AUTOMATING THE SEGMENTATION OF X-RAY IMAGES WITH DEEP NEURAL NETWORKS

Cassandra Desvig-Forss, Ludvík Petersen & Ulrika Woulhøj Jakobsen

s194115, s194613, s194124

ABSTRACT

The contents of this report detail the construction of a U-Net model that can automatically segment tomographic X-ray images. The model was tested with two different dataset sizes, as well as different variations of noise implementation. The baseline model was found to have an accuracy of $\sim 97\%$ for the small dataset and 99% for the large dataset. With a combination of noise and a larger dataset, it was possible to achieve accuracies between $\sim 94\%$ and 96%. It was shown that up to a noise level of 30000 it still performed better than randomly guessing the classification would, when training with 50% noisy data.

1. INTRODUCTION

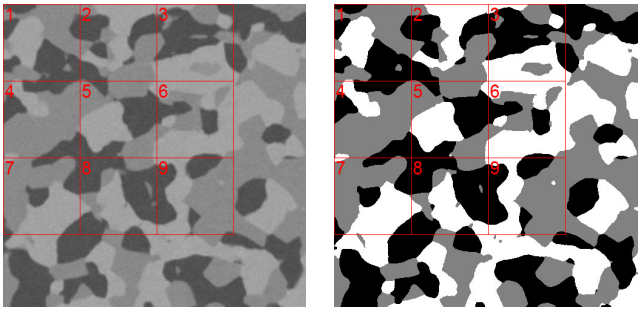
The domain of X-ray physics is currently undergoing rapid advancements. The impact of this growth is notable in several sectors of society, most notably within scientific and medical domains. Tomographic X-rays, in particular, have become especially indispensable for analytical applications. The broad utilization of tomographic X-rays underlines the necessity for developing tools that can be applied within the industry, which may help speed up this analytical process. One significant challenge lies in the segmentation of the images. This is a very cumbersome task to perform manually, where both time expenditure is vast and human error is easy to come by. The industry would therefore benefit greatly from a fully automatised process for such tasks. In this paper, we showcase how we may utilize deep learning models to help develop such an image segmentation tool.

1.1. Problem definition

The main goal of this project was to create a deep neural network, which can segment X-ray tomography images into three distinct classes automatically. Additionally, the effect of noise added to the dataset, as well as the utilization of an augmented dataset is investigated.

2. THE DATASET

A dataset was provided which consists of 500 tomographic X-ray samples, with 500 raw input images and their corresponding label images. An example of both images can be seen in Figure 1. The labels are made up of three different classes namely black, grey, and white. The label images have been created manually, and thus it is important to note that errors can occur even in the label image, due to simple human error. However, in our case, these labels are considered as the ground truth for which our model will be trained on. Some discrepancies between the original image and the label are visible in Figure 1. The original images are of size 501×501 and are 16-bit grey-scale, which implies that each pixel value ranges between 0 and 65536.



(a) The raw/input image (b) The labeled image

Fig. 1. One of the images in the dataset (a) and the corresponding manually labeled image (b), which is used as the ground truth. The red lines and numbers indicate the croppings that are made and are not part of the original images.

2.1. Cropping and normalization

To reduce the computation time the images and their labels are cropped to sizes of 128×128. At first, only the upper left corners of the images were used, which is the square marked as 1. With this method, there are still 500 images in the dataset, simply smaller than the originals. It is possible to do so because of the scale and distribution of the features

in the images. They are very similar and at the same time random.

In order to obtain more data the images are cropped into nine, small 128×128 sections, instead of just one. The full 501×501 pixel images are not used since it does not match up perfectly with 128×128 when executing the cropping. Cropping all 500 images and their respective 500 labels results in 4500 small images and their corresponding labels which can then be used for further training. At last, all pixel values in the images are normalized such that their values are between 0 and 1. The label image is also "converted" to three masks, where all black pixels are equal to 0, all grey pixels are equal to 1 and all white pixels are equal to 2.

3. THE NEURAL NETWORK

When designing a neural network, it is important to make valid model decisions, based on the data that you are currently working with. In this case, as the report concerns itself with image data, a model based on a regular convolutional neural network framework is desirable. This is due to the capabilities of a CNN network in regards to dimensionality, downsizing, and feature mapping. In this specific case, it was chosen to utilize the **U-Net** architecture, as proposed by Ronneberger et. al.[1], which is based on this exact framework with some distinct features that can produce good results for image segmentation tasks.

