discuss the two image segmentation methods and why they could not be implemented. Section 2.4 discusses label creation and model training based on the CNN approach. Section 2.5 describes the integration of the trained model into the existing labeling software. The results, including analysis of segmentation accuracy and time efficiency, are presented in Sections 2.6 and 2.7. Finally, Section 3 concludes the thesis with a summary and a discussion of potential future developments.

2. Practical Part

2.1 Data

The dataset consists of images captured using a scanning electron microscope (SEM). As said before these images are part of a study focused on the thickness of coating cladding layers applied to various materials. In Figure 2.1, the coating layers are shown in cropped SEM images. The right side of each image highlights the coating layer with a red border, while the left side shows the original, unmarked region for comparison.

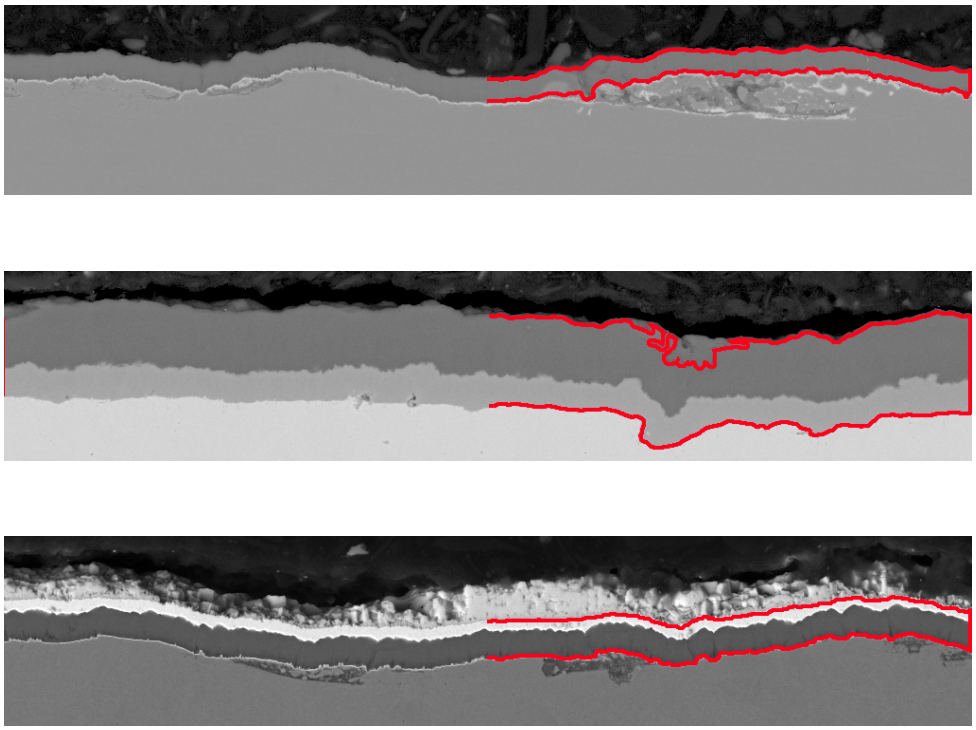


Figure 2.1: Cropped SEM images with partially highlighted coating layers with red border.

The material beneath the coating in the samples varies, and the applied coating layers differ in both composition and physical characteristics. Variations are often observed at the interface between the coating and the underlying material, as well as along the top surface, where signs of oxidation may appear. Additionally, surface damage, such as

fractures in the coating layer, is commonly present. These factors complicate the task of clearly identifying where the coating begins and ends, as the visual boundaries are not always well-defined. As a result, manual labeling becomes a time-consuming process.

2.1.1 Manual Data Processing

The images are initially obtained in *.tif format after the measurements with the SEM. Then they are processed using Fiji, an open-source image processing software that is a distribution of ImageJ [13]. A predefined template consisting of 20 lines is used for measurements, each corresponding to a specific measurement region. The first ten expert-made measurements, indexed 1 to 10, are employed to measure the thickness of the coating layer, while the second set of ten lines, located beneath the first, is used to measure oxidation, which is not that common for images with coating layer. The lines are evenly distributed across the image's width and share a constant x-coordinate. The choice of ten lines was a compromise between accuracy and efficiency, while using more lines could improve precision, ten provided a good balance between informative value and the time required for manual adjustment.

Subsequent to the initial measurements, each of the 20 lines is manually adjusted to align with either the oxidation or the coating layer. After all adjustments are completed, the length of each line is exported to Excel for statistical analysis. If a layer is absent at the location of a predefined line, the line is left in its position but is later marked as 0.0 in the corresponding Excel spreadsheet.

The images are grouped into sets of approximately 30, with each set containing images of the same material. Different parts of the same material are captured in each set, leading to high similarity among the images in each set. The initial image of each set typically takes longer to process since the line adjustments from the previous image are reused for the other images in the same set. As the remaining images within a set are more similar, they require less time for adjustment. The primary challenges arise in images with significant oxidation, which may destroy the coating layer.

Figure 2.2 illustrates the measurements performed in Fiji. Lines 1 to 10, used for mea-

suring the coating layer, are shown. In this example, no oxidation layer is present.

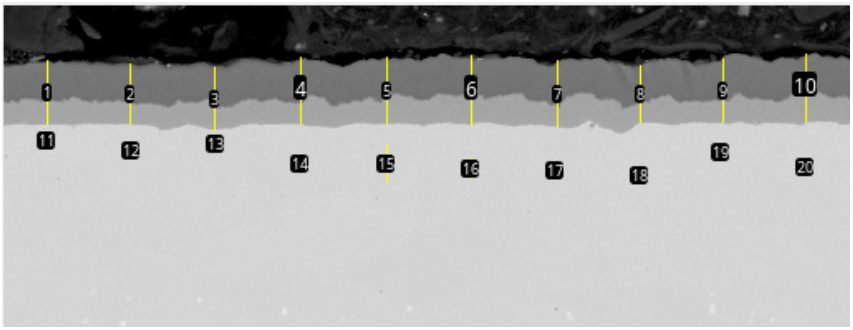


Figure 2.2: Measurement in Fiji software. Only the region of interest is shown; the bottom part of the image is omitted for clarity.

Although oxidation and coating layers are both critical for material characterization, this thesis focuses specifically on automating the measurement of coating layer thickness in SEM images. Coating layers prevent underlying materials from oxidizing, making oxidation layers less common in samples that feature coatings.

2.2 K-means Algorithm

The first method considered for automating the measurement process is the K-means algorithm. K-means [8] is a clustering technique used in unsupervised machine learning. It works by minimizing a function that measures the quality of clusters. The algorithm starts by randomly selecting k points from the dataset as cluster centers. Each data point is then assigned to the nearest center based on distance. The centers are updated by calculating the geometric mean of the points in each cluster, and this process repeats until convergence.

K-means is useful for separating image regions based on color similarity, which matches the goal of image segmentation in this work. One of its main advantages is that it does not require pre-existing labels, which makes it a good starting point when labels are unavailable.

However, the algorithm has several limitations. Since it relies mainly on color, it struggles with images where regions have similar colors. For example, in the third image in Fig-

ure 2.1, oxidation appears under or over the coating layer, making it difficult to segment. Also, the same K-means settings cannot be used for all coatings, as shown in Figure 2.1; different coatings have different numbers of layers and colors.

The value of k needs to be manually set for each image set, and additional post-processing is often required to isolate the desired cluster. As shown in Figure 2.3, K-means often fails to separate oxidation from the coating layer due to their similar pixel values, and it frequently clusters them together.

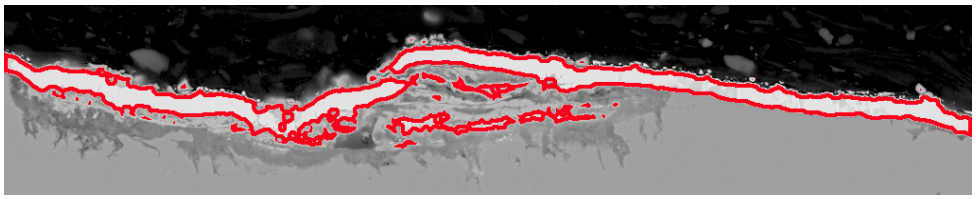


Figure 2.3: Highlighted coating layers predicted using K-means.

