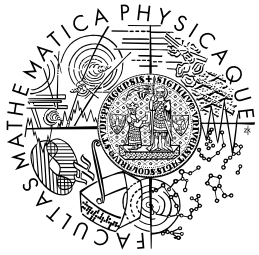


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Jan Palášek

Detection of grids on nuclear fuel set images

Department of Software and Computer Science Education

Supervisor of the master thesis: RNDr. Jan Blažek, Ph.D.
Study programme: Computer Science
Study branch: Artificial Intelligence

Prague 2021

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

To my fiancée Anastasia, my sister Martina and my parents for their support during writing this thesis and my studies.

Title: Detection of grids on nuclear fuel set images

Author: Jan Palášek

Department: Department of Software and Computer Science Education

Supervisor: RNDr. Jan Blažek, Ph.D., Department of Software and Computer Science Education

Abstract: Visual inspection of fuel assemblies is necessary to identify potential anomalies in their behaviour associated with their condition and their future usage. One of the possible findings are foreign objects caught on the fuel spacer grid which can disrupt the cladding of fuel rods during the operation. The goal of this thesis is to accurately segment the spacer grid from an image, which is a task dual to the foreign object detection, and therefore to automate visual inspection process in this area. We created new datasets covering typical problems appearing on the fuel assembly. To perform the segmentation, we employed neural networks. We increased performance by data augmentation techniques and domain-specific output post-processing. We also measured the algorithm's performance by a newly introduced Line Distance metric, computing the size of the maximum uncertain area between the actual and the predicted transition between grids and rods. In the experiments, we found the best hyperparameters and reached very good results, outperforming our predecessor's algorithm by having three times lower Line Distance metric.

Keywords: nuclear fuel, image segmentation, grid

Contents

Introduction

Visual nuclear fuel inspection is necessary in order to ensure reliable performance of the fuel design and the facility in which it operates. The subject of such inspection is a fuel assembly.

The fuel assembly consists of spacer grids and fuel rods. The purpose of the grids is to hold the fuel rods in a specific shape, where any two neighbouring rods have equal distance between one another.

In this thesis, we work with a fuel assembly of VVER-1000 design. Such design is selected because CVR, in whose direct cooperation this thesis is created, has its mock-up available in their laboratory and it is widely used in eastern types of power reactors. To simplify our nomenclature in the rest of the text, we use the term fuel assembly synonymously with the term mock-up unless stated otherwise.

Figure 1: Fuel assembly. The picture shows the subjects of the visual inspection: fuel rods and spacer grids. Some of the rods are extracted to reveal the assembly's inner structure. The assembly in the picture is a mock-up of VVER-1000 fuel assembly [7].

The visual inspection of the fuel assembly in CVR is performed using a radiation-resistant camera controlled by operators. They scan the assembly looking for defects that could disrupt reliable operation of the fuel [7]. During the inspection, only a sample of fuel assemblies are inspected due to the limited time reserved for them (between 70 and 120 hours yearly) [9]. An automation of the inspection using Digital Image Processing methods could

- reduce a human error caused by subjectivity or fatigue,
- reduce the necessary time to perform such inspection [7].

One of the goals of the visual inspection is to detect anomalies on the fuel assembly. An example of such an anomaly could be a foreign object caught on the spacer grid. This object could cause fretting which could subsequently damage the cladding of the rods.

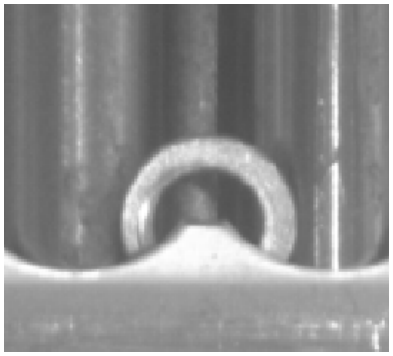
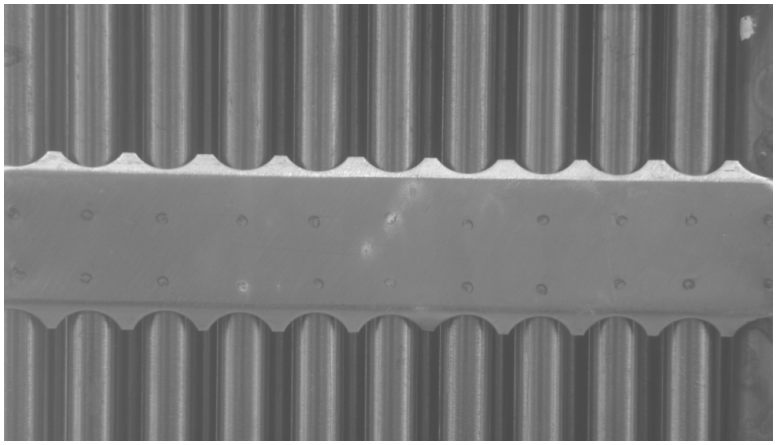


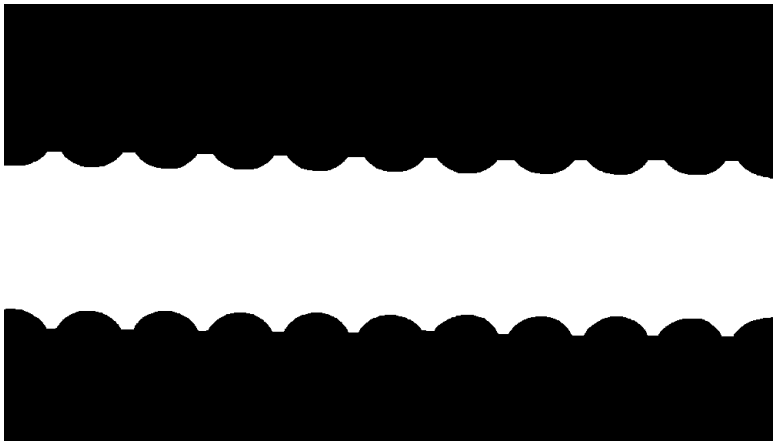
Figure 2: An example of a defect: a foreign object caught on the spacer grid.

The first step towards the detection was done by Knotek [7]. His efforts were successful in the rods area but had difficulties in the transition area between the grid and the rods. In order to perform detection there, he attempted to segment the grid's mask. This proved to be difficult due to e.g. irregular lighting [7, §4.5].

The goal of this thesis is to continue the efforts of Knotek and accurately segment the spacer grids from an image.



(a) The spacer grid and the rods screened from the front.



(b) Segmented mask of the grid. White pixels indicate where the grid is.

Figure 3: The fuel assembly screened from the front and its segmentation.

Chapters

This thesis comprises of five chapters:

- Problem analysis: the goal of this chapter is to analyse the problem of segmenting spacer grids and select the best method to solve it.
- Related work: in the second chapter, we describe related work important for this thesis.
- Method: the third chapter describes our contribution, specifically the data augmentation, used models and the prediction's post-processing.
- Experiments: in the fourth chapter, we analyse importance of individual hyperparameters, attempt to find the best model and evaluate it.
- Discussion and future work: in the last chapter, we discuss our findings, outline our algorithm's future usage in the CVR framework and suggest a promising way of increasing its performance.

1. Problem analysis

This chapter describes visual conditions of screening the fuel assembly and its possible difficulties. Lastly, it specifies our goal and outlines our method reaching it.

1.1 Description of the fuel assembly

The fuel assembly consists of fuel rods and hexagonally-shaped spacer grids. Both are made of zirconium alloy resembling stainless steel. The surface of the grid is flat, except for possible scratches and engraved text [7]. The grids are placed on top of each other with a specific distance between them. The rods are inserted into the grids. Each of the six sides of the fuel assembly has 11 rods. There are usually 10 grid's teeth separating the rods, see figure 3. The fuel assembly, however, have a special side excepting this rule. The two rods in the middle have no grid tooth between them, as seen in figure 1.1, resulting in this side having only 9 teeth.

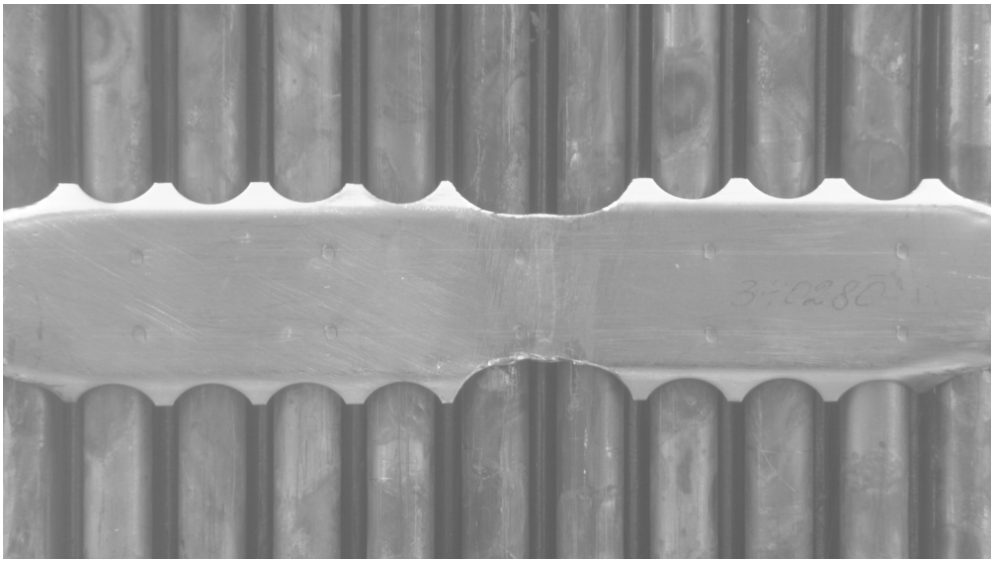


Figure 1.1: The special side of the fuel assembly. It has 11 rods but only 9 teeth because there is a groove in the middle and hence there is no tooth between the sixth and the seventh rod.

In the nuclear power plant, the fuel assembly is inspected by CVR operators using a radiation-resistant camera. The only illumination source is point lights placed around the camera. The camera is remotely controlled by an operator during the inspection, allowing him to focus on various problematic parts. The fuel assembly is submerged into water. It is approximately 4.5 m tall with grids being placed every 0.5 m. They limit the bend of the rods.

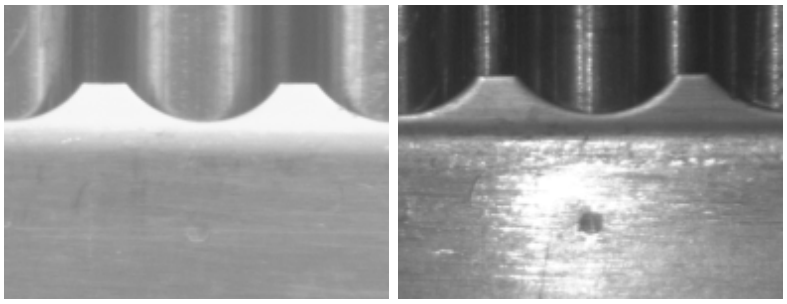
Our screening was done in CVR laboratory, where our main goal was to simulate the power plant screening conditions using available tools. This allowed us to have a better control of the environment. We could have inserted foreign objects more easily and also progressively increased the difficulty of our screening

conditions. This method also allowed us to verify that our algorithm worked on the easier data at start while being able to gradually make the data more challenging and closer to the real power plant data. Instead of the nuclear power plant fuel assembly, we used a shortened fuel assembly's mock-up without the fuel. It was 1.25 m tall with 3 spacer grids.

1.2 Difficulties

A simple separation of the grid and rods is complicated by many difficulties. All of them suggest that the grid pixels cannot be simply inferred using their close neighbours, but they need a larger context to be able to perform a precise segmentation.

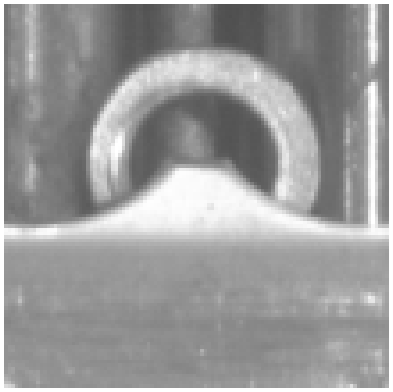
Lighting Lighting, especially using point lights, can be irregular and create glares or shades. These effects can confuse a simple edge detector, or even hide part of the transition between the grid and the rods.



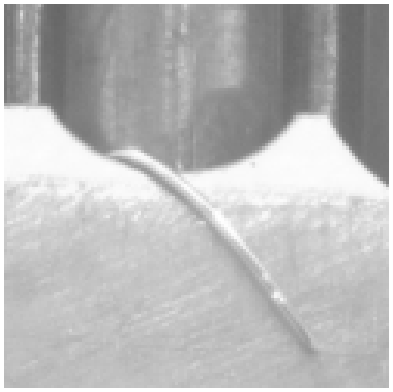
(a) Tube-light illumination. (b) Point-light illumination.

Figure 1.2: Example of tube-light and point-light illumination. The former has slightly lighter grid's teeth, the latter has a big glare in the grid part.

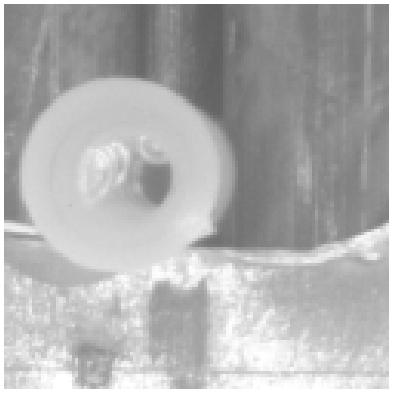
Debris A foreign object can be found usually caught on the spacer grid. This can cover part of the transition between the grid and the rods and also potentially create new edges in a simple edge detector.



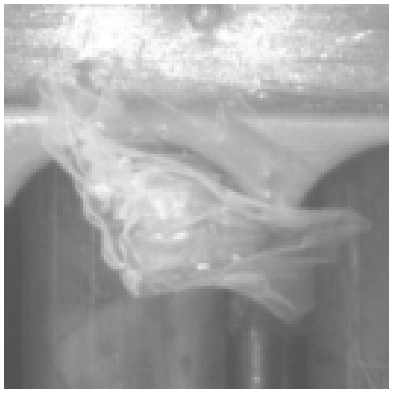
(a) Washer debris.



(b) Wire debris.



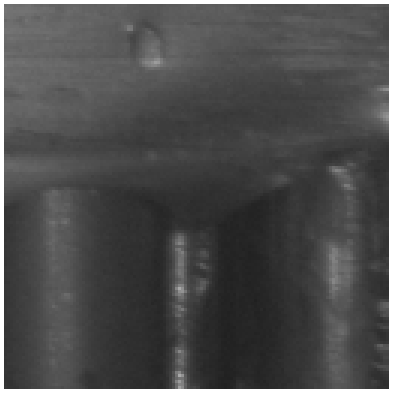
(c) Silicone debris.



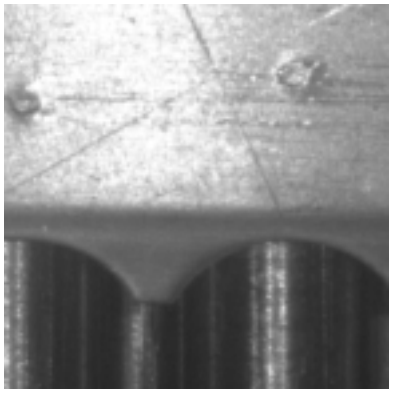
(d) Plastic debris.

Figure 1.3: Example of debris that can occur caught on the grid. The silicone and plastic bags do not appear in the power plant, we added them to increase the difficulty of our data.

Damaged grid The grid can have its teeth bent or broken off. This results in the grid having an irregular shape. The grid and rods can be also scratched.



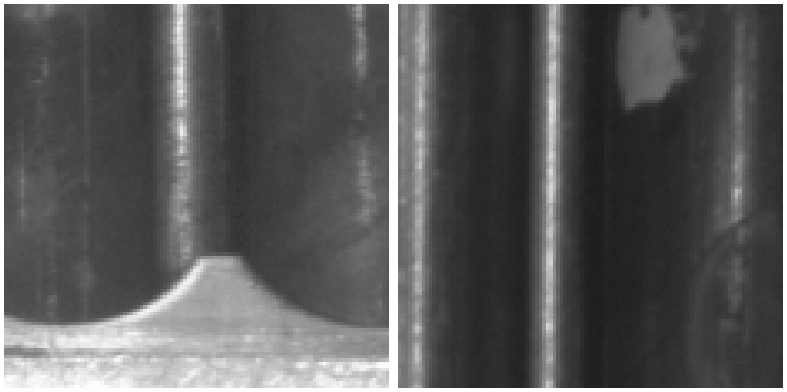
(a) Bent grid's tooth.



(b) Grid scratch.

Figure 1.4: Examples of a damaged grid.

Oxide A surface-level oxide on a rod manifests itself as a dark part of the rod. The rod loses its reflective visual properties and has a dark gritty look. A deep oxide looks like a white spot on the rod.



(a) Surface-level oxide. (b) Deep oxide.

Figure 1.5: Examples of oxide manifestation on the rods.

1.3 Datasets

We created three datasets distinguished by their screening visual conditions: Easy, Medium and Hard. Every subsequent dataset built upon the difficulty of the previous one, adding new problems needed to be solved.

The precise screening procedure is described in appendix A. The manual preprocessing is discussed in appendix B.

