ENS Data Challenge - CorroSeg: corrosion detection in steel pipes

Hippolyte Pilchen[*], Lucas Gascon[†] – Presented on March, 20^{th} 2024

Abstract

In this paper, we introduce CorroSeg, an approach to detect corrosion in steel pipelines using advanced image segmentation techniques.

This paper delves into the specifics of ResNet and U-Net architectures, providing a rationale for our CorroSeg model choice based on their unique strengths in image segmentation tasks. We discuss the challenges posed by the dataset, including the handling of outliers and the imbalance between corrosion and background pixels. We explain our strategies for addressing these issues through preprocessing and the design of a tailored loss function.

1 Introduction

Advancements in machine learning and image processing have dramatically enhanced the field of infrastructure inspection. Central to this progress are Convolutional Neural Networks (CNNs), which are pivotal in image segmentation tasks—classifying every pixel or group of pixels within an image. Two notable architectures, UNet [Ronneberger et al. 2015] and DeepLab [Chen et al. 2017], stand out for their high-precision segmentation capabilities. UNet, with its unique U-shaped design, combines downsampling and upsampling paths to effectively capture image context and detail at multiple scales. DeepLab, on the other hand, integrates CNNs with atrous convolution and Conditional Random Fields to produce high-resolution segmentations.

Our project is dedicated to identifying corrosion in steel pipelines, employing an image segmentation technique where pixels are classified as either 0 or 1 to distinguish between background and corrosion areas, respectively.

This challenge is presented by SLB, a globally renowned French company committed to energy innovation. In the oil and gas industries, over 25% of well failures are attributed to corrosion-related issues, highlighting the critical need for effective detection methods. Beyond the financial ramifications, corrosion can lead to significant environmental consequences, including groundwater contamination.

The challenge categorizes three types of defects (corrosion and wear) as follows:

1. Pitting corrosion;
2. Localized defects;
3. Axial grooves.

For training our segmentation model, we have been provided with ultrasonic images from 15 wells. These images are accompanied by binary masks of identical dimensions, representing the segmentation of the corrosion we seek to predict. Given the large size of the original images (one image per well), we are provided with 36x36 image patches for our analysis. Code and resources are available on the page of the project [1].

[*]hippolyte.pilchen@polytechnique.edu
MVA, ENS Paris-Saclay
[†]lucas.gascon@polytechnique.edu
MVA, ENS Paris-Saclay

[1] https://github.com/HipPilchen/ENS-data-challenge-CorroSeg

2 Segmentation challenge

2.1 Dataset analysis

Before delving into building a model, our initial approach involved conducting a comprehensive data analysis. This step was crucial as it provided us with a deep understanding of the underlying complexities and intricacies of the task. Through rigorous examination and interpretation of the available data sets, we were able to uncover valuable insights. By prioritizing data analysis, we ensured a well-informed approach that laid the groundwork for addressing the challenge with precision and efficacy.

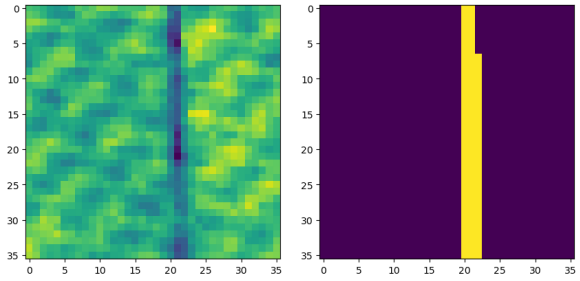


Figure 1: *Examples of raw data: an image and the associated mask.*

The images of the dataset comes from ultrasonic imagers. They monitor pipe thickness to inspect steel pipelines and detect corrosion along the wellbore. Corrosion can be noticed by thinner localized part which corresponds to the darker part in Fig. 1. The annotations are patches of the same size as the well images with 0 and 1 where the 1s indicate the presence of corrosion (in yellow in the figure).
The training set is composed of 15 wells which are divided in patches (one channel image) of size 36. Each well has a different length so all wells do not have the same number of patches. There are 9670 training patches and 2538 testing patches, which come from 5 different wells.

Firstly, we noticed that they were more annotations than images in the train dataset. After discussing with the challenge organizers and the company we removed the additional masks. Then, we observed that some images were corrupted by outliers. These outliers arise due to telemetry, where processing errors can occur, corrupting certain data points. As a result, we have decided to remove them from our training set to prevent them from compromising our models.

Secondly, we performed pre-processing tasks on the training set. We removed the outliers and normalized the training images. We used the RobustScaler from scikit-learn[2] which is a scaler robust to outliers. Once fitted on the training images, we transform both the training and the testing patches. The data distributions before and after the pre-processing are highlighted in Fig. 2. The normalization enables to avoid discrepancy between the training and the testing phase for the model. It generates smoother images as shown in Fig. 3.

