Segmentation dataset for reinforced concrete construction

Schmidt, Patrick; Nalpantidis, Lazaros

Published in:
Automation in Construction

Link to article, DOI:
10.1016/j.autcon.2025.105990

Publication date:
2025

Document Version
Publisher's PDF, also known as Version of record

Automation in Construction 171 (2025) 105990

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Segmentation dataset for reinforced concrete construction

Patrick Schmidt *, Lazaros Nalpantidis

DTU - Technical University of Denmark, Department of Electrical and Photonics Engineering, Elektrovej Bygning 326, 2800, Kongens Lyngby, Denmark

ARTICLE INFO

Dataset link: https://doi.org/10.11583/DTU.26213762, https://github.com/DTU-PAS/ConRebSeg

Keywords:
Dataset
Construction robotics
Segmentation
Rebar detection
Shotcrete
Digitization

ABSTRACT

This paper provides a dataset of 14,805 RGB images with segmentation labels for autonomous robotic inspection of reinforced concrete defects. Baselines for the YOLOv8L-seg, DeepLabV3, and U-Net segmentation models are established. Labeling inconsistencies are addressed statistically, and their influence on model performance is analyzed. An error identification tool is employed to examine the error modes of the models. The paper demonstrates that YOLOv8L-seg performs best, achieving a validation mIOU score of up to 0.59. Label inconsistencies were found to have a negligible effect on model performance, while the inclusion of more data improved the performance. False negatives were identified as the primary failure mode. The results highlight the importance of data availability for the performance of deep learning-based models. The lack of publicly available data is identified as a significant contributor to false negatives. To address this, the paper advocates for an increased open-source approach within the construction community.

1. Introduction

The construction industry will be facing a big challenge in the upcoming years. In the European Union (EU), the number of people employed in the construction industry has steadily increased since 2015 [1]. However, so did the number of hours worked per person, except for a little, probably COVID-related dip in 2020 [2]. Both of these statistics suggest that the construction industry will be facing major labor shortages in the future. In addition, the real labor productivity per hour worked has decreased compared to 2015 [2]. This particular trend does not apply to all industries, as the industrial sector without construction experienced an increase in real labor productivity [2]. A major difference between other industries and construction is the lack of adoption of digitization and automation in the construction industry. Construction companies see big potential in the adoption of these technologies, yet seem to have little or no implementation of these in place, according to a 2016 study conducted by Roland Berger [3]. Another study focused on the United States construction industry, conducted by McKinsey&Company, confirms this. In the study, construction is shown to be one of the sectors with the lowest adoption rate of digital technologies [4]. Furthermore, it shows a correlation between the adoption rate of digital technologies and productivity growth, where construction is attested with the most negative growth rate between 2005 and 2015 [4].

Another particularly detrimental fact is that construction consistently shows the highest incidence rate for work-related non-fatal accidents with 3,151.9 accidents per 100,000 people employed in 2021 [5] and the second-highest incidence rate for work-related fatal accidents with 6.32 deaths per 1000 people employed in 2021 [6]. Both issues, labor shortage, and workers' safety, can be alleviated by adopting digital technologies, especially developing autonomous robotic systems for construction activities. These systems would replace workers, addressing the labor shortage situation, and taking workers out of dangerous activities.

One of the key challenges faced during the development and deployment of autonomous robotic systems in construction is the nature of the environment the systems need to operate in. For example, manufacturing environments, where robotic systems are widely adopted nowadays, are structured environments with clear, designated spaces for people, objects, etc. In addition, a manufacturing process's outcome is usually known in advance, making it easier to program robots to follow specific, pre-planned activities. This is generally not the case in construction, and this discrepancy is shown in [7].

To counterbalance the unstructured and highly dynamic nature of construction sites, autonomous robotic systems need to rely on advanced perception capabilities, integrating sophisticated systems of sensors and algorithms to e.g., detect irregularities and navigate safely through a construction site. Most often, these detection algorithms are based on neural networks and require abundant data to train on. However, unlike other industries, the construction industry lacks this abundance of data. Our work addresses this issue and provides a new dataset focused on reinforced concrete construction processes, specifically the inspection and repair of reinforcement bars. The dataset

* Corresponding author.
E-mail addresses: pasch@dtu.dk (P. Schmidt), lanalpa@dtu.dk (L. Nalpantidis).

https://doi.org/10.1016/j.autcon.2025.105990
Received 16 July 2024; Received in revised form 7 January 2025; Accepted 17 January 2025
Available online 28 January 2025

consists of 14,805 RGB images, both from our own capturing sessions and publicly available data from YouTube. The images show different scenarios of shotcrete construction in both construction and repair phases, such as the construction of ground support walls, tunnels, and culverts, as well as the repair of piles and beams. With these images, we tackle the task of image segmentation for robotic inspection, repair, and construction of reinforced concrete structures. The images are provided with labels, which can be used to train deep learning-based segmentation algorithms for solving the segmentation task, which is a crucial element in enabling robots to autonomously perform construction tasks where reinforced concrete structures are involved. Such autonomous systems are currently in development, for example by the RobétArmé project [8], which is developing an autonomous robotic solution to shotcrete construction and benefits from datasets like ours for training deep learning-based methods for semantic understanding of the construction environment [9]. We demonstrate that the dataset can be used to train a working segmentation model by establishing baselines for the state-of-the-art models DeepLabV3, U-Net and YOLOv8L-seg. We perform experiments analyzing the influence of data availability on the performance of YOLOv8L-seg. We show that the model can be successfully used on an embedded GPU typically found in robotic platforms. Furthermore, as opposed to classic Computer Vision (CV) datasets, our dataset is scene-based and of sequential nature, i.e., the images have a temporal order, reflecting the spatio-temporal nature of robotic tasks. This property is used to identify label inconsistencies and analyze the effect of these on the performance of the model. Based on these experiments, we identify construction-related challenges that hinder the application of these technologies in the construction domain and propose steps to alleviate them.

To summarize, the contributions of our work are as follows:

- A new open source dataset of construction imagery for reinforcement concrete construction with segmentation labels for four categories: exposed reinforcement bars, persons, cars, and trucks.
- An analysis protocol for label inconsistencies to analyze the influence of multiple valid labeling styles on model performance, i.e., opinions of different engineers regarding the extent of a defect.
- Introducing emphasis on spatio-temporal properties when employing CV techniques in a construction robotics context
- Investigating the influence of data availability and pre-training on the performance of the YOLOv8L-seg segmentation model.
- A discussion about challenges faced when applying deep learning-based methods in the construction sector, based on our experimental findings.

2. Related work

The adoption and development of digital technologies in the Architecture, Construction, and Engineering (ACE) sectors is an ongoing and active field of research. The usage of autonomous systems relies on trustworthy and reliable perception capabilities of robots, which nowadays mostly rely on deep learning-based CV.

2.1. General overview

CV can be employed for various tasks, starting from general-purpose site understanding tasks for robot navigation [10], site monitoring for safety and management purposes [11–14] to more specific process-related tasks: Numerous works cover the task of crack segmentation [15–18], where traditional and deep learning-based computer vision techniques and models are deployed to do a pixel-level (dense) detection of cracks on surfaces, mostly concrete surfaces. A broader, associated task to this is the detection of damaged, reinforced concrete, where the goal is to detect not only surface cracks but also spallations, efflorescences, and exposed reinforcement bars (rebars) [16,19].

2.2. Reinforcement bar detection

Rebars are an important element at all lifecycle stages of structures, as they are of vital importance for their integrity. In early lifecycle stages, i.e., when structures are to be newly built, CV can be used to locate and assess the quality of rebar lattices/mats [20–23], specifically the spacing and linkage between rebar elements as a crucial parameter to their contribution to the structural integrity of bridges and buildings. While lattices have a characteristic regular structure, there also exists work on locating individual bars, e.g., in precast reinforced concrete [24]. During the lifetime of a building or other infrastructure, it is important to monitor structural health. Defects can either appear on the surface in the form of spallations and exposed rebars—as mentioned previously, or underneath the surface. Sub-surface structural health can be surveyed using CV techniques on Ground Penetrating Radar (GPR) data [25–27] as opposed to commonly used RGB cameras and 3D Light Detection and Ranging (LiDAR) sensors.