Easy dataset The first dataset was taken using a professional photographic equipment. The lighting was done from one side using a photographic lamp and from the other side using a light reflector. Every other light source was covered. The photos were taken by Canon EOS D500 (appendix C).

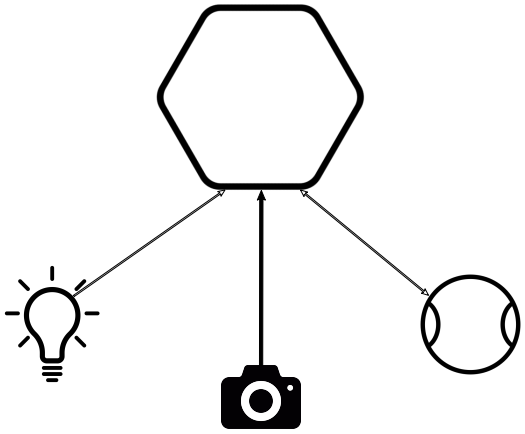


Figure 1.6: Schema showing the screening setup from above. The hexagon represents the fuel assembly, the lighbulb represents the photographic lamp and the circle shape represents the light reflector. There is a big angle between the camera and the lamp and the camera and the reflector to eliminate glares. Consequently, it also highlights scratches.

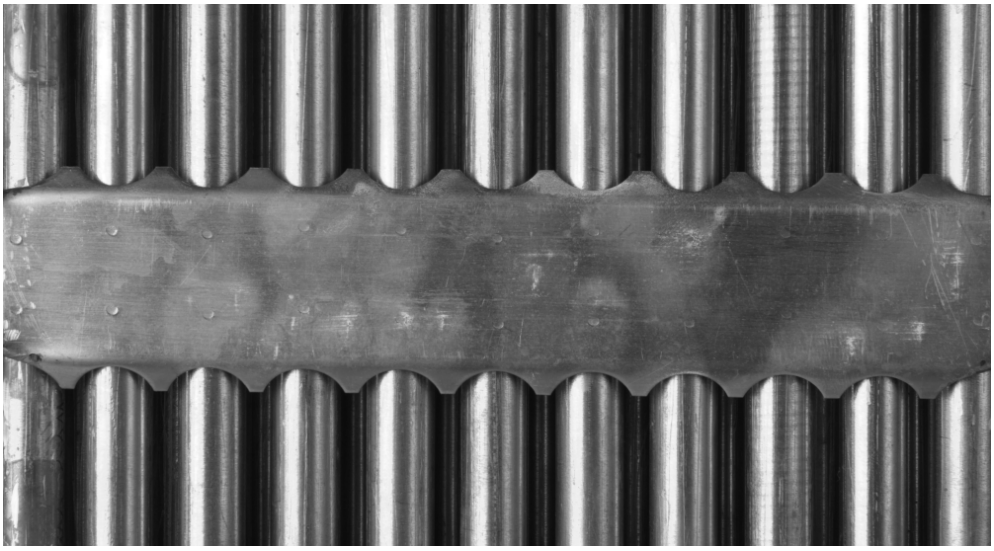


Figure 1.7: Example of a photograph from Easy dataset.

The dataset is the easiest because it allows us to illuminate arbitrary part of grids or rods as well as we can.

To be able to solve it, our algorithm needs to be able to ignore grid scratches and handle bent grid's teeth. Therefore, it cannot completely rely on smoothness of the grid and regularity of the grid's teeth.

Medium dataset Medium dataset was taken using Olympus TG-5 (appendix C). The only illumination source were the tube lights on the ceiling of the laboratory room. The screened parts of the fuel assembly and the camera were submerged in the water. The debris was also added, potentially covering part of the transition between the grids and the rods.

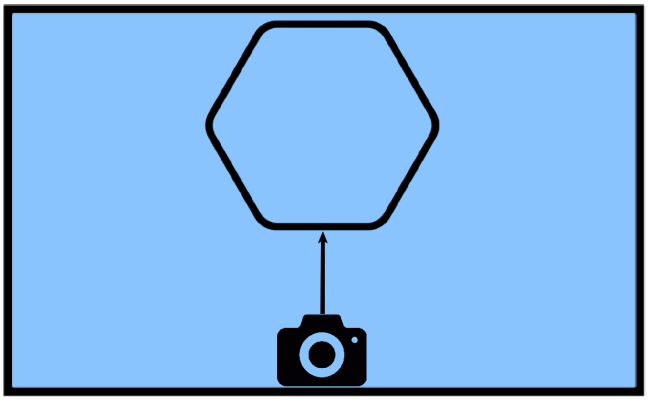


Figure 1.8: Diagram of screening Medium dataset from above. The rectangle with blue filling represents the water tank, the hexagon is the fuel assembly.

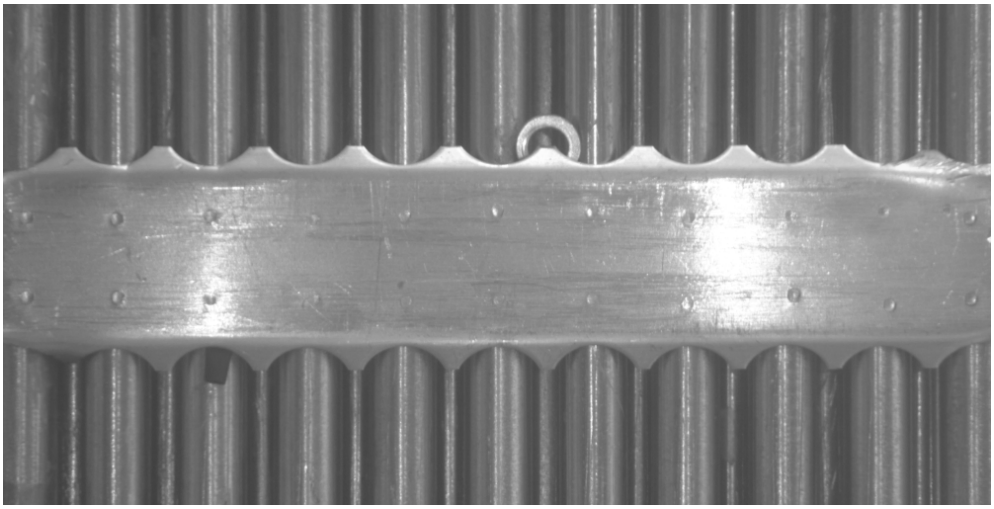


Figure 1.9: Example from Medium dataset.

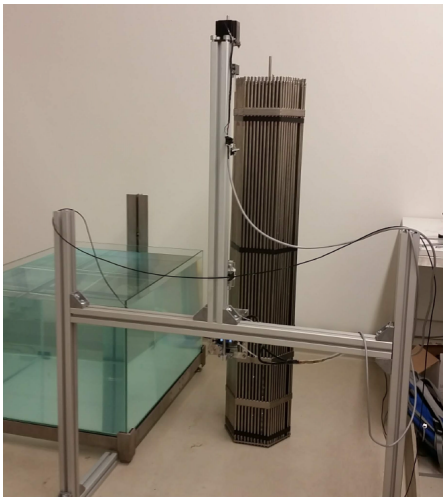


Figure 1.10: The water tank and the fuel assembly in CVR laboratory room [7].

The added difficulty of the dataset is following:

- Upper grid's teeth are more illuminated because the light source is above the fuel assembly.
- The underwater screening adds a blur effect.
- The debris adds another layer of difficulty because the model needs to learn to ignore it.

Hard dataset This dataset was taken in the water tank using the Olympus camera, too (appendix C). However, instead of the tube lights, we used point lights attached to the camera stand.

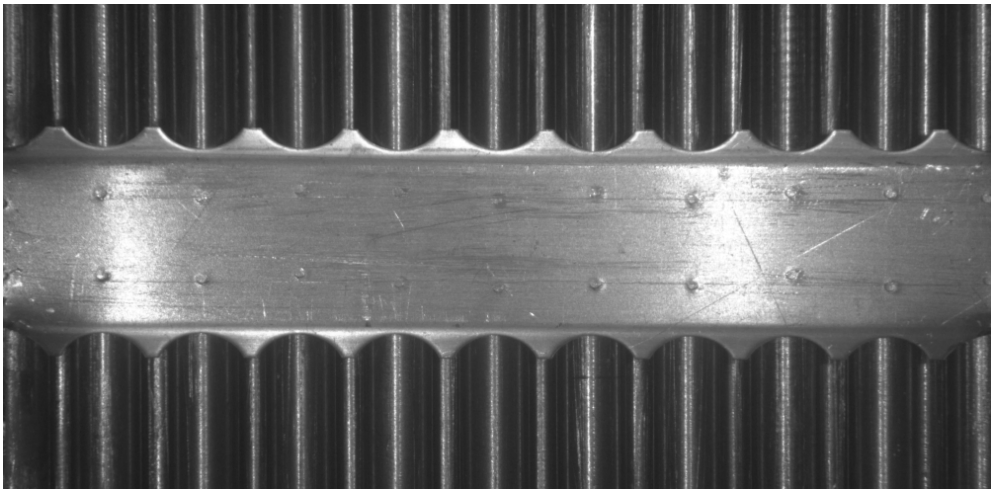


Figure 1.11: Example of a photograph from Hard dataset.

Upon the previous dataset difficulties, Hard dataset adds the intricacy of irregular lighting, converging much closer to the visual conditions of nuclear power plant data.

1.4 Choosing the approach

Our goal is to segment the grid's mask. This is a task dual to detecting the defects in the transition area between the grids and the rods.

Classical digital image processing methods have shown to have problems especially with irregular lighting. This has already been explored by Knotek [7]. He took a pre-defined mask and attempted to fit it onto the image so that it filled out missing edges.

The method worked with limited accuracy on the images illuminated by tube lights but failed completely on the images illuminated by point lights. This happened due to many anomaly edges caused by irregularity of point lights.

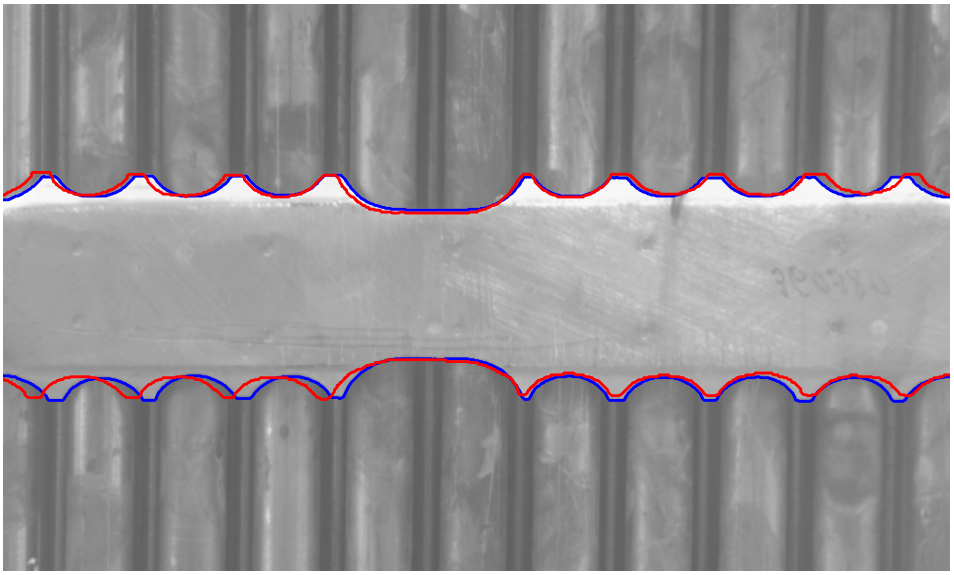


Figure 1.12: Example of Knotek's segmentation. The red coloured curve indicates the predicted boundary between rods and the grid, the blue curve is the true boundary. The tooth is missing between the fifth and the sixth rod. This does not occur in our datasets, we flipped the image along the vertical axis due to requirements of Knotek's to have the groove between the fifth and the sixth rod. The prediction is not very accurate because it relies on a pre-defined mask universal for all images.

We choose a different approach. We use artificial neural networks to perform semantic segmentation and thus predict the grid's mask. This can help us solve difficulties mentioned in section 1.2, which previously proved to be hard to tackle.

One of the disadvantages of neural networks is their lack of interpretability. We cannot decide, whether an obscure prediction is caused by the neural network making an error or e.g. by a deformed grid. To solve this, we employ a post-processing of the network's predictions using our domain knowledge. It attempts to either fix the network's prediction, if it is sure the anomaly is an error, or wrap the problematic part into a bounding box.

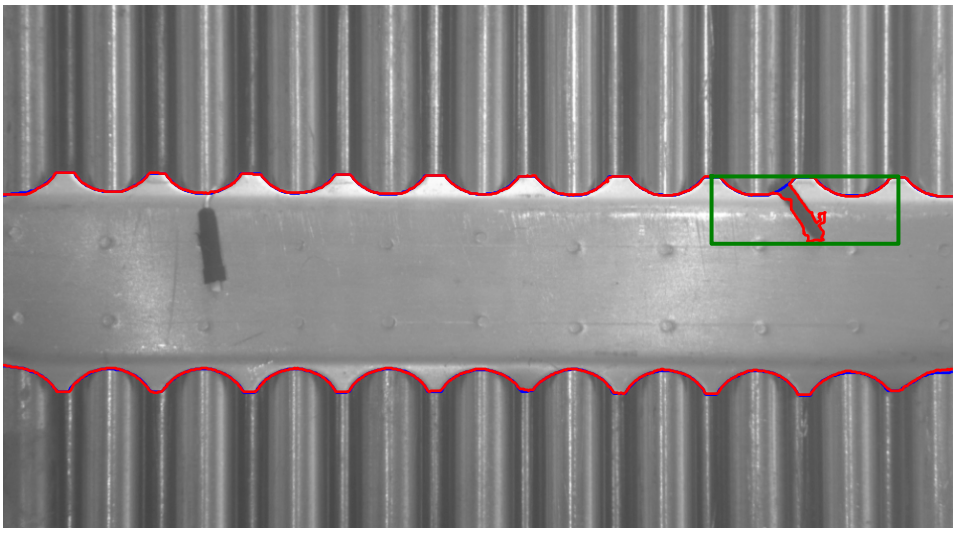


Figure 1.13: Problematic input with a bounding box surrounding the problem. The red coloured curve denotes the predicted boundary between rods and the grid, the blue curve is the true boundary and the bounding box is coloured in green.

Another problem that arises is small amount of data. This stems from two sources:

1. The laboratory we cooperated with has only one fuel assembly mock-up. This leads to us having at most 18 different grid photos because the hexagonally-shaped fuel assembly has 6 sides with 3 grids each. Unfortunately, the number of rods is limited. We have enough of them to only fill at most 3 sides out of the total 6. Hence, we can have at most 9 images. The rods could be moved to screen other sides too, but it is very time-consuming and it further damages the fuel assembly by scratching it.

2. Setting up the scene, screening and creating the labels is also very time-consuming.

We try to overcome this obstacle by using data augmentation.

2. Related work

This chapter discusses previous attempts to segment the grid and relevant deep learning semantic segmentation architectures.

2.1 Previous grid segmentation attempts

Knotek [7] attempted to extract grid's mask. He assumed all grids' masks were identical and prepared one mask. Then he attempted to fit it onto the new grid and refined the fit using Canny Edge Detector. Unfortunately, as mentioned in section 1.4, this method did not succeed on point-light illuminated images.

2.2 Relevant architectures

We considered the following four architectures: U-Net, ResU-Net, ResU-Net++ and DeepLab V3+. U-Net [10] achieved state of the art results in medical semantic segmentation using only few image inputs. ResU-Net [12] improved U-Net's training speed while not hindering its accuracy. It also tackled the problem of high resolution images. ResU-Net++ [5] followed up on the recent successes of the attention modules and used it to improve performance of the network. Finally, DeepLab V3+ [2] has been a state of the art semantic segmentation architecture on high resolution real-life images from streets.

3. Method

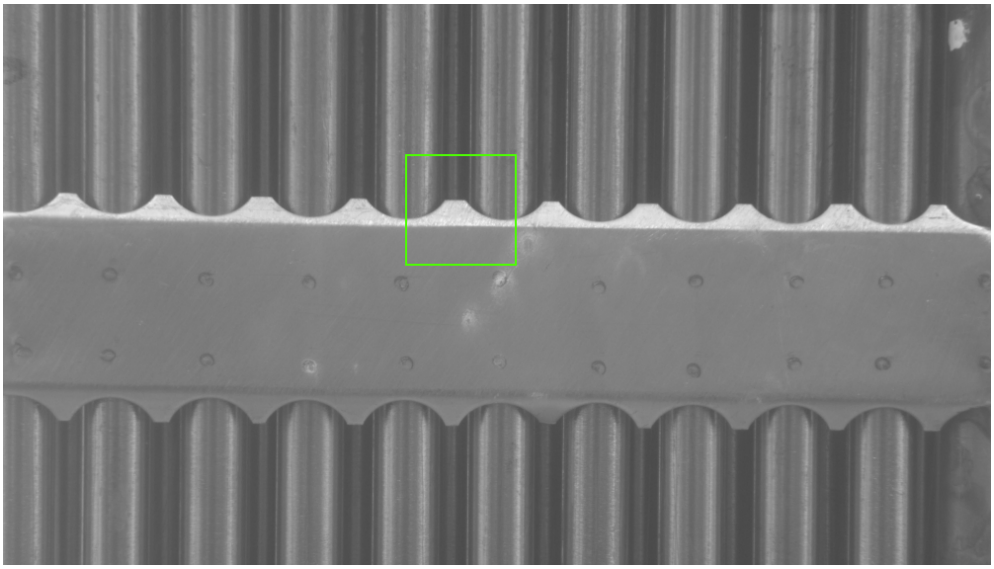
In this chapter, we describe our solutions to the problems introduced in chapter 1. In the first section, we describe our data augmentation. The second section looks into the model. The third section explains post-processing of the model's prediction.