3.1. U-Nets

The U-Net is an architecture, which was originally designed for biomedical image segmentation. Its design resembles that of an auto-encoder, as it has an encoder-decoder structure with a bottleneck at the bottom. The latent space that one reaches at the bottleneck of an auto-encoder is slightly different in the U-Net architecture. A U-Net instead tries to capture the last hierarchical features of an image at the bottleneck. It then proceeds to the decoder, which is in part connected to the encoder through the so-called 'skip-connections' of the network.

The encoder and decoder capture different features of an image, whereby the features that the encoder finds are denoted as being 'hierarchical' features. This explains the exact form of objects (e.g. the wheel of a bicycle), whilst the decoder extracts 'semantic' features, that describe spatial relationships and contextual information, indicating regions where the objects identified in the encoder are located. These properties together allow for pixel perfect representations of desired objects, indicating the use case for image segmentation tasks.

U-Nets are based on blocks, where each block contains two convolutional layers followed by a ReLU activation function and a max-pooling layer for the contracting path (encoder) and an up-convolution layer for the expanding path (decoder). Hence, the structure is symmetric, which

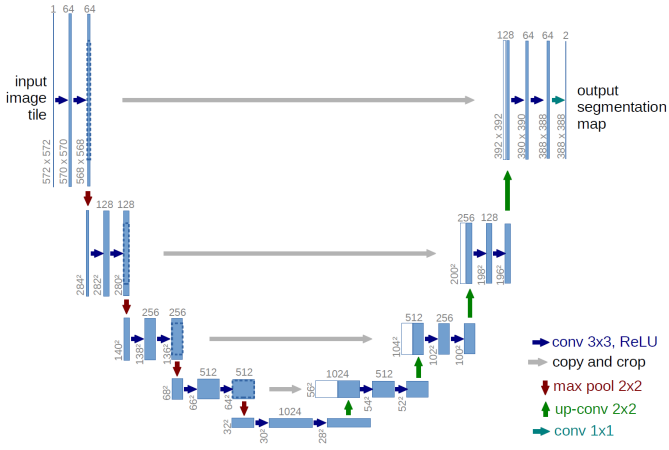


Fig. 2. The original proposed U-Net architecture [1].

can be attributed to the name **U**-Net. The two paths are joined through the skip-connections, as referenced previously, where the features of the encoder are appended onto the features of the decoder, to then be able to predict where in the image the given feature is located. The output layer is located at the end of the decoder, having the final shape `NUM_CLASSESxHEIGHTxWIDTH`, where `HEIGHT` and `WIDTH` is the original x,y-shape of the image and `NUM_CLASSES` is the number of classes for which it is wanted to segment the image into - in the case of this paper, this equates to three.

3.2. Training, validation and testing

When segmenting the data into training-, validation- and testing data, it was mainly chosen to segment this such that 70% of all data was used for training, 15% for validation, and consequently 15% for testing. To ensure that no prior bias would be contained within this segmentation, the images were shuffled, whilst still ensuring that the input to label image mapping was kept. The size of the training/validation batches varied depending on the size of the dataset. For the original dataset batches of size 25 were utilized and for the augmented dataset batches of size 64 were used, which was found to give the best performances. For all datasets the model was trained over 15 epochs.

3.3. Computing the accuracy

The accuracy for the predicted images was computed by comparing them with a pixel's ground truth label. In other words, a pixel at position $(45, 45)$ of the predicted image should be assigned the same label as its counterpart in the label image. Different approaches might have been preferable, such as utilizing the intersection over union (IoU) metric, however, for this particular data the method used was deemed valid as an

accuracy metric. The ground truth masks span the whole area of the image, which deviates from the bias that surrounds the pixel accuracy metric, which suffers when the class representation is small within an image.[2]

3.4. Performance of the baseline

The model was trained using cross-entropy loss and the *Adam* optimizer with a learning rate of $lr = 10^{-3}$. No *L1* or *L2* regularization was used for the baseline. However, batch normalization was used, which did result in a better model performance in contrast to when it was not utilized. The validation accuracies for the two versions (using the small dataset) were without batch-norm: $\sim 98.0\%$ and with batch-norm: $\sim 99.0\%$. Figure 3 shows the training and validation loss and accuracies for the baseline with batch norm. Table 1 shows the validation and test results for the baseline, when trained on the small and the large dataset.