2.3. Construction robots

The previously mentioned related work focuses on inspection of structures and site monitoring and shows plenty of research activity in these areas. They provide a good basis for the adoption of autonomous systems, which rely on semantic understanding capabilities to navigate construction sites and carry out construction activities. Examples of such systems are excavators capable of autonomously building stone walls [28] or autonomous surface excavation [29]. Other examples are autonomous rebar tying robots [30,31] and autonomous 3D concrete printing robots with rebar placement [32]. A pipeline detecting exposed rebars from RGB images and reconstructing a 3D model from stereo images for autonomous robotic placement of shotcrete is presented in [9,33]. [34] developed a proof of concept showcasing the robotic application of shotcrete, and [35] shows an algorithm to identify and reconstruct steel rebar meshes. A thorough review of the state of construction robotics can be found in [36].

2.4. Datasets

The lack of publicly available datasets in the ACE domain is a common problem and poses a barrier to the development of deep learning-based solutions. The few (semi-)publicly available datasets addressing the ACE domain range from general-purpose object detection datasets for site monitoring and navigation to more task-specific datasets. SODA [37] is such a general-purpose dataset consisting of around 20,000 images with around 268,200 bounding boxes, depicting 15 classes of categories *Person*, *Material*, *Machine* and *Layout*. In [37], other general-purpose datasets are listed as well, such as MOCS [38], a dataset for detecting moving machinery and workers, ACID [39] for detecting construction machinery, and CHV [40] for detecting personal protective equipment (PPE). CIS [11] is another general-purpose dataset for detecting people, machines, and materials. However, as opposed to the aforementioned datasets, this dataset features instance segmentation annotations instead of just bounding boxes. Out of these five datasets, only CHV and CIS are publicly available, while the others are available only upon request.

Task-specific datasets are more tailored to solving a specific construction task. Table 1 gives an overview of datasets used in the works in Section 2.2 depicting rebars for various tasks. As the table shows, availability of data is low, which poses a barrier to the development of robust, deep learning-based solutions. Not every work makes a data availability statement, and more often than not, the data is only available upon request. Only [41] provides their data without any additional barriers. We want to overcome this barrier by releasing our dataset to the public, hoping that it will motivate other researchers to make also their data publicly available.

Table 1
Examples of rebar detection datasets.

	[42]	[25]	[22]
No. images	256	3992	240
No. objects	more than 10,000 crosspoints	13,026	N/A
Modality	RGB-D	Ground penetrating radar	RGB + Stereo
Annotation type	Keypoints	Bounding Boxes	Instance Segmentation
Image sizes	N/A	N/A	1280 × 720
Domain	real	real	real
Task	Crosspoint detection for rebar tying	Non-invasive localization of rebars in concrete	Quality assessment of rebar installations
Availability	No availability statement	No availability statement	No availability statement
	[41]	[21]	[23]
No. images	2500	192	3130
No. objects	N/A	N/A	N/A
Modality	RGB	RGB + LiDAR	RGB
Annotation type	Instance Segmentation	N/A	Semantic Segmentation
Image sizes	1280 × 720	N/A	N/A
Domain	synthetic	real	real
Task	Instance Segmentation of Rebars	Instance Segmentation of Rebars for spacing analysis	Segmentation of Rebars for Quality Control
Availability	public	Upon request	Upon request

3. The ConRebSeg dataset

The following section introduces and carries out a deep analysis of our dataset. We will start with general properties such as image sizes, number of objects, and object sizes, and will also analyze aspects that are more specific to our dataset.

3.1. Overview

We have collected a dataset of 14,805 images captured mostly in construction environments. The main purpose of this dataset is to collect a broad set of images depicting exposed rebars in the context of shotcrete construction, aimed at autonomous inspection, repair, and construction of reinforced concrete structures. Our dataset contains scenes from various shotcrete construction tasks, such as the construction of ground support walls, tunnels, and culverts as well as the repair of piles and beams. This captures the diverse appearance of exposed rebars. The scenes consist of temporally ordered images, reflecting the robotic application this dataset targets.

We provide segmentation masks for four different classes:

- **Exposed rebars:** These can be whole rebar lattices, i.e., in ground support wall construction, or partially exposed lattices and single bars in defective concrete.
- **Persons:** These will usually be construction workers in PPE.
- **Cars:** Those are regular-sized passenger cars.
- **Trucks:** This is usually heavy construction machinery or material delivery trucks.

Given the embedding of this work in the RobétArmé project,[1] which develops an autonomous robotic system for shotcrete inspection, repair and construction, we put our main focus on the exposed rebars class since it is important to identify these areas as they need to be filled with shotcrete. The other classes were chosen as sanity check classes, aimed at supervising the annotation process and empirically assessing the quality of the annotations.

In total, there are 54,115 instances, distributed as follows: 41,237 exposed rebars, 11,275 persons, 330 cars and 1273 trucks. Our dataset is a mix of self-collected data, for which the setup is described in Section 3.2, and YouTube videos.

3.2. Acquisition of data and labeling process

The scenes in the dataset stem from two sources: self-collected data we acquired during construction site visits and data from YouTube. The camera we used in our data collection visits is a Basler a2A1920-160ucBAS RGB camera fitted with a Basler C125-0418-5M-P f4 mm lens. The camera's exposure time is set to 5 ms, with automatic white balancing and the light source preset is set to `Tungsten2800K`. To avoid noisy images, we set the noise reduction and sharpness enhancement parameters to −10,000. We performed two data collection sessions for our self-captured data. The collection has been conducted in a hand-held manner, so as not to introduce any platform-specific characteristics into the data, possibly jeopardizing a general applicability of the dataset on different robotic platforms. A summary of the two recording sessions is included in the first two columns of Table 2.

Arranging construction site visits always relies on partners with connections to the construction industry, which can be a great barrier in the process, since not every site owner is willing to let outsiders in. That is why in addition to the self-collected data, we have also identified publicly accessible videos on YouTube depicting shotcrete construction activity, which has allowed us to rapidly grow the dataset. However, the frames of those videos are not directly included in the dataset, but rather their annotations together with a mapping to a specific frame of the video. This is due to legal uncertainties regarding the distribution of downloaded YouTube videos.[2] Instead, we provide

[1] https://www.robetarme-project.eu/

[2] In their service terms, YouTube states that "*You are not allowed to:* [...]*1. access, reproduce, download, distribute, transmit, broadcast, display, sell, license, alter, modify or otherwise use any part of the Service or any Content except: [...] (c) as permitted by applicable law;*" and "[...] *access the Service using any automated means (such as robots, botnets, or scrapers) except*: [...] *(c) as permitted by applicable law;*" [43]. We stipulate that the EU DSM directive [44] and its national implementation in Denmark [45] are fulfilling the conditions on which users are allowed to download data from YouTube as per their service terms.

Table 2
Metadata of the three constituent parts of ConRebSeg.

Location	Vester Sogade, 1601 Copenhagen, DK	Langebro, 1219 Copenhagen, DK	Web (YouTube)
Short Description	Balcony repair works with exposed rebars, both individual and in lattice format	Repair of bridge piles and beams in challenging lighting conditions	Mix of ground support walls, piles and beams, and tunnels and culverts
Environmental conditions	outdoors, cloudy, naturally illuminated	indoors, dark, artificially illuminated	varying
Number of scenes	12	14	51
Number of frames	4911	7270	2624
Average frame rate	1.8 FPS	3.8 FPS	N/A
Resolution (width/height)	1920 × 1200	1920 × 1200	varying

Fig. 1. Visual examples provided as annotation guidelines.

a way for users to obtain the data themselves in an automated manner for scientific/research purposes. Users should check the legality of this process in their respective jurisdictions. Once we have collected the data, we annotated them to obtain masks for the previously listed object classes. To make the annotation process privacy-preserving, we have treated all scenes with the face-blurring algorithm Deface [46], based on the CenterFace [47] face detector. Furthermore, we have treated all frames of the scenes gathered at Langebro (see Table 2) with histogram equalization and AI-denoising based on NAFNet [48] to alleviate the rather low-illumination, low-contrast frames. This facilitates the annotation process and avoids low-quality annotations due to invisible image features. However, we provide the original, but anonymized images in our dataset to give users the freedom to pre-process the frames according to their needs while maintaining the needed level of privacy. The experiments conducted in Section 4 use the raw version of the images without any level of anonymization, and we stipulate that the effect on the outcome of the experiments remains minimal.

While classes such as *person*, *truck*, and *car* have some common sense associated with them, the *ExposedBars* class is rather domain-specific and not easy to describe. We set the following description for the annotation process: **area** with exposed rebars, where the tips of the bars should be polygon points and roughly outline the area. Along that, we gave some visual examples, as illustrated in Fig. 1.