[2] scikit-learn

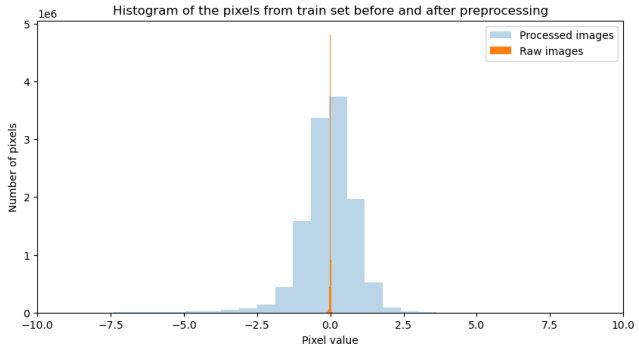


Figure 2: *Train images distribution before and after preprocessing with RobustScaler. Outliers from the raw images have been removed for display purpose.*

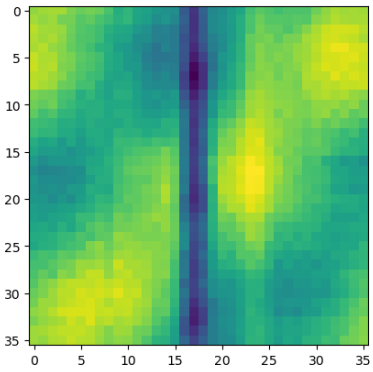


Figure 3: *Example of a pre-processed train image*

Finally, black masks are associated to corrupted test images, which represents 6.8% of the test dataset against 3.5% for the training patches.

We obtaixn from our data analysis two major insights which lead us in our designing of the model. First, corrosion often appears in large horizontal or vertical grooves which overlap on several patches. Therefore, if there is a horizontal axial corrosion on a patch from a well, the next patch will most likely have a corrosion mark following on from that on the previous patch. This is why we include the number of the well for each patch in the dataloader we designed. Secondly, there is a great imbalance between the number of pixels with corrosion (label 1) and the background (label 0). On the training dataset, 80.078% of the patches contain corrosion but it represents 7% of the pixels. This leads us to design particular loss functions to deal with this class imbalance.

2.2 The evaluation metric

The evaluation metric used for scoring is the Intersection over Union (IoU). This metric is a widely used evaluation measure in image segmentation tasks. It quantifies the overlap between the predicted segmentation mask and the ground truth mask. The IoU is calculated by dividing the area of overlap between the predicted and ground truth masks by the total area encompassed by their union. Mathematically, the IoU is expressed as:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{TP}{TP + FP + FN} \quad (1)$$

where TP represents the true positive pixels (correctly segmented pixels), FP represents false positive pixels (incorrectly segmented pixels), and FN represents false negative pixels (missed pixels). The IoU metric provides a value between 0 and 1, where 1 indicates perfect overlap between the predicted and ground truth masks. It serves as a comprehensive measure of segmentation accuracy, capturing both localization and shape accuracy of the segmented objects.

3 Model architectures

3.1 First models

Our first approach was to reproduce the model described in the baseline with 5 convolution layers and two fully connected linear layers at the end to obtain the required dimension. Starting from this baseline, we implemented more complex architectures. Then, we tried a Fully Convolutional Network (FCN) based on [Long et al. 2015a]. This architecture consists in several blocks of convolution layers, pooling and activation function (ReLU). It enables to compute nonlinear filters which are applied successively to the input. The feature map from the encoder is then upsampled with transposed convolutions. It consists in applying a convolution with a certain kernel on a modified input, zeros have been inserted between rows and between columns (depending on the stride).

We also tried another FCN with a *bilinear* upsampling at the end instead. After further deepening our review of the literature, it comes up that U-Nets are the most accurate models to perform segmentation and they are actually used to perform defaults and corrosion detection as described in [Nash et al. 2022]. Thus we decided to focus on this type of architecture.

3.2 U-Net

Introduced in 2015, the U-Net architecture [Ronneberger et al. 2015] advances the concept of a "fully convolutional network" as first proposed in [Long et al. 2015b]. Initially designed with a focus on biomedical imaging to address the issue of data scarcity, U-Net excels in image segmentation through detailed pixel-level classification.