3.1 Data augmentation

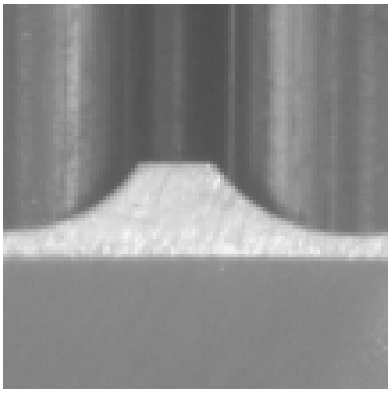
To compensate for our very limited number of data, we employed data augmentation. We used the following techniques: cropping, mirroring, noise addition, rotation, zooming and cutout.

3.1.1 Cropping

Instead of using whole image as an input of the model, we split the image into many square-shaped sub-images (crops) of the same size and performed predictions on them. Then, we combined those small predictions into a whole image-level prediction.



(a) Whole image with a highlighted sub-image: a crop.



(b) Extracted crop.

Figure 3.1: Example of a whole image and one of its possible crops.

Note that the cropping was the only augmentation technique performed on the evaluation data as well. Since the model is trained on the crops, it also needs to be evaluated on them.

Deterministic cropping In this thesis, we used Deterministic cropping method [12]. It splits the whole image into square-shaped crops. The overlay parameter o is the length of the overlapping region between two subsequent crops. We can observe from figure 3.2 that larger o leads to more crops obtained from the image.

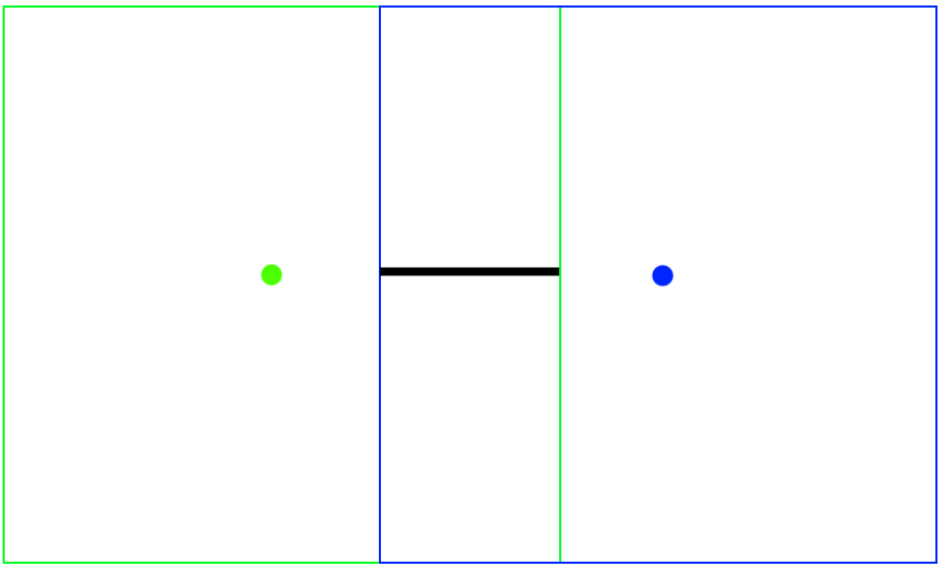
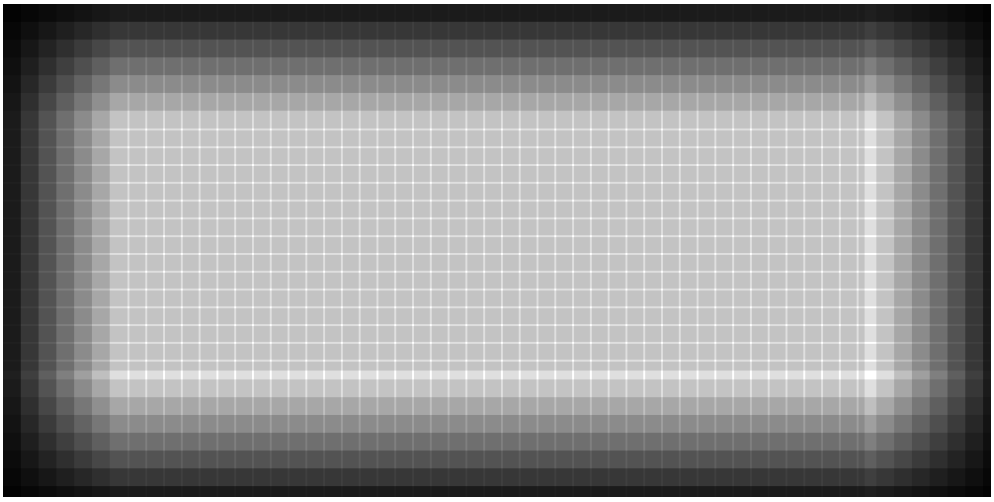
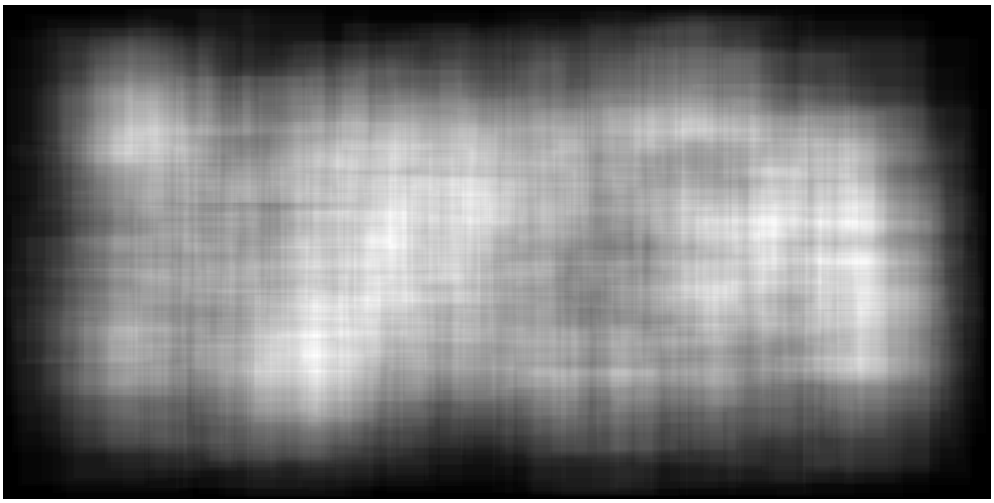


Figure 3.2: The two crops are denoted as the blue and green squares. They have identical sizes s_c. Their centers c_{c1} and c_{c2} are depicted as the blue and green dots. The overlay of the length o is shown as the black line.

This approach works better for semantic segmentation task than Random cropping [8] because it covers the image (roughly) uniformly and does not force our model to train on many crops from the same area while omitting other regions entirely.



(a) Deterministic crop. The crops are organised with equal distances between one another. The edge areas are less covered, but there is always at least one crop intersecting every pixel.



(b) Random crop. The crops randomly cover some areas more and some less. The edge areas are less covered because the probability of selecting them is lower. Many of the pixels on the border of the image are not covered at all.

Figure 3.3: Coverage of the whole image by crops using the two methods. Lighter pixels mean that more crops intersect them. Both coverages have been made using the same number of crops.

Suppose we have overlay $o = 1$. This means that two subsequent crops overlap by one pixel. If we wanted to infer pixel in the overlapping area, the prediction would only have context from the one side because the crops are given to our model independently. Therefore, it could have been easily missclassified.

To solve this, we introduce a term sub-crop s: a smaller crop cut from the crop c. The sub-crop represents the minimum context for its center pixel needed to perform the prediction. We want every pixel to be a center of a sub-crop, whose all pixels are entirely covered by one crop. We denote the sub-crop's size as s_s.

We can enforce this property by setting up the overlay o. To establish its lower boundary, imagine the worst-case scenario: inferring the pixel p right between two subsequent crops. If $o < s_s$, then pixels of sub-crop with center p are not all

covered by the same crop. This problem is depicted in figure 3.4. Therefore, we want the following inequality to hold:

$$o \geq s_s \,. \tag{3.1}$$

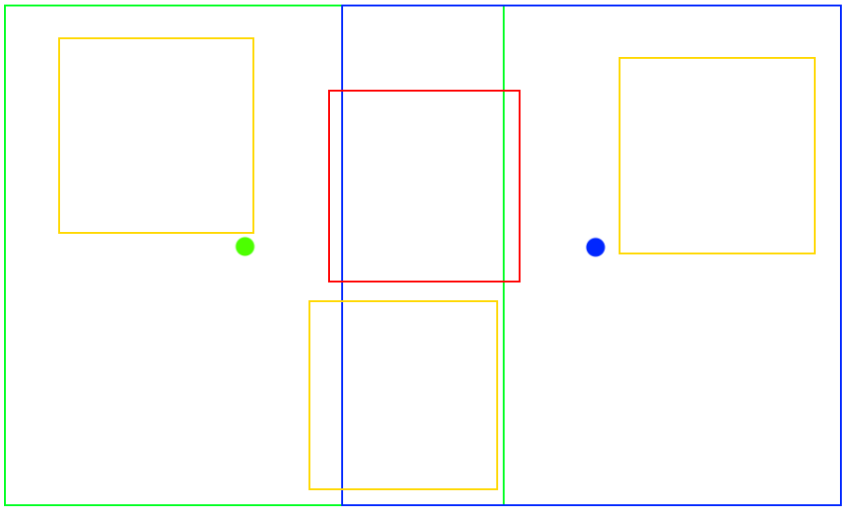


Figure 3.4: Crops are denoted as the blue and green coloured squares with their centers being the blue and green dots. The yellow squares are sub-crops entirely covered by one of the crops. The red square is a sub-crop that is not entirely covered by one of the crops. Such sub-crop can exist because size of the sub-crop is larger than the size of the overlay.

However, often it is useful to set o to a higher value because higher overlay leads to more data for the model training. This results in some p's sub-crop to be entirely contained by multiple crops. In such cases, we need to determine, which one of those crops should handle the prediction. In other words, we need to decide how to combine the crops' predictions into the whole image prediction.

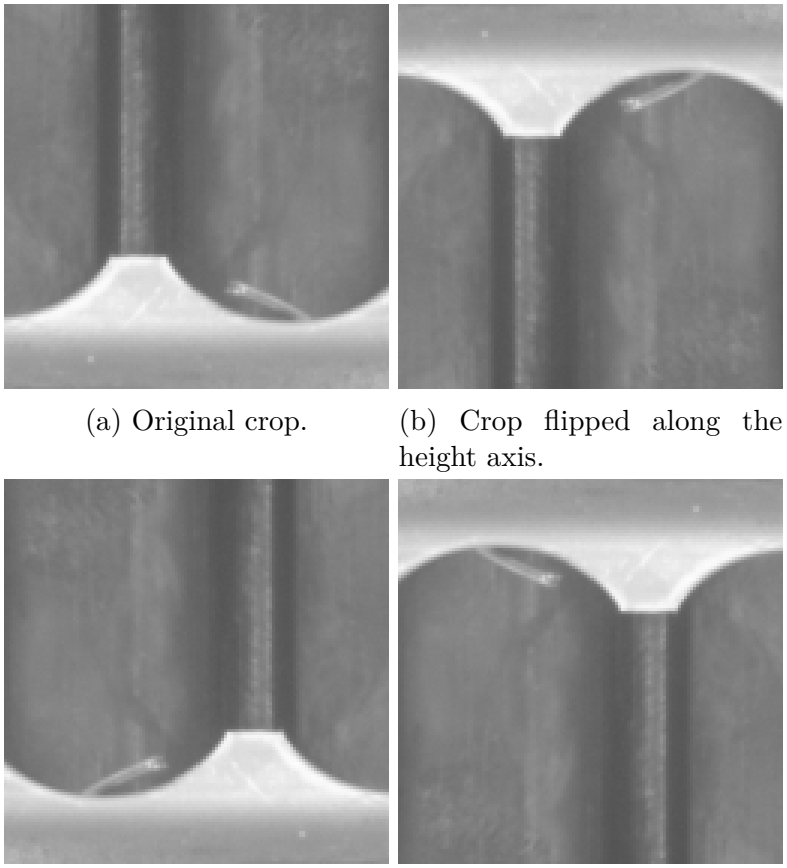
Zhang et al. [12] computed the value of pixel p by averaging all of the crop's predictions intersecting p. In our experience, if the average is unweighted, this does not yield the best results. The crops, where p is close to their edge, have a same impact as those, where p is in their center. Such p can have context mainly from one side. We believe that having context from all sides equally is generally better, if we do not know which side is more important. Therefore, we choose to use such crop for prediction of p, whose center c is closest to p.[1]

3.1.2 Mirroring

We performed mirroring by flipping the crop along the width axis, the height axis and both the width and the height axes, making three new images out of the original one. This is demonstrated by the example in figure 3.5. The mirroring is performed both on the crops and their masks.

[1]The distance between the points is measured in Euclidean metric.

The flipping operation modifies position of the light source in the crops. This can be helpful in making the model more robust to changes in lighting, especially in case of Hard dataset.



(a) Original crop.

(b) Crop flipped along the height axis.

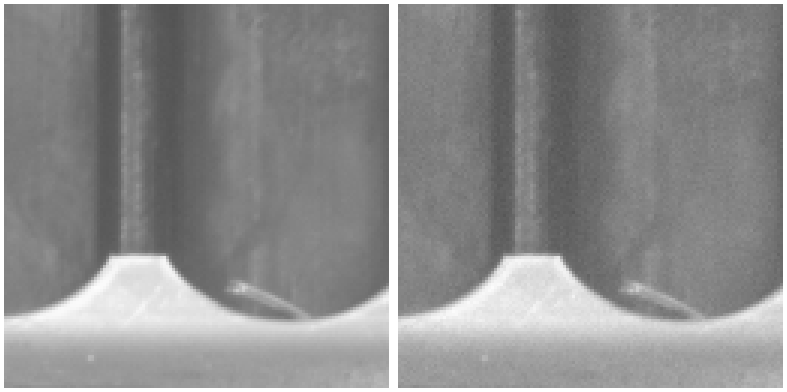
(c) Crop flipped along the width axis.

(d) Crop flipped along the height and the width axis.

Figure 3.5: Mirror operation performed on the original crop. The original crop comes from Medium dataset. The light source is above the assembly, which can be noticed on the teeth's light intensity. Flipping it along the width axis results in seemingly changing position of the light source to being below the fuel assembly.

3.1.3 Noise addition

We randomly added noise to some crops to make the model more resilient. We decided to use Gaussian noise because it frequently emerges due to bad illumination or high temperatures [1]. The noise addition only influences the crop, not its mask.

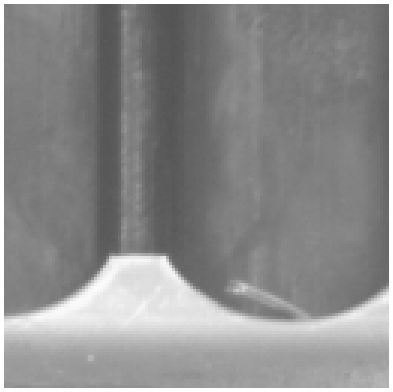


(a) Original crop. (b) Crop with added noise.

Figure 3.6: Adding noise to a crop.

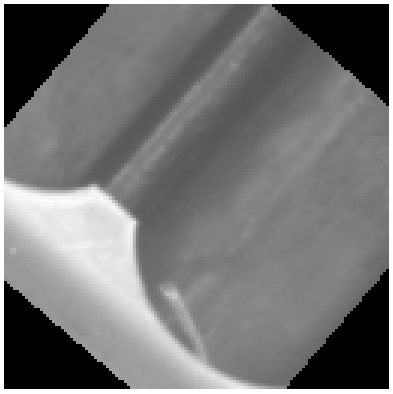
3.1.4 Rotation and zoom

To tackle the possibility of some images being slightly rotated, we added rotation augmentation. With a certain probability, it changes the original crop, rotating it by an angle uniformly sampled from $[-7°, 7°]$, the minimum and maximum angles allowed by the assembly's construction. We selected nearest neighbour method to fill in the invalid pixels.



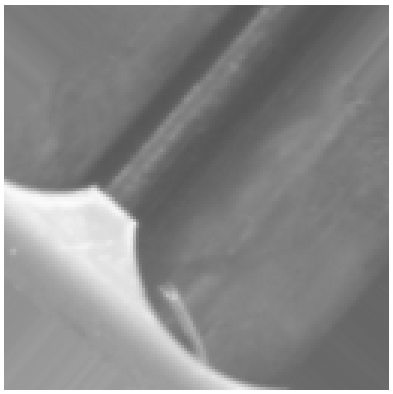
(a) Original crop.

(b) Original crop's mask.



(c) Rotated crop with no filling.

(d) Rotated crop's mask with no filling.

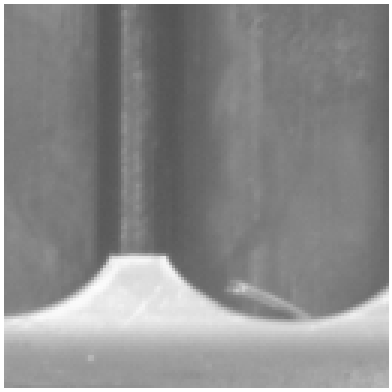


(e) Rotated crop with nearest neighbour filling.