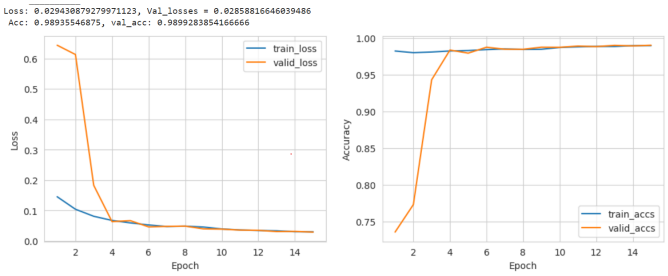


Fig. 3. Baseline performance of our model with batch normalization. The left graph illustrates the loss as a function of epochs, whilst the right depicts the accuracy as a function of epochs.

	Validation (small)	Test (small)	Validation (large)	Test (large)
Accuracy	0.990	0.968	0.991	0.990

Table 1. The resulting validation and test accuracies of the baseline model. "Small" represents the accuracies found using the small dataset with only 500 images, whereas "large" represents the large dataset with 4500 images.

To visualize the results of the baseline Figure 4 shows a test/input image, its label, and the predicted label using the baseline. It is clear that the segmentation overall is very good and corresponds well with the label. One noticeable part is the small grey segment next to the large black segment in the upper middle of the label and the predicted image. There is a significant difference since they are connected in the prediction but not in the label. Looking at the input image it might actually seem like they are indeed connected, which leads to some uncertainty regarding the label, and the accuracy may be higher than expected.

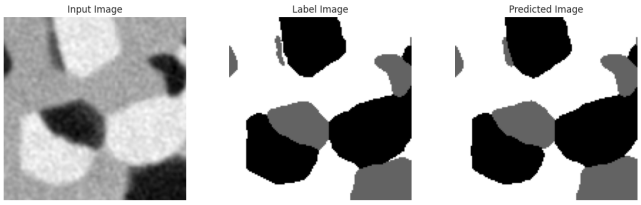


Fig. 4. Comparison of the input image (left), the label image (middle), and the predicted image (right) for the baseline model. Please note that the colors of the input image and the label image do not correlate, but the label image and the predicted image do.

4. CASEWORK

The U-net model performed well on the original data with accuracies in the nineties both in testing, validation, and training. Therefore, a case was set up to test the capabilities of the implemented U-net model. The case that was investigated is how the model handles different levels of noise on the test data and how large of a percentage of the training data needs to have noise added for the model to be able to segment the test data to a satisfactory degree.

4.1. Adding noise

The first step to challenge the baseline was to implement noise in specific parts of the dataset. This was done by adding a random number to each pixel of the original image, where the random number was sampled from a Gaussian distribution. The Gaussian distribution had a mean of zero and the standard deviation was not fixed, meaning that it could be changed in order to gauge the impact of varying levels of noise. It was implemented with the `torch.randn` function, which creates random numbers to form a normal distribution with a mean of zero and a standard deviation of 1. Thus, in order to achieve different levels of noise the function was multiplied by the wished value for the standard deviation.

The noise was added to all of the test data with the same noise level (standard deviation) on all the images. For the training and validation data, it was a bit different. Here only some images had noise added. This was controlled by a `rand` function where the noise was only added to the individual images if the random number was below a specified threshold. Thus if the threshold was set to 0.7, around 70% of the training and validation data would have noise added. The noise level was also varied for the training and validation data such that each image had a random noise level (standard deviation). This was done by multiplying a random number between 0 and 1 with a standard deviation of 10000. Thus the training and validation images could have noise levels between 0 and 10000.