At its core, U-Net enhances a traditional contracting network by incorporating layers that utilize upsampling techniques instead of conventional pooling operations, thus improving the resolution of the output. This enhancement is further refined by additional convolutional layers that compile a complex output from the upscaled features. A key feature of U-Net is its abundance of feature channels in the upsampling phase, ensuring smooth transmission of context across layers of varying resolutions. This balance between the expansive and contracting pathways forms the characteristic U-shaped architecture of the network. Uniquely, U-Net relies solely on the effective area of each convolution, foregoing fully connected layers to maintain efficiency and focus. For predicting pixels at the edge, it ingeniously extrapolates missing context by mirroring the input image. This is done by concatenating the upsampled feature map with the corresponding cropped feature map from the encoder (contracting path). This strategy that is crucial for the network's adaptability to large-scale images without the typical limitations imposed by GPU memory.

Due to its results on IoU score on other datasets, we rapidly converged towards this type of architecture. We tried to adapt DeepLabV3 [Chen et al. 2017] model to our problem since it was another state-of-the-art segmentation model. However, the small size of our input images does not enable to leverage the advantages of atrous convolution.

Our first model based on U-Net architecture is made of five convolutional blocks (SAdd-UNet). Skip connections are implemented by adding feature maps from the encoder to those in the decoder. These long skip connections restore spatial information that may have been lost during downsampling.

We also implemented a smaller U-Net architecture using concate-

nation to perform long range skip connections (XSCat-UNet). This method allows even less spatial information to be lost. However, since the input size increases, this method is more costly computationally.

Models	Train IoU	Valid IoU
Baseline CNN	0.770	0.659
SAdd-UNet	0.7644	0.707
XSCat-UNet	0.670	0.674
Add-CorroSeg	0.781	0.719
Cat-CorroSeg	**0.827**	**0.738**

Table 1: *Comparison of different architectures trained with the IoU smooth loss during* 100 *epochs, on the data transformed with the first 4 transformations, without any pre-trained backbone and without dropout. The learning rate is set to* 10^{-4} *and the batchsize to* 64.

3.3 Our final model: CorroSeg

The final model we have implemented (Figure 4) adopts the overall architecture of U-Net, but with a twist: it incorporates ResNet50 layers [He et al. 2015] for the Encoder section, aiming to leverage advantages of the ResNet and enabling to use a pre-trained Encoder. ResNet was one of the first model to introduce skip connections. Such skip connections in the Encoder are able to maintain a continuous gradient flow from the initial to the final layer, addressing the issue of vanishing gradients. Short skip connections seem to contribute to stabilizing gradient updates within deep architectures. Therefore, with this architecture, the model takes advantage of both short skip connection in the Encoder and long range skip connection to avoid loosing spatial information, as introduced with U-Net.

The Upsampling and Downsampling layers have been redefined; they are now implemented using Convolution and Transpose Convolution layers directly instead of nearest neighbours interpolation for upsampling in [Ronneberger et al. 2015]. We tried several options for the Decoder. With additive skip connections, our experiments show that using batch-normalization on the decoder contributes to a more unstable learning with slightly lower results on the validation set in the not pre-trained case (see Fig. 11 in appendix). The same goes for the model with pre-trained backbone. With concatenation, batch-normalization slows convergence but the final validation score reached is higher than without.

4 Loss

4.1 Binary Cross Entropy (BCE)

In binary segmentation tasks, the goal is to classify each pixel in an image as belonging to one of two classes, typically corrosion or background. BCE loss, also known as logistic loss, is well-suited for such tasks due to its ability to measure the difference between predicted probabilities and ground truth labels for each pixel. Mathematically, BCE loss calculates the cross-entropy between the predicted probability distribution and the true binary labels. This loss function penalizes the model more heavily for predicting probabilities far from the ground truth, thus encouraging the model to learn accurate pixel-wise classifications. Furthermore, BCE loss provides a smooth gradient for backpropagation, facilitating efficient optimization during training. Model does learn how to increase IoU score while trained with the BCE loss. However, since this loss focuses on a pixel level with a great imbalance in labels and is not designed directly to maximize the IoU, we believed that another loss could be more accurate.

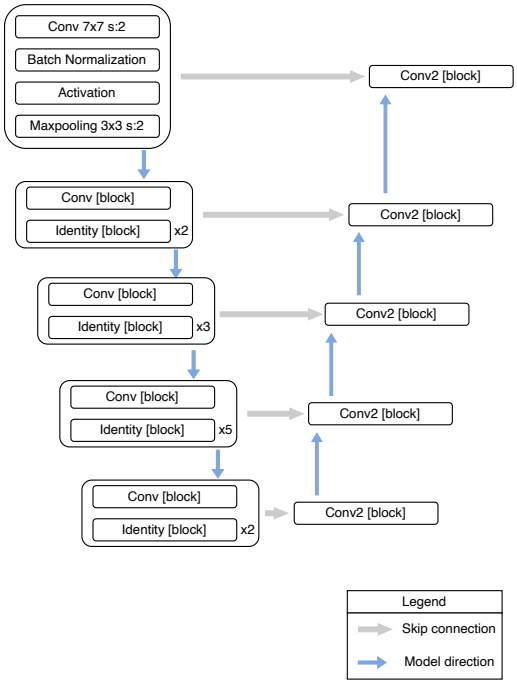


Figure 4: *Architecture of our model CorroSeg. ResNet50 encoder on the left part and customed decoder on the right part forming a U-Net like architecture.*