2.3 Segment Anything Model

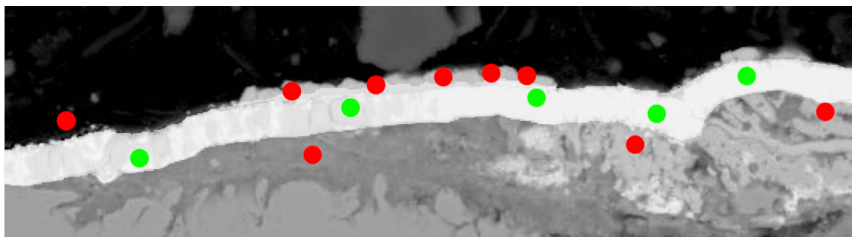
The Segment Anything Model (SAM) [7] is a state-of-the-art tool for image segmentation. It is designed as a general-purpose segmentation model that works across a wide range of images. However, the samples used in this work are very specific, and pixel-level precision is crucial.

As shown in Figure 2.4, even when using five positive and nine negative points to guide the segmentation, SAM struggles to distinguish between the coating and the upper oxidation layer. This is not ideal, as researchers would need to manually place many guidance points for each measurement and still often need to refine the mask using a brush tool.

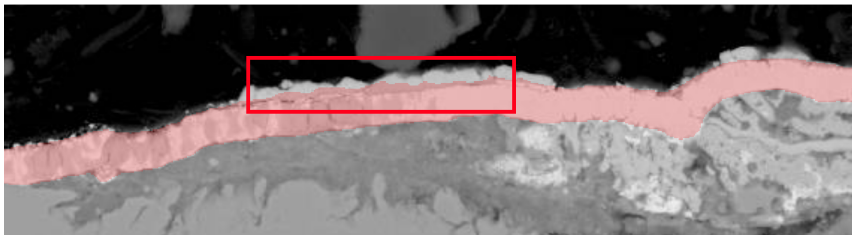
Another key limitation of SAM is its high hardware requirements. Running it on a standard office computer, essential for automating the measurement process, is not feasible. Unlike smaller models that can operate locally, SAM demands either a dedicated GPU or a remote server setup, neither of which is a viable option in this case.

Additionally, SAM requires user input through interactive prompts, creating a significant technical hurdle, particularly since the model needs to be integrated into Fiji. Adapting

it within Fiji would require considerable adjustments to external software, introducing unnecessary complexity to the workflow.



(a) User-provided positive (green) and negative (red) points used to guide the SAM segmentation.



(b) Output mask generated by SAM. The red rectangle highlights the difficulty distinguishing between the oxidation and coating layers.

Figure 2.4: Top: User-provided points (input). Bottom: SAM-generated output mask.

2.4 Convolutional Neural Networks

To address the limitations of K-means and SAM, Convolutional Neural Networks (CNNs) [10] were considered due to their strong ability to capture spatial relationships between pixels, which is critical for distinguishing between the coating and oxidation layers in complex samples.

This makes them a good candidate for solving segmentation problems in images where traditional methods fail. An important advantage of CNNs is that they can be trained specifically on the dataset used in this project, with the ability to adapt to the particular features of the coating layers. They also require less computational power than SAM, making them more suitable.

However, CNNs require ground truth labels for training, which were not initially available.

2.4.1 Label Reconstruction Strategies

As mentioned in Section 2.1.1, the only available data were microscope images and an Excel spreadsheet with the lengths of the expert-made measurements. Unfortunately, this was not enough to fully reconstruct the original measurements. Since labeling all data manually would have been too time-consuming, three alternative strategies for dataset creation were explored.

First, a few minor adjustments were made to the labeling tool Fiji to allow researchers to create data that could later be converted into 2D polygon labels. While researchers were labeling the initial set of samples to obtain a reasonable dataset, two additional approaches were explored in parallel.

The first approach was to reuse the K-means segmentation results from the previous experiment. The second approach involved manually refining the K-means labels in collaboration with researchers to create high-quality ground truth labels. These refined labels are especially important, as they represent the most accurate annotations available and are used for evaluation on the test set.

2.4.2 Fiji Adjustments

The whole manual labeling is done in Fiji. To make the data collection possible, a few important adjustments were made. The adjustments involved the customization of the 'StartUpMacro' in Fiji, which runs automatically each time the program is launched. Three new buttons were incorporated into the interface: one for saving measurements, another for adjusting the scale, and a third for displaying the default lines, which represent the default expert-made measurements.

The `Save Measurement` button was implemented to store data from the 20 lines required to create the dataset. This feature ensures that the required measurements are automatically saved.

The `Adjust Scale` button was added to streamline user interaction with Fiji, addressing

the need to manually adjust the scale at the start of each session. Since the scale, visible at the bottom of the image, remains constant across all images, this functionality simplifies the process.

The `Set Default Lines` button was introduced to allow the researcher to quickly access the prepared default 20 lines for the new set.

Furthermore, the extraction of the final measured lengths into an Excel file was simplified.

2.4.3 Types of Labels

The following three sections present strategies of label creation mentioned in 2.4.1 in more detail.

Polygon labels

This set of labels was generated using measured data from expert-made measurements. The upper ten points of each line were connected to form a boundary curve, and the same process was applied to the lower points. The space between these two curves was then filled to generate the label. However, due to missing information at the edges of the image, cropping of both the images and the labels was needed. Figures 2.5a illustrate that the labels do not extend to the edges of the images. While these labels are not highly precise, their creation process is highly efficient, focusing on the 20 key points defined by the researcher. Due to their efficiency, they are considered valuable for testing in the training process, as they provide a balance between accuracy and computational effort.

Linear interpolation was employed to connect the boundary points. The upper and lower boundary points, stored in arrays, created a boundary curve by connecting each pair of neighboring expert-made measurement points with a straight line. The region between the interpolated upper and lower curves was then filled, resulting in a binary label.

Linear interpolation involves estimating values between two known points. It assumes a linear relationship and is defined by the following equation:

$$y(x) = y_1 + \frac{(x - x_1)}{(x_2 - x_1)}(y_2 - y_1) \tag{2.1}$$

Here, (x_1, y_1) and (x_2, y_2) are two neighboring known points, and $y(x)$ represents the interpolated value at a given x such that $x_1 \leq x \leq x_2$.

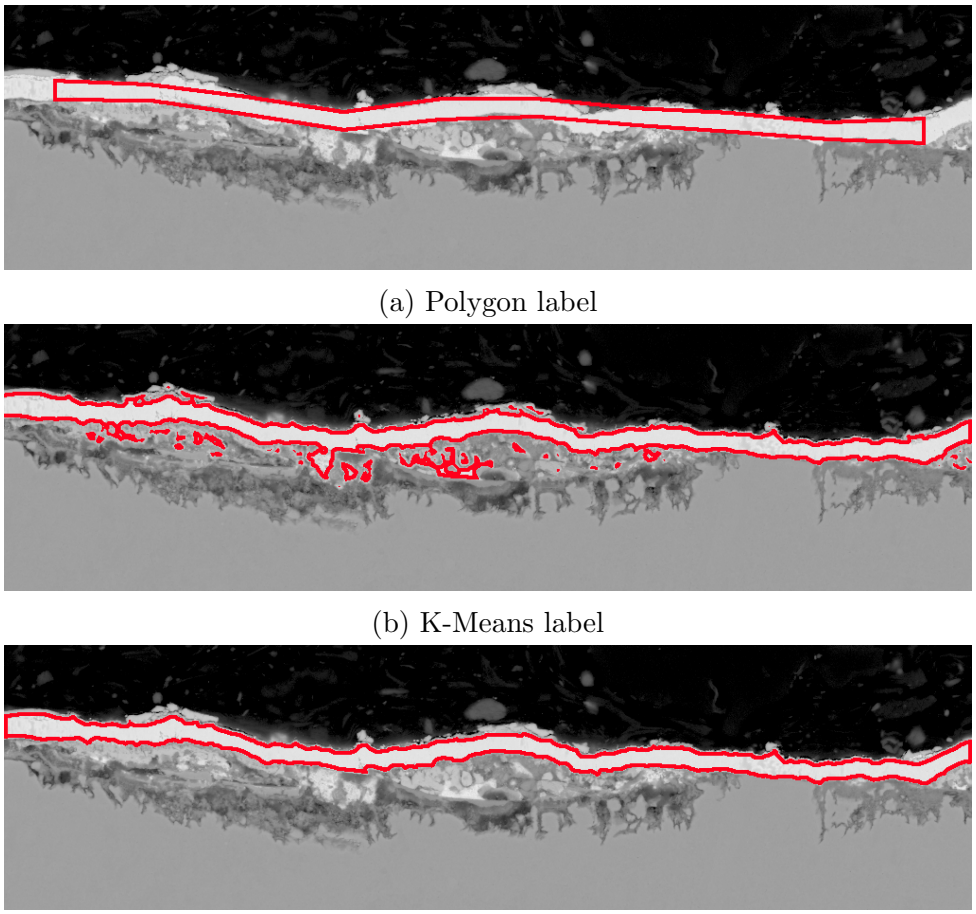
This approach enabled the automated generation of labels from the measured lines, ensuring that the labels were consistent with the scientific data.