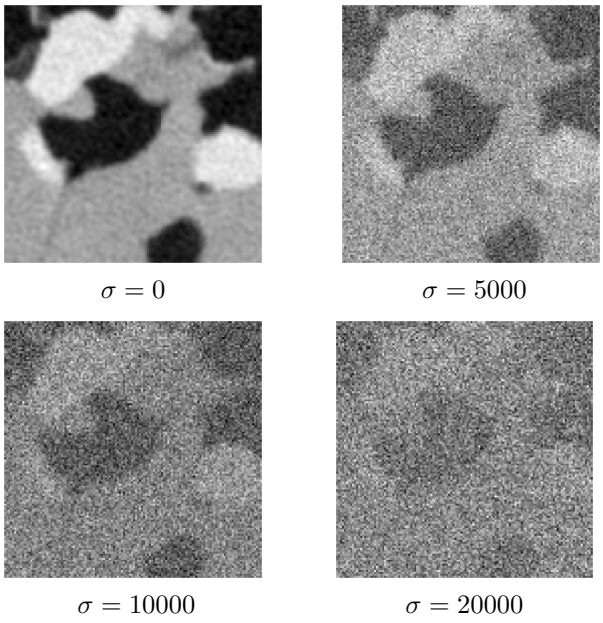


Fig. 5. Figure showing the effect of the different noise levels on the datasets. As can be seen, the "white" and "grey" areas are almost indistinguishable from each other at the highest noise level.

4.2. Adding data

Since the dataset was fairly small, the effect of a larger dataset on the network was also an interesting aspect. Thus, extra data was added in the manner described in Section 2.1. After this process, the dataset had a total of 4500 images, with their own corresponding labels.

4.3. Effect of training with noise

The first test was to see how large a percentage of the training data needed to have noise added for the model to segment noisy test data. Five different cases were studied. This was where 0% of the training (and validation) data had noise implemented, where 25% of the training (and validation) data had noise implemented, and likewise for 50%, 75% and 100%. The test was done both when using the original small dataset with 450 images in total and the large dataset with 4500 images in total to see if that would make an impact too. The resulting accuracies of the predicted labels can be seen in table 2 below.

As can be seen from row one, it had a negative impact on the accuracy when noise was added. As the training set gets more and more noisy data, this accuracy is increased by a small amount, but still quite a bit from the original accuracy obtained with the baseline without noise. Implementing the larger dataset didn't improve the performance, when there was not any noise on the training data. The accuracy in this case was only at $\sim 55\%$, which is even worse than

	0%	25%	50%	75%	100%
Testing (small)	0.844	0.863	0.841	0.866	0.869
Valid (small)	0.987	0.977	0.972	0.970	0.967
Testing (large)	0.545	0.936	0.954	0.957	0.960
Valid (large)	0.990	0.982	0.981	0.977	0.977

Table 2. Table showing the obtained accuracies in the various evaluations/tests. The percentages represent the amount of training (and validation) data that have noise added. Small and large represents the size of the dataset.

for the small dataset. This could be due to overfitting since the model got even more training data without noise. But it has not been investigated further, since that was not the main purpose of this test. After implementing noise on the training data, however, there was a vast improvement in the accuracy of the model. Obviously, the best case for the scenario where there is noise on all of the test data is when there is noise on 100% of the training data. Yet, even for the cases with 25% - 75% the accuracy is still excellent. In Figure 6 an example of a test image with $\sigma = 5000$ is segmented using a model trained on the large dataset where 50% of the training data has noise added. It is clear that even though this "only" has an accuracy of around 95% (as stated in Table 2) the overall segmentation seems good.

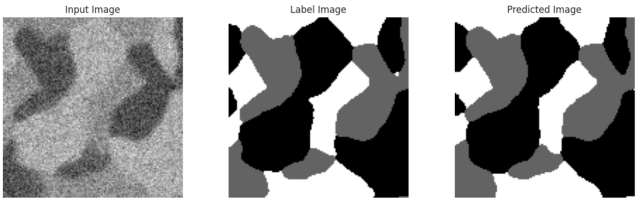


Fig. 6. Comparison of the input image (left), the label image (middle), and the predicted image (right). The model used is trained with noise on 50% of the training data and the input image used in this case has a noise level of $\sigma = 5000$. Please note that the colors of the input image and the label image do not correlate, but the label image and the predicted image do.