4.2 Focal loss

To limit the impact of the imbalanced dataset on learning, the focal loss has been tested. It is derived from the Cross-Entropy loss with additional terms to deal with the class imbalance problem. Focal Loss [Lin et al. 2018] is a modification of the standard cross-entropy loss function, designed to address the problem of class imbalance in object detection tasks. It introduces a modulating factor that down-weights the loss assigned to well-classified examples, thus focusing the training on hard, misclassified examples.

The formula for Focal Loss is given by:

$$FL(p_t) = \begin{cases} -\alpha(1-p_t)^{\gamma} \cdot \log(p_t) & \text{when } y = 1 \\ -p_t^{\gamma}(1-\alpha) \cdot \log(1-p_t) & \text{when } y = 0 \end{cases}$$

where p_t is the predicted probability of the corrosion class, α weights each class and γ is a tunable focusing parameter. However, even though the two additional hyperparameters should help to tackle class imbalance and misclassified examples, the results of the model trained on this loss were not satisfying.

4.3 Smooth IoU loss

In our work, we have implemented a variant of the Intersection over Union (IoU) loss, termed as "smooth IoU loss". The IoU metric measures the overlap between the predicted segmentation and the ground truth, where higher values signify improved performance. As our objective is to minimize the loss, we employ IoU as our loss

function. For computing IoU, we utilize the following formulation:

$$\text{IoU} = \frac{\text{Area of Overlap} + \text{Smoothing Parameter}}{\text{Area of Union} + \text{Smoothing Parameter}} \quad (2)$$

This addition of a smoothing parameter enhances numerical stability by averting division by zero and contributes to gradient stabilization. Specifically, in situations with very small intersections and unions, both the IoU and its gradient can exhibit high volatility. Furthermore, this modified loss is differentiable, in contrast to the traditional IoU metric, making it well-suited for gradient-based optimization techniques prevalent in training deep neural networks.

From the perspective of our problem, this loss function is particularly beneficial as it directly targets the optimization of our primary metric of interest, the IoU score. Consequently, we anticipate superior results with this approach compared to previous methodologies.

5 Learning methods

5.1 Transformations

Due to the relatively small size of the dataset, we decided to implement data transformations. By doing so, the goal is to perform data augmentation, adding new examples to the available ones. We tested the following transformations: *HorizontalFlip*, *VerticalFlip*, *RollTransform*, and *ComposedFlips*. "None" means that the raw image is used. *RollTransform* is a vertical and horizontal random shift and spaces left empty are replaced by pixels disappearing at the other borders. *ComposedFlips* is a composition of an horizontal and a vertical flip of the image.

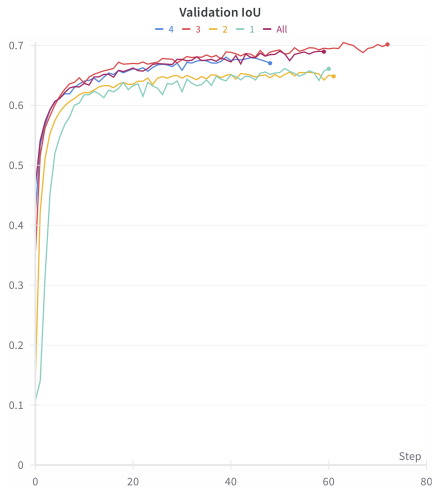


Figure 5: *Experiments on the learning rate on our CorroSeg model, not pretrained and without dropout. The Smooth IoU loss is used to train the model with a batchsize of 160 images. The validation Smooth IoU score is displayed during learning for various number of transforms from the list in section 5.1.* 1 *means only raw image,* 2 *corresponds to raw images concatenated with horizontally flipped images, and so on for* 3 *and* 4 *in the order in which the transformations are listed in paragraph, and* All *corresponds to the whole list of transforms.*

In Fig. 5 it can be observed that adding more than one transform enables to reach higher validation IoU scores. Flipping enables to transform a vertical axial corrosion to a horizontal one and *RollTransform* just shifts the corrosion. Therefore, these transformations preserve the type of the corrosion enabling to avoid discrepancies between learning and infering. In order to have enough transformations to reach high validation scores without having too many which could lead to overfitting and increase the computational cost by increasing the dataset size, we choose 4 as the best number of transforms. Then, we tested all the combinations of 4 transformations, the one which enabled to reach the highest score on the validation set is the one without *RollTransform*.