(f) Rotated crop's mask with nearest neighbour filling.

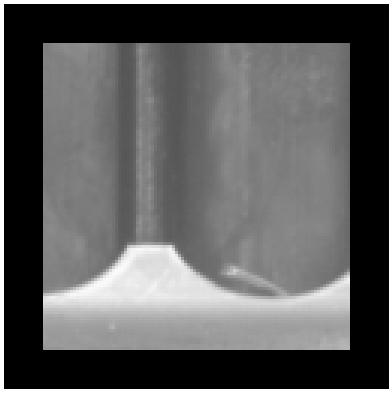
Figure 3.7: Rotating crop and its mask with and without filling invalid pixels. The crop is rotated by 45° to highlight the filling effect.

Similarly, we also employed random zooming with filling in invalid pixels using nearest neighbour method. The filling is needed only in case of zooming out. When zooming in, no invalid pixels are created.



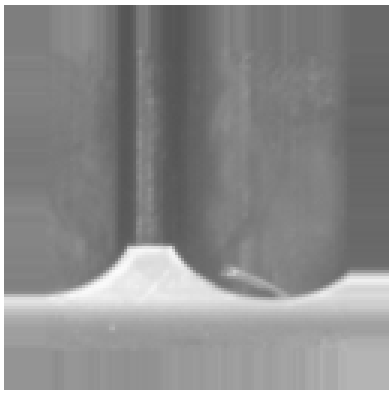
(a) Original crop.

(b) Original crop's mask.



(c) Zoomed crop with no filling.

(d) Zoomed crop's mask with no filling.



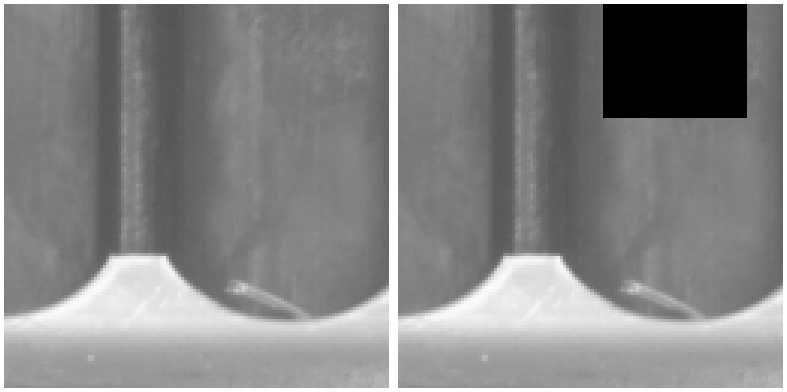
(e) Zoomed crop with nearest neighbour filling.

(f) Zoomed crop's mask with nearest neighbour filling.

Figure 3.8: Zooming of a crop and its mask with and without filling invalid pixels.

3.1.5 Cutout

The last performed augmentation was Cutout [3]. It randomly selects a pixel in the crop and marks it as a center of a square of a defined size. Then it recolours the square's pixels as black. Note that part of the cutout's area can be outside of the crop's borders. We hoped that it would simulate part of the grid's and rods' transition being covered by debris and thus help the model to ignore the debris more easily. The crop's mask was left unchanged by the cutout.



(a) Original crop.

(b) Crop with part of it being cut out.

Figure 3.9: A crop with and without part of it being cut out.

3.2 Model

We selected U-Net [10] and ResU-Net [12] architectures.

Input The input of the model is expected to be a gray-scale crop of a shape $(H, W, 1)$, where H is the height of the crop and W is the width of the crop. Its values should be normalised in the interval $[0, 1]$.

Output Output of the model has shape (H, W, C), where C is the number of output channels, H and W are the height and the width of the input crop. We used $C = 2$ with softmax instead of $C = 1$ with sigmoid due to technical reasons. The first output channel corresponds to the segmented rods, the second output channel corresponds to the segmented grids. All output values are also in the interval $[0, 1]$ and sum up to 1 over C, giving them interpretation as a probability that given prediction's pixel is a grid.

Training To train the model, we used categorical cross-entropy loss. We took only pixels that had ratio of the white pixels between 0.2 and 0.8, ensuring that we only used crops containing the transition of the grids and rods. The idea is that this greatly reduces the training time while not sacrificing any accuracy thanks to the post-processing.

3.3 Post-processing

The goal of the post-processing is to try to polish the network's prediction. If the polishing is too dificult in that particular instance, then to at least highlight the problematic areas. We call the former process Prediction cleaning, the latter Problem detection.

All Prediction cleaning and Problem detection methods work by slicing up the prediction horizontally in the middle, performing the operations on the two

halves separately and joining them back in the end. When we explain any post-processing bellow, to simplify it, we deal only with the upper half. Processing the lower half is equivalent after flipping it along the horizontal axis.

3.3.1 Prediction cleaning

Prediction cleaning has two phases: Stain removal and Bridge removal.

Stain removal A white stain, resp. black stain, is a small independent white component, resp. black component. The goal of Stain removal is to remove it. We do this by finding two largest components in the image and removing all other smaller components.

(a) Mask with black and white stains.

(b) Mask with removed black and white stains.

Figure 3.10: Example of mask with stains and its cleaned counterpart.

Bridge removal Imagine we have a black stain that is connected by a thin black line, where the width of this line is one pixel. We call such a stain a *connected stain* and such a line a *bridge*. The stain cannot be removed by Stain removal because it is connected. We need to remove the bridge first.

(a) Example of predicted mask containing a bridge.

(b) The bridge and a few other connected stain pixels are highlighted by red colour. Removing the red pixels creates a new black stain.

Figure 3.11: Example of a bridge.

We can identify such bridges by walking over the black part of black and white components' boundary. If such walk is forced to go over the same pixel twice, the walk encountered a bridge.

Imagine we stand on the border black pixel and have a current direction vector pointing along the border to the right. Assume we have $\theta \in [-179°, 179°]$ representing the angle between the current direction and the direction to a neighbouring black pixel. If we ignore the pixel that leads us backwards, we have up to eight possible directions and therefore eight angles (including the direction right ahead with angle 0°). The relative left neighbours have positive angles, the relative right neighbours have negative angles.

Assume we start on the leftest border black pixel. Then, the walk is realised by following these rules in the specific order:

1. If you reach the right border of the prediction, quit.

2. Find a neighbouring not-yet visited black border pixel with minimum θ. Update your current direction to be heading towards the chosen neighbour. Move to the pixel, updating your current position.

3. If there is no such pixel, go back to the last visited pixel. This step is called backtracking.

If there is a bridge, then it is part of a detour. Step 2 ensures that we go over all detours because it prioritizes neighbouring not-yet visited black border pixels that are the rightest, relatively to the current direction. Assume a detour goes over a bridge. An example of such bridge is depicted in figure 3.11. When we go over the detour, we eventually stumble into a pixel we have already visited. This happens because there is no other way to return from the detour other than going through the narrowest part of the protrusion: the bridge. Step 3 then forces us to backtrack until there is another black pixel on the border that it has not visited yet. Such pixel is always before the bridge because all border pixels of the

connected stain have already been visited. Therefore, if there is any bridge, the algorithm backtracks over it.

Removing the backtrack pixels removes some of the connected stain's pixels and the bridge and it can create new stains. Therefore, after Bridge removal, we need to perform Stain removal once more. Figure 3.12 shows the example from figure 3.11 after removing the bridge and subsequent Stain removal.

Figure 3.12: Image from figure 3.11 after removing the bridge and the stain.

3.3.2 Problem detection

Sometimes we cannot fix the prediction. In such cases, we want to at least detect if there is a problem in it. We implemented three methods of Problem detection: Frequency analysis, Value analysis and Shape analysis. All three methods detect *problem intervals*: intervals in which a problem was found. In the end, the intersecting intervals found by all the methods are merged.

Frequency analysis The first approach assumes and utilises the periodicity of the masks. We can observe that the rods and the grid's teeth have similar distances between one another. We can detect lower peaks, which are the rods' middle parts and upper peaks, which are the grid's teeth. By joining all those peaks and sorting them by their width coordinate, we get *all peaks*.

The peaks can be computed from the function given by the average number of white pixels in each column. We filter out peaks whose height difference between the peak and its lowest contour line is too low. This difference is called *prominence*.

More formally, the average number of pixels in the column j is given by the following function:

$$\mu(j) = \frac{1}{H} \sum_{i=1}^{H} x_{i,j} \,, \tag{3.2}$$

where

- H is the height of the prediction,
- $x_{i,j}$ is the value of the pixel on the coordinates (i, j), where i is the row coordinate and j is the column coordinate or in other words i is the height coordinate and j is the width coordinate.

The upper peaks U are peaks of minimum prominence r found on the function given by equation 3.2, whose distance between one another is at least n. The lower peaks L are defined similarly, but they are found on function $-\mu(j)$. All peaks A are defined as

$$A = \{j \in L \cup U | \text{all j values are sorted}\}\,. \tag{3.3}$$

Let D be a sorted set of A's differentials. If we view A and D as functions of indices, then

$$D(i) := A(i+1) - A(i)\,. \tag{3.4}$$

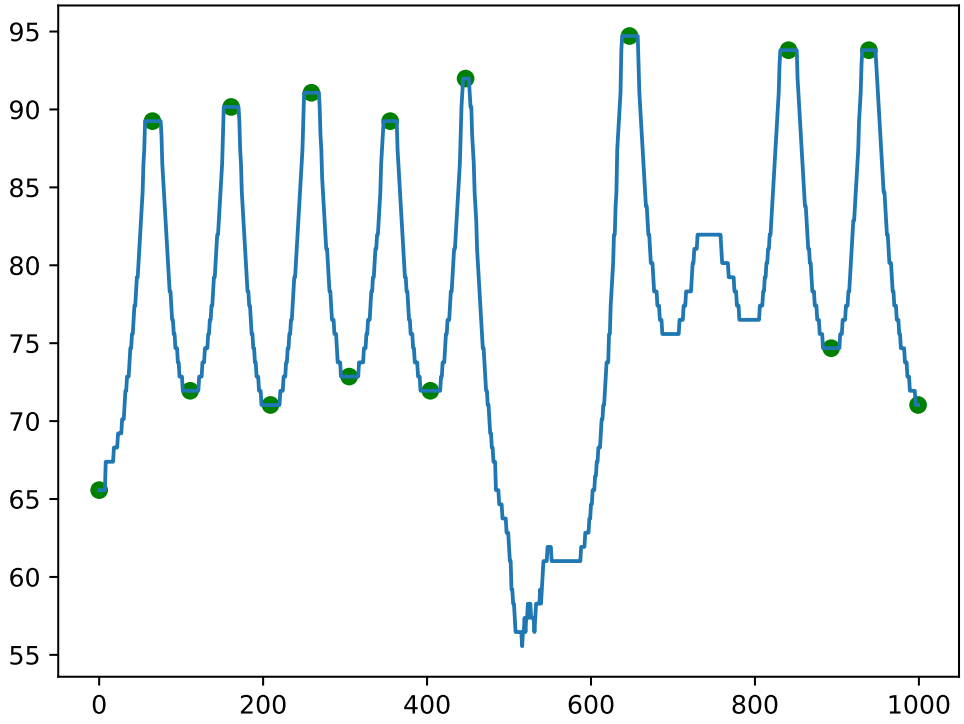
We expect the values of D to be similar, formally for all j

$$D(j) \approx \operatorname*{med}_i D(i)\,. \tag{3.5}$$

All j's, for which equation 3.5 does not hold, are problem points. Their complement are valid peaks. If we detect problem at the column j, we can compute nearest left and right valid peak and claim the problem is between them. This is useful because the problem point tends to indicate a larger problem in that area, rather than only one column being predicted wrongly.

However, there can be a peak missing in the middle because the fuel assembly has a special side with the groove, where the grid's tooth is missing. This can be seen in figure 1.1. Hence, we ignore problem points found in the area of the sixth and the seventh rod.

(a) Example of predicted mask. The seventh grid's tooth is smoothened out in the prediction. This type of an error can happen for example if the area is covered by debris.

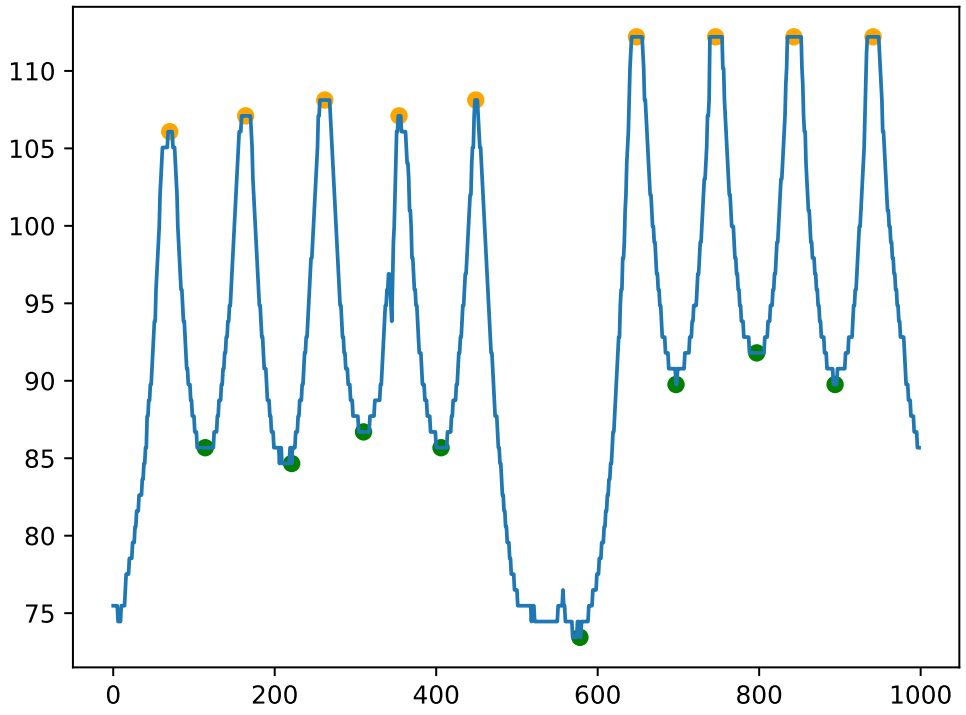


(b) The blue curve shows the average number of white pixels for each column, the green peaks are all valid peaks.

Figure 3.13: Frequency analysis performed on a prediction.

The weakness of this method is that there can be an issue that is not related to the peaks, as shown in figure 3.14. It also assumes that the prediction's shape visually resembles a grid. For example, if there are no white grid's teeth in the prediction, the method fails.

(a) A problematic protrusion is between the fourth and the fifth rod. Frequency analysis cannot detect this problem because it does not relate to peaks.



(b) The blue curve is the average number of white pixels for each columns, the orange peaks are upper peaks, the green peaks are lower peaks. In this image, the lower peak on the hasn't been filtered out, yet.

Figure 3.14: Example of problems that cannot be detected by Frequency analysis.

Value analysis Value analysis partially solves the main weakness of Frequency analysis. It computes the average number of white pixels for each column and detects outliers using these values directly. Then they are matched to their corresponding problem intervals using valid peaks from Frequency analysis.

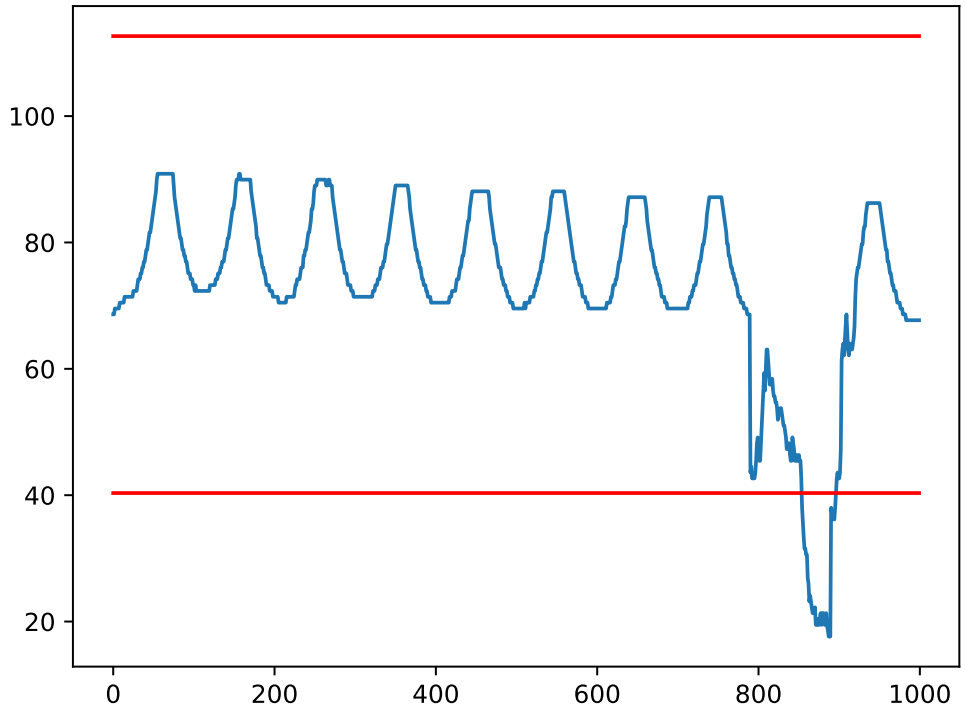
Formally, we expect the following formula to hold for all width coordinates:

$$\forall w_1 \in [1, W] : \underset{w_2 \in [1,W]}{\text{med}}(\mu(w_2)) \approx \mu(w_1)\,, \tag{3.6}$$

where W is the width of the prediction.

