Seafloor Pipeline Detection With Deep Learning

Vemund S. Schøyen*[1], Narada D. Warakagoda[1,2], and Øivind Midtgaard[2]

[1]University of Oslo, Oslo, Norway
[2]Norwegian Defence Research Establishment, Kjeller, Norway.

Abstract

This paper presents fast, accurate, and automatic methods for detecting seafloor pipelines in multibeam echo sounder data with deep learning. The proposed methods take inspiration from the highly successful ResNet and YOLO deep learning models and tailor them to the idiosyncrasies of the seafloor pipeline detection task.

We use the area between lines and Hausdorff line distance functions to accurately evaluate how well methods can localize (pipe)lines. The same functions also show promise as loss functions compared to standard mean squared error, which does not include the regression variables' geometrical interpretation.

The model outperforms the highest likelihood baseline by more than 35% on a region-wise F1-score classification evaluation while being more than eight times more accurate than the baseline in locating pipelines. It is efficient, operating at over eighteen 32-ping image segments per second, which is far beyond real-time requirements.

1 Introduction

Seafloor pipelines are critical infrastructure to transport oil and gas. As pipeline failures can result in high economic and environmental costs, the pipeline's integrity must be verified through inspection. Specifically, the objective of external inspection of seafloor pipelines is to determine the degree of burial and to detect potential free spans, buckles, debris, or damages from human activities such as trawling and anchoring [1].

Traditionally, Remotely Operated Vehicles (ROVs) perform external pipeline inspection, but Autonomous Underwater Vehicles (AUVs) are emerging as a more efficient and less costly solution, particularly when the AUV operates without constant supervision from a mothership [2]. Typical payload sensors for these AUVs include Multi-Beam Echo Sounder (MBES), Side Scan Sonar (SSS), and an optical camera. In order to collect high-quality sensor data for inspection, the AUV must be able to follow the pipeline at the specified relative distance and height. Global position estimates from the vehicle's inertial navigation system will not be sufficient due to inevitable drift and to uncertainties in prior pipeline position.

One solution is to automatically detect pipelines in the sensor data to provide real-time input to the vehicle's control system to maintain the desired relative position, and orientation [2], [3]. Considerable variability in the appearance of pipelines in the payload sensor data, however, makes designing automatic detection algorithms a challenging task. There can be substantial variations in, e.g., data quality, sensing range and geometry, pipeline diameter and coating, deliberate or natural pipeline burial, marine growth, and seafloor characteristics. The data may also contain multiple pipelines.

This work investigates whether we can use deep learning methods to detect seafloor pipelines in MBES data. Ideally, we would use all payload sensors simultaneously during a seafloor pipeline inspection mission. However, the MBES and SSS have complementary fields of view while the MBES and the optical camera supplement each other. We expect the MBES data to facilitate more robust detection than the camera images because MBES provides a significantly wider swath, and the water visibility does not limit its data quality.

To the best of our knowledge, deep learning has never been used for object detection in MBES data before. Moreover, we define a novel label and pre-

*Corresponding Author: vemund@live.com

https://doi.org/10.7557/18.5699

diction format, representing the pipeline location with line segments. Therefore, we need a measure for line similarity to evaluate our model. We propose to use the Hausdorff Line Distance (HLD) or the area between lines (AbL) functions.

The paper is organized as follows. In section 2, we introduce the dataset used to train and evaluate the model. Section 3 presents the model, along with the necessary details for training the model. Finally, sections 4 and 5 evaluate and summarize how well our proposed model solves the seafloor pipeline detection task.

2 Dataset

2.1 Multibeam Echo Sounder

A multibeam echo sounder (MBES) is an active sonar used to gather information on the seafloor. A MBES consists of a transmitter and a receiver array, which is mounted below a vehicle. The sensor transmits a broad across-track and narrow along-track, fan-shaped sound pulse, called a *ping*. Through beamforming [4], in both the transmitter and receiver array, the MBES can capture information at multiple angles in a swath below the vehicle, called *beams*.

The MBES records the time delay between sound pulse transmission and echo reception giving the relative *depth* to the seafloor from the AUV. The sensor also records compensated echo strength, called *reflectivity*.