K-means labels

Despite these limitations mentioned in Section 2.3, the labels generated by the K-means algorithm were still used to create a second dataset. This approach was faster than expert-made manual measurements and, in many cases, provided a more precise boundary for the coating layer, something the polygon labels couldn't offer. However, one downside was that the oxidation areas were often grouped with the coating layer due to their similar pixel values. This can introduce noise into the data, potentially affecting the model's ability to generalize.

Manually Refined Labels

To improve the accuracy of the K-means labels, the entire dataset was reviewed in collaboration with the researcher, and the labels were manually refined using image editing software. Additionally, the manual revision process was much less time-consuming than the complete manual annotation of every image, due to the predefined labels from K-means. The final version of these labels was then used as a test dataset for further automation. This label is shown in Figure 2.5c.



(a) Polygon label

(b) K-Means label

(c) Manually refined label

Figure 2.5: Examples of polygon, K-Means, and manually refined labels, shown as red-highlighted borders over the original SEM image.

2.4.4 Dataset Splitting Strategy

The full dataset consists of 177 annotated images. Out of these, 130 images were used for model development and split into training and validation subsets using a 75:25 ratio. The remaining 47 images were reserved as a separate test set. This test set was selected manually prior to any model training, with the goal of ensuring it represented a diverse and challenging subset of the data. Random splitting was avoided in this case due to the limited dataset size and the risk of ending up with a test set that lacked difficult examples. By deliberately including images with visible oxidation and other complex features, the test set was designed to better reflect real-world variability and to more rigorously evaluate the model's generalization capabilities.

2.4.5 U-Net Architecture

After collecting the entire dataset, a suitable convolutional neural network (CNN) architecture for segmentation had to be selected. Although originally developed for biomedical image segmentation, U-Net [11] remains a widely used and practical starting point for a variety of segmentation tasks. While the original version is now considered somewhat outdated, modern U-Net implementations often include enhancements such as pretrained encoders and others to improve performance. In this work, a U-Net-based model was chosen as a strong and accessible baseline. Despite the availability of more advanced architectures, it ultimately proved sufficient for the problem at hand.

A 2023 survey [14] highlights U-Net as an effective model for microscopic image segmentation. This success is attributed to its ability to perform well with limited training data and its relatively quick training process. The survey also emphasizes the use of the original, unmodified U-Net, which has been widely applied in fields such as cytology and geology [3] (e.g., using scanning electron microscopy). Given that the images in this dataset share similar characteristics, the U-Net architecture is considered suitable for this task.

Unlike traditional CNNs designed for classification, U-Net performs pixel-wise classification, making it particularly well suited for tasks requiring precise spatial localization, such as segmenting structures in microscopy images. The architecture consists of two main components:

- **Contracting path (Encoder)**: The left side of the U-Net consists of a series of convolutional, activation, and pooling layers that progressively reduce the spatial dimensions of the input while capturing contextual information. In this work, the encoder is implemented using a pretrained ResNet34 backbone, as provided by the Segmentation Models library [9]. This allows the model to benefit from features learned on large-scale datasets.
- **Expanding path (Decoder)**: The right side of the U-Net gradually upsamples the feature maps back to the original image size. This step recovers spatial resolution lost during downsampling and ensures precise pixel-wise segmentation.

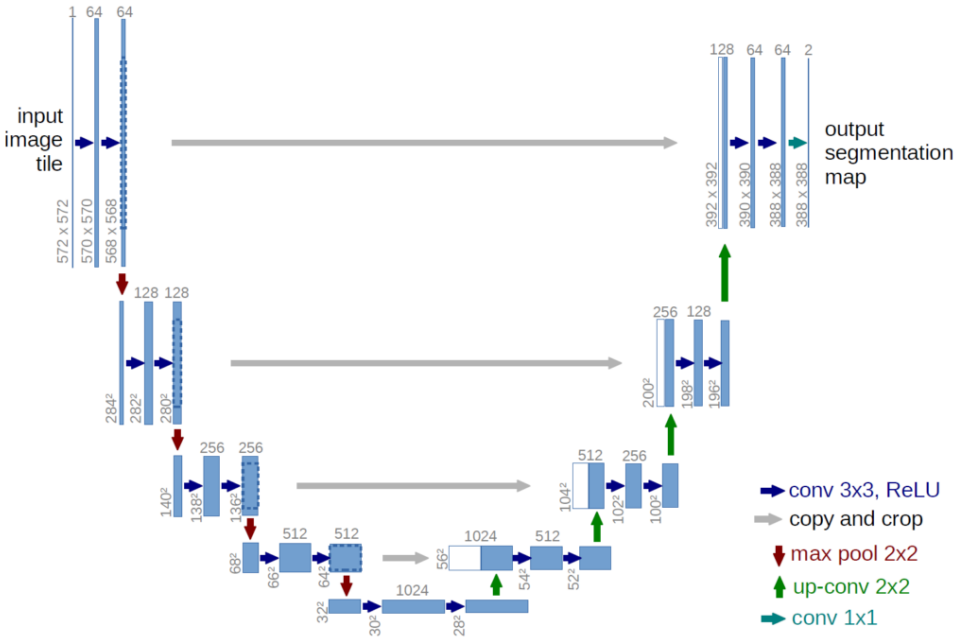


Figure 2.6: U-Net architecture [12]

The input to a U-Net model is typically an image of size $H \times W \times C$, where H is the height, W is the width, and C is the number of input channels. In Figure 2.6, the input consists of grayscale images, where $C = 1$. For RGB images, $C = 3$. The output is an image of size $H \times W \times N$, where N is the number of segmentation classes. In this case, there are two classes (binary segmentation), so $N = 2$.

Upsampling with Transpose Convolutions

The expanding path uses transpose convolutions (also called deconvolutions). Convolution reduces the image to a feature map, while transpose convolution upsamples the feature map back into an image.

Skip Connections

During the downsampling process, the network learns various features. However, as the feature maps shrink, details are lost. Skip connections allow the network to preserve these details by carrying them over from the earlier layers in the contracting path to the corresponding layers in the expanding path. This process helps retain important spatial

information, improving segmentation accuracy. These connections are represented by the horizontal arrows in Figure 2.6.

Key Steps in U-Net Processing

1. In the contracting path, feature maps are progressively downsampled until the lowest resolution is reached.
2. Before each downsampling step, feature maps are saved.
3. During the upsampling process, saved feature maps are combined with the upsampled maps to recover spatial details [12].

2.4.6 Data Preprocessing – Augmentation

A said in Section 2.4.4 that the dataset used for training and validating consists of just 130 images. Because of the limited data, augmentation is applied to the training set to better showcase potential deviations in visual representation. Thus, the model can learn to generalize better to variations it might encounter during future predictions.

The first step involves either cropping or resizing the image to a size of $(n \times 1.5) \times (n \times 1.5)$, with equal probability, where n is the patch size and a model hyperparameter. Cropping keeps the original resolution, while resizing includes more context to reduce overfitting. Cropping is done using the *CropNonEmptyMaskIfExists* function [2], ensuring the region contains part of the coating. After additional augmentations, the image is cropped to a final size of $n \times n$ to match the model input.

Additional augmentations include:

- **Brightness and contrast adjustment:**This simulates lighting variations.
- **Sharpening:** This enhances the details in the image.
- **Horizontal flipping:** The image is mirrored along the horizontal axis to reduce positional bias.

- **Elastic transformation:** A non-linear deformation is applied to simulate distortions and improve generalization.
- **Rotation:** The image is rotated randomly within the range of -90° to 90° to simulate different orientations.

The effects of these augmentations are illustrated in Figure 2.7, where different transformations are applied to a sample image.