5.2 Hyperparameters

Learning parameters

First, we studied through cross-validation the impact of the learning rate on the training. A balance must be reached between not getting stuck in spurious minima with a too small learning rate, learning in a reasonable time and not plateauing at low score values with too high learning rate (see Fig. 12 in the appendix). The result of our study indicated that $1 * 10^{-4}$ is a good candidate to obtain the best model.

In the same way, the batchsize impact on model learning has been tested. Batchsizes from the following list [16, 32, 64, 128, 160, 256] have been tested. Too large batchsize leads to GPU memory error in some settings, and also to more instabilities during learning on the validation loss. Therefore, increasing the batchsize will not enable to obtain everytime a better IoU score on the test set. Inversely, a batch of 16 images does not learn how to reach a high IoU score even on the training set. Batchsize also affects the learning speed, higher is the batchsize slower is decreasing the loss. In this particular context, we've opted for a batch size of 128. This choice strikes a balance between achieving the highest score on the validation set and ensuring that GPU memory doesn't become overwhelmed.

Avoiding overfitting

After a few experiments we noticed that due to the size of our CorroSeg model we were often overfitting on the training data. We adopted several strategies to avoid overfitting.

Several dropout's policy were implemented. The first one which consists in applying a dropout of parameter 0.1 between the two first and two last layers and a dropout of parameter 0.2 in the middle of our U-Net model. The second strategy is similar to the previous one but with parameters of 0.2 and 0.3 respectively. In the model without pre-training, adding dropout seems to slightly improve learning by bringing the validation loss closer to the training loss. However, for the model with a pre-trained backbone, dropout is crucial to reach high scores and avoid plateauing.

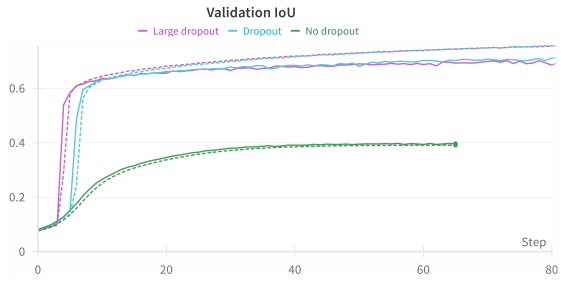


Figure 6: *Experiments on dropout for our CorroSeg model with the pretrained backbone. The IoU smooth loss is used to train the model with a batchsize of 160 images with a learning rate of* $5 * 10^{-5}$*. The weight-decay is set to* 10^{-2}*. The validation IoU score is displayed in plain lines and the train score in dashed lines during learning for various dropout (None, with dropout of 0.1 and 0.2 and with dropout between 0.2 and 0.3).*

In the same way, weight decay of the Adam optimizer [Kingma and Ba 2017] has been studied. This parameter enables to add a $\mathbf{L_2}$ regularization to the loss in order to prevent overfitting. The weights of the neural network are then more or less regularized. While regularizing too much hinders the model from learning, not using enough weight decay leads to overfitting. This is particularly true for the pre-trained model. Indeed, in Fig. 7 weight decay larger than 10^{-3} implies a poor learning which get stuck at low value of the IoU score. Subsequently, we opted for the highest weight decay parameter (to mitigate overfitting), which allowed us to reach a plateau at high score values, specifically 10^{-3}.

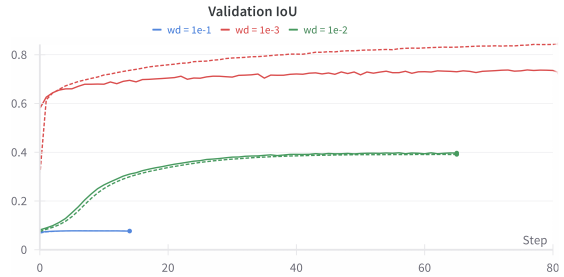


Figure 7: *Experiments on weight decay for our CorroSeg model with the pretrained backbone. The IoU smooth loss is used to train the model with a batchsize of 160 images with a learning rate of* $5 * 10^{-5}$. *The validation IoU score is displayed in plain lines and the train score in dashed lines during learning for various weight decay.*

Finally, weight-decay and dropout seem to play a similar role in enabling the pre-trained backbone to adapt to the new task. Indeed, by optimizing weight decay or dropout, we start from a model that usually plateaus at validation scores of 0.4 to a model that reaches up to 0.7 IoU score.