2.2 Data and labels

The dataset is a collection of fifteen seafloor pipeline inspection missions gathered from different HUGIN AUVs by Kongsberg Maritime and the Norwegian Defence Research Establishment at various locations around the world [1], [3]. The relevant information can be considered a three-dimensional tensor of pings, beams, and channels. The number of pings varies between missions, depending on, for example, mission length. The number of beams is constant within missions but can vary depending on sensor type and configuration. Fourteen missions have 400 beams per ping, while one mission has 254 beams. We consider the *reflectivity* and *depth* channels. Figure 1 shows a 1000 ping image segment example with superimposed labels for both channels, courtesy of Kongsberg Maritime.

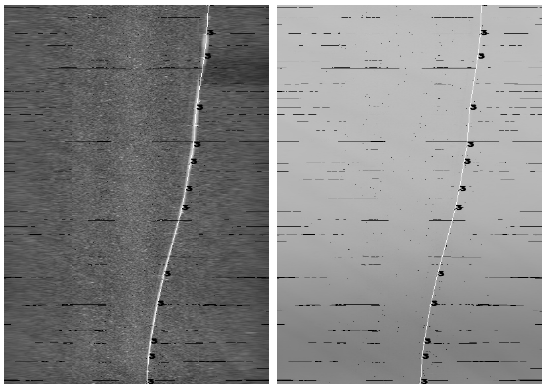
(a) Reflectivity channel (b) Depth channel

Figure 1: Example of a rescaled 1000-ping image segment with a superimposed labeled pipeline. The ping number, and equivalently time, is read from top to bottom, while the beam number goes from left to right. Here, the depth ranges from 3.5-6.5 meters.

The labels are created by manually marking points in MBES image segments, typically 1000 subsequent pings. The points are selected to represent the top of a pipeline, spanning a single beam for each ping. The annotation process was made more efficient and coherent by letting marked points represent endpoints of consecutive line segments. This makes the labeled pipeline top intrinsically locally linear, which we exploit when defining the pipeline localization task and the corresponding loss and evaluation functions.

2.3 Training, validation and test sets

The dataset is split into three, a training set, a validation set, and a test set. The test set contains three missions, while the training and validation sets contain the rest.

The validation set is created by extracting 10% of consecutive pings at a random ping offset within each mission of the training set.

There are 3607340 pings in total, where the training and test set contains approximately 10% of pings each.

3 Model and Implementation

3.1 Architecture

We use ResNet50 [5] without the last two layers, average pooling, and dense layer, as the backbone to our pipeline detection model. Other computationally cheaper or more costly backbone networks, however, can replace ResNet50 with a possible trade-off between detection accuracy and computational cost.

Our model's single input has shape $32 \times 400 \times 2$, representing pings, beams, and channels, respectively. We consider 32 pings because this is the amount of spatial reduction imposed on the input to ResNet50.

Object detection and tracking with deep learning commonly use bounding boxes to infer object location. As explained in Subsection 2.2, however, there is inherent linearity in our labels, i.e., that subsequent annotated points define endpoints of line segments. Therefore, instead of making our model infer location by predicting coordinates to bounding boxes, we make the same coordinates represent endpoints of line segments. This makes our localization contain more information than a bounding box for two reasons; (i) the predicted line segment is a dense representation of the pipeline top (ii) the relative orientation of the pipeline is also indirectly available.

The ResNet50 backbone is fully-convolutional and therefore commutes with translation [6]. In other words, shifting a pipeline in the input will shift its feature representation by an amount proportional to its striding factor in the feature map. We call a mutually exclusive region of the input, corresponding to a feature vector in the feature map for a *grid cell*.

Our model's final layers infer information on each grid cell through five output variables in a similar fashion to single-shot detection models such as YOLO [7], and SSD [8]. One variable is used for the binary classification task c on whether the grid cell contains pipeline or not, while the remaining four regression variables represent two two-dimensional (pipe)line segment endpoints $\underline{x}_1 = (x_1, y_1)$ and $\underline{x}_2 = (x_2, y_2)$ for locating the pipeline. We restrict the model to one detection per grid cell because multiple pipelines are rare, especially in nearby beams. Our model is summarized in Figure 2.