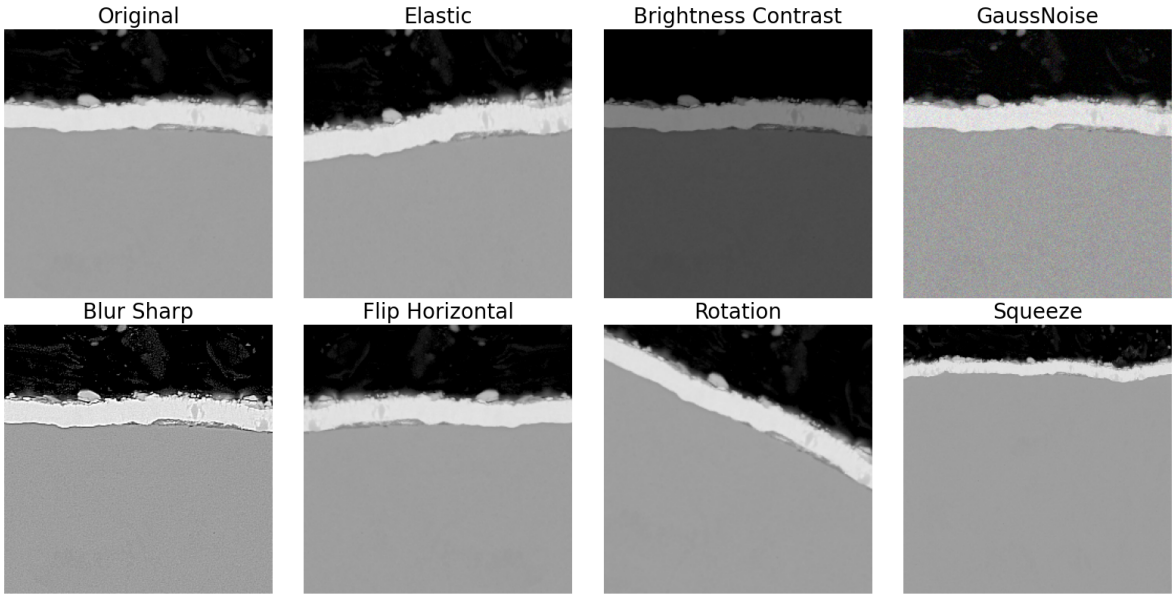


Figure 2.7: Illustration of various augmentation techniques applied to an image from the dataset.

2.4.7 Technical Details

The `segmentation_models` library [5] was used for the U-Net implementation. Documentation for this implementation can be found here [9]. After initializing the training functions and other helper functions, hyperparameter tuning can begin. The main hyperparameters include depth, patch_size, filters, and learning rate, all of which will be tested.

Depth

The depth of a U-Net is determined by the number of encoder and decoder blocks. This influences the model's ability to capture features within the input image. For instance, in Figure 2.6, the depth is 5.

A deeper network (higher depth) can handle complex data with high variability, but may suffer from excessive downsampling and overfitting. In contrast, a shallower network may struggle to extract relevant features and differentiate between noise and useful information.

Patch Size

Patch size plays a crucial role in how well the model learns different features from the image.

Smaller patches allow the model to focus on fine details but might miss the broader context, making it harder to understand relationships between different parts of the image.

Larger patches capture more context and larger structures, but may lose fine details.

To ensure that the region of interest, specifically the coating layer, is always included, the images are cropped in a way that retains this key feature.

Filters

The filters parameter determines the number of filters used in each encoder and decoder layer, which affects the number of feature maps generated by the model. In the code, the number of filters in the decoder is calculated as follows:

```
decoder_channels = [filters * 2**i for i in range(depth, 0, -1)]
```

This calculation, inspired by the original U-Net architecture, doubles the number of filters at each downsampling step [1]. For example, with `filters = 8` and `depth = 3`, the

number of filters would be:

Encoder: 8, 16, 32

Decoder: 32, 16, 8

Using more filters allows the model to learn more complex patterns. However, it can also increase the risk of overfitting when there is insufficient training data. On the other hand, using fewer filters might prevent the model from capturing important features, reducing its performance.

Hyperparameter values

In this project, all combinations of the following hyperparameters were tested:

- **Patch size:** 128, 256
- **Depth:** 3, 4, 5
- **Filters:** 8, 16, 32

2.4.8 Learning Rate and Schedulers

During training, the model's parameters are adjusted to find the optimal solution. The **Adam optimizer** [6] is used for this purpose. Adam (Adaptive Moment Estimation) calculates adaptive learning rates for each parameter.

Although Adam adapts the learning rate internally, the initial learning rate (often referred to as the base or maximum learning rate) still significantly affects performance.

- **Linear Scheduler:** The learning rate starts at the set value and decays linearly. The **end factor** multiplies the learning rate to finish at **end factor * lr**.
- **Warmup Cosine Scheduler:** The learning rate starts at a predefined value and decreases over `T_0` epochs, following a cosine annealing schedule. After reaching the minimum, the learning rate resets and begins from the previous value, restarting the cycle.

The following schedulers and values were tested:

- **LR values**: 0.01, 0.001, 0.0001
- **Linear scheduler**: With end factors of 0.01 and 0.001
- **Warmup Cosine scheduler**: With `T_0`= 25 and `T_0`= 50

Figure 2.8 shows the learning rate schedulers. On the left, the linear scheduler spans 200 epochs, starting at 0.01 and ending at 0.001. On the right, the warmup cosine scheduler starts at 0.01 and ends at 0.00, with a warmup phase during the first 50 epochs.

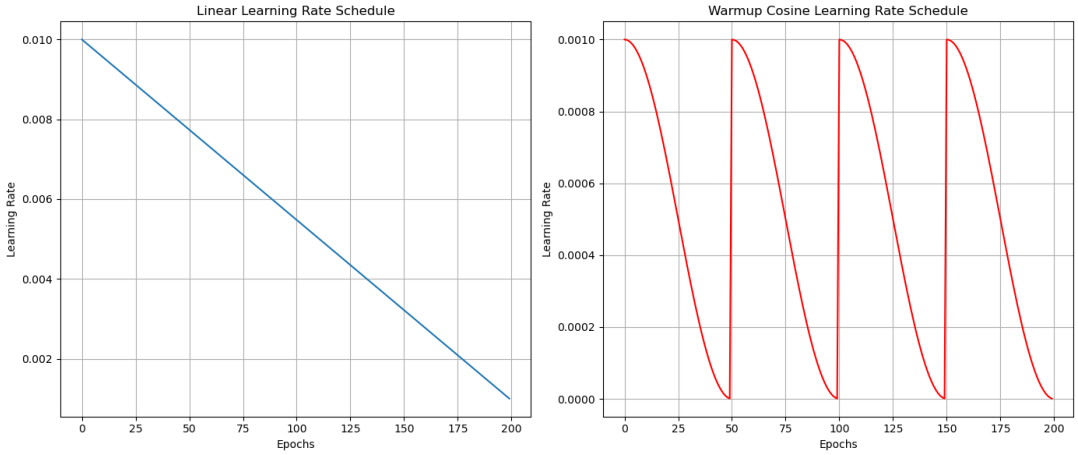


Figure 2.8: Learning Rate Schedulers (LRS)

2.4.9 Loss Function

The loss function $L(Y, \hat{Y})$ is used to evaluate prediction errors. Since the model's performance on test data is assessed using the Intersection over Union (IoU) metric, IoU is also used as the loss function during training and validation.

The IoU is defined as:

$$\text{IoU} = \frac{\text{intersection} + \text{smooth}}{\text{union} + \text{smooth}} \tag{2.2}$$

Where:

- The intersection is the number of pixels correctly predicted as the foreground (True Positives).
- The union consists of all pixels predicted or labeled as foreground. It equals the sum of True Positives (TP), False Positives (FP), and False Negatives (FN).

Thus, the IoU formula becomes:

$$\text{IoU} = \frac{\text{TP} + \text{smooth}}{\text{TP} + \text{FP} + \text{FN} + \text{smooth}} \tag{2.3}$$

The smoothing constant (smooth = 1) prevents division by zero. The loss function returns $1 - \text{IoU}$, ensuring that lower values indicate better model performance.

2.4.10 Training

After defining the key components, model training begins. The dataset is divided into 98 training images and 32 validation images. The training, validation, and testing labels come from the manually refined labels dataset, with the validation set remaining fixed as a single batch. During training, only the training data is shuffled each epoch to reduce the risk of overfitting. The DataLoader then creates batches of 32 augmented images from the shuffled data. Training proceeds for 200 epochs, with each epoch consisting of both training and validation phases.

Training Phase

In each epoch, the training dataset is shuffled, and three batches of 32 images are formed. The model predicts the binary labels for each batch and computes the training loss. Back-propagation adjusts the model's parameters using the optimizer based on the computed loss.