This method only detects big problems such as a large unexpected mass of black or white pixels. It fails if the prediction has a lot of such masses because it ruins the median value in equation 3.6.

(a) A prediction with a large mass in-between the peaks.



(b) The blue curve is a mean of white pixels for each column of the prediction, red lines demarcate valid values.

Figure 3.15: Visualised Value analysis on a flawed prediction.

Shape analysis The last method of Problem detection is Shape analysis. It utilises the following property given by the shape of the true masks: if we perform a border-walk (section 3.3.1) over the true masks, every step on average increases the width coordinate by 1. The same, however, cannot sometimes be said about the prediction. The walk on the border of the prediction can go over many detours, performing many steps while not increasing the width coordinate at all.

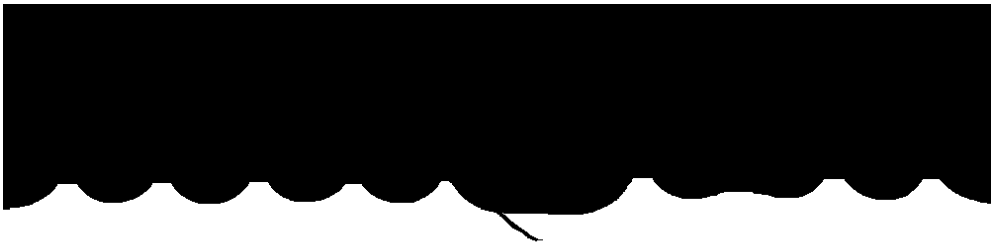
Formally, suppose we have a border-walk $e = ((h_1, w_1), ..., (h_k, w_k))$. If we take any slice of the length n, denoted as $e_{i:i+n} = ((h_i, w_i), ..., (h_{i+n}, w_{i+n}))$, then we expect the following formula to hold true:

$$\forall i \in [1, W - n] : w_{i+n} - w_i \approx n \,, \tag{3.7}$$

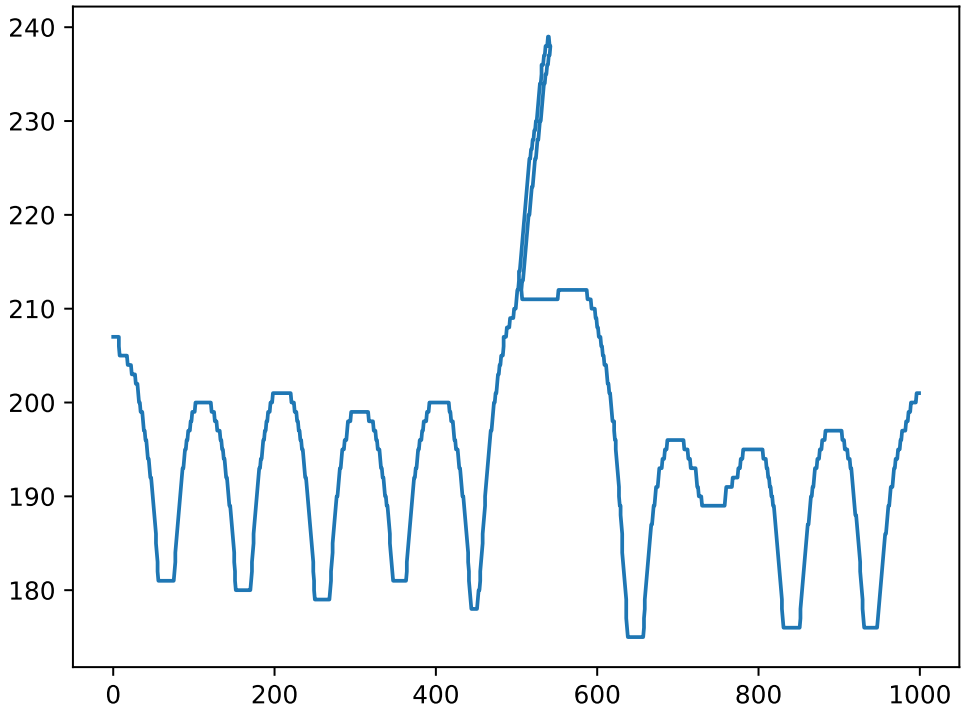
where W is the width of the prediction.

If equation 3.7 does not hold for some i, then we discovered a problem interval $[i, i + n]$.

The weak spot of this approach is that erroneous shapes that more remind standard shape of the grid, even though they may be faulty, are not detected by this approach.



(a) Prediction with an erroneous shape in the middle and one missing grid's tooth in the right area.



(b) The blue curve is coordinates of the border-walk. The problem in the middle part is detected because there are many steps that do not change the width coordinate but change the height coordinate. However, the error on the right is not detected because its shape is not anomalous. This can be detected by Frequency analysis.

Figure 3.16: Visualised Shape analysis on a flawed prediction.

3.4 Metrics

We categorise the metrics by the measured subject: whole image metric and crops metric. Whole image metric is measured on combined predictions as described in section 3.1.1, whereas crops metric is measured on the individual crops.

Unfortunately, even though we ultimately want to optimise the whole image metric, computing it is significantly slower than crops metrics due to our inability

to evaluate it quickly on the GPU. However, we discovered a strong Pearson's correlation [4] between whole image metrics and crops metrics. Hence, we only use the crops metrics during the training and the whole metrics are used during the evaluation.

Metric	Correlation
Mean IoU	0.994
Mean Absolute Error	0.975

Table 3.1: Pearson's correlation between crop and whole metrics. All correlations were found statistically significantly different than 0 with the significance level $\alpha = 0.05$.

To evaluate the model, we used three metrics: Mean IoU, Mean Absolute Error and Line Distance.

Mean IoU Mean IoU [11] computes, what percentage of the correctly predicted pixels overlap both true and predicted pixels. Formally, IoU is defined as

$$IoU = \frac{|T \cap P|}{|T \cup P|}, \tag{3.8}$$

where P is the set of predicted pixels and T is the set of true pixels, and Mean IoU is created by averaging IoU over all channels.

Mean Absolute Error Mean Absolute Error tells us, what percentage of pixels per image (or crop) our model predicted incorrectly.

Line Distance Line Distance metric computes the maximum absolute distance between two border-walks performed on true masks and predictions for each width coordinate. It indicates, how inaccurate the model is at its worst, or in other words, the size of the maximum uncertain area between the true and the predicted transition between grids and rods. It requires the prediction to be aliased and cleaned (section 3.3.1), which is making this metric even slower to compute than the other whole image metrics.

Formally, suppose we have a true walk (gold) $e_g = ((h_{1g}, w_{1g}), ..., (h_{kg}, w_{kg}))$ and a predicted walk $e_p = ((h_{1p}, w_{1p}), ..., (h_{kp}, w_{kp}))$. May $H_i(w)$ be the set of all height coordinates of the walk i for given width coordinate w, formally

$$H_i(w) = \{h | (h, w) \in e_i\}. \tag{3.9}$$

Then the Line Distance between two walks e_p and e_g is defined as

$$d_l(e_p, e_g) := \max_w \Big(\max_{h_p \in H_p(w), h_g \in H_g(w)} |h_p - h_g| \Big). \tag{3.10}$$

Line Distance highly penalises model's inconsistent performance. If the model utterly fails at prediction in one area in the image, the metric value is high no matter how accurate it is in other areas.

Even though we compute Line Distance metric as the maximum error for each sample individually, we average it over all samples. We think the average encapsulates model's performance better than the maximum because high Line Distance in one sample does not totally ruin the metric's value of all the samples while still highly influencing it. Furthermore, a metric computed as a mean allows us to perform hypothesis tests because their random variables are estimates of a mean.

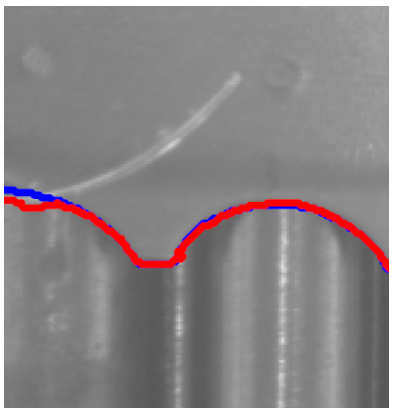


Figure 3.17: Line Distance. The predicted walk is shown as the red curve and the true walk as the blue curve. Line Distance computes the maximum (absolute) height distance of the predicted and the true walk. We show only part of the entire grid for the demonstration purposes, the metric is in fact computed on the whole prediction and mask.

4. Experiments

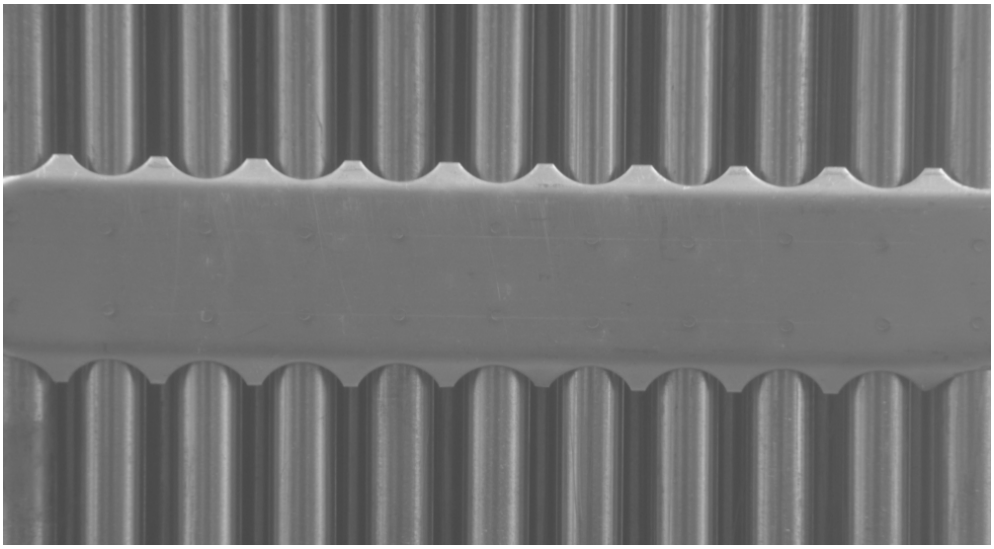
In this chapter, we initially describe our data split strategy into training, validation and testing groups. Then, we analyse relevant hyperparameters and their relation to model's performance. This gives us insight into their importance and possibly allows us to train better models much faster. Lastly, we attempt to find the best model and evaluate it.

When we perform hypothesis tests in this chapter, we set the significance level to $\alpha = 0.05$. To ensure correctness for a testing with multiple groups, we divide the level by the number of groups. When we say that a result is statistically significant, we make such claim for an already properly divided significance level. Furthermore, the correlation computed in this chapter is Pearson's correlation [4]. We denote H_0 as the null hypothesis and H_a as the alternative hypothesis. For purposes of this chapter, we also modify the terminology used to describe the datasets. Instead of the three datasets, we define five out of the original ones: Easy, Medium, Medium with debris, Hard and Hard with debris. More precisely, we denote the datasets with debris as the ones having images both with and without the debris. This division allows us to better understand the influence of the debris on model's performance.

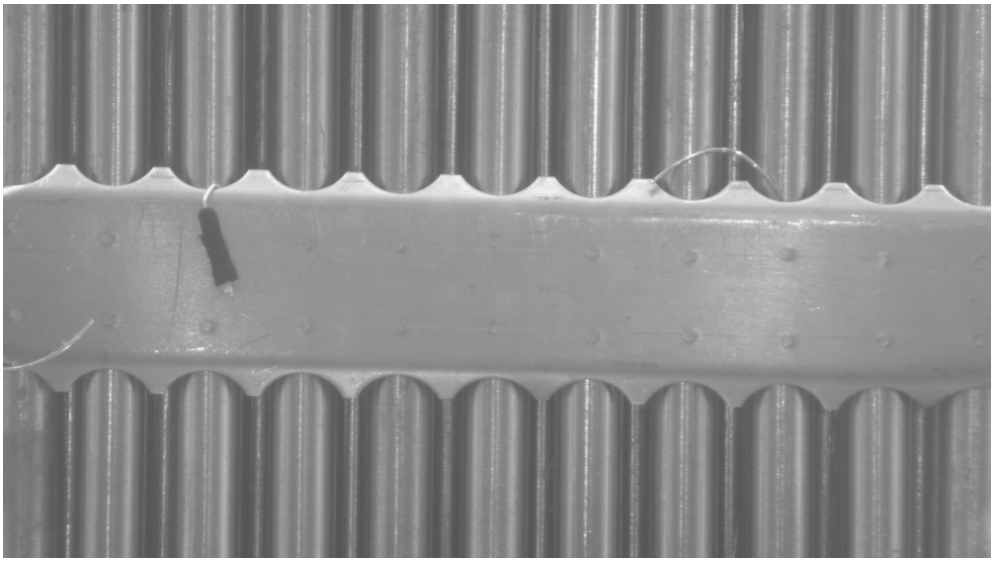
4.1 The splitting problem

To be able to select the best model and later evaluate it, we need to divide each dataset into three subsets: training, validation and test. This is usually done by randomly splitting the data in a desired ratio. If we choose this approach and do this with our input crops (small sub-images cut out of the whole images), it creates a bias because close crops can appear in different sets. Such crops have a very similar lighting, leading our model to have seemingly better performance.

To combat this, instead of splitting the crops, we decided to split the whole images. We split the photos without debris and their variations with the debris the same way. For example, if a grid photograph appears in the training set, then all of its versions with the debris also appear in the training set. This eliminates possible bias resulting from the same grid being both in the training and the testing data. We performed one deterministic split in which we attempted to cover most of the defects in the training sets while still leaving enough of them for the validation and the testing sets.



(a) Image of the grid.



(b) A variation of the image.

Figure 4.1: Photograph of a grid and its variation. The variant depicts the same grid, but it can have various debris added. The photograph and its variations appear in the same split to prevent bias.

4.2 Hyperparameter analysis

In this section, we analyse hyperparameters, trying to get knowledge to narrow-down the search for their optimal values.

4.2.1 Setup

During the experiments, we varied a subset of hyperparameters while leaving the others as reasonable constants.[1] We experimented only on Hard dataset with

[1]The reasonable constants were chosen by us before the training. Their exact values are listed in appendix D.

debris to reduce the number of training runs. We used the most difficult dataset in order to be able to see a significant difference in performance and also to be as close as possible to the real power plant's images. The best model for each training run was chosen as the one with the best performance metrics on the validation crops. We did not evaluate validation metrics on the whole images during the training due to its time complexity described in section 3.4. After choosing the best model for the run, its result performance was evaluated on the validation whole images.

The best model for each training run was selected as the one with the minimum Mean Absolute Error on the validation crops. We found Mean IoU to be inappropriate because its values are unstable for crop masks with only a small number of pixels of one of the classes.

To evaluate the selected model, we mainly used Cleaned Mean IoU (section 3.4). Mean IoU, unlike Mean Absolute Error, is not biased when the model overfits on one class. This property is useful because in our masks, more pixels can potentially be black than white (or reversely) depending on the particular whole image. We used primarily the cleaned metric to more accurately reflect performance of our entire algorithm (including the post-processing), rather than focusing only on performance of the model. For further analysis, we also computed Line Distance and Mean IoU. Line Distance allowed us to inspect the model's stability as stated in section 3.4 and Mean IoU improved our understanding of the model.

For our experiments, we set the following hard limits:

1. maximum allowed time limit: almost 24 hours,
2. maximum 300 epochs,
3. maximum allowed memory: 80 GB.

Additionally, if Mean Absolute Error of the validation crops had not improved over 80 epochs, we stopped the training. The number (80) was chosen this high to stabilise model's performance. Lastly, if the time limit was surpassed, we selected and evaluated the best model out of last 80 epochs. If memory was surpassed, we could not perform the training and ignored these hyperparameters.

Overlay, step and crop size We inspect performance for crop sizes 64, 128 and 256, and overlay sizes ranging from 20 up to 240.[2] Since the overlay describes the number of pixels shared by two subsequent crops, having different overlays leads to a different number of crops for different crop sizes. Therefore we analyse the relation between the overlay and model's performance for all crop sizes individually.

To further examine this phenomenon, we introduce a derived parameter called *step*, denoted as Δ. It represents a distance between two subsequent crops' starting coordinates and allows us to analyse the relation between overlay and crop size for all crop sizes at the same time. Formally

$$\Delta = s_c - o\,, \tag{4.1}$$

where

[2]Invalid combinations of crop sizes and overlay were skipped.

- s_c is the size of the crop,
- o is the overlay,
- Δ is the size of the step.

Furthermore, we try to find out, which crop size leads to better model's performance. We split our data points by the crop sizes into three categories and compare them.

We hypothesise that smaller step (or larger overlay for crop sizes individually) and larger crop sizes lead to better performance.

Border crops In section 3.2, we claim that we only use such crops for the training that have ratio of black pixels in [0.2, 0.8]. Since such crops are located only at the border of grids and rods, we call them a *border crops*. The main idea is following: if we correctly predict the transition between grids and rods, the cleaning (section 3.3.1) fixes the problematic grid-only and rod-only areas. Additionally, this approach solves a potential problem of imbalanced classes in the dataset (e.g. more black pixels than white) and saves us a lot of training time.

We hypothesise that border-only crops do not worsen the cleaned metrics but only Mean IoU metric.