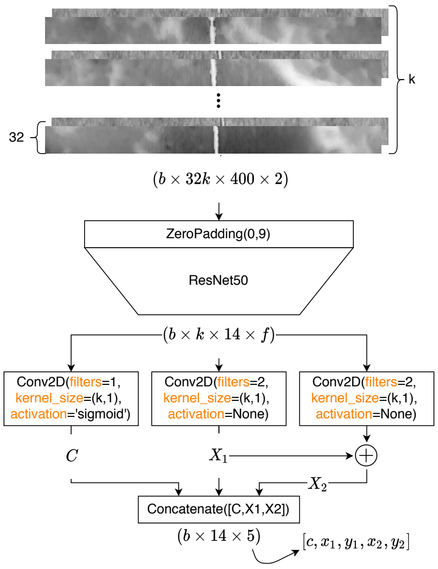
k
32
$(b \times 32k \times 400 \times 2)$
ZeroPadding(0,9)
ResNet50
$(b \times k \times 14 \times f)$
Conv2D(filters=1, kernel_size=(k,1), activation='sigmoid')
Conv2D(filters=2, kernel_size=(k,1), activation=None)
Conv2D(filters=2, kernel_size=(k,1), activation=None)
C
X_1
X_2
Concatenate([C,X1,X2])
$(b \times 14 \times 5)$
$[c, x_1, y_1, x_2, y_2]$

Figure 2: Illustration of the pipeline detection model. k denotes the input image segment length, in factors of 32 pings. b denotes the mini-batch size. $f = 2048$ is the number of features after the final layer of ResNet50.

3.2 Loss function

Similarly to one-shot detection, we want to solve both classification and localization simultaneously. Therefore we need both a classification and a regression loss term and a method to combine them, collectively called a multi-task loss function.

The multi-task loss function in object detection commonly combines cross-entropy, and some variation of mean squared error (MSE) [8], [9]. Cross entropy is used to learn classification, including a no-object class, while mean squared error measures the dissimilarity between predicted and labeled regression variables.

However, individual regression variable dissimilarity measures, such as MSE, do not consider the dissimilarity between the regression variables' collective representation. This makes MSE a poor measure for model evaluation. Furthermore, while a zero MSE would ultimately lead to the desired model prediction independent of the regression variables' geometrical interpretations, we imagine that this is a too crude description of the model's objective. Hence we also propose two additional functions for measuring the dissimilarity of labeled and predicted (pipe)lines.

The classification part of our task is binary, i.e., does a grid cell contain a pipeline or not? Hence we can use the binary cross-entropy (BCE) as our classification loss function $\mathcal{L}_C$. However, pipelines are unevenly distributed among grid cells, and most grid cells are biased towards not containing pipeline. Therefore, we dynamically rescale the BCE loss to be equal in size for both classes, independent of each class's frequency, based on grid cell pipeline occurrence within mini-batches.

The regression loss function $\mathcal{L}_R$ can be formulated through mean squared error as

$$\text{MSE} = \frac{1}{2N}\sum_{i=0}^{B-1}\sum_{j=0}^{S-1} c_{ij}\left(d_1^2 + d_2^2\right) \tag{1}$$

where N is the number of non-zero c_{ij}'s aggregated across the fourteen grid cells S and mini-batch size B. d_1 and d_2 are the Euclidean distances between predicted and labeled line segment end points, as illustrated in Figure 3 and used in Equation 1.

As argued at the start of this Subsection 3.2, MSE does not consider what the regression variables represent. The regression variables represent lines, and we therefore instead propose to use a variant of the Hausdorff Line Distance (HLD) [10], which we express as

$$\text{HLD} = \frac{1}{N}\sum_{i=0}^{B-1}\sum_{j=0}^{S-1} c_{ij}\max_{x\in X}\tilde{d}_x \tag{2}$$

where $X = \{\underline{x}_1, \underline{x}_2, \hat{\underline{x}}_1, \hat{\underline{x}}_2\}$ are the end points of the line segments, and the corresponding distances $\tilde{d}_x$ are as illustrated in Figure 3. Specifically, by letting l and $\hat{l}$ be line segments on a vector space, we can find the projection and subsequently the rejection of each $x \in X$ onto the opposite lines spanned by l and $\hat{l}$. The length of the rejection vector then gives the distances $\tilde{d}_x$.