Validation Phase

At the end of each epoch, the validation set is evaluated using the loss function. The model's parameters are not updated during this phase. If the validation loss is lower than the previously recorded minimum, the model and its parameters are saved, and the minimum validation loss is updated.

2.4.11 Optimization Strategy

1. Three optimization runs were performed to determine the optimal learning rate (LR) and learning rate scheduler. Details about the hyperparameters tested are provided in section 2.4.8.

2. The main optimization focused on the three key hyperparameters, using the best LR and LR scheduler from the initial optimization. The specific hyperparameters optimized are discussed in section 2.4.7.

3. After identifying the best hyperparameters, two additional runs were conducted. One used polygon labels, while the other used K-means labels.

2.4.12 Optimization results

The optimization of the learning rate and learning rate schedulers is discussed first. The schedulers were tested with the hyperparameters **depth = 3**, **patch size = 128**, and **filters = 8**. These values were chosen as they offered the best computational efficiency among the tested ones.

Table 2.1 summarizes the best results for different learning rate schedulers. The linear scheduler achieved the lowest objective function value of 0.06429 with a learning rate of 0.001 and an end factor of 0.01.

Table 2.1: Best results for different Learning Rate Schedulers.

Schedule	Learning Rate (lr)	End Factor / T_0	Val Loss
Warmup Cosine	0.001	50	0.06527
Linear	0.001	0.01	0.06429
None	0.001	—	0.06474

It is evident that a learning rate of 0.001 consistently outperformed the others across all three optimization experiments. In contrast, the choice of scheduler did not lead to significant differences in the validation loss values, indicating that it had only a minor influence on the training process.

Figure 2.9 illustrates the validation loss progression over 200 epochs for three different learning rates, without the use of any learning rate scheduler.

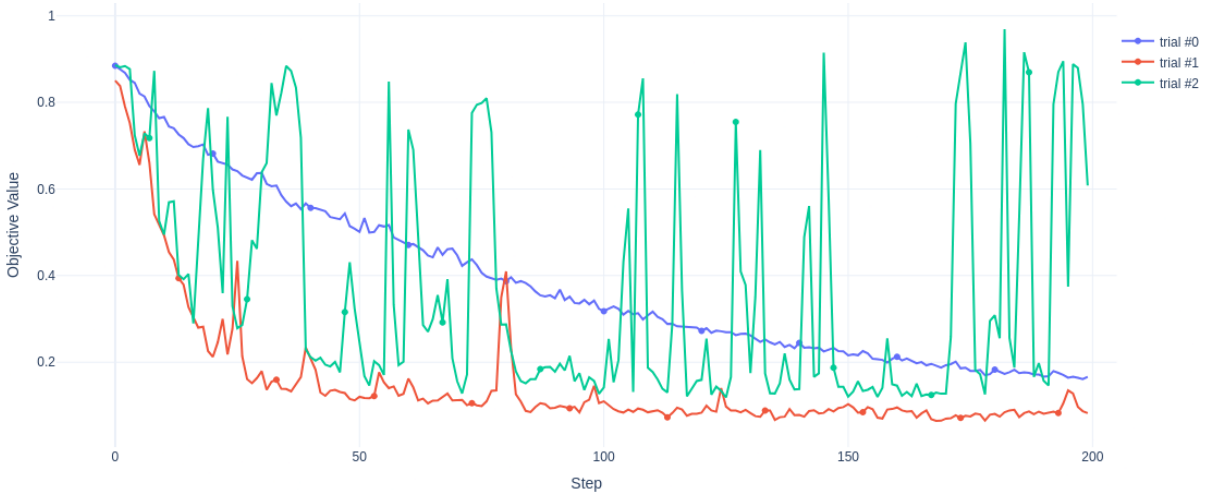


Figure 2.9: Validation loss values for three different learning rates over 200 epochs.

Trial 0, corresponding to a learning rate of 0.0001, shows a very stable but very slow decay in validation loss, suggesting that the learning rate is too low to enable efficient convergence. Trial 1, with a learning rate of 0.001, demonstrates a faster decrease in loss and achieves relatively low values, however, with some minor fluctuations. This indicates that it provides a good balance between convergence speed and stability. In contrast, trial 2, using a learning rate of 0.01, is highly unstable with large oscillations in the validation loss. This instability likely results from the learning rate being too high, causing the

model to overshoot optimal points and fail to converge effectively. These results clearly highlight the critical role of learning rate selection in the model's learning ability.

Next, the main optimization was conducted using the linear scheduler and the best values from the previous step. The hyperparameters discussed in section 2.4.7 were optimized, resulting in 18 combinations ($2 * 3 * 3 = 18$). The complete results are shown in Table 2.2, ordered from lowest to highest validation loss, with the best result appearing at index 8.

Table 2.2: Validation loss and corresponding hyperparameters across trials

Idx	**Val Loss**	**Depth**	**Filters**	**Patch Size**
15	0.0657	5	16	128
4	0.0643	3	8	128
0	0.0643	5	8	128
7	0.0636	4	8	128
2	0.0631	5	32	128
9	0.0601	4	16	128
10	0.0600	3	32	128
5	0.0574	4	32	128
11	0.0573	3	16	128
6	0.0444	5	8	256
12	0.0427	3	8	256
13	0.0424	4	8	256
16	0.0407	5	16	256
3	0.0401	3	16	256
17	0.0401	4	16	256
14	0.0392	5	32	256
8	0.0389	3	32	256
1	**0.0382**	**4**	**32**	**256**

Figure 2.10 presents the validation loss for the best (trial 1) and worst (trial 15) values

over 200 epochs. The plot shows that both losses decrease rapidly during the first 50 epochs, after which they oscillate around the same value. Despite this, trial 1 achieves a lower minimum loss compared to trial 15.

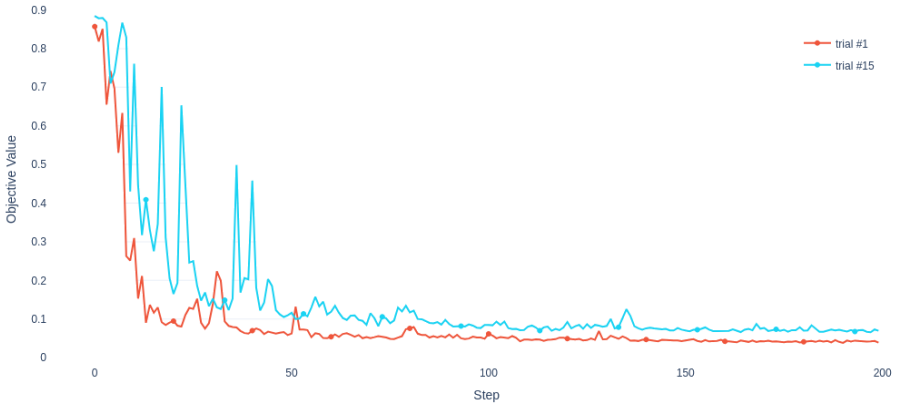


Figure 2.10: Validation loss over 200 epochs

Figure 2.11 shows the importance of each hyperparameter, providing valuable insights into the model training process. As shown in Table 2.2, larger patch sizes and a higher number of start filters tend to achieve better results compared to smaller ones. This explains why their importance is higher than that of the model depth, as they have a greater influence on the final loss value.

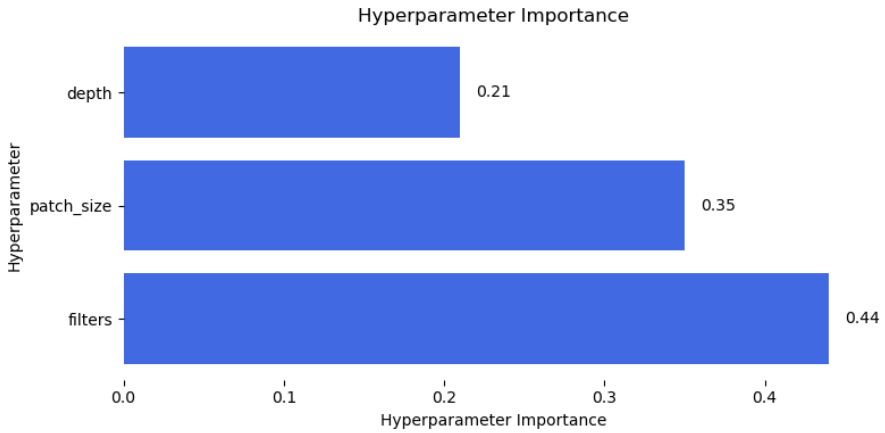


Figure 2.11: Hyperparameter importance

The final two runs were conducted using the best hyperparameters, but with datasets that included labels generated by polygons and K-means clustering. The models with the best validation losses were chosen.