Cutout We explore Cutout augmentation as a method to help the network to learn to ignore debris. We perform experiments for a crop of size 128 with cutout probability ranging from 0.1 up to 1 and cutout size from 16 up to 96. We use cutout only on the training crops, the validation crops are left unchanged. Using cutout on the validation data would have added an unwanted randomness to the selection of the best model due to cutout's stochastic nature.

We hypothesise that cutout helps the model up to a certain threshold probability and size, we do not expect any direct correlation showing that larger cutout probability and size leads to better performance.

Model We inspect, whether model's performance depends on its architecture and size. The size is regulated by two parameters:

1. Root's filters: this parameter regulates the model's width. It specifies the number of filters in the root convolutional layer.
2. Layer depth: specifies, how many blocks the encoder part of the network has.

We try U-Net [10] and ResU-Net [12] architectures, and root's filters varying from 8 up to 64 and layer depth from 2 up to 8. To be able to compare two different architectures, we also compute the number of parameters of the model.

We hypothesise that ResU-Net is better than U-Net and increasing the number of model's parameters up to a certain point improves its performance.

4.2.2 Results

Overlay and step For the crop sizes individually, we found a statistically significant correlation between overlay and Cleaned Mean IoU.

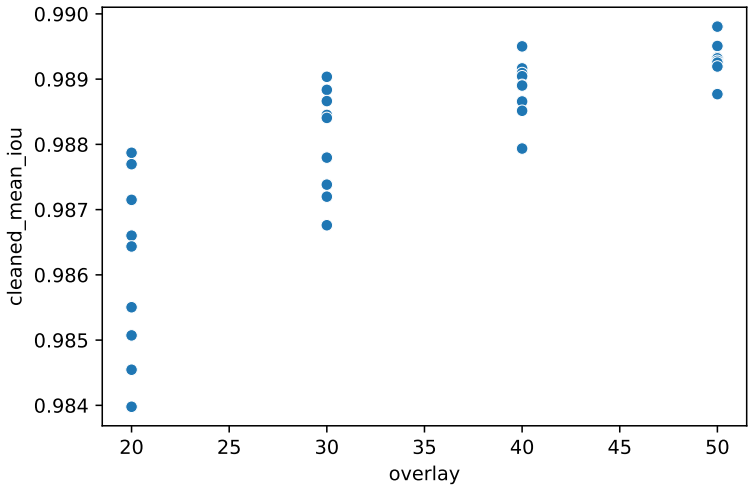


Figure 4.2: Scatter plot depicting dependency between overlay and Cleaned Mean IoU for crop size 64. Increasing overlay leads to an increase in Cleaned Mean IoU.

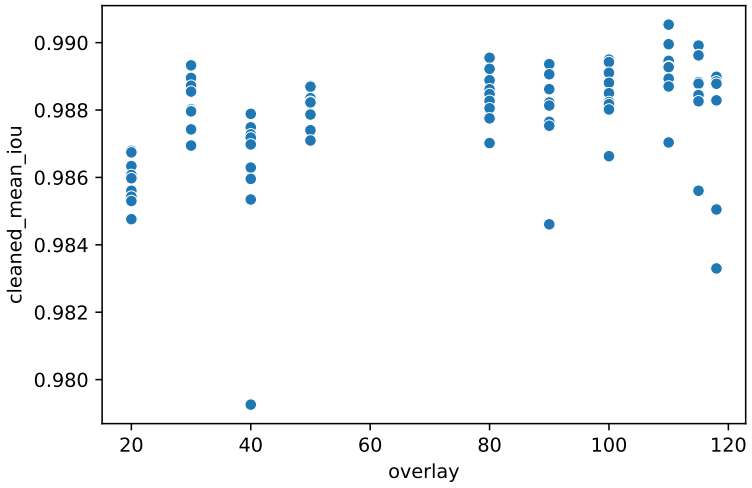


Figure 4.3: Scatter plot depicting dependency between overlay and Cleaned Mean IoU for crop size 128. Increasing overlay leads to an increase in Cleaned Mean IoU, though not as strongly as in the case of crop size 64.

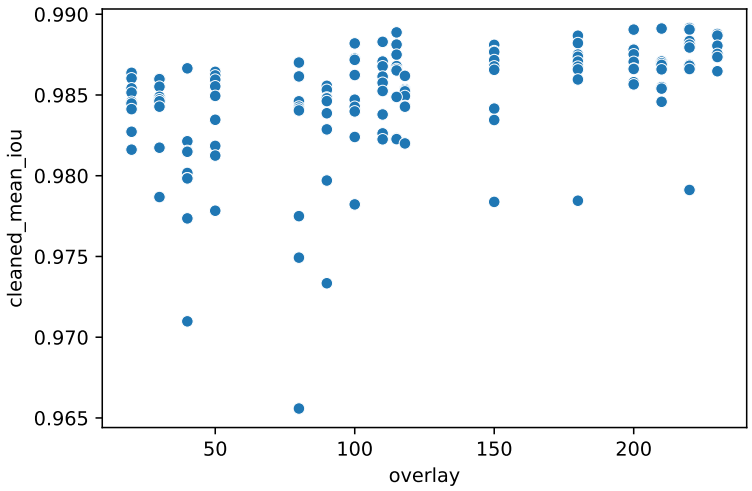


Figure 4.4: Scatter plot depicting dependency between overlay and Cleaned Mean IoU for crop size 256. There seems to be a correlation between overlay and Cleaned Mean IoU. Lower values of overlay seem to have bigger variability.

According to section 4.2.1, step enables us to aggregate results of overlays of different crop sizes into one value. Hence, we analysed dependency between step and Clean Mean IoU for all crops at the same time.

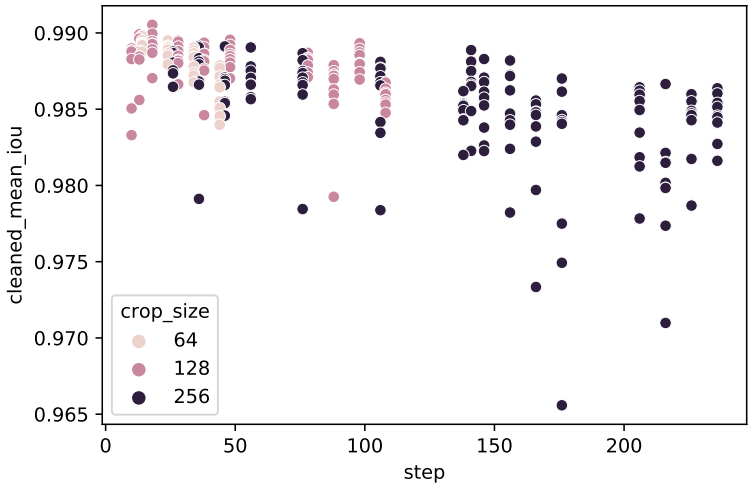


Figure 4.5: Scatter plot of step and Cleaned Mean IoU. Smaller step generally leads to better Cleaned Mean IoU. Additionally, larger step seems to lead to bigger performance variance.

We found a statistically significant negative correlation between the size of the step and Cleaned Mean IoU. These results make an intuitive sense, since the smaller step (or larger overlay) leads to more crops and therefore more training data.

Crop size	Correlation r	p-value
64	-0.807	0.000
128	-0.55	0.000
256	-0.478	0.000
all	-0.638	0.000

Table 4.1: Correlation r between step and Cleaned Mean IoU and its p-value for $H_0 : r = 0, H_a : r \neq 0$. Note that for crop sizes individually, the correlation between step and Cleaned Mean IoU is equal to the negative correlation between overlay and Cleaned Mean IoU.

Crop size We examine our hypothesis that larger crop leads to better performance. Unfortunately, during the experiments, crop size 256 and overlays 230 and 240, or steps 26 and 16 respectively, surpassed our memory limit. Since smaller step leads to better performance, this puts crop size 256 into a disadvantage. Furthermore, larger crop size can have steps of a larger size, e.g. crop size 128 can have step 100, whereas crop size 64 cannot. To prevent potential bias resulting from this, we only compare crop sizes 64 and 128 in more detail and we only take steps in the interval [10, 50]. The data shows that there is no evidence of a difference between Cleaned Mean IoU of crops size 64 and 128.

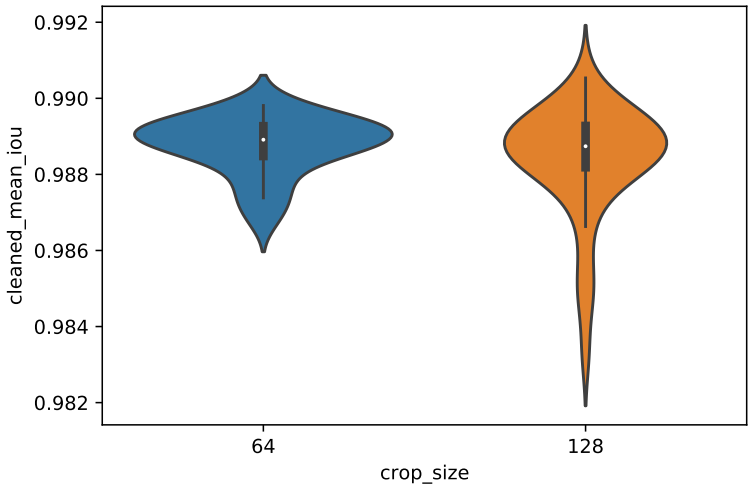


Figure 4.6: Violin plot of Cleaned Mean IoU for crop size 64 and 128. The two groups seem to have very similar performance.

Border crops Next, we inspect, whether taking border-only crops hinders performance.

Figures 4.7 and 4.8 indicate that taking border-only crops does not affect Cleaned Mean IoU. The results are similar for Line Distance.

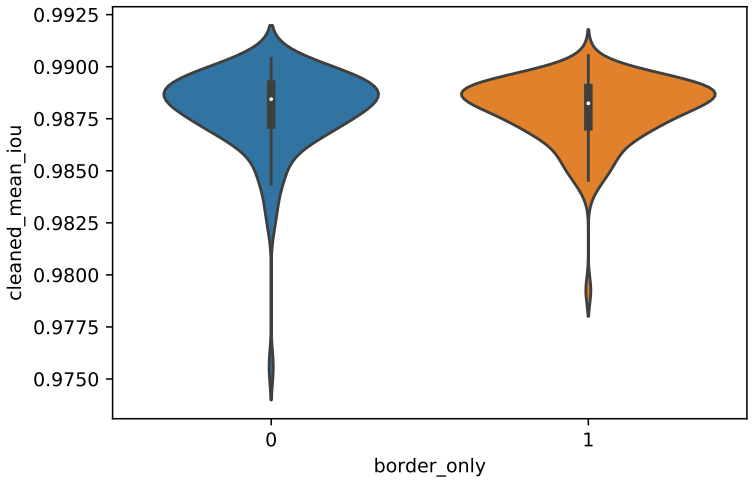
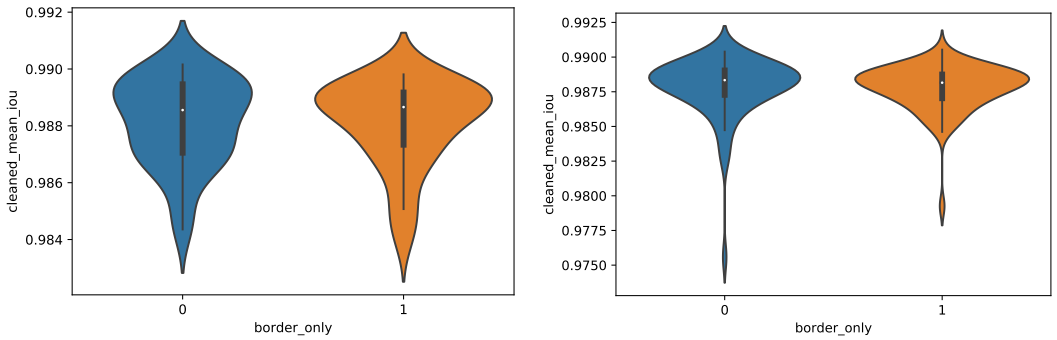


Figure 4.7: Violin plot of Cleaned Mean IoU for all crop sizes for two groups: all crops (0) and border-only crops (1). The two groups seem to have similar performance.



(a) Crop size 64.

(b) Crop size 128.

Figure 4.8: Violin plot of Cleaned Mean IoU for multiple crop sizes for two groups: all crops (0) and border-only crops (1). The two groups seem to have similar performance.

However, if we look at Mean IoU in figure 4.9, performance of model trained on all crops appears to be significantly better.

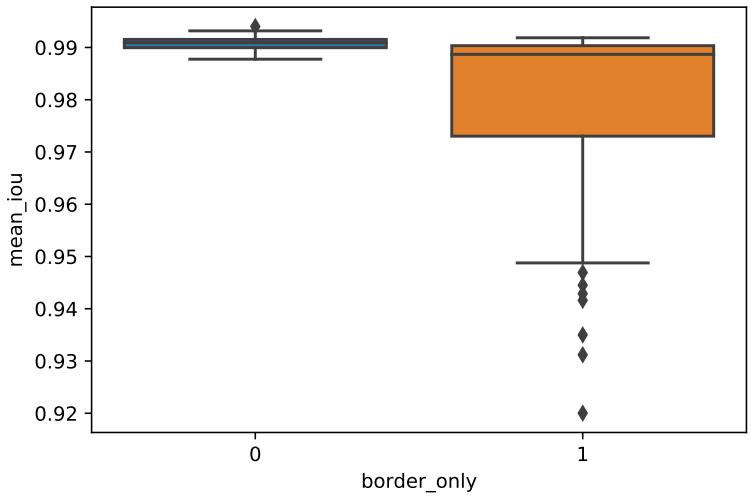
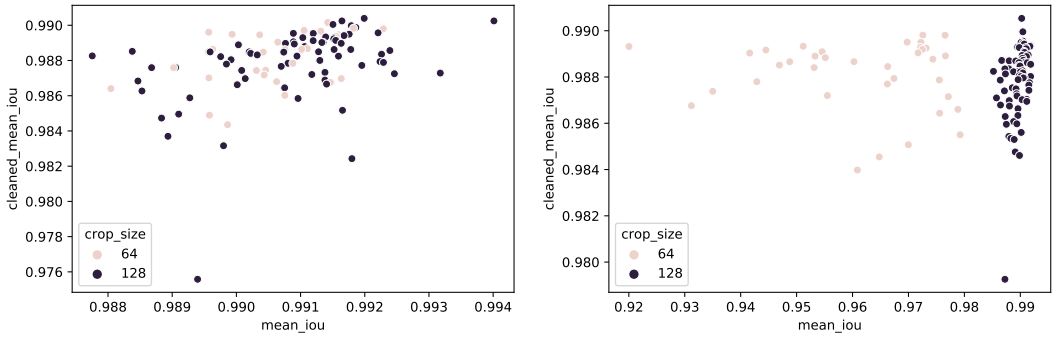


Figure 4.9: Comparison of all (0) and border-only crops' (1) Mean IoU. Mean IoU of all crops (0) is significantly better and has significantly smaller variability.

To explore this further, we analysed relation between Mean IoU and Cleaned Mean IoU. Figure 4.10 suggests that models trained on all and border-only crops have different relation between the two metrics.



(a) All. (b) Border-only.

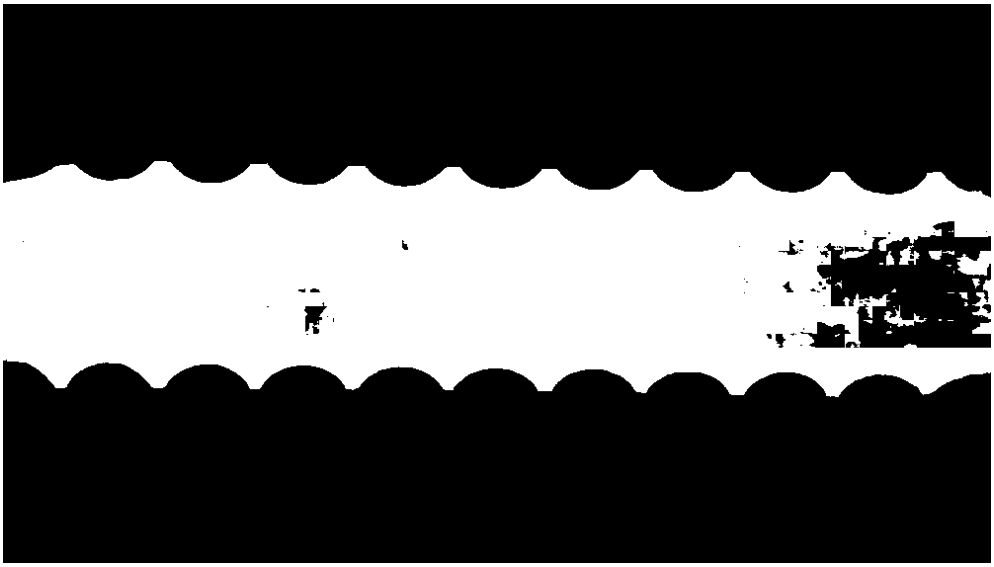
Figure 4.10: Scatter plot depicting Mean IoU and Cleaned Mean IoU values for all (a) and border-only (b) crops.

To inspect this in more detail, we computed correlations between Mean IoU and Cleaned Mean IoU for relevant hyperparameters. The results can be found in table 4.2.