4.4. Performance at different noise levels

To see how the noise affects the accuracy Figure 7 was created. Here the model was trained on a training set where 50% of the data had a randomly varying standard deviation between 0 and 10000. The model was only trained once. Different test sets were then made, where they each had a differ-

ent noise level. The noise levels on the test data varied from $\sigma = 0$ up to $\sigma = 30000$ (one test set with $\sigma = 1000$, one with $\sigma = 2000$, etc.). The resulting accuracies at the different noise levels in the test data are showcased in Figure 7 below.

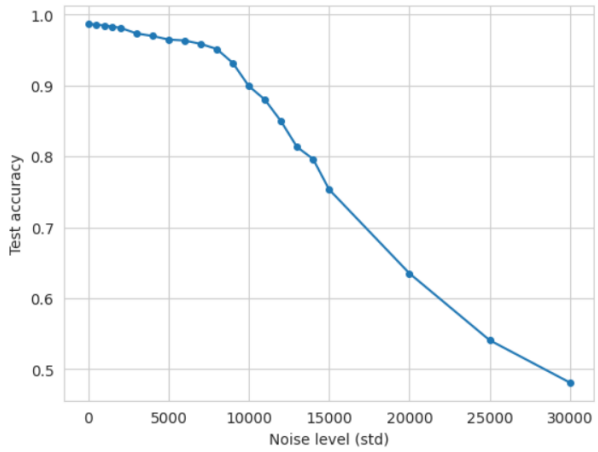


Fig. 7. Figure showing the accuracy for the model as a function of the noise level on the test data. The model was trained on data with varying noise levels on 50% of the training data data. The model was run several times on the different test datasets, varying the noise level each time.

The accuracy is relatively stable up until $\sigma \sim 9000$, where after this it has a fairly steep drop-off. The accuracy does not decrease in a line, however, and almost seems to flatten out a bit at the higher noise levels. Even at the very high noise level of $\sigma = 30000$ the model is still able to classify the pixels with an accuracy of a little under 50%. It is not an impressive accuracy in itself, but considering that there are three classes, where guessing would only yield $\sim 33\%$, it is acceptable.

5. DISCUSSION

In this section, the various facets that encompass this work are briefly commented on.

5.1. Viability of the U-Net model for image segmentation

It is wanted to discuss the rationale behind the choice of model. The initial premise for this choice was, that the U-Net has been especially built for semantic segmentation. Other models, such as deeplab [3], have also been built for this same purpose, however, given the widespread popularity of the U-Net, this was chosen as the baseline for the project. The performance of the model also suggests that the description of the U-Net holds, as it was able to achieve very good baseline results. This suggests that a change in the infrastructure will only at most make a fractional change to the model performance. In general, this report managed to show how well such a U-Net model is able to perform for semantic segmentation problems.

5.2. Model hyperparameters

In regards to the model, it was decided not to utilize any weight decay for the Adam optimizer, based on the performance of the regular dataset, which did not attribute any need for weight decay. As detailed previously in Section 4.3, the baseline model trained with the large dataset overfit the data to a large degree. By adding weight decay, it was possible to achieve better results, reaching a high of 66.5% accuracy for $weight\_decay = 10^{-2}$, which can be found in figure 8.

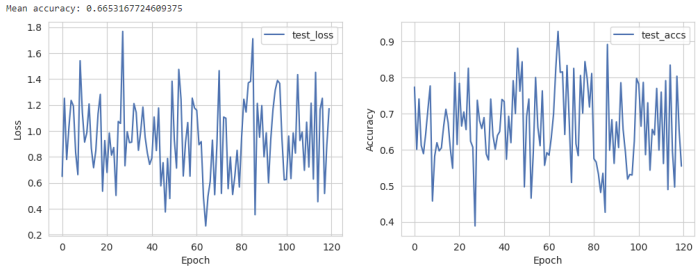


Fig. 8. Test results of the model when adding $weight\_decay = 10^{-2}$ to the optimizer.

However, as this was later in the process and as this only became an issue once noise was added, the baseline remains without weight decay. The model does not contain any dropout either, which was excluded due to the use of the batch norm as well as the insufficient nature of dropout when used for convolutional neural networks [4].

5.3. Generalization to full-sized images

When doing image augmentation over the whole dataset, it cannot be presumed that if the whole dataset has been augmented in some way, then when given an original image as input, it will naturally unconditionally generalize to this. The structure of the U-Net allows for generalization through its skip connections, which perform as intended, which can be perceived in Figure 9.