Since both validation sets use labels derived from their respective training sets, the resulting performance metrics may not be fully reliable. A more meaningful assessment can be made by evaluating the model on unseen test images using manually refined labels. Although neither the K-means nor polygon labels are intended for use in this final workflow, this evaluation can provide insight into whether either labeling method is sufficient for future training.

2.4.13 Evaluation on Test Data

Four primary evaluation metrics were used:

- **Mean IoU**: The intersection over union (IoU) is calculated and averaged across all test data.
- **Min IoU**: The minimum IoU value is used to assess the worst-case scenario.
- **Mean Hausdorff Distance**: Measures the average greatest distance across all test images.
- **Max Hausdorff Distance**: Represents the maximum of these distances, giving an indication of the worst mismatch in a picture.

Evaluation metrics

The IoU metric is calculated as shown in Equation 2.3, without the transformation $1-\text{IoU}$, meaning that a higher IoU indicates better results.

The Hausdorff distance [4] (HD) quantifies how far two shapes (or point sets) are from each other. It is defined as the greatest of all the distances from a point in one set to the nearest point in the other set. It captures the largest error between the two sets, highlighting the worst-case mismatch.

Results

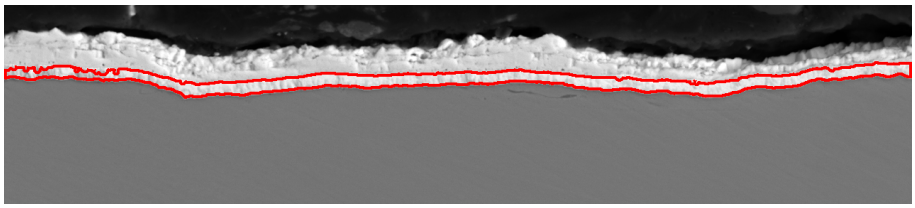
The evaluation results are presented in Table 2.3:

Table 2.3: Evaluation Results on Test Data

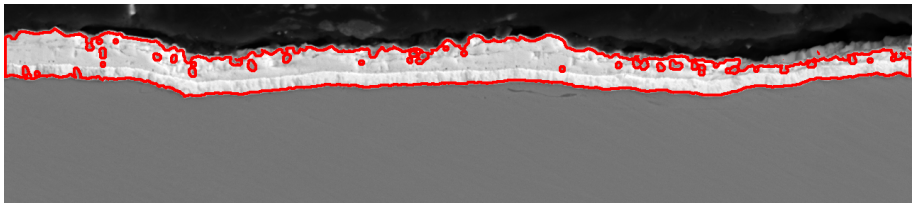
Metric	**Manually Refined**	**K-means**	**Polygon**
Mean IoU	0.95155	0.84833	0.89800
Min IoU	0.64854	0.18043	0.72229
Mean HD	53.83457	75.51385	91.68906
Max HD	284.51977	288.06249	337.45242

The results reveal some key patterns. The model trained with polygon labels performed surprisingly well, with good overall IoU, although the Hausdorff Distance (HD) was relatively poor. On the other hand, the model trained with K-means labels achieved the lowest IoU values. This could be explained by the fact that the K-means labels didn't differentiate well enough between oxidation and coating areas, leading to poorer segmentation. The figure 2.12 highlights the comparison of the three types of predictions made by models trained with different kinds of labels, alongside the original ground truth label, on a particularly complex sample. The best results across all four metrics were achieved with the manually refined labels, as expected.

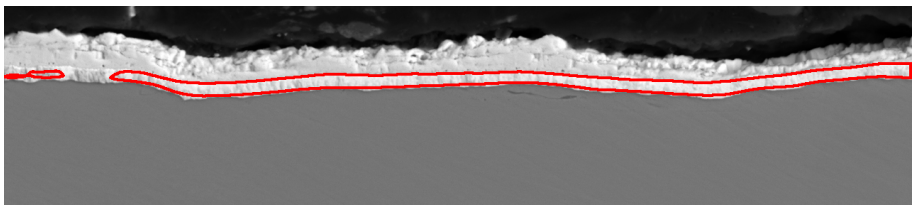
Figure 2.12b illustrates that the model trained with K-means labels struggles to differentiate between the oxidation and coating layers, leading to poor performance. In contrast, Figures 2.12c and 2.12a show that the models trained on polygonal and manually refined labels can accurately predict the coating layers, despite the complexity of the images and the challenge of distinguishing between the layers. However, Figure 2.12c still struggles with finer details, particularly at the boundaries of the layers.



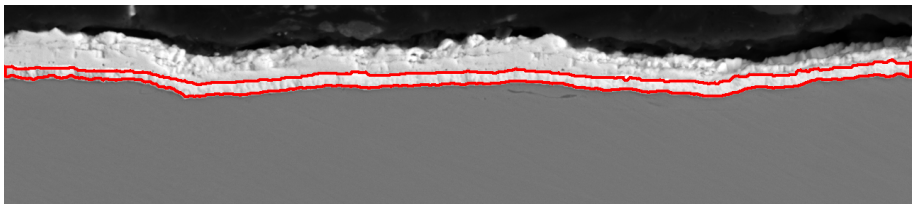
(a) Manually refined label prediction



(b) K-means label prediction



(c) Polygon label prediction



(d) Original label

Figure 2.12: Comparison of predictions made by models trained on different datasets, alongside the original ground truth label on a complex sample.

To highlight the superiority of the U-Net model over the K-means algorithm in segmenting the image, it is important to note that the K-means method struggled to distinguish between the oxidation layer and the coating layer, as shown in Figure 2.5b. In contrast, as shown in Figure 2.13, the model trained on the manually refined labels was able to successfully learn this pattern.

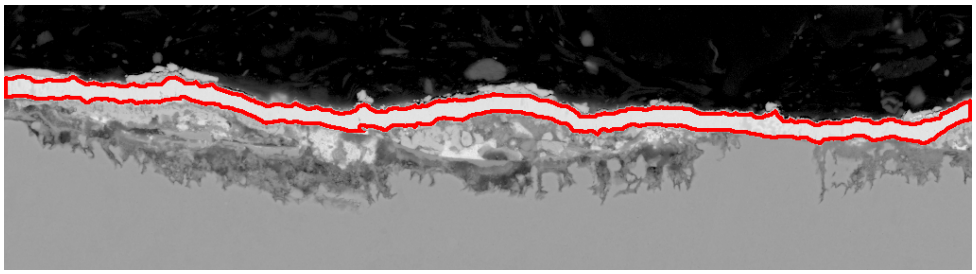


Figure 2.13: Prediction labels shown in red border on the original image with extensive oxidation

An important part of the evaluation is also to inspect the images that caused the minimum IoU and maximum Hausdorff Distance (HD). The lowest IoU occurred on an image with the thinnest coating layer in the entire training or testing dataset, as can be seen in Figure 2.14. This highlights the need for a more robust and heterogeneous dataset, as small datasets, like the one used here, can lead to issues where less frequent or extreme cases, such as images with very thin coatings, cause problems during model evaluation.

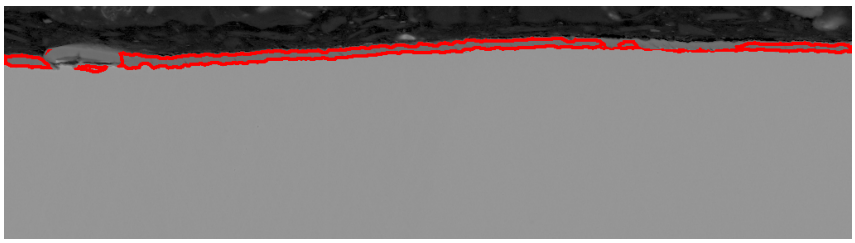


Figure 2.14: Example image with minimum IoU

On the other hand, the lowest HD was observed on an image containing a dark defect located at the bottom, as shown in Figure 2.15. However, this deviation is not particularly concerning, as such anomalies can be effectively addressed with simple postprocessing steps. These types of deviations are relatively easy to detect and remove.