Alternatively, we can measure line segment similarity as the area between the labeled and predicted lines on the image segment interval (AbL). We calculate this by converting the line representation to the slope-intercept form and integrating the difference between the two terms, giving

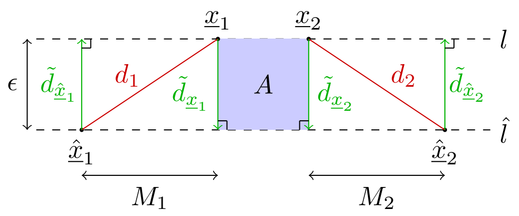
$\underline{x}_1$
$\underline{x}_2$
l
ϵ
$\tilde{d}_{\hat{\underline{x}}_1}$
d_1
$\tilde{d}_{\underline{x}_1}$
A
$\tilde{d}_{\underline{x}_2}$
d_2
$\tilde{d}_{\hat{\underline{x}}_2}$
$\hat{l}$
$\hat{\underline{x}}_1$
$\hat{\underline{x}}_2$
M_1
M_2

Figure 3: Illustration of a predicted line $\hat{l}$ which is similar to a labeled line l, but with dissimilar points defining the lines. Assuming that ϵ is small and that M_1 and M_2 are large, then, MSE (red) is large $\approx (M_1^2 + M_2^2)/2$, while HLD (green) is small $= \epsilon$. A gives the area between the two lines l and $\hat{l}$, here at the interval defined by the endpoints of the line segment of l.

$$\tilde{A} = \frac{1}{2}(a - \hat{a})(\tilde{x}_2^2 - \tilde{x}_1^2) + (b - \hat{b})(\tilde{x}_2 - \tilde{x}_1) \tag{3}$$

where a and b denotes the slope and intercept, respectively. $\tilde{x}_1$ and $\tilde{x}_1$ is the start and end ping (zero and 31) of the image segment. See Figure 3 for an illustration, but note that the endpoints defining l do not have to be equal to the start and end pings of an image interval. It is also necessary to check whether the lines intersect within the image interval, in which case we have to calculate $\tilde{A}_1$ and $\tilde{A}_2$ for the intervals $[\tilde{x}_1, x^*]$ and $[x^*, \tilde{x}_2]$, where x^* is the point of intersection. Subsequently, we add the two absolute areas and aggregate across mini-batches and grid cells, which can be summarized as

$$\text{AbL} = \frac{1}{N}\sum_{i=0}^{B-1}\sum_{j=0}^{S-1} c_{ij}\left(\left|\tilde{A}_1\right| + \left|\tilde{A}_2\right|\right) \tag{4}$$

3.3 Combining Regression and Classification

There are several options for aggregating the loss from multiple objective functions dynamically [11], [12]. However, the dominant approach to multi-task learning in object detection with deep learning is to select a scaled linear combination of the losses manually [7], [8], [13], [14]. In the two-task learning scenario with a classification $\mathcal{L}_C$ and regression loss $\mathcal{L}_R$ function, this can be written as

$$\mathcal{L} = \lambda_1 \mathcal{L}_C + \lambda_2 \mathcal{L}_R. \tag{5}$$

where λ_1 and λ_2 are some chosen scalars. Ren, He, Girshick, *et al.* [13] sets $\lambda_1 = 1$ and $\lambda_2 = 10$ to roughly give equal weight to both terms. Following and expanding upon this equal-weighting principle, we set $\lambda_1 \mathcal{L}_C = \lambda_2 \mathcal{L}_R$ which enforces equal loss magnitude for each term dynamically throughout the training. Assuming that the scales sum to one, the multi-task loss function becomes the harmonic mean which we can write as

$$\mathcal{L} = \mathrm{sg}\left(\frac{\mathcal{L}_R}{\mathcal{L}_C + \mathcal{L}_R}\right)\mathcal{L}_C + \mathrm{sg}\left(\frac{\mathcal{L}_C}{\mathcal{L}_C + \mathcal{L}_R}\right)\mathcal{L}_R \tag{6}$$

where $\mathrm{sg}(\cdot)$ is the stop gradient operator. This is necessary to prevent the model from minimizing the loss $\mathcal{L}$ by minimizing one term while maximizing the latter.

Although interesting, we consider an extensive comparison of multi-task loss scaling methods outside the scope of this work.

3.4 Additional details

A 5x1x1 median filter smoothes image segments before inputting it to the deep learning model.

The diversity of the training data is enhanced with two *data augmentation* methods. First, image segments have a static start index, but training examples are drawn with a discrete, uniformly sampled offset to the start index. Second, each image segment is augmented with Gaussian noise sampled from $\mathcal{N}(0, \alpha)$, where $\alpha \sim \mathcal{U}\{0, 10\}$ which is redrawn for each image segment. Before applying Gaussian noise, each channel of an image segment is standardized, i.e., shifted and scaled with its channel sample mean and standard deviation.