The predicted image shows that the baseline model is able to perform well on the original data, concluding with a mean accuracy of 99.0% over a small sample of 10 images.

6. CONCLUSION

In order to solve the problem of the time-consuming segmentation process of tomographic X-ray images, a U-Net model has been implemented. Instead of only working with a ”small” dataset of 500 images, an additional larger dataset was created such that there was a total of 4500 images. The

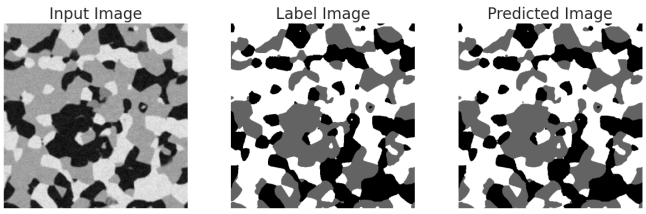


Fig. 9. Model generalization to the full-sized image. Note, that the colour scheme for the label/predicted image is not mapped correctly relative to the input image (between the two they are however).

baseline was found to have an impressive accuracy for both the small and the larger data set, being $\sim 97\%$ and $\sim 99\%$ respectively.

To challenge the model noise was implemented, which at first confused the model. To overcome this confusion it was tested how large of a percentage of training data needed to have noise added for the model to segment noisy test images. Here it was found that when the test data has noise, implementing noise on between 25% and 100% of the training data improves the accuracy. For the small dataset, this improvement was not very large, but for the larger dataset, it had a vast significance. Different standard deviations for the noise on the test data were also tested. Here it was found that up until a value around $\sigma = 9000$ there was not a large impact of the noise. Once above this value, the accuracy of the model declined rapidly.

Altogether it was found that the implemented U-Net model was able to successfully segment the tomographic X-ray data automatically.

7. THE CODE

The code for the model can be found on GitHub here. In the script there are three scenarios. There is one with the small dataset, the large dataset. Both of these are without noise. The last scenario is the large dataset with noise on 50% on the training/valdiation data, as well as noise on 100% of the test data.

8. FUTURE WORK

While the model was shown to work to a very satisfactory level, there is still room for improvement if the model should work to the optimal potential. There are several steps which can be taken in the future to improve the model.

The tests run in Section 4.3 were only performed with noise on all of the test data. Thus, the optimal middle point for the accuracy where it is able to perform well on both non-noisy and noisy data has not been found. To rectify this several tests should be run with a completely random amount of noise on the test data, such that it is tested on both non-noisy and noisy data. It could also be investigated whether noise on the training data could improve the segmentation of non-noisy test data.

Furthermore, it would be interesting to analyse whether there are better ways to calculate the accuracy than the currently used method. As mentioned above in Section 3.3 the IoU approach could have been beneficial. In this method, it is the overlap divided by the area of union which determines the accuracy [5]. This method gives a better understanding of how close the predicted image is to the ground truth since it is not simply a wrong or correct label.

It would also be interesting to look into the effect of an even larger dataset. This could be accomplished by rotating the images or making the cropped images a bit smaller such that the whole original image is used (see Figure 1 where the current cropping is shown). A larger dataset could improve the accuracy just a bit more but there is of course a trade-off with computation time which needs to be taken into account. At the moment the model is trained in under 15 minutes (with GPU) and with a larger dataset this could increase significantly which may not be worth it compared to the improvement. This trade-off could be investigated.

Lastly, another noise model could be implemented to investigate any differences from the Gaussian model. It could be that fx uniform noise would impact the model and thus the results differently. If multiple types of noise were used the model could potentially also generalize better and thus have a better performance.

9. REFERENCES

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany. 2015.

[2] Jeremy Jordan. *Evaluating image segmentation models*. 2018. URL: https://www.jeremyjordan.me/evaluating-image-segmentation-models/. accessed: 15.12.2023.

[3] Liang-Chieh Chen et. al. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: 2017.

[4] Harrison Jansma. *Don't Use Dropout in Convolutional Networks*. 2023. URL: https://www.kdnuggets.com/2018/09/dropout-convolutional-networks.html. accessed: 15.12.2023.

[5] Adrian Rosebrock. *Intersection over Union (IoU) for object detection*. 2016. URL: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. accessed: 20.12.2023.