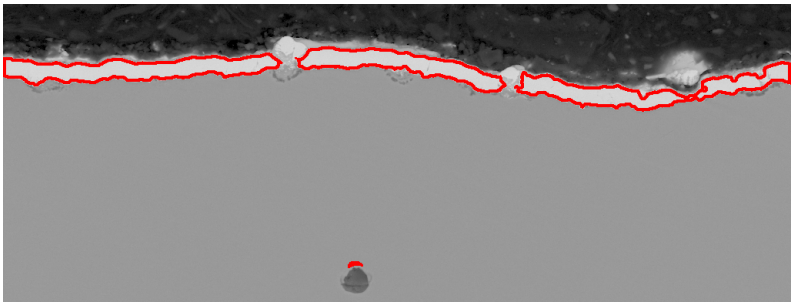


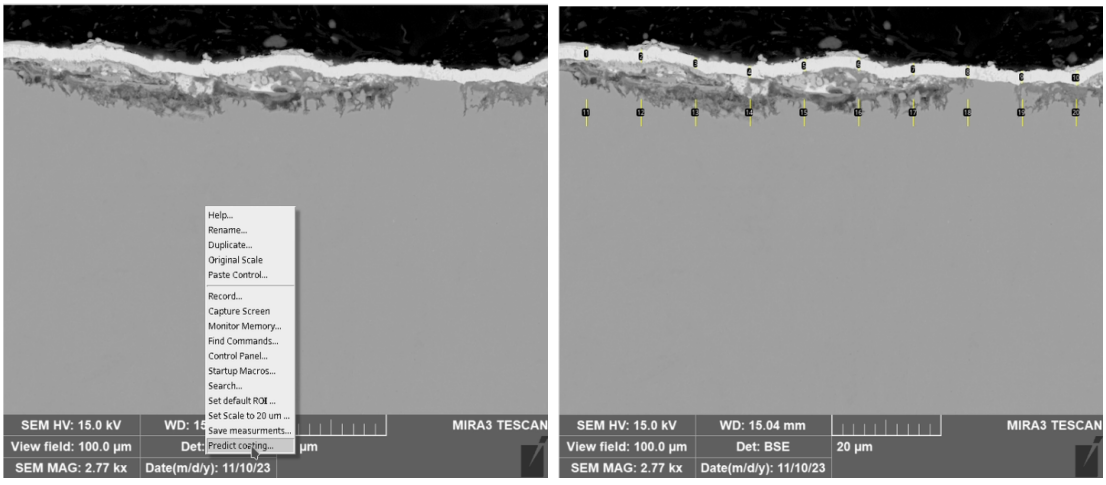
Figure 2.15: Example image with maximum HD

2.5 Integration of the Model

After training the model, it was crucial to integrate it smoothly into the existing workflow to avoid requiring the researcher to learn new software. The model was integrated into the Fiji software, which was already in use, instead of creating a new application.

The first step involved adding a new button to the Fiji interface, as shown in Figure 2.16a. Pressing this button triggers a Python script from the 'StartupMacro', with the path to the image being analyzed passed as an input argument. The script, stored in the Fiji application folder along with the macros and model parameters, loads the image and crops the lower part to remove irrelevant information. It then predicts a binary mask, processes it, and extracts the ten most relevant lines.

By default, another set of ten lines is added beneath the first set. The resulting data, containing the twenty lines, is saved in a zip folder. The path to this zip folder is then returned, and Fiji opens the folder to display the predicted lines to the researcher. If the researcher is dissatisfied with the predictions, they can modify the lines as needed.



(a) Menu with *Predict coating...* button (b) Predicted lines

Figure 2.16: Example workflow in Fiji

2.6 Results - Time Experiment

Following the model integration and the update to the Fiji setup, a time measurement study was conducted to assess time savings. Eight images were selected from four image sets, forming four groups, with two images chosen from each set. The experiment was designed as follows:

The first image in the group is measured manually, while the second is analyzed using the prediction model. In the first image from the first group, only the predefined default lines were used, simulating the first image in each set, where the researcher does not have lines similar to the coating. The first image from the second group used predefined lines from another image in the same set. Since the images in each set are similar, the researcher typically spends less time adjusting these lines. The third group followed the same approach as the second, but with an image where the model did not predict all the lines correctly. The fourth group included a challenging image with complex oxidation, where the model's predictions were less accurate, though the same approach as the second group was used on the first image.

The "Model" column in Table 2.4 indicates whether the model was used for line prediction. The model was not used for the first image in each group, but was used for the second image.

The "Predict" column shows the time from pressing the "Predict coating" button until the predicted lines were displayed, which took approximately 14 seconds, depending on the computer hardware.

The "Measure" column represents the total time from starting the prediction (or manual measurement with predefined lines) to when the researcher finished editing.

The "Add_Excel" column reflects the time the researcher took from the start until adding the measurements to the Excel file. When using the model, the researcher worked with an updated version of Fiji, which included the improvements described in *Fiji adjustments* (2.4.2). Consequently, Fiji, with the model, was also able to export to Excel faster.

Table 2.4: Time measurement results for four groups of images. All times are in min:s:ms.

Type	**Model**	**Predict**	**Measure**	**Add_Excel**
First in set	N	/	01:12.00	01:30.00
First in set	Y	00:14.00	00:28.00	00:38.44
Normal	N	/	00:57.32	01:26.00
Normal	Y	00:14.00	00:25.20	00:38.34
Model Wrong	N	/	01:09.00	01:32.00
Model Wrong	Y	00:14.00	00:57.32	01:10.00
Hard Oxidation	N	/	01:05.00	01:26.00
Hard Oxidation	Y	00:14.00	00:45.24	00:59.00

2.6.1 Time per one set

A theoretical set of 30 images was considered for this experiment. It was assumed that the first image in the set would take the longest to process, as is explained in 2.1.1. Additionally, two images were expected to experience significant model errors, while two others presented challenging oxidation conditions. The time spent on measurement with the model was approximated as:

$$\text{Time with model} = 28+(25\times25)+(57\times2)+(45\times2) = 857 \text{ seconds} = \frac{857}{60} \approx 14.28 \text{ minutes.}$$

The results of the time calculations are summarized in Table 2.5.

Table 2.5: Comparison of time taken in different scenarios.

Scenario	**Time (seconds)**	**Time (minutes)**
Model - Measure	857	$\frac{857}{60} \approx 14.28$
Manual - Measure	1765	$\frac{1765}{60} \approx 29.42$
Model - Total	2596	$\frac{2596}{60} \approx 43.27$
Manual - Total	1246	$\frac{1246}{60} \approx 20.77$

The percentage reductions in time are presented in Table 2.6:

Table 2.6: Percentage reduction in measurement and total time.

Scenario	**Percentage reduction**
Reduction in measurement time	$\frac{29.42-14.28}{29.42} \times 100 \approx 51.5\%$
Reduction in total time	$\frac{43.27-20.77}{43.27} \times 100 \approx 52.0\%$

Thus, using the model results in significantly faster processing times for measurement and saving results. Although these measurements are approximations, the results clearly demonstrate that the model improves the speed of the process.

2.7 Results - Precision

After the model was integrated into the workflow, data were collected from the research process. A total of 90 images were analyzed using the model. For each image, two folders were generated: one containing the predicted lines and another for the lines that could be manually adjusted by the researcher. The results from three separate sets are summarized below.

The first and second sets showed high accuracy, requiring minimal manual adjustments. However, the third set exhibited reduced performance. This decline was attributed to the introduction of different types of layers. Notably, the model also detected the white oxidation layer above the coating layer, as shown in Figure 2.17. This layer is visually similar to other structural layers, as seen in Figure 2.1 (bottom image).

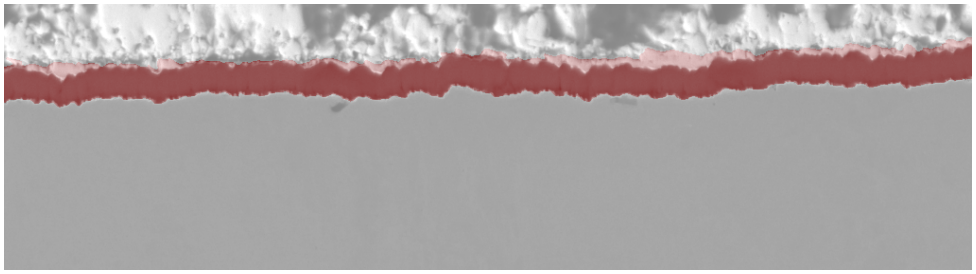


Figure 2.17: Image from the third set, highlighting a poor prediction.