The model is trained with the Adam optimizer [15] with default parameters from TensorFlow 2.0.0, except for the initial learning rate, which we set to 0.2. The learning rate is halved every tenth epoch through a step decay learning rate scheduler.

Image segments are assumed to be independent, which allows us to shuffle image segments between epochs. Every model (trained with different loss functions) are trained for a fixed 50 epochs, with mini-batches of size 64. Our model can learn and infer information on image segments of variable length, as indicated by the k variable in Figure 2. We restrict ourselves to $k = 1$.

4 Experiments and Results

All experiments and results are run on the AI HUB provided by the University of Oslo, which consists of 2 x 14 core Intel CPUs, 4 NVIDIA RTX 2080 Ti GPUs (we only use one), 128GiB RAM, using Ubuntu 16.04. The model is implemented with TensorFlow 2.0.0.

The model evaluation takes into account both classification and regression separately. Classification uses standard evaluation measures such as accuracy, true positive rate, precision, and F1-score, all averaged across image examples and grid cells.

For regression evaluation, we use the regression loss functions MSE, HLD, and AbL. We especially emphasize HLD and AbL because they are more appropriate for evaluating line similarity than MSE, as illustrated in Figure 3.

We create a baseline method or summary statistics that always predict the most likely pipeline scenario. Specifically, the baseline always predicts that pings contain one pipeline at beam 201. This is because 80% pings contain one pipeline, and beam 201 is the most likely location. Beam 201 corresponds to the seventh grid cell, which is also the only grid cell with a higher than 50% chance of containing pipeline in the training set.

Because the baseline method only predicts one pipeline, it is non-trivial to evaluate localization on image segments containing multiple pipelines. Here, we give the baseline an unfair advantage by only evaluating it to the closest labeled pipeline. Thus, the baseline localization evaluation can be interpreted as the distance to the closest labeled pipeline.

The results for classification is summarized in Table 1, and in Table 2 for the regression.

We see from Table 1 that all model entries, including the baseline, achieve high accuracy. The high baseline accuracy confirms that grid cells seldom contain a pipeline, except for grid cell seven. Furthermore, the baseline gets a relatively low TPR, indicating that it fails to predict that a grid

	Acc	TPR	Precision	F1-score
MSE	0.9732	0.8584	0.8422	0.8494
HLD	**0.9734**	**0.8619**	0.8420	**0.8510**
AbL	0.9729	0.8474	**0.8477**	0.8466
B*	0.9173	0.4375	0.5350	0.4809

Table 1: Region-wise **classification** summary results. The three first rows show the model trained with different regression loss functions, while the last row B^* is the baseline. The columns express different classification evaluation functions. Boldfont display the best performance in each evaluation function.

	MSE	HLD	AbL
MSE	**22.955**	2.1757	47.917
HLD	411.81	**1.9664**	**43.576**
AbL	802.42	2.1725	44.696
B*	504.22	13.086	377.66

Table 2: Equivalent to Table 1, but for **localization**. The columns express different regression evaluation functions, as discussed in Subsection 3.2. Regression units are in pixels. Specifically, pings and beams, interchangeably.

cell contains pipeline. Finally, the baseline obtains a relatively high precision which reveals the proportion of test set examples containing a pipeline in grid cell seven.

Because accuracy depends on class distribution which can give a vague classification performance impression, we instead consider TPR and precision to summarize model classification performance. F1-score calculates the harmonic mean between TPR and precision and therefore captures both, making it the favoured classification evaluation function.

All three deep learning models outperform the baseline in every classification evaluation measure. In particular, the deep learning models achieve roughly 35% higher F1-score than the baseline. The model trained with the HLD regression functions attains the highest F1-score.

Similarly, for the regression evaluation, Table 2 shows that the deep learning models outperform the baseline in both HLD and AbL by a factor of about six and eight times, respectively. Conversely, the baseline achieves a lower MSE evaluation than the model trained with AbL. In contrast to HLD and AbL, a high MSE evaluation does not necessarily correspond to line-segment predictions which are dissimilar to labeled line-segments, only that their corresponding line-segment endpoints are dissimilar, as illustrated in Figure 3. We, therefore, emphasize using HLD or AbL for evaluating regression performance.