5.3 Transfer learning techniques

Transfer learning capitalizes on insights obtained from a source task to improve the efficiency on a subsequent, similar target task. This process involves adapting a pre-trained model, initially developed for a distinct task, to a new context. The essence of this strategy lies in utilizing the pre-trained model as the structural "backbone" for the new application. The adaptation typically necessitates modifications to the model's final layers, known as the "model head", to tailor it for the target task. Various strategies for this adaptation include maintaining the original weights of the backbone while training only the model head, retraining the model in its entirety, or progressively "unfreezing" the backbone layers, beginning with the lowermost layers.

In our experiments, we chose the ResNet50 architecture as the backbone for our model. ResNet50 is known for its balance between accuracy and computational efficiency. We use it as the encoder in our CorroSeg model, providing a strong foundation for our transfer learning efforts.

Ultimately, it is observed that within the hyper-parameter configuration employed in these experiments, and for our CorroSeg model specifically, utilizing a pretrained ResNet50 backbone yields higher performance compared to randomly initializing the weights when subsequently training these backbones. The tested defreezing strategies, which involve starting with a pretrained backbone with all its blocks frozen and then defreezing one block every N epochs, lead to varying degrees of success. The most effective strategy appears to be defreezing a block every 10 epochs; however, given that these defreezing strategies do not appear to significantly impact the

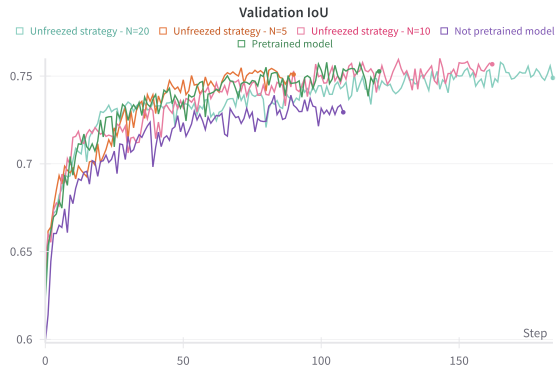


Figure 8: *Experiments on unfreezing strategy for our CorroSeg model with the pretrained backbone, and one experience with not pretrained backbone to have a baseline. The IoU smooth loss is used to train the model with a batchsize of 64 images with a learning rate of* 10^{-4}.

model's performance compared to using the pretrained backbone and retraining it entirely from the first epoch, we have, in our subsequent experiments, only considered models where the backbone is unfrozen from the beginning. This approach corresponds to the "not pretrained model" and "pretrained model" experiments within Fig. 8.

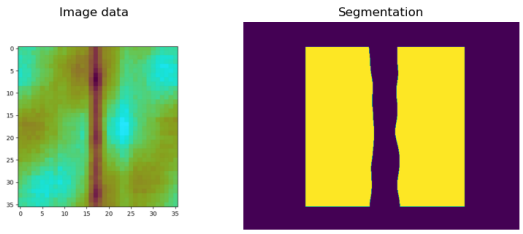


Figure 9: *Illustration of the random walker algorithm. Training image on the left and the result of the random walk with labeled pixels given by the neural network on the right.*

6 Random Walker

Due to the special features of our images, we thought that post processing could lead to better results. Indeed, corrosion can be noticed on an image by the contrast in color with the background. Therefore, the random walker algorithm introduced by [Grady 2006] seems to leverage these characteristics. The algorithm takes an image with some pixels marked with a label as an input. Then, for each unmarked pixel, a random walk starting at that pixel is defined, moving from neighbor to neighbor. The probability of reaching a neighbor depends on the intensity difference between the two pixels, the contrast, which is represented by a cost function. For each unlabeled pixel, probabilities for this random walk to reach a marked pixel are examined, giving for each pixel, a probability of belonging to each label. It is then sufficient to conclude that this pixel belongs to the label for which it has the highest probability of belonging. However, this process is very time-consuming as presented here. [Grady 2006] states in his paper that this problem is actually equivalent to a linear algebra problem, and that solving a linear system is sufficient to obtain all the desired probabilities. It leads to an efficient algorithm which classifies pixels based on a few labelled pixels. Pixels labeled 0 or 1 by the deep learning model with the highest probability (above 0.9) are given as marked pixels

to the random walker. An example of the output mask is given in Fig. 9.

This method improves the results of some models on the test set. With the small U-Net model the results goes from 0.55 IoU to 0.57. However, our best models are not improved by this post-processing method.