These results suggest that the model performs well overall, requiring only minor adjustments in most cases. However, the findings also highlight the need to obtain a larger dataset. The current dataset is still relatively small. These new data can be incorporated into the dataset using polygon labels, which can be efficiently created using automated scripts. After this, the model can be re-trained to make it more robust. This will create a feedback loop where labels can be generated faster, leading to a larger dataset, which, in turn, after re-training, improves overall precision and results in time savings.

The precision of the model's predictions was evaluated by counting the number of points that required manual adjustment after the predictions were made. Table 2.7 presents the summary statistics for the three analyzed sets.

Table 2.7: Analysis of model predictions across three sets.

Set	**Mean difference (pixel)**	**Unchanged Points (%)**
1	1.11	93.3% (560/600)
2	0.57	97.3% (584/600)
3	5.58	22.7% (136/600)

As shown in Table 2.7, the first two sets display a high percentage of unchanged points, indicating that the model's predictions are highly accurate. In contrast, the third set shows a significantly higher mean difference and a lower percentage of unchanged points. This highlights the need for additional data to improve the model's performance.

3. Conclusion

This thesis successfully addresses the challenge of automating a time-consuming manual process in the nuclear industry. Researchers are often required to perform repetitive, labor-intensive measurements. The primary goal was to develop a solution that integrates seamlessly into the researcher's workflow without causing disruptions. The thesis focused on a specific category of coating-layer images with relatively homogeneous properties, in contrast to other types of images in the dataset. It was shown that segmentation of more complex, heterogeneous images is a viable direction for future work.

Three approaches were evaluated. First, the classical K-means algorithm was found to be computationally efficient, but it lacked universal applicability due to the variability of coating layers. This made it unsuitable for a streamlined workflow without extensive post-processing. Second, the Segment Anything Model (SAM) was considered, but hardware and precision limitations made it infeasible for the given problem. Finally, Convolutional Neural Networks (CNNs) were explored. This solution required labeled data. Three types of labels were developed. The model was trained on the most precise set, but it was found that even the least manually intensive labels were effective for training, although not as good as the most precise type.

The optimization and testing of the model were thoroughly discussed, with results measured using the Intersection over Union (IoU) metric alongside the Hausdorff distance. The model achieved a mean IoU of 95.155%, demonstrating its high accuracy in segmenting coating layers.

The practical impact was clear after integrating the model into the researcher's workflow. The time required for measurement tasks was reduced by half. For images similar to the training dataset, over 90% of predicted measurements required no further manual adjustments. However, for images from sets with significantly different characteristics, performance declined, highlighting the need for further re-training.

With the code infrastructure in place, future improvements can focus on streamlining the training process and enhancing prediction accuracy. The current model was trained on fewer than 100 images from a limited number of sets, which is insufficient for creating a fully robust model. Nevertheless, even when incorrect, the model's average deviation was approximately five pixels, resulting in substantial time savings. This efficiency will accelerate the generation of additional labeled data, creating a feedback loop for continuous improvement. The iterative workflow is illustrated in Figure 3.1.

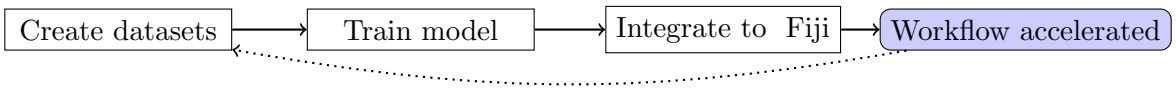


Figure 3.1: Workflow of model integration and performance improvement.

In conclusion, this thesis automates a portion of the researcher's measurement process by addressing the challenges of applying deep learning to image analysis tasks. By refining traditional methods for label reconstruction, this work establishes a cost-effective approach to preparing high-quality ground truth labels, enabling the successful application of deep learning. The improvements brought by this work will lead to significant time savings and enhanced efficiency, paving the way for continuous model retraining and further advancements.

4. Appendix

The complete implementation of this thesis, including preprocessing, label generation, model training, and evaluation scripts, is available at GitHub (`https://github.com/emmateki/coating_detection`)

This project benefited from various open-source libraries that facilitated image processing, deep learning model training, and evaluation. A detailed list of dependencies and libraries used can be found in the repository's environment file.

5. Disclaimer

I hereby declare that I have completed this thesis independently, with the assistance of artificial intelligence tools used for specific tasks throughout the writing process. I have thoroughly reviewed all AI-generated content to ensure its accuracy and correctness.

I take full responsibility for the information presented in this thesis, guaranteeing its reliability. The use of AI tools was conducted in compliance with ethical standards and academic integrity, ensuring that the work remains original and authentic.

The following tools were used, along with their respective purposes:

- **Grammarly** (https://www.grammarly.com) - for text corrections
- **ChatGPT-4** (https://chat.openai.com/) - for rephrasing and text corrections
- **GitHub Copilot** (https://copilot.github.com) - for code corrections

Bibliography

[1] A. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin. Albumentations: fast and flexible image augmentations. *ArXiv e-prints*, 2018.

[2] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.

[3] Zhuoheng Chen, Xiaojun Liu, Jijin Yang, Edward Little, and Yu Zhou. Deep learning-based method for SEM image segmentation in mineral characterization, an example from Duvernay Shale samples in Western Canada Sedimentary Basin. *Computers & Geosciences*, 138:104450, February 2020.

[4] Jeff Henrikson. Completeness and total boundedness of the hausdorff metric. *MIT Undergraduate Journal of Mathematics*, 1:69–80, 1999.

[5] Pavel Iakubovskii. Segmentation models pytorch. `https://github.com/qubvel/segmentation_models.pytorch`, 2019.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[7] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[8] Stuart Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[9] Segmentation Models. Segmentation models - documentation. `https://smp.readthedocs.io/en/latest/models.html#id22`, 2025.

[10] Keiron O'Shea and Ryan Nash. An Introduction to Convolutional Neural Networks, December 2015.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015.

[13] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, et al. Fiji: an open-source platform for biological-image analysis. *Nature methods*, 9(7):676–682, 2012.

[14] Jian Wu, Wanli Liu, Chen Li, Tao Jiang, Islam Mohammad Shariful, Yudong Yao, Hongzan Sun, Xiaoqi Li, Xintong Li, Xinyu Huang, and Marcin Grzegorzek. A state-of-the-art survey of U-Net in microscopic image analysis: from simple usage to structure mortification. *Neural Computing and Applications*, 36(7):3317–3346, March 2024.

List of Figures

1.1 Examples of coating and oxidation layers in material samples. 2

2.1 Cropped SEM images with partially highlighted coating layers with red border. 4
2.2 Measurement in Fiji software. Only the region of interest is shown; the bottom part of the image is omitted for clarity. 6
2.3 Highlighted coating layers predicted using K-means. 7
2.4 Top: User-provided points (input). Bottom: SAM-generated output mask. 8
2.5 Examples of polygon, K-Means, and manually refined labels, shown as red-highlighted borders over the original SEM image. 12
2.6 U-Net architecture [12] 14
2.7 Illustration of various augmentation techniques applied to an image from the dataset. 16
2.8 Learning Rate Schedulers (LRS) 19
2.9 Validation loss values for three different learning rates over 200 epochs. . . 22
2.10 Validation loss over 200 epochs 24
2.11 Hyperparameter importance 24
2.12 Comparison of predictions made by models trained on different datasets, alongside the original ground truth label on a complex sample. 27
2.13 Prediction labels shown in red border on the original image with extensive oxidation 28
2.14 Example image with minimum IoU 28
2.15 Example image with maximum HD 28
2.16 Example workflow in Fiji 29
2.17 Image from the third set, highlighting a poor prediction. 32

3.1 Workflow of model integration and performance improvement. 35

List of Tables

2.1 Best results for different Learning Rate Schedulers. 22

2.2 Validation loss and corresponding hyperparameters across trials 23

2.3 Evaluation Results on Test Data 26

2.4 Time measurement results for four groups of images. All times are in min:s:ms. 31

2.5 Comparison of time taken in different scenarios. 31

2.6 Percentage reduction in measurement and total time. 32

2.7 Analysis of model predictions across three sets. 33

6. List of Abbreviations

- **TP** - True Positive
- **TN** - True Negative
- **FP** - False Positive
- **FN** - False Negative
- **CNN** - Convolutional Neural Network
- **LR** - Learning Rate
- **LRS** - Learning Rate Scheduler
- **SEM** - Scanning electron microscope
- **IoU** - Intersection over union
- **SAM** - Segment Anything Model
- **HD** - Hausdorff Distance