3.3. Statistics

Our dataset is split into a training (train), validation (val), and testing (test) set. The split was done ensuring an equal distribution of class instance counts. Fig. 2 shows the relative distribution of the count of object instances in the respective splits. Fig. 2 also shows a clear dominance of the *ExposedBars* class, followed by the *person* class. We neglect this class imbalance as we are mainly interested in the ExposedBars category, given that there exists a plenitude of segmentation models trained on general-purpose datasets like MS COCO [49], Pascal VOC [50] or Cityscapes [51] for the remaining categories in our dataset. In total, there are 8570, 3543, and 2692 samples for the train, val, and test split respectively. Table 3 shows key statistics about the splits and their annotations.

Table 3
Label statistics, per dataset split.

	train	val	test
Average number of objects per image	3.79	3.01	4.08
Average mask size in pixels	77,151	47,408	115,337
Average bounding box size in pixels	141,703	133,770	200,729

Table 3 shows that the average object count per image stays roughly the same throughout the splits, however, the mask and bounding box size vary. The train set is a balance point, as it lies between the val and test split in terms of object count and size. Fig. 3 shows histograms of the centroids of bounding boxes on the left and masks on the right respectively, for the train, val, and test split in the first, second, and last row respectively. Especially for the train split we can observe a clear bias towards the center of the picture, which is not as expressed in the val and test splits, where the centroids lie still vertically centered, however with some horizontal spread.

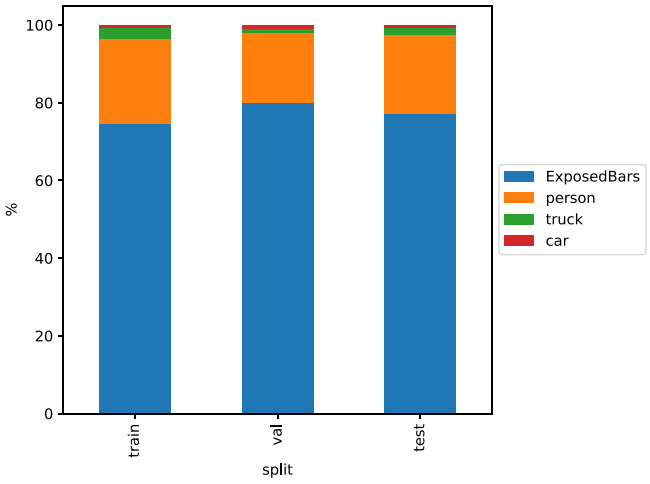


Fig. 2. Class distribution within each split.

3.4. Distribution and structure of the dataset

We provide a public repository that contains our data and other elements that facilitate viewing, exploring, and experimenting with the dataset. We use the FiftyOne dataset management framework [52] that provides end-users with a web interface for viewing samples and managing metadata such as train/test/val tags as well as other tags describing certain properties of samples. Our repository contains scripts that set up the dataset for the user, especially obtaining the data from YouTube, as we do not provide the frames directly for reasons specified in Section 3.2. In general, the data is organized as listed in Fig. 4.

The angle brackets in Fig. 4 are placeholders and denote the meaning of the fields. `dddd` are decimal seconds, `ns_epoch_timestamp` denotes a timestamp in nanoseconds in Unix Epoch time (since Jan. 1, 1970), `%04d` means that frame numbers up to 1000 are zero-padded.

The YouTube frames are downloaded as whole videos and then are converted to frames, however, only every 200th frame is used since the informational value of the frames in between is minimal due to minimal variations of the depicted scene. For integrity purposes, we save a MD5 checksum for every sample in the dataset, which is saved as a metadata field in the dataset management tool. At import time, every frame is checked against this checksum to ensure the integrity of the dataset.

3.5. Annotation styles/cleanup

A phenomenon we observed while analyzing the segmentation masks is that the same object has been annotated in different ways. This was to be expected since we shuffled the frames within scenes collected at Langebro (see Table 2) to remove the temporal dependencies within frames since we wanted to analyze how different annotators will label the *ExposedBars* category. Fig. 5 shows three consequent frames with minimal changes depicting the same object, along with the mask.

It can be seen that the mask drastically changes in appearance, moving from a continuous "area" annotation to a more fine-grained, "rebar" annotation. While our label description explained that "areas" should be annotated, we conclude that both styles are valid annotations that serve different purposes. The styles could reflect "expert" opinions, which both need to be respected. However, we assume that two vastly different masks for very similar images might impede the training of deep learning-based segmentation models. We will elaborate on this in Section 4.4 in more detail, and describe an approach to separate the different styles in the following.

Fig. 5 suggests an indicator being the temporal development of the number of objects (per-class) in an image. The object count from the left picture in Fig. 5 to the middle one suddenly increases from 1 to 7 and then goes back to 1 in the last picture. We, therefore, extract the number of unconnected masks of the same class per frame and create a time series from it, respecting the temporal order of the frames. A local inconsistency is then determined by a rolling window approach of taking $n = 10$ frames centered around the current frame, determining their 0.90 quantiles $q_{.90}(t)$ of the object count, and marking a frame as an anomaly if the number of objects in the frame exceeds $q_{.90}(t)$. Formally, the rule is expressed as follows:

$$\text{anomaly}(t) = \begin{cases} 1, & \text{if } n_{\text{obj}}(t) > q_{.90}(t) \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

with 1 indicating an anomaly and 0 indicating that something is adhering to the "common" style. We now use the train scene `run_2023_06_07_10_05_31` from the Langebro scenes as an example. Fig. 6 shows a plot of the time series, with the number of objects $n_{obj}(t)$ in a frame noted on the y-axis. The green dashed line indicates $q_{.90}(t)$, and any point above this line is considered an anomaly and is denoted as an orange dot. Based on this rule, we tag the frames within a scene with an *anomaly* tag if they fulfill it.

Figs. 7(a) and 7(b) show the result of the tagging process on Langebro scene `run_2023_06_07_10_05_31`. We have randomly sampled four frames from samples without (Fig. 7(a)) and with the anomaly (Fig. 7(b)) tag. Clearly, the samples tagged as anomalies show the more fine-grained, single-bar annotation style as opposed to an area annotation. We therefore proceed to use the tags in Section 4.4 to distinguish between the labels and carry out experiments related to this phenomenon.

Table 4
Hardware Setups used in our experiments.

	A	B	C
CPU	Intel Xeon Gold 6126/6142/6242	Intel Xeon Gold 6226R/6326	AMD Ryzen Threadripper PRO 5945WX
GPU	NVIDIA Tesla V100	NVIDIA Tesla A100	NVIDIA GeForce RTX 3090
OS	Scientific Linux 7.9	Scientific Linux 7.9	Ubuntu 22.04 LTS

4. Experiments

This section presents the various experiments conducted on the dataset. We will first present our infrastructure used in the experiments. Then, we will establish some baselines for different semantic and instance segmentation models, and carry out experiments regarding data efficiency and the influence of certain labeling styles on the performance of the models.

4.1. Setup

For our experiments, we use three different setups, as listed in Table 4. All the models used in the experiments are based on PyTorch [53] and trained on CUDA-enabled GPUs. To ensure consistent and comparable results within and across models, we enable deterministic training and initialize the random number generators with fixed seeds.

4.2. Baseline training

This section establishes baselines for various models to compare future experiments against. We will establish baselines for 3 models: YOLOv8L-seg, DeepLabV3, and U-Net. U-Net and DeepLabV3 are semantic segmentation models, i.e., produce a dense pixel map for the whole image. YOLOv8L-seg is an instance segmentation model, i.e., it