7 Final results

After experimenting several combinations of parameters, testing their results on the validation set, we were able to reach 0.6770 of IoU score on the test dataset, ranking 1^{st} on academic public leaderboard. We ran our experiments on NVIDIA RTX A5000 GPU's. Our best results were reached with the following combination of model's architecture and hyperparameters:

- Our CorroSeg architecure with concatenation skip-connections and batch-normalization in decoder blocks using the pre-trained backbone and a not pre-trained one, since difference in the validation score are of the order 10^{-3} only.
- Training on the IoU smooth loss, without using dropout but with the appropriate weight decay.
- Using a learning rate of $1 * 10^{-4}$ and a batch-size of 128.

Finally, to stabilize our results and get robust predictions we performed ensembling. It consists in averaging the output of several models with different random seeds in order to averaged the results and get a more robust segmentation.

8 Conclusion

In conclusion, our model CorroSeg has made significant strides in addressing the issue of corrosion detection in steel pipelines through advanced image segmentation techniques. By leveraging the strengths of ResNet and U-Net architectures and by optimizing hyperparameters of our pipeline, we were able to design a robust model that effectively handles the intricacies and challenges presented by the dataset, such as outliers and class imbalance.

The implementation of a smoothed IoU loss function, in particular, has shown promise in directly optimizing our primary metric of interest, leading to potentially superior results compared to conventional approaches. Furthermore, the exploration of various learning methods, including data transformations and hyperparameter optimization, has contributed to refining our model's performance.

References

CHEN, L.-C., PAPANDREOU, G., SCHROFF, F., AND ADAM, H., 2017. Rethinking Atrous Convolution for Semantic Image Segmentation, Dec. arXiv:1706.05587 [cs].

GRADY, L. 2006. Random Walks for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 11 (Nov.), 1768–1783. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

HE, K., ZHANG, X., REN, S., AND SUN, J., 2015. Deep Residual Learning for Image Recognition, Dec. arXiv:1512.03385 [cs].

KINGMA, D. P., AND BA, J., 2017. Adam: A Method for Stochastic Optimization, Jan. arXiv:1412.6980 [cs].

LIN, T.-Y., GOYAL, P., GIRSHICK, R., HE, K., AND DOLLÁR, P., 2018. Focal loss for dense object detection.

LONG, J., SHELHAMER, E., AND DARRELL, T., 2015. Fully convolutional networks for semantic segmentation.

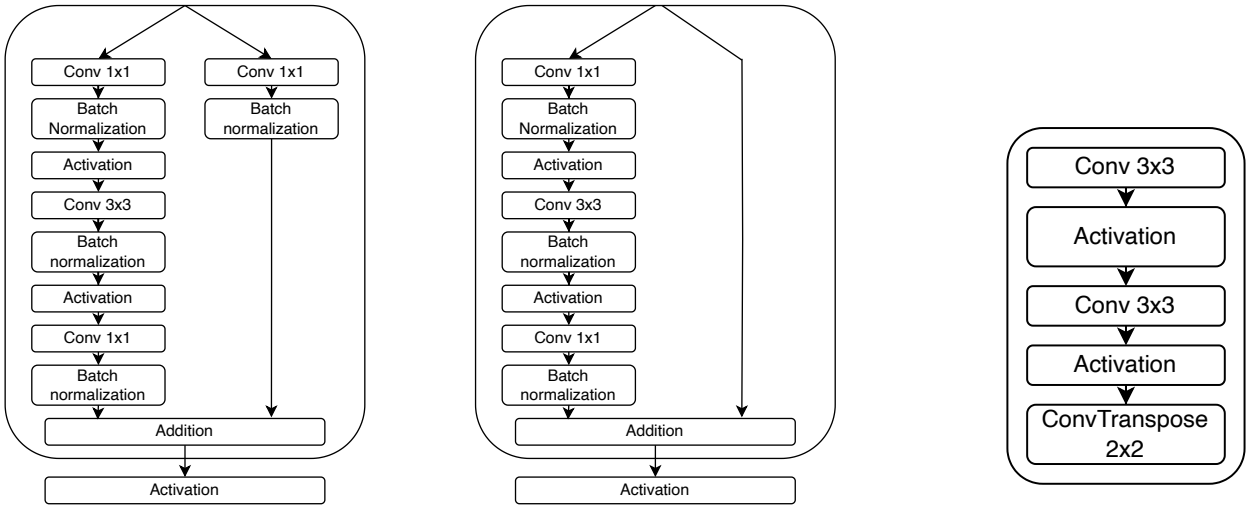
LONG, J., SHELHAMER, E., AND DARRELL, T., 2015. Fully Convolutional Networks for Semantic Segmentation, Mar. arXiv:1411.4038 [cs].