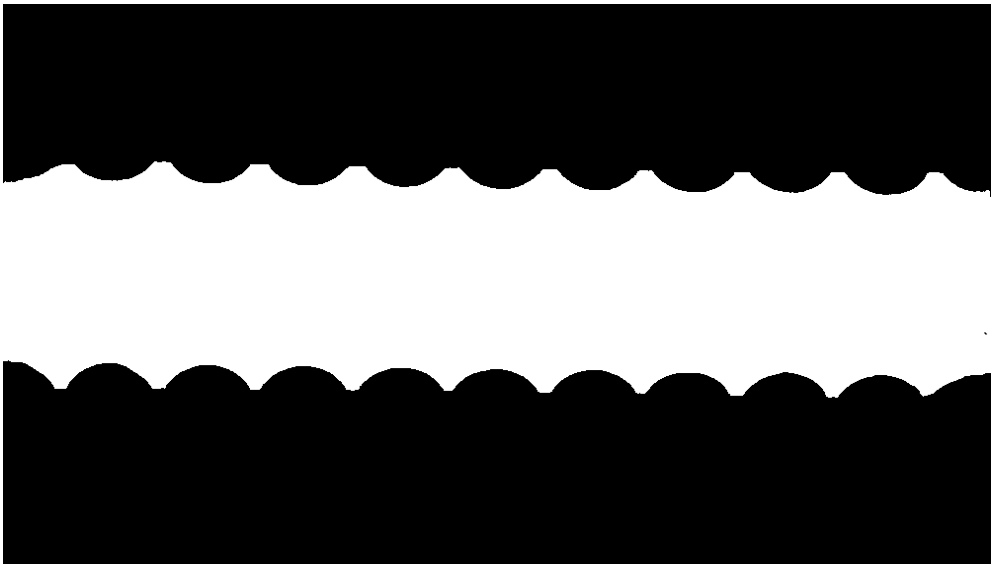
Crop size	**Correlation** r	p**-value**
Non border-only:		
64	0.437	0.016
128	0.373	0.001
Border-only:		
64	-0.084	0.624
128	0.413	0.0003

Table 4.2: Correlations r between Mean IoU and Cleaned Mean IoU with its p-values for $H_0 : r = 0, H_a : r \neq 0$.

We suspect that the model trained on border-only crops is unable to accurately predict the grid-only and the rod-only part of the data. We can see this in figure 4.11. Models trained on crops with lower sizes are especially sensitive to this effect because their border-only training crops do not reach far to the grid-only or rod-only area during the training. This phenomenon does not influence the cleaned metrics because the cleaning polishes this kind of an error.



(a) Border-only.



(b) All.

Figure 4.11: Non-cleaned aliased prediction of model that has been trained on border-only images and model that has been trained on all images.

Cutout Surprisingly, cutout probability does not seem to affect Cleaned Mean IoU (see figure 4.12), it only affects Line Distance. This hints that it reduces the maximum error, making the prediction more reliable, but it also seems to distribute the error across the whole image because Cleaned Mean IoU is the same.

Figure 4.13 suggests that some form of cutout helps the model to have smaller Line Distance. We found that models with cutout probability larger than zero are significantly better than models with no cutout.

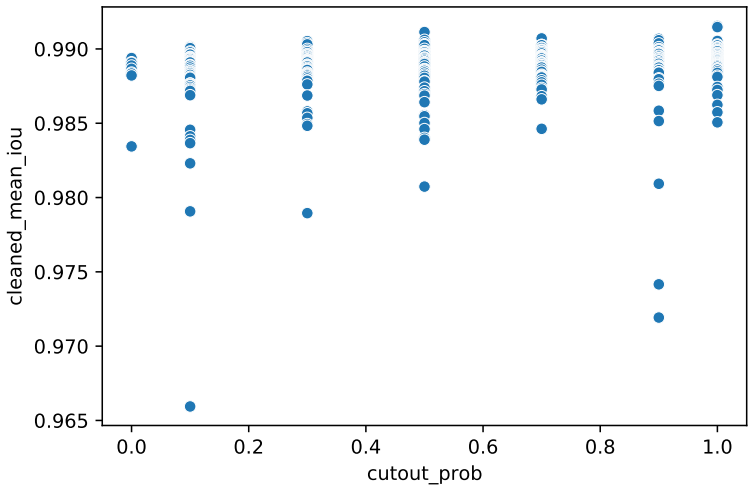
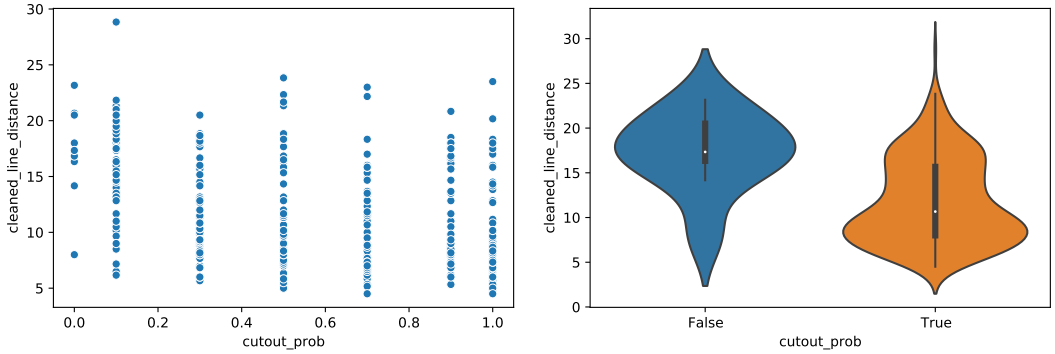


Figure 4.12: Scatter plot between cutout probability and Cleaned Mean IoU.

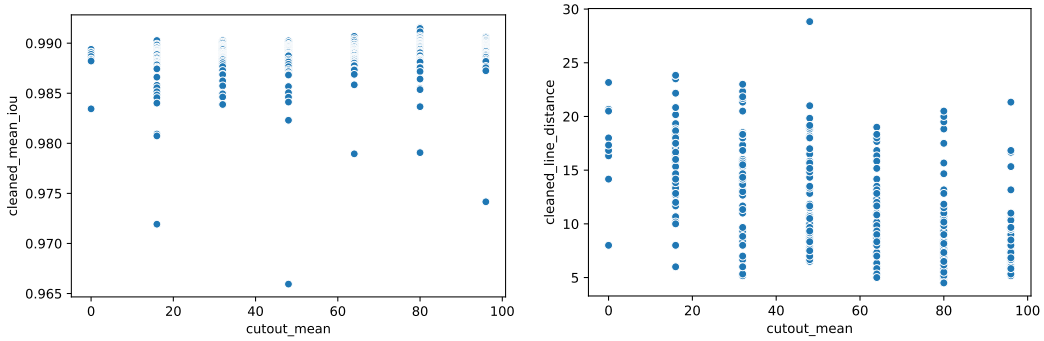


(a) Scatter plot of cutout probability and Line Distance.

(b) Violin plot of Line Distance for two groups: cutout probability equal to zero (False) and cutout probability larger than zero (True).

Figure 4.13: Plots of cutout probability and Line Distance.

Furthermore, we explored the size of the cutout. Figure 4.14b suggests larger cutout size leads to smaller Line Distance.



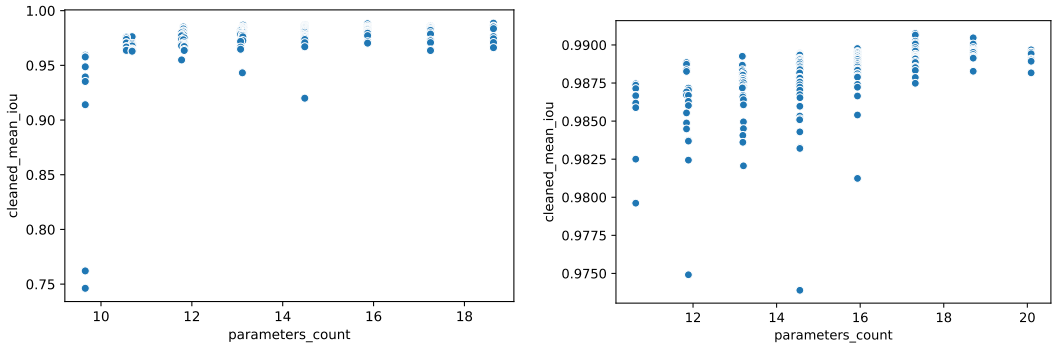
(a) Scatter plot of cutout size and Cleaned Mean IoU.

(b) Scatter plot of cutout size and Line Distance.

Figure 4.14: Dependency between cutout size and model's performance.

Even cutout probability 1 and cutout mean 96 does not hinder performance at all. This is surprising because it covers up to 75% of the crop's area in all training inputs. In fact, it is one of the best performing configurations.

Model First, we analyse dependency between the number of parameters and Cleaned Mean IoU. We examine the two architectures separately. In figure 4.15, we can see that increasing the number of parameters generally leads to better performance of the model.



(a) U-Net. (b) ResU-Net.

Figure 4.15: Scatter plots depicting dependency of the number of model's parameters and Cleaned Mean IoU. The number of model's parameters is shown on a logarithmic scale.

This can be further supported by its correlations in table 4.3.

Model architecture	**Correlation** r	p**-value**
U-Net	0.368	0.000
ResU-Net	0.532	0.000
All	0.359	0.000

Table 4.3: Correlations r between the number of parameters and Cleaned Mean IoU with its p-values for $H_0 : r = 0, H_a : r \neq 0$.

Figure 4.16 seems to suggest that ResU-Net has better performance than U-Net. To be able to more formally compare the two architectures, we need to make sure that we compare them on roughly the same number of parameters. Therefore, we remove the leftest points of U-Net and the rightest points of ResU-Net from figure 4.16. We can look at 4.17 to better visualise this comparison. We found ResU-Net to have statistically significantly higher Cleaned Mean IoU than U-Net.

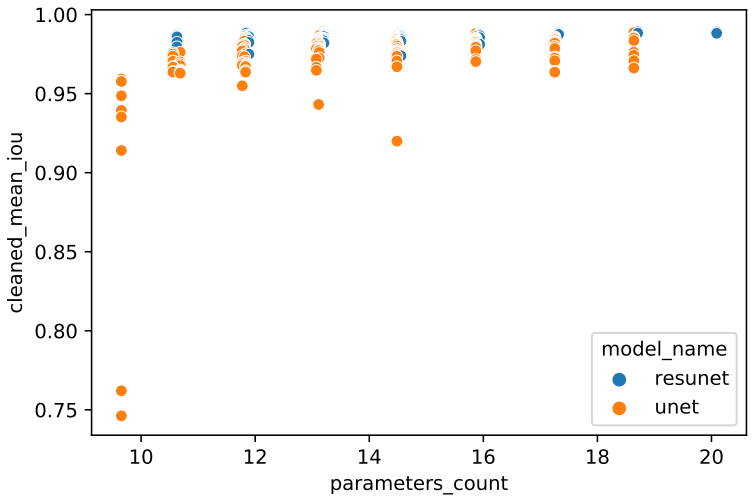


Figure 4.16: Scatter plot of the number of model's parameters and Cleaned Mean IoU. The number of model's parameters is shown on a logarithmic scale.

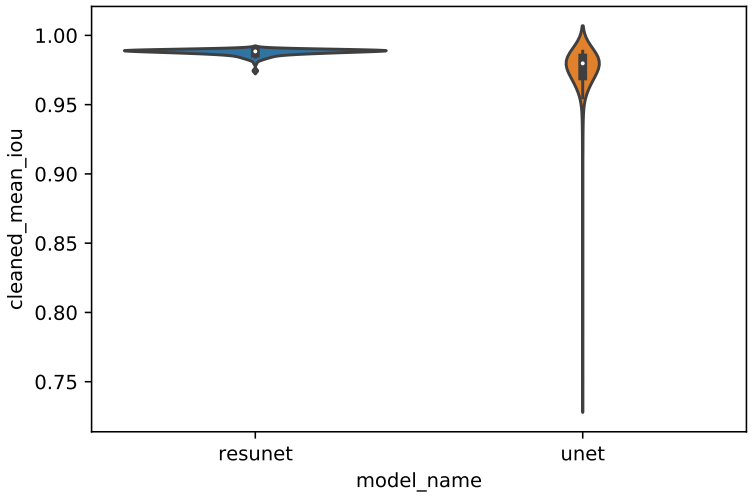
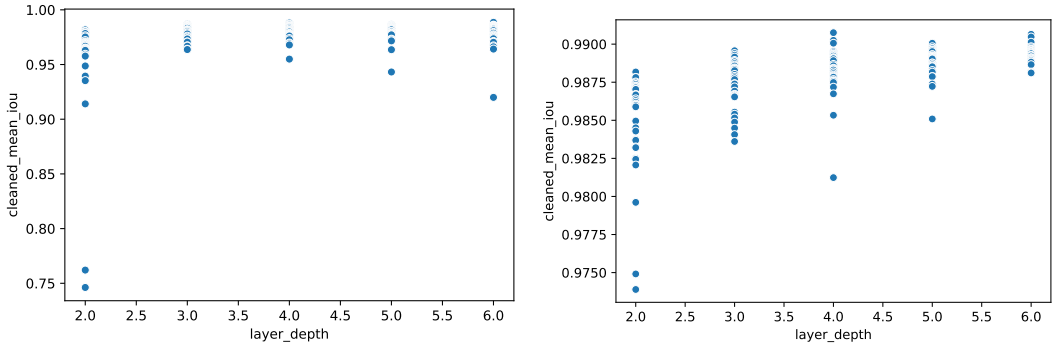


Figure 4.17: Violin plot demonstrating difference in performance between the two architectures.

Furthermore, we inspect the architecture's behaviour in relation to layer depth parameter. In case of U-Net, there seems to be much weaker correlation between the number of layers and Cleaned Mean IoU than in case of ResU-Net. This supports the basic idea of residual networks: greater network's depth does not lead to a performance degradation because the model can mimic simpler models by using the residual connections.



(a) U-Net. (b) ResU-Net.

Figure 4.18: Scatter plot of layer depth and Cleaned Mean IoU for the two explored architectures.

4.3 Best model search

In this section, we attempt to find the best model for each dataset and evaluate it.

4.3.1 Setup

We use the same hard limits and selection of the best model as in the hyperparameter analysis experiments. We train the models for all the datasets separately.

We use our knowledge obtained from the hyperparameter analysis and reduce the number of necessary training run. We fixate the size of the crop to 128, since crop size 64 does not offer any performance improvements and we also believe that size 64 is insufficient for handling debris of larger size. We set overlay as 110 and use border-only crops to fit into our time limit. We employ only ResU-Net model because it proved to be better. We vary multiple cutout and model hyperparameters. All hyperparameter values for Best model search can be seen in appendix E.

To select the best model, we combine the following three metrics: Cleaned Mean IoU, Line Distance and Mean IoU. Cleaned Mean IoU expresses well overall performance of our algorithm. Line Distance highly penalises the algorithm's inconsistencies, as described in section 3.4. Finally, Mean IoU notices improvements in model's predictions, even though they are smoothed by the cleaning process and therefore are not reflected in the cleaned metrics.

To be able to aggregate the three metrics, we need them to have comparable scales. Since Line Distance can have values from $[0, H]$, where H is the height of the whole image, but the other two metrics take values from $[0, 1]$, we need to rescale them. To achieve that, we decided to transform all metrics into their z-scores and combine them by their weighted average.

$$score = w_{\mathrm{cm}} m_{\mathrm{cm}} - w_{\mathrm{l}} m_{\mathrm{l}} + w_{\mathrm{m}} m_{\mathrm{m}} \,, \tag{4.2}$$

where

- m_{cm} is the z-score of Cleaned Mean IoU metric and w_{cm} is its weight,

- m_{l} is the z-score of Line Distance metric and w_{l} is its weight,
- m_{m} is the z-score of Mean IoU metric and w_{m} is its weight, and
- the weights sum up to one.

The weights denote the importance of the given metric. Using the reasoning above, we formalise the weights by the following formula:

$$w_{\mathrm{cm}} \gg w_{\mathrm{l}} \gg w_{\mathrm{m}} \,. \tag{4.3}$$

To reflect the fact that we minimise the Line Distance (not like the other two metrics), we use a negative value for its weight. Furthermore, we compute the z-scores separately for each dataset because the datasets separately seem to form a normal distribution.

4.3.2 Results

We found the best models for each dataset separately and evaluated them on the training, validation and test sets. The results are shown in table 4.4.

Dataset	Cleaned Mean IoU	Mean IoU	Line Distance
Training dataset:			
Easy	0.9997	0.9997	2.2
Medium	0.9991	0.9995	2.2
Medium + debris	0.9956	0.9964	3.15
Hard	0.9962	0.9978	3.4
Hard + debris	0.9962	0.9969	2.93
Validation dataset:			
Easy	0.9957	0.9973	2.5
Medium	0.9961	0.9978	3
Medium + debris	0.9952	0.9919	3.25
Hard	0.9908	0.9944	4
Hard + debris	0.9906	0.9924	5
Test dataset:			
Easy	0.9949	0.9969	2.5
Medium	0.9959	0.9971	2
Medium + debris	0.9936	0.9950	6.625
Hard	0.9911	0.9951	8.5
Hard + debris	0.9886	0.9939	7.83

Table 4.4: Table showing performance of the best models on the training, validation and test sets. A model that has Line Distance approximately 3 and Cleaned Mean IoU 0.9960 has perfect performance because the masks were labelled by a human hand (and therefore are not 100% accurate) and the pixels on the border of the grid and the rods are slightly blurred, making the true border inaccurate.

Even though the Easy and Medium dataset models have near perfect test performance, the same does not hold for the Hard dataset model. Its error is located in a seemingly easily segmentable part.