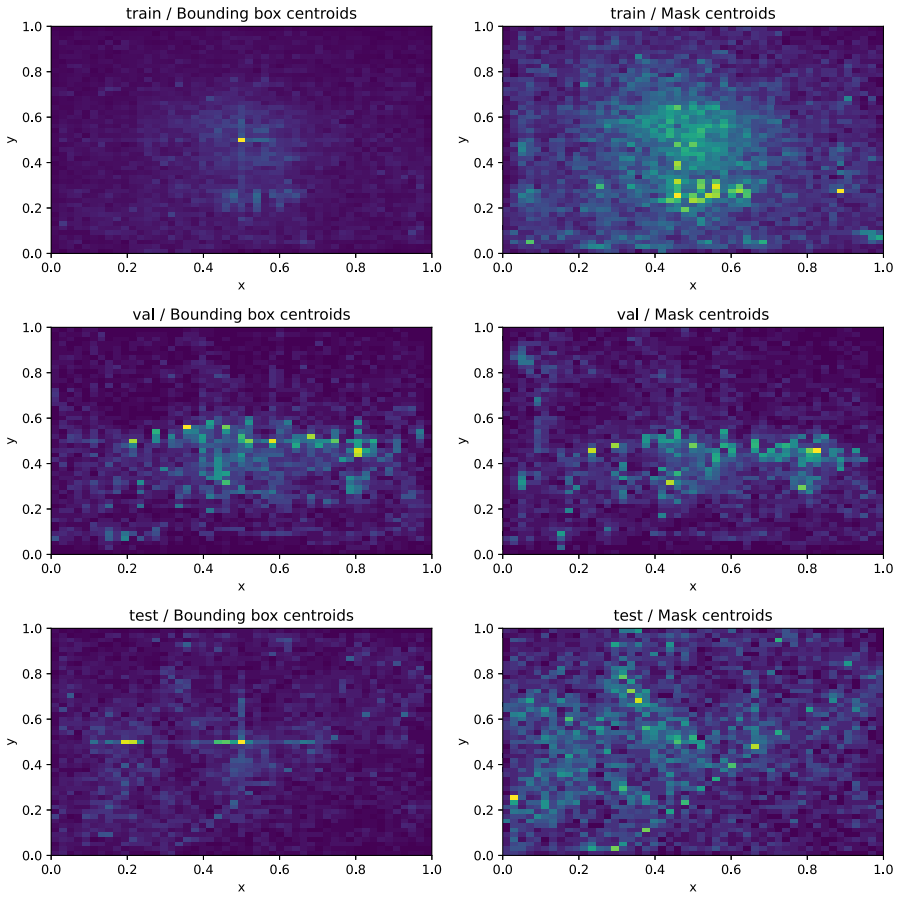


Fig. 3. Histogram of normalized label centroids in the train (upper row), val (middle row), and test split (bottom row) of the ConRebSeg dataset, for the mask bounding boxes (left) and for the mask itself (right).

```
data/
├── langebro/
│   └── run_<year>_<month>_<day>_<hh>_<mm>_<ss>/
│       └── _<year>_<month>_<day>_<hh>_<mm>_<ss>.<dddd>.jpg
├── vester_sogade/
│   └── _<year>-<month>-<day>-<hh>-<mm>-<ss>/
│       └── <ns_epoch_timestamp>.png
└── youtube/
    └── youtube_id/
        └── frame_<%04d>.jpg
```

Fig. 4. File system structure of the dataset.

is based on an object detection model and produces a binary mask within each bounding box detection. The aforementioned models are all CNN-based models.

Regarding the handling of the output of instance segmentation models, we assemble an image-level mask by pooling the instance predictions and their corresponding masks together, i.e., we perform a logical OR operation. If two detections occupy the same pixel and have different class labels, we resolve this by assigning the pixel the class label of the prediction with the higher confidence score. Unless otherwise specified, we train each model with the recommended default hyperparameters, and only keep detections and their corresponding masks with a confidence score equal to or greater than 0.25, and report macro-level metrics. This means the metric is computed per category and then averaged with equal weighting. All models except U-Net have around 45 million trainable parameters, making it fair to compare them against each other.

In the first round, we train from scratch with default optimizer hyperparameters, and we only consider data from the first 100 epochs. For all models, we use an image size of 640 × 640 pixels. YOLOv8L-seg uses extensive image augmentations, whereas DeepLabV3 and U-Net use only `RandomHorizontalFlip (RHF)` and `RandomVerticalFlip (RVF)` augmentations, both with $p = 0.5$. Table 5 summarizes the hyperparameters used for each model.

Table 6 shows the results of the baseline experiments. While DeepLabV3 and U-Net achieve similar performances in regard to mean Intersection Over Union (mIOU) on the val split (mIOU ≈ 0.3), YOLOv8L-seg almost performs a magnitude better with a val mIOU of approximately 0.5. Given the superior performance of YOLOv8L-seg compared to the other two models, we proceed to conduct further experiments based on the YOLOv8L-seg model. A straightforward experiment is to start the training from pre-trained weights provided by the developers of the model. Those pre-trained weights stem from pre-training on the large-scale MS COCO dataset, which consists of 330,000 images [49]. The result of this experiment is listed in Table 7 and shows improvement compared to the model trained from scratch. Therefore, trainings we conduct in further experiments will also be started from pre-trained weights.

Fig. 5. Sequence of successive frames with their labels (in red) showing an inconsistency in labeling style. The left and right frame share the same annotation style, the center frame has a different style.

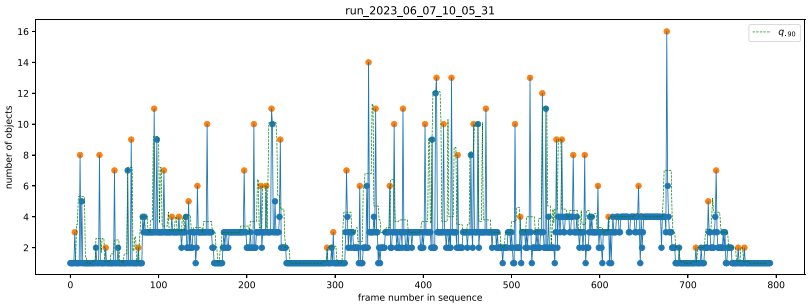


Fig. 6. Plot of the number of objects $n_{obj}(t)$ (on y-axis) per frame (on x-axis) in the Langebro scene `run_2023_06_07_10_05_31`. Orange dots indicate anomalies. The dashed green line indicates the local .90 quantile $q_{.90}(t)$.

(a) Randomly sampled non-anomaly frames (b) Randomly sampled anomaly frames

Fig. 7. Comparison between frames and their labels (in yellow) without the anomaly tag (left) and with the anomaly tag (right)

Table 5
Training hyperparameters used in establishing the baselines.

	DeepLabV3	U-Net	YOLOv8L-seg
Image size	640 × 640	640 × 640	640 × 640
Batch size	8	8	16
Augmentations	RHF and RVF with $p = 0.5$	RHF and RVF with $p = 0.5$	recommended default[a]
Weight Initialization	scratch	scratch	scratch
Optimizer	Adam with `PolynomialLR` scheduler	Adam	SGD with momentum
Optimizer learning rate	0.001	0.001	0.01, momentum 0.9
Epochs	200	200	200

[a] https://docs.ultralytics.com/usage/cfg/?h=augmentation#augmentation-settings.

4.3. Effect of withheld training data

With the established baselines being trained on 100% of the training data, we want to analyze the influence of withheld data on the performance of the models. This gives an intuition about how much variation the data introduces and how diverse our dataset is. There are two ways to withhold the data; one can either withhold whole training scenes, or withhold samples from every training scene. We opted for the former to actually reduce the variation in data.

It is important to ensure that the withheld data does not change the data distribution too much, as this drift would introduce performance implications already. Furthermore, it makes comparing the various experimental results unfair. Fig. 8 shows the instance class distributions for each new split `train_wX`, where `X` is the percentage of withheld training data. The percentage is calculated on the number of samples, not the number of instances. The left plot shows the decreasing number

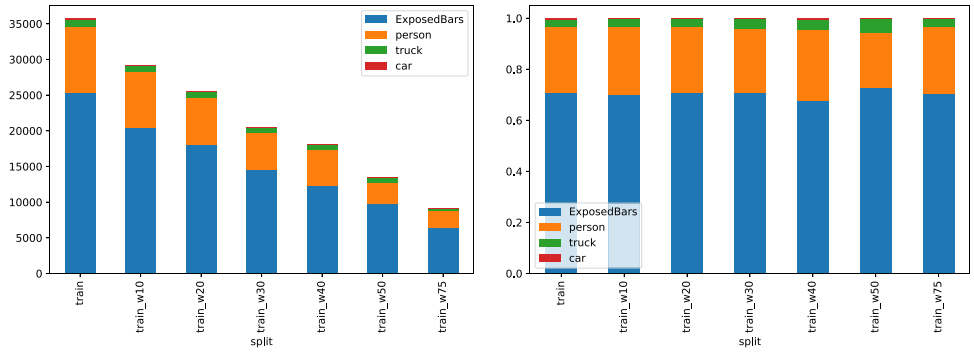


Fig. 8. Instance class distribution of train split, in absolute values (left) and relative values (right) for different amounts of withheld training data.

Table 6
Baseline results.

	DeepLabV3	U-Net	YOLOv8L-seg
Checkpoint Epoch	44	78	100
mIOU (train)	0.705	0.574	0.687
mIOU (val)	0.341	0.301	0.516
Mask mAP_{50-95} (val)	N/A	N/A	0.216
Box mAP_{50-95} (val)	N/A	N/A	0.282

Table 7
Influence of pre-trained weights on model performance.