NASH, W., ZHENG, L., AND BIRBILIS, N. 2022. Deep learning corrosion detection with confidence. *npj Materials Degradation 6*, 1, 26.

RONNEBERGER, O., FISCHER, P., AND BROX, T., 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation, May. arXiv:1505.04597 [cs].

Supplementary information

S.1 CorroSeg Model blocks



(a) *conv [block]* **(b)** *identity [block]* **(c)** *conv2 [block]*

Figure 10: *Blocks from our final version U-Net model.*

S.2 Supplementary plots

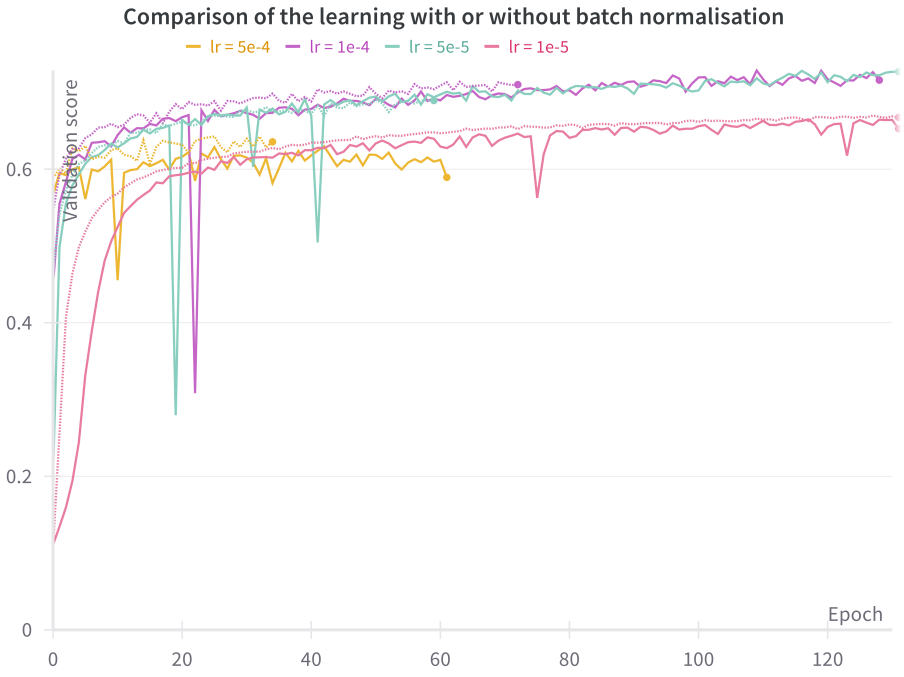


Figure 11: *Experiments on the decoder on our CorroSeg model, not pretrained and without dropout. The IoU smoothloss is used to train the model with a batchsize of 160 images. The validation IoU score is displayed during learning for various learning rate for decoder with (in plain lines) and without (in dashed lines) batchnormalization in the convolutional blocks.*

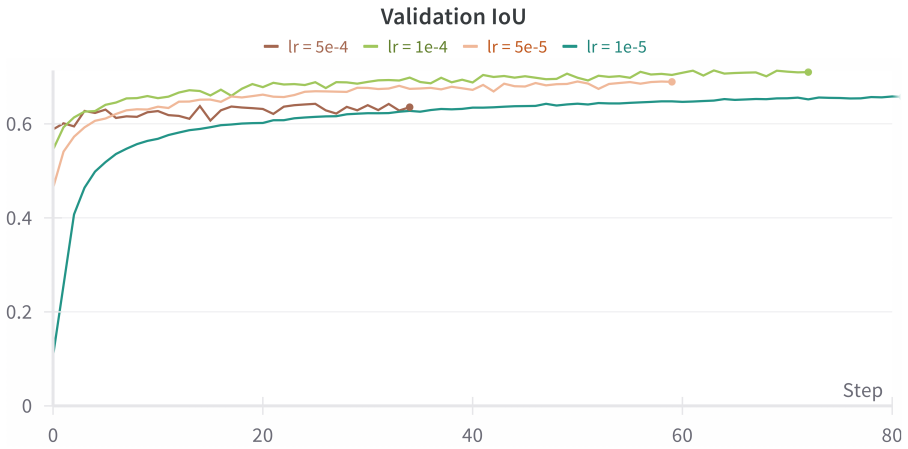


Figure 12: *Experiments on the learning rate on our CorroSeg model, not pretrained and without dropout. The IoU smoothloss is used to train the model with a batchsize of 160 images. The validation IoU score is displayed during learning for various learning rate.*