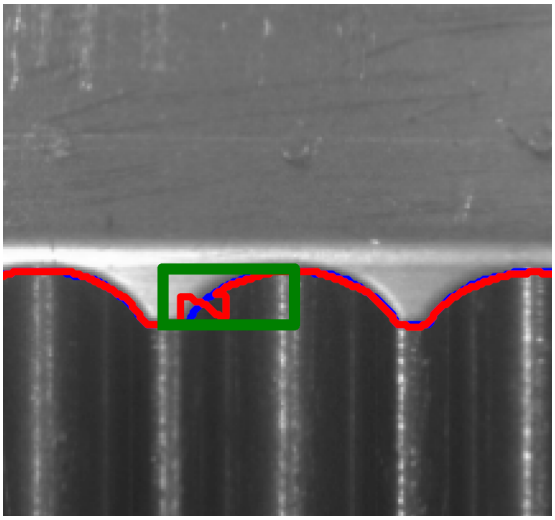


Figure 4.19: A problem in a prediction of the Hard dataset's model. The prediction is denoted as the red curve, the true mask as the blue curve. The problem is surrounded by a green bounding box. Easy and Medium models do not make the same mistake in that area.

Interestingly, the Hard with debris model predicted this area perfectly. We hypothesise that this is due to more training data with more variability in the lighting.

Unfortunately, the Hard with debris model made mistakes in a different area. We can see it in figure 4.20. There are two problems: the first is located directly on the silicone and the second is right from it. Both, however, relate to the silicone because even the second one has it in its prediction context (see figure 4.21). We believe that this inaccuracy is caused by the fact that there is no silicone or a similarly looking debris in the training data and therefore it confuses the network. Note that the Medium with debris model also made a very similar mistake.

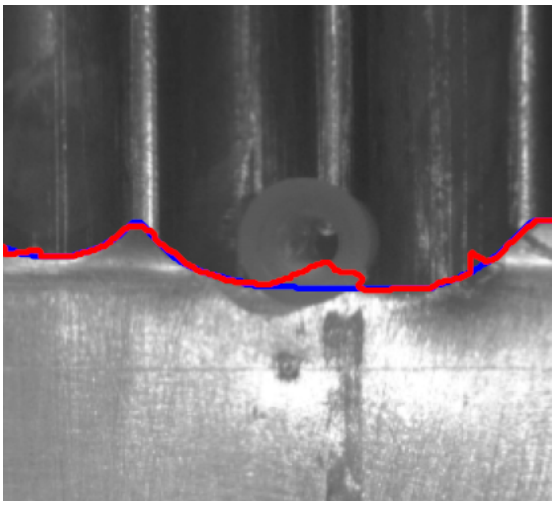


Figure 4.20: A problematic input for the Hard + debris dataset's model. The prediction is denoted as the red curve, the true mask as the blue curve. The problem was not recognized by our Problem detection.

Figure 4.21: A problematic input for the Hard + debris dataset's model. The image is made of three sub-images: the left is the input, the middle is the expected mask and the right is the prediction.

5. Discussion and future work

This chapter discusses our results, outlines the future usage of our algorithm and suggests how to further improve it.

5.1 Comparison to previous work

We compare our segmentation to Knotek's segmentation. Unfortunately, his algorithm only works on Medium dataset for the side with the groove. Furthermore, he expects the grid to have the groove between the fifth and the sixth rod, even though our images have the groove between the sixth and the seventh rod. To remove this difference, we flip our data along the vertical axis. This operation does not hinder our model's performance due to the cropping and mirroring data augmentation.

We compare our segmentation to Knotek's on three grids of the grooved side. In each case, our model outperformed its predecessor significantly. Note that the first two images appear in our model's training set, only the third image is from the test set.

Image name	**Knotek's**	**Ours**
Cleaned Mean IoU:		
1	0.9481	0.9954
2	0.9256	0.9969
3	0.9475	0.9940
Line Distance:		
1	20	3
2	30	3
3	17	6

Table 5.1: Table showing performance comparison of our method to Knotek's method. Note that the first two inputs are in the training dataset of our model.

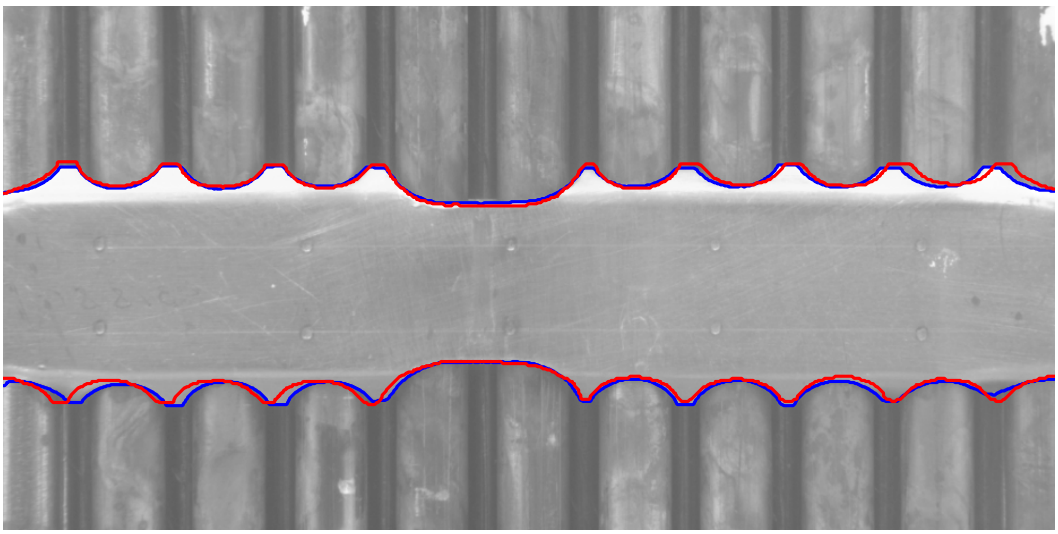


Figure 5.1: Comparison of Knotek's and our prediction on the third sample. The red curve is the predicted boundary between grids and rods of Knotek's algorithm, the blue curve is ours.

5.2 Integration to the framework

Our method remains to be tried on the nuclear power plant data. The nuclear power plant data can differ in the following ways:

1. There are more types of fuel assemblies.
2. The lighting conditions are various. Some images have very poor illumination, but later captured images have it close to our Hard dataset.

Our hypothesis is that on images with lighting of a similar quality, our models could perform very well after retraining with modified hyperparameters. Our algorithm could help to detect defects. Currently, the defect detection in the grid part is done by finding a sharp uncontinuous light transition. The defect could then be highlighted to help the operator to speed up his work. The problem previously lied in finding the grid accurately. Our thesis could solve that part.

5.3 Future work

Even though we reached very good results in the experiments, we realise a few things could be further improved.

5.3.1 Model selection

We selected the best model as the one with the lowest Mean Absolute Error in 300 epochs on the validation crops. If we had not improved over 80 epochs, we finished the training and selected the best model.

We can see from the metric curves in figure 5.2 that this can lead to selecting a high peak with big gaps around it. Such behaviour can cause unstable model's test performance.

We propose to solve this by selecting the best model from the smoothed metric over last n values, where higher n leads to prioritizing more stable model.

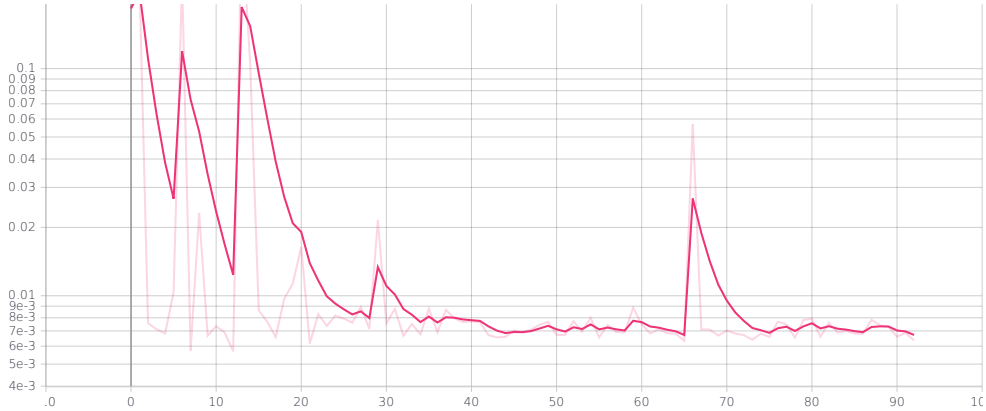


Figure 5.2: Mean Absolute Error for the validation data on the logarithmic y scale. The unsmoothed curve is the one with the lower opacity. The training ended preemptively because the lowest value on the unsmoothed curve was found very early.

5.3.2 Post-processing improvements

Although we implemented a basic post-processing, it does not cover all the problems that could arise. In our opinion, the most promising way to continue this effort is to improve Shape analysis.

The border-walk gives us a lot of insight. We can transform it from list of coordinates to list of angles and use it to create e.g. if-then rules to find invalid shapes or a machine learning model. This method could be much more sensitive than the current approach.

Conclusion

Our goal was to segment out the spacer grid of the nuclear fuel assembly. To achieve that, we prepared new datasets covering typical problems of the fuel.

To perform the segmentation, we used a deep neural network. We increased its performance by data augmentation techniques. We used two techniques suitable for segmentation: Deterministic cropping and Cutout. By Deterministic cropping, we increased the number of data substantially by splitting every image into sub-images. We introduced a new approach of combining the sub-images' predictions. Cutout augmentation covered part of the input image and this way increased network's robustness to debris.

We employed post-processing methods to increase the reliability of our algorithm: Prediction cleaning and Problem detection. The former polished the network's output and the latter wrapped anomalies into bounding boxes.

To measure our algorithm's performance, we proposed Line Distance metric, computing the size of the maximum uncertain area between the actual and the predicted transition between grids and rods. This metric is crucial to minimise to enable an automated evaluation.

In the experiments, we analysed the hyperparameters and verified our augmentation's effectiveness. Then, we found and evaluated the best model for each dataset. We reached very accurate segmentations and outperformed Knotek's algorithm by having three times lower Line Distance. Finally, we outlined, how this thesis will be used on the real power plant data.

Acknowledgements

I would like to express my sincere gratitude to my supervisor RNDr. Jan Blažek, Ph.D. for his guidance and interest.

Special thanks go to Ing. Marcin Kopeć and Ing. Ondřej Pašta, who helped me immensily in obtaining data, especially in these difficult times.

I gratefully acknowledge Profinit EU s.r.o. and Centrum Výzkumu Řež for allowing me to create this thesis.

Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

Bibliography

[1] Philippe Cattin. Image restoration: Introduction to signal and image processing, 2013.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation, 2018.

[3] Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout, 2017.

[4] David Freedman, Robert Pisani, and Roger Purves. Statistics (international student edition). *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*, 2007.

[5] Debesh Jha, Pia H. Smedsrud, Michael A. Riegler, Dag Johansen, Thomas de Lange, Pal Halvorsen, and Havard D. Johansen. Resunet++: An advanced architecture for medical image segmentation, 2019.

[6] Noel E. O'Connor Kevin McGuinness. Interactive segmentation evaluation: User-driven evaluation and an approach to automation, 2009. URL `http://web.archive.org/web/20110827180159`.

[7] Jaroslav Knotek. Automation of nuclear fuel visual inspection. Master's thesis, Charles University: Faculty of Mathematics and Physics, May 2020.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[9] Martina Malá. Seven years of inspections on tvsa-t fuel assemblies at temelín npp, 2017.

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

[11] Tanimoto TT. An elementary mathematical theory of classification and prediction, 1958.

[12] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net, 2017.

Appendices

A Data screening protocol

All three datasets are created by the following procedure:

1. Position the fuel assembly in front of the camera from one of the six sides.
2. Point the camera at the highest grid, move it closer or further from the fuel assembly.
3. Sharpen the image and take the photo.
4. Move to the lower grid and repeat the previous step. Repeat this until all grids of this side have been screened.
5. Turn the fuel assembly to another side and continue with step 2. Repeat this until all sides have been screened.

B Manual data pre-processing

Although our input was specified as a camera shot of the relevant side of the spacer grid, in reality, the camera usually depicts its surroundings, too.

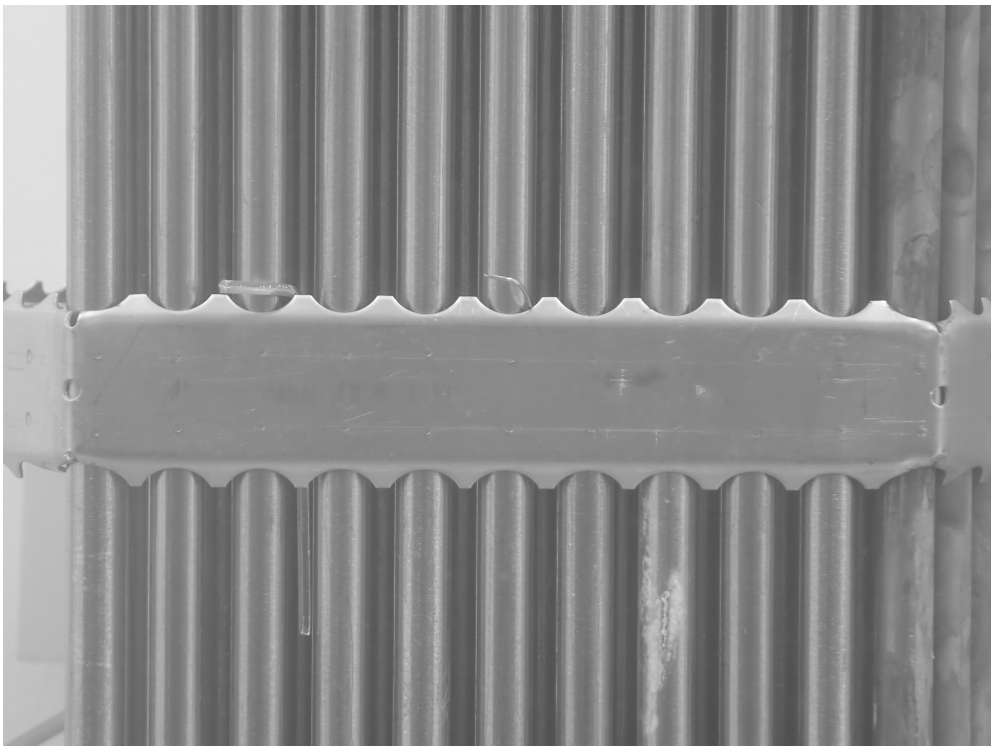


Figure B.1: The original input before manual pre-processing.

We used manual pre-processing to select the relevant part. We could have used the algorithm implemented by Knotek [7], but this approach saved us development time.

After cropping, we manually rescaled the image to have a width of 1000 pixels. This particular number of pixels was selected using the following reasoning:

1. The tool we used to create grid's mask [6] has a performance limitation given by the tool's algorithm and our hardware. If too large mask is submitted, then the tool gets stuck during the evaluation and the process of creating mask becomes even more time-consuming.

2. The camera used in the nuclear power plant can have significantly lower resolution than cameras used in the laboratory.

3. We want to fixate the size of the spacer grid in the image because it helps the neural network.

C Cameras parameters

The table below lists cameras that were used during the screening.

	Canon EOS D500	**Olympus IM005**
Product type	SLR camera	Digital camera
Image resolution	Up to 4752 x 3168	Up to 4000 x 3000
Lens	17 to 55 mm	4.5 to 18.00 mm
Shutter speed	30 to 1/4000 sec.	4 to 1/2000 sec.
Waterproof	no	yes

Table C.1: Cameras. We use Canon EFS 17-55mm f/2.8 IS USM lens in the Canon camera.

D Default experiment hyperparameters

In order to be able to analyse only a subset of the hyperparameters, we need to make others constant, since we do not have enough computing power to be able to perform the full grid search.

Hyperparameter	Value
Crop size	128
Overlay	110
Border-only crops	Yes
Cutout probability	0
Cutout size	0
Model name	ResU-Net
Root's filters [1]	32
Layer depth [2]	4
Learning rate	0.001
Beta 1 [3]	0.9
Beta 2 [4]	0.999
Rotation probability	0.3
Zoom probability	0.15
Noise probability	0.15
Seed	70, 700, 7000, 8, 18, 180, 42, 142, 1142

Table D.2: Default hyperparameter values for the experiments.

E Best model search hyperparameters

The table below specifies the tested hyperparameters in our search for the best model for each dataset. Many hyperparameters are constant to reduce the size of the search.

[1]The number of filters in the root convolutional layer.

[2]The number of blocks in the encoding part of the neural network. The number of blocks in the decoding part is the same.

[3]Parameter of Adam optimizer.

[4]Parameter of Adam optimizer.

Hyperparameter	Value
Crop size	128
Overlay	110
Border-only crops	Yes
Cutout probability	0.5, 0.9
Cutout size	0, 48, 80
Model name	ResU-Net
Root's filters	8, 16, 32
Layer depth	4, 5, 7
Learning rate	0.001
Beta 1	0.9
Beta 2	0.999
Rotation probability	0.3
Zoom probability	0.15
Noise probability	0.15
Seed	700, 7000

Table E.3: Hyperparameters for the best model search.

List of Figures

List of Tables

List of Abbreviations

CVR Centrum Výzkumu Řež

IoU Intersection over Union

Attachments

The thesis contains electronic attachments having the following structure:

- `/data`: datasets used for the training and evaluation in this thesis,
- `/experiments`: records obtained from experiments used for analyses in experiments section and best models that were found,
- `/source`: source code of this thesis with the main program in Python files and analyses captured in Jupyter notebooks,
- `/thesis.pdf`: text of this thesis.