	YOLOv8L-seg (scratch)	YOLOv8L-seg (pre-trained)
Checkpoint Epoch	100	58
mIOU (train)	0.687	0.681
mIOU (val)	0.516	0.544
Mask mAP_{50-95} (val)	0.216	0.317
Box mAP_{50-95} (val)	0.282	0.374

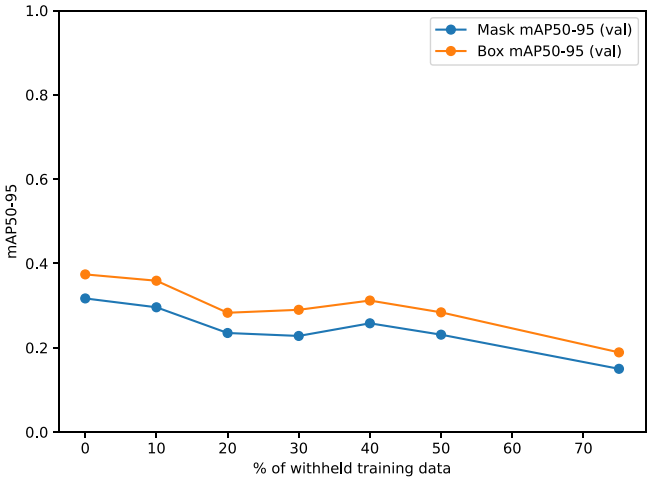


Fig. 9. Best mask mAP_{50-95} value (y-axis) achieved during training, measured on the val split, per portion of withheld training data (x-axis).

of instances with more and more withheld data, while the right plot shows the approximately equal class distribution, with the first bar showing the class distribution of the original train split without any withheld data.

We train the YOLOv8L-seg instance segmentation model on the training data with X percent of it withheld. We start our training from pre-trained weights provided by the developers of YOLOv8L-seg and use default hyperparameters. Figs. 9 and 10 show the results of this experiment.

In Fig. 9, the blue line shows the best value of the mask mean Average Precision $(mAP)_{50-95}$ metric achieved during training on the *y*-axis, measured on the val split of the dataset at a certain percentage of withheld training data on the *x*-axis. The first point on the line shows the performance of the baseline with no withheld training data, corresponding to a value of 0.317. We observe a significant drop in performance at 20% of withheld training data and again at 75%. In Fig. 10, we also report the mIOU metric per class and the percentage of withheld training data. The right plot (results on val split) shows a similar behavior, with the mIOU dropping significantly at 75% of withheld training data. This is particularly visible for the *person* class. We also observe a high discrepancy in performance for the *car* class compared to the rest of the classes. However, a quick peek at the left plot of Fig. 8 reveals a negligible amount of instances for the *car* class, explaining the poor performance of this class in general. Given the prominence of the *ExposedBars* class in the dataset, the performance rather remains constant when withholding training data, which indicates that the data captures enough variation to train a robust model. The performance of other classes could for example be improved by using common segmentation datasets like MS COCO and including those in the training of the model.

4.4. Effect of different labeling styles

Starting from the tagging process for anomalous samples as outlined in Section 3.5, we analyze the influence of the presence of different annotation styles associated with visually similar samples. We withhold Langebro training samples tagged as anomalies, followed by a training of the YOLOv8L-seg model with the same training settings as the established baselines. To ensure a fair comparison, we also consider the amount of withheld data in the train and val split and use the results from Section 4.3 to have a second, more appropriate baseline to compare against. In general, when analyzing the effect of different labeling styles on model performance, we propose the following protocol for fair comparison:

1. Perform experiments withholding certain amounts of training data while ensuring similar class distributions across the original and the smaller datasets.
2. Identify deviating labeling styles in training and validation data and remove frames containing these.

    (a) Possibly, remove more samples to ensure a similar class distribution compared to the original training and validation data

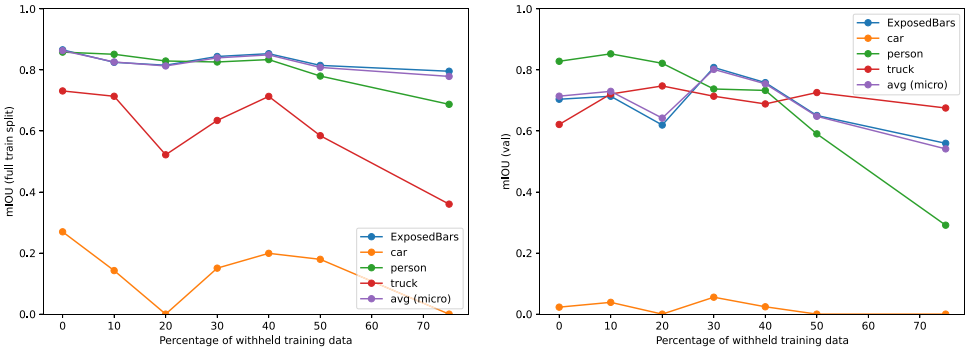


Fig. 10. mIOU value (y-axis) of models trained on a percentage of withheld training data (x-axis), on the full train split (left) and val split (right). The purple line shows the micro average (class-agnostic mIOU), the other lines per-class mIOU.

Table 8
Instance counts of baseline train split and with anomalous samples removed.

	ExposedBars	person	truck	car	Sum
baseline	24,985	9,105	996	180	35,266
non-anomaly	23,598	9,047	996	180	33,821
Difference to baseline	−1,387	−58	0	0	−1,445
relative	−5.5%	−0.6%	0	0	−4.1%

Table 9
Instance counts of baseline val split and with anomalous samples removed.

	ExposedBars	person	truck	car	Sum
baseline	8,660	3,225	243	91	12,219
non-anomaly	8,415	3,133	243	91	11,882
Difference to baseline	−245	−92	0	0	−337
relative	−2.8%	−2.9%	0	0	−2.8%

3. Re-train segmentation model on modified training data
4. Evaluate model on modified validation data
5. Compare obtained metrics to an appropriate baseline. Here, select the metrics of a model from Step 1 where the amount of training data is roughly the same as the amount of training data used in Step 3.

In total, applying the rule outlined in Section 3.5 and following the previously mentioned protocol, we omit 150 samples from the train and 27 samples from the val split respectively. The effect of the instance counts on the train split are outlined in Table 8, for the val split in Table 9. Relatively speaking, the different annotation styles are more explained in the train split than in the val split, however, the differences are not too big, making it fair to conclude the general effect of this phenomenon in our dataset. The results of this experiment are listed in Table 10.

The mIOU performance on both the train and val split has increased slightly. The same is valid for the val mask and box mAP_{50-95}. The results of the training together with the small number of redacted samples and instances lead to the conclusion that the effect of these anomalous labels is minimal and can be disregarded.

We verify this by conducting another experiment, constructing a batch of seven consecutive samples (frame-label pairs). From the frames, we create a mean image to have a common input. We then pass this input through a DeepLabV3 model with a ResNet50 backbone, and use pre-trained weights (ImageNet) to create outputs. We then calculate the cross-entropy loss and evaluate the gradient of each loss with respect to all the model's parameters and investigate the cosine similarity of the respective gradients. Conflicting samples would then show a strong negative cosine similarity, whereas supporting samples would exhibit a strong positive cosine similarity. Fig. 11 shows the data used for this experiment. Fig. 12 shows the mean cosine similarity between the gradients for each sample loss.

Table 10
Comparison between pre-trained baseline and pre-trained model with anomalous samples removed for training.

	YOLOv8L-seg (pre-trained)	YOLOv8L-seg (pre-trained, non-anomaly)
Checkpoint Epoch	58	18
mIOU (train)	0.681	0.711
mIOU (val)	0.544	0.572
Mask mAP_{50-95} (val)	0.317	0.304
Box mAP_{50-95} (val)	0.374	0.385

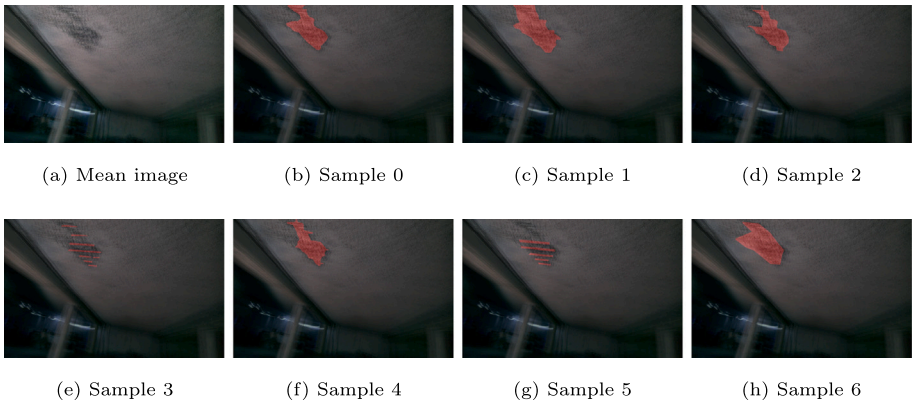
All values are positive, meaning that there is no gradient conflict among samples' loss gradients. There are also no values that strike out from the general picture. Therefore, this experiment supports the previously drawn conclusion that the influence of these different labeling styles is minimal and can be disregarded.

5. Results

In this section, we report results for the YOLOv8L-seg model trained from pre-trained weights on 100% of the training data using default hyperparameters, as it was the best-performing model in our experiments conducted in Section 4. For this final evaluation, we are also using the test split of the dataset. We will also identify common modes of failure and present ways to mitigate these. To verify the results, we have conducted additional verification runs with different seeds and report the results (min, mean, max) in Table 11. Additionally, to reflect the robotic nature of the task, we have exported the best performing model of the verification runs to a TensorRT engine and tested the average FPS on an NVIDIA Jetson Xavier NX embedded GPU, with results reported in Table 12, as high performance desktop GPUs are often not feasible on robotic platforms.

For all metrics, the performance is roughly stable within a split, with only small fluctuations. When measuring the same metrics on the different splits of the dataset, the performance naturally decreases. This is a sign that the model is overfitting and is usually alleviated by introducing more variation in the training data, accompanied by regularization techniques. Table 11 also shows that the drop from val to test is not as drastic as from train to val, indicating that the validation split is representative of the test split as well.

When analyzing mean mask precision and recall, we see that the model performance is heavily impacted by a significant drop in recall.



(a) Mean image (b) Sample 0 (c) Sample 1 (d) Sample 2

(e) Sample 3 (f) Sample 4 (g) Sample 5 (h) Sample 6

Fig. 11. Input data (labels in red) used to analyze gradient interactions between samples.

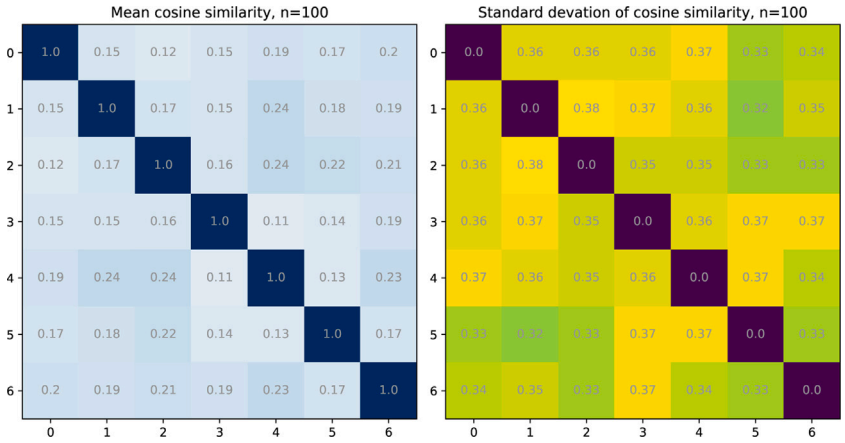


Fig. 12. Cosine similarities between sample loss gradients, left mean cosine similarity, right standard deviation of cosine similarity. The tick labels are sample indices as listed in Fig. 11.

Table 11
Final results, descriptive statistics based on four verification runs with equal configuration and different seeds.

	min			mean			max		
	train	val	test	train	val	test	train	val	test
mIOU	0.660	0.555	0.329	0.684	0.571	0.345	0.702	0.590	0.367
Mask mAP_{50-95}	0.487	0.287	0.169	0.514	0.292	0.174	0.532	0.298	0.184
Mask mean precision	0.718	0.582	0.348	0.762	0.626	0.373	0.803	0.692	0.390
Mask mean recall	0.616	0.305	0.201	0.634	0.315	0.209	0.653	0.322	0.219

While the drop in precision is about half a magnitude when going from the train split to the test split, it is roughly two-thirds for the recall. This means that the model often fails to localize the objects in general. The root cause of this is usually an insufficient number of instances in the dataset. Since Table 11 shows macro metrics, it is imperative to recall that our dataset is imbalanced and does not have a lot of instances for classes other than the *ExposedBars* category. Thus, it makes sense to separately calculate these metrics solely for this category, which is listed in Table 13. We can see that the *ExposedBars* category performs better than the macro average in terms of mIOU — the test mIOU shows an improvement of almost a magnitude. The mAP_{50-95} on the test split is slightly higher than the macro average, probably due to the increased precision for the *ExposedBars* category. Compared to the macro average mean value, the mean mask precision is almost double as high when solely considering the *ExposedBars* category. The mean mask recall is roughly on par with the macro average mean value. When converting the model to a TensorRT engine for deployment on an embedded GPU such as the NVIDIA Jetson Xavier NX in our case, performance drops are to be expected and also observed, as indicated in Table 12. The difference in segmentation metrics between the full floating-point precision (FP32) and half floating-point precision (FP16) are minimal, however, there are significant performance drops when using integer quantization (INT8) of the model weights. The performance drops of FP16 and FP32 are minimal, with a relative decrease of 3% in validation mIOU, while achieving a frame rate of 12.8 and 4.6 FPS respectively. We therefore recommend using the FP16 engine as a good compromise between model accuracy and frame rate. Since we used the YOLOv8L-seg model, an instance segmentation model, we also have detected bounding boxes at our disposal. We therefore use these to analyze the failure modes and feed them into the TIDE tool [54], a tool for identifying object detection errors. The TIDE tool categorizes detection errors into five main categories and assigns them a ΔmAP value, which reports the increase in mAP that one could expect if all errors of a type are resolved. The categories are as follows:

Table 12
Results of the best performing model of Table 11, when converted to TensorRT engines of different weight types (integer, half precision and full precision). FPS are reported on Jetson Xavier NX in 20 W hexacore mode.

	INT8			FP16			FP32		
	train	val	test	train	val	test	train	val	test
mIOU	0.507	0.259	0.146	0.723	0.571	0.322	0.724	0.571	0.322
Δ%	−27.8	−56.1	−60.2	3.0	−3.2	−12.3	3.1	−3.2	−12.3
Mask mAP_{50-95}	0.367	0.264	0.121	0.518	0.296	0.171	0.518	0.297	0.171
Δ%	−31.0	−11.4	−34.2	−2.6	−0.7	−7.1	−2.6	−0.3	-7.1
Mask mean precision	0.729	0.701	0.434	0.756	0.653	0.374	0.754	0.653	0.375
Δ%	−9.2	1.3	11.3	−5.9	−5.6	−4.1	−6.1	−5.6	-3.8
Mask mean recall	0.337	0.068	0.070	0.637	0.327	0.208	0.64	0.327	0.207
Δ%	−48.4	−78.9	−68.0	−2.5	1.6	−5.0	−2.0	1.6	-5.5
Avg. FPS	20.92			12.79			4.61		

Table 13
Final results, descriptive statistics based on four verification runs with equal configuration and different seeds, on the *ExposedBars* category only.

	min			**mean**			**max**		
	train	val	test	train	val	test	train	val	test
mIOU	0.884	0.607	0.564	0.889	0.675	0.617	0.893	0.716	0.718
Mask mAP_{50-95}	0.590	0.182	0.257	0.618	0.194	0.267	0.638	0.211	0.273
Mask mean precision	0.787	0.470	0.669	0.822	0.512	0.679	0.854	0.533	0.691
Mask mean recall	0.775	0.161	0.199	0.783	0.176	0.210	0.793	0.196	0.218

- **Classification errors**, i.e., properly localized bounding box, but wrong class label
- **Localization errors**: a bounding box has an IOU with a detection bigger than a background IOU threshold, but lower than a foreground IOU threshold
- **Duplicate detection errors**: Multiple predicted bounding box have an IOU bigger than the foreground IOU threshold with the ground truth bounding box
- **Background errors**: A false positive detection, i.e., a predicted bounding box has an IOU smaller than the background threshold with a ground truth box
- **Missed detection**: There is no matched predicted bounding box for a ground truth bounding box (false negative)

Fig. 13 shows the results of the bounding box error analysis conducted with the TIDE tool. Throughout the splits, one big source of errors are missed detections, i.e., the model fails to produce a prediction for some ground truth instance. After this type, false positive detections are also a significant source of errors, i.e., the model recognizes unimportant backgrounds as an object. Localization and classification errors are, compared to the other error types, minimal. We can also see that ΔmAP value for missed detections is almost double compared to the train split, which is in line with the mask mean recall values reported in Tables 11 and 13.

To conclude, the results show that our trained model is precise and show good segmentation performance for the *ExposedBars* class, but struggles to generalize across splits. This indicates that our dataset has qualitative annotations, but still has room for improvement in terms of the amount of data and variety captured.

6. Discussion

Our study was aimed at introducing a new dataset for reinforced concrete construction, with a focus on public availability, contrary to most other similar works. Furthermore, it was desired to create a dataset with spatiotemporal relationships, reflecting the realistic robotic application of this dataset. The following summarizes our key findings, which will be elaborated later in this section:

- We have found that the performance of the segmentation model YOLOv8L-seg decreases with more and more withheld training data.
- We have showed a positive influence of pre-training on model performance, even when trained on unrelated, general data like the MS COCO dataset.
- We have showcased labeling inconsistencies in our dataset, and found that removing these inconsistencies does not alter model performance significantly.
- We have also showed that our data has a certain center bias, i.e., the mask centroids are concentrated in the middle of the image.
- Our data shows high discrepancies in model performance when evaluated on a per-class level, with the *car* class showing low performance in terms of mIOU.
- An error analysis of our model revealed that false negative errors are by far the biggest error mode.
- On an embedded GPU, using half-floating point precision engines yields a good compromise between model accuracy and frame rate.

Our experiment showing the decrease in model performance with more and more withheld training data suggests also the opposite, i.e. model performance increases with more and more data availability. Given the lack of publicly available datasets for construction applications, we identify this property as one of the main factors why training well-performing deep learning-based models for construction applications is challenging. We therefore urge future work to make data publicly accessible to alleviate this barrier.

Because our data has a center bias in its annotations, the superiority of the YOLOv8L-seg over DeepLabV3 and U-Net can easily be explained, since YOLOv8L-seg uses a sophisticated set of image augmentations, making the model more robust to these peculiarities. Furthermore, our dataset is biased towards the *ExposedBars* class, with a very minimal

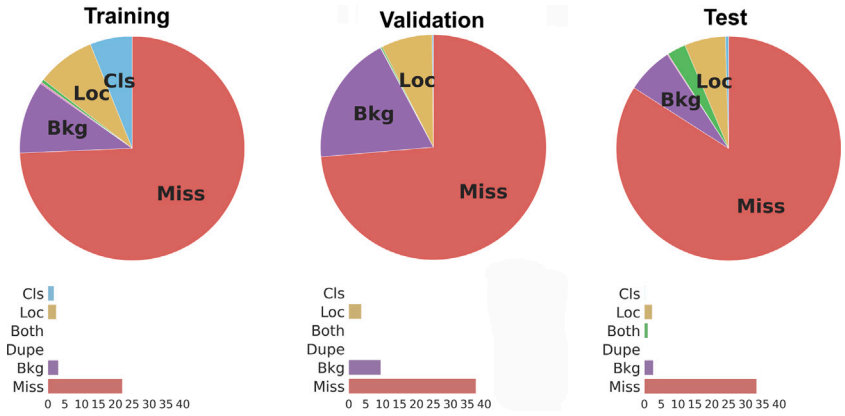
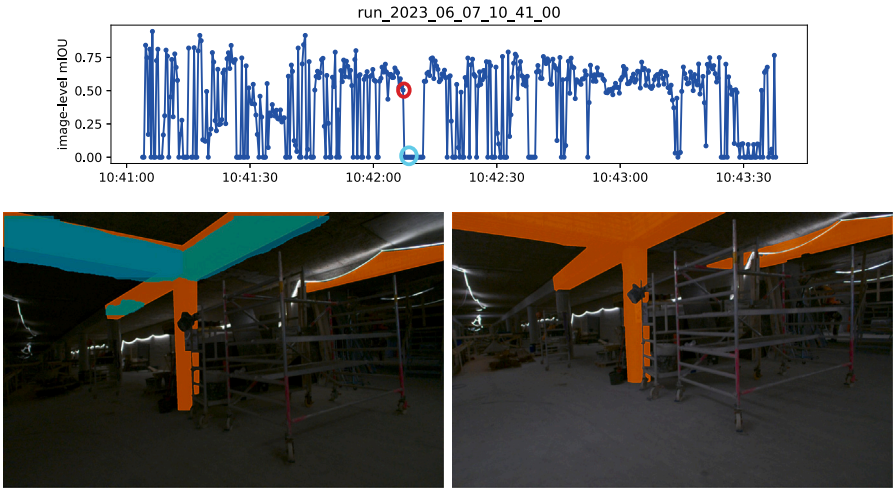


Fig. 13. Error analysis of bounding box predictions, per split. Output from TIDE tool.



(a) Reference frame with existing prediction, referring to the red circle in the upper plot

(b) Frame with missing prediction of same object, referring to the cyan circle in the upper plot

Fig. 14. Illustration of model instabilities. Model instabilities often include a significant drop in mIOU, followed by a sudden increase in mIOU. Predictions are in cyan, ground truth annotations in orange.

amount of instances of the *car* class. This explains the discrepancies in per-class performance. This limitation can be overcome by including data from other datasets that include these classes.

The false negative errors are a clear limitation of our model. We have further analyzed this phenomenon by investigating the development of mIOU values over time in a scene and could see that some false negatives are just model instabilities, i.e. the model fails to produce a prediction for an object previously detected. Fig. 14 illustrates this. This problem has already been discussed and analyzed in [9]. We stipulate that these errors are due to minor perturbations of the input resulting in big distances in feature space, thus failing to produce predictions. We therefore recommend the usage of mask stabilization techniques, as outlined in [9]. This also supports the approach of having a spatiotemporal dataset, since models are applied in real life, where one can leverage these dependencies to increase system performance.

While our research showed minimal influences of different labeling styles on label performance, we acknowledge that this insight cannot be generalized given the low number of samples in our dataset showing this phenomenon. However, given the positive gradient interaction presented in Section 4.4, we recommend analyzing this phenomenon in other settings to verify our findings, if applicable. Furthermore, we stipulate that deep learning-based segmentation models in construction are a good use case for probabilistic segmentation models, such as Probabilistic U-Net [55], as those try to model different, valid expert opinions on how to segment a certain object. Further research should investigate the application of such models in a construction context.

7. Conclusion

This paper presented a dataset consisting of temporally ordered RGB images with segmentation labels for reinforced concrete construction. The dataset is, as opposed to most datasets in construction, made publicly available to boost research in the area and contains 14,805 images in total, bigger than most task-specific datasets that focus on the construction sector. We have carried out an extensive analysis of our dataset and specifically analyzed variations in annotation styles. Furthermore, we have established and compared three baseline segmentation models: U-Net, DeepLabV3, and YOLOv8L-seg. Our study revealed that the YOLOv8L-seg model trained from pretrained weights is superior in performance compared to U-Net and DeepLabV3. Moreover, we have explored the effects of withholding training data while maintaining an equal class distribution and have found that the performance of the model increases with more available training data. In addition, we have explored the influence of different annotation styles on the model performance by removing anomalous annotations from the training data and have found no significant

change. In addition, we implemented the model on an embedded GPU and found a good compromise between model accuracy and frame rate using FP16 precision. Lastly, our final evaluation showed that the model is precise in predicting areas with exposed reinforcement bars, but fails to generalize and recognize these areas in scenes that deviate from those depicted in the train split.

Our study concludes that there is still a significant lack of CV data for construction tasks. Unlike other sectors, where digitization has been adopted swiftly in the past decades, construction has been traditionally slow to adopt digital technologies. This has resulted in low data availability and, thus, a challenging environment to develop deep learning-based solutions to be used in autonomous robotic systems in construction. These robotic systems will be a necessity in the future, as the construction sector is facing labor shortages in the upcoming years. We think the community will benefit from more publicly available datasets and a common data lake, where researchers will upload data along with annotations, even if annotations styles within a category are vastly different, as our study showed the minimal impact of this phenomenon.

We hope that our work will motivate other researchers to contribute their data and thus work towards a large data pool, paving the way for better model performance for computer vision tasks in construction.

CRediT authorship contribution statement

Patrick Schmidt: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Lazaros Nalpantidis:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been funded and supported by the EU Horizon Europe project “RobétArmé” under the Grant Agreement 101058731. We extend our gratitude to Christiansen & Essenbaek A/S (CEAS) and Markus Schmidtke, for sharing their domain expertise and organizing access to their construction sites for data-capturing purposes.

Data statement

The self-collected scenes of the dataset can be found on DTU Data [56]. The code to initiate the dataset including the dataset management tool can be found on GitHub in our repository.[3]

Data availability

The dataset is available at the following DOI: https://doi.org/10.11583/DTU.26213762 – the supplementing code can be located at https://github.com/DTU-PAS/ConRebSeg.

[3] https://github.com/DTU-PAS/ConRebSeg

References

[1] Eurostat, Labour Input in Construction - Annual Data, Eurostat, 2022, http://dx.doi.org/10.2908/STS_COLB_A, URL https://ec.europa.eu/eurostat/databrowser/product/page/STS_COLB_A.

[2] Eurostat, Productivité du travail et coût salarial unitaire par branche d’activité, Eurostat, 2024, http://dx.doi.org/10.2908/NAMA_10_LP_A21, URL https://ec.europa.eu/eurostat/databrowser/product/page/NAMA_10_LP_A21.

[3] D.K.-S. Schober, Digitization in the construction industry, Think: Act Mag. (2016) URL https://www.rolandberger.com/publications/publication_pdf/tab_digitization_construction_industry_e_final.pdf.

[4] J. Manyika, S. Ramaswamy, S. Khanna, H. Sarrazin, G. Pinkus, G. Sethupathy, A. Yaffe, Digital America: A Tale of the Haves And Have-Mores, Tech. rep., McKinsey&Company, 2015, URL https://www.mckinsey.com/~/media/mckinsey/industries/technology%20media%20and%20telecommunications/high%20tech/our%20insights/digital%20america%20a%20tale%20of%20the%20haves%20and%20have%20mores/digital%20america%20full%20report%20december%202015.pdf.

[5] Eurostat, Non-Fatal Accidents at Work by NACE Rev. 2 Activity and Sex, Eurostat, 2022, http://dx.doi.org/10.2908/HSW_N2_01, URL https://ec.europa.eu/eurostat/databrowser/product/page/HSW_N2_01.

[6] Eurostat, Fatal Accidents at Work by NACE Rev. 2 Activity, Eurostat, 2022, http://dx.doi.org/10.2908/HSW_N2_02, URL https://ec.europa.eu/eurostat/databrowser/product/page/HSW_N2_02.

[7] S. Parascho, Construction robotics: From automation to collaboration, Annu. Rev. Control. Robot. Auton. Syst. 6 (1) (2023) 183–204, http://dx.doi.org/10.1146/annurev-control-080122-090049, URL https://www.annualreviews.org/doi/10.1146/annurev-control-080122-090049.

[8] I. Kostavelis, L. Nalpantidis, R. Detry, H. Bruyninckx, A. Billard, S. Christian, M. Bosch, K. Andronikidis, H. Lund-Nielsen, P. Yosefipor, U. Wajid, R. Tomar, F.L. Martínez, F. Fugaroli, D. Papargyriou, N. Mehandjiev, G. Bhullar, E. Gonçalves, J. Bentzen, M. Essenbæk, C. Cremona, M. Wong, M. Sanchez, D. Giakoumis, D. Tzovaras, RoBétArmé project: Human-robot collaborative construction system for shotcrete digitization and automation through advanced perception, cognition, mobility and additive manufacturing skills, Open Res. Eur. 4 (2024) 4, http://dx.doi.org/10.12688/openreseurope.16601.1, URL https://open-research-europe.ec.europa.eu/articles/4-4/v1.

[9] P. Schmidt, D. Katsatos, D. Alexiou, I. Kostavelis, D. Giakoumis, D. Tzovaras, L. Nalpantidis, Towards Autonomous Shotcrete Construction: Semantic 3D Reconstruction for Concrete Deposition Using Stereo Vision and Deep Learning, Lille, France, 2024, http://dx.doi.org/10.22260/ISARC2024/0116, URL http://www.iaarc.org/publications/2024_proceedings_of_the_41st_isarc_lille_france/towards_autonomous_shotcrete_construction_semantic_3d_reconstruction_for_concrete_deposition_using_stereo_vision_and_deep_learning.html.

[10] K. Asadi, P. Chen, K. Han, T. Wu, E. Lobaton, LNSNet: Lightweight navigable space segmentation for autonomous robots on construction sites, Data 4 (1) (2019) 40, http://dx.doi.org/10.3390/data4010040, URL https://www.mdpi.com/2306-5729/4/1/40.

[11] X. Yan, H. Zhang, Y. Wu, C. Lin, S. Liu, Construction instance segmentation (CIS) dataset for deep learning-based computer vision, Autom. Constr. 156 (2023) 105083, http://dx.doi.org/10.1016/j.autcon.2023.105083, URL https://linkinghub.elsevier.com/retrieve/pii/S0926580523003436.

[12] C.D. Casuat, N.E. Merencilla, R.C. Reyes, R.V. Sevilla, C.G. Pascion, Deep-hart: An inference deep learning approach of hard hat detection for work safety and surveillance, in: 2020 IEEE 7th International Conference on Engineering Technologies and Applied Sciences, ICETAS, IEEE, Kuala Lumpur, Malaysia, 2020, pp. 1–4, http://dx.doi.org/10.1109/ICETAS51660.2020.9484208, URL https://ieeexplore.ieee.org/document/9484208/.

[13] R.N. Bhadeshiya, K.N. Brahmbhatt, J.R. Pitroda, Hard-hat detection using YOLOv4, in: 2021 Second International Conference on Electronics and Sustainable Communication Systems, ICESC, IEEE, Coimbatore, India, 2021, pp. 1114–1120, http://dx.doi.org/10.1109/ICESC51422.2021.9532896, URL https://ieeexplore.ieee.org/document/9532896/.

[14] S. Du, M. Shehata, W. Badawy, Hard hat detection in video sequences based on face features, motion and color information, in: 2011 3rd International Conference on Computer Research and Development, IEEE, Shanghai, China, 2011, pp. 25–29, http://dx.doi.org/10.1109/ICCRD.2011.5763846, URL http://ieeexplore.ieee.org/document/5763846/.

[15] Z. Zhu, S. German, I. Brilakis, Visual retrieval of concrete crack properties for automated post-earthquake structural safety evaluation, Autom. Constr. 20 (7) (2011) 874–883, http://dx.doi.org/10.1016/j.autcon.2011.03.004, URL https://linkinghub.elsevier.com/retrieve/pii/S0926580511000318.

[16] P. Hüthwohl, R. Lu, I. Brilakis, Multi-classifier for reinforced concrete bridge defects, Autom. Constr. 105 (2019) 102824, http://dx.doi.org/10.1016/j.autcon.2019.04.019, URL https://linkinghub.elsevier.com/retrieve/pii/S092658051831269X.

[17] Z. Huang, W. Chen, A. Al-Tabbaa, I. Brilakis, NHA12D: A New Pavement Crack Dataset and a Comparison Study of Crack Detection Algorithms, 2022, http://dx.doi.org/10.35490/EC3.2022.160, URL https://ec-3.org/publications/conference/paper/?id=EC32022_160.

[18] C. Tang, S. Jiang, H. Li, D. Huang, X. Huang, Y. Xiong, Lightweight concrete crack segmentation method based on deeplabv3+, in: Z. Shen (Ed.), Proc. Spie - Int. Soc. Opt. Eng. 12754 (2023) 127541M, http://dx.doi.org/10.1117/12.2684291, ISBN: 1510667563, 1510667571, 9781510667563, 9781510667570 Medium: article Publisher: SPIE.

[19] M. Mundt, S. Majumder, S. Murali, P. Panetsos, V. Ramesh, Meta-learning convolutional neural architectures for multi-target concrete defect classification with the concrete defect bridge image dataset, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, Long Beach, CA, USA, 2019, pp. 11188–11197, http://dx.doi.org/10.1109/CVPR.2019.01145, URL https://ieeexplore.ieee.org/document/8953853/.

[20] X. Yuan, A. Smith, F. Moreu, R. Sarlo, C.D. Lippitt, M. Hojati, S. Alampalli, S. Zhang, Automatic evaluation of rebar spacing and quality using LiDAR data: Field application for bridge structural assessment, Autom. Constr. 146 (2023) 104708, http://dx.doi.org/10.1016/j.autcon.2022.104708, URL https://linkinghub.elsevier.com/retrieve/pii/S0926580522005787.

[21] C.-C. Chang, T.-W. Huang, Y.-H. Chen, J.J. Lin, C.-S. Chen, Autonomous dimensional inspection and issue tracking of rebar using semantically enriched 3D models, Autom. Constr. 160 (2024) 105303, http://dx.doi.org/10.1016/j.autcon.2024