Finally, we measure the average ($N = 1000$) amount of 32-ping image segments the model can forward pass in one second to 18.7. By assuming an MBES operating at 32Hz, and hardware as described at the start of this section, our model can operate over eighteen times above the real-time requirements.

5 Conclusion & Future work

This work demonstrates that deep learning with ResNet50 and single-shot detection can efficiently and accurately detect pipelines in multibeam echo sounder data. Furthermore, we present two regression functions that can be used both for training and evaluating line prediction. The HLD function also shows promise as a loss function by slightly outperforming the model trained with MSE and AbL.

Our model achieves an F1-score of 85%, which is 35% higher than the most likely baseline, on the region-wise pipeline classification task. Moreover, our model is eight times more accurate than the baseline in locating pipelines. Finally, the model can operate at over eighteen 32-ping image segments per second, making it feasible for real-time operations.

In further work, we envision combining the classification and regression evaluation into a unified function. Similar to mean average precision in main-steam object detection, but replacing the intersection over union with HLD or AbL for classifying when a prediction is a true or false positive. Further work can also extend the detection model to combine consecutive 32-ping detections into a coherent track over arbitrary many pings, for example, by setting $k > 1$ or, by adding RNNs.

References

[1] P. E. Hagen, E. Børhaug, and Ø. Midtgaard, “Pipeline Inspection With Interferometric SAS,” *Sea Technology*, vol. 51, no. 6, 2010.

[2] J. Evans, P. Patron, B. Privat, *et al.*, “AUTOTRACKER: Autonomous inspection - Capabilities and lessons learned in offshore operations,” *MTS/IEEE OCEANS, Biloxi*, 2009.

[3] Ø. Midtgaard, T. Krogstad, and P. E. Hagen, “Sonar detection and tracking of seafloor pipelines,” in *Proc UAM conf*, 2011.

[4] B. D. Van Veen. and K. M. Buckley, “Beamforming: a versatile approach to spatial filtering,” *IEEE ASSP Magazine*, vol. 5, no. 2, pp. 4–24, 1988.

[5] K. He, X. Zhang, S. Ren, *et al.*, “Deep residual learning for image recognition,” *Proc. CVPR*, pp. 770–778, 2016. DOI: 10.1109/CVPR.2016.90. arXiv: 1512.03385.

[6] L. Bertinetto, J. Valmadre, J. F. Henriques, *et al.*, “Fully-convolutional siamese networks for object tracking,” *LNCS*, vol. 9914, pp. 850–865, 2016. DOI: 10.1007/978-3-319-48881-3_56. arXiv: 1606.09549.

[7] J. Redmon, S. Divvala, R. Girshick, *et al.*, “You only look once: Unified, real-time object detection,” *Proc. CVPR*, pp. 779–788, 2016. DOI: 10.1109/CVPR.2016.91. arXiv: 1506.02640.

[8] W. Liu, D. Anguelov, D. Erhan, *et al.*, “SSD: Single shot multibox detector,” *LNCS*, vol. 9905, pp. 21–37, 2016. DOI: 10.1007/978-3-319-46448-0_2. arXiv: 1512.02325.

[9] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” *Proc. CVPR*, pp. 6517–6525, 2017. DOI: 10.1109/CVPR.2017.690. arXiv: 1612.08242.

[10] S. Wirtz and D. Paulus, “Evaluation of established line segment distance functions,” *9th Open German-Russian Workshop on Pattern Recognition and Image Understanding*, pp. 89–93, 2014.

[11] R. Cipolla, Y. Gal, and A. Kendall, “Multitask Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics,” *Proc. CVPR*, pp. 7482–7491, 2018. DOI: 10.1109/CVPR.2018.00781. arXiv: 1705.07115.

[12] O. Sener and V. Koltun, “Multi-task learning as multi-objective optimization,” *CoRR*, vol. abs/1810.04650, 2018. arXiv: 1810.04650.

[13] S. Ren, K. He, R. Girshick, *et al.*, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017. DOI: 10.1109/TPAMI.2016.2577031. arXiv: 1506.01497.

[14] V. N. Nguyen, R. Jenssen, and D. Roverso, “LS-Net : Fast Single-Shot Line-Segment Detector,” arXiv: arXiv:1912.09532v2.

[15] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *Proc. ICLR*, 2015. arXiv: 1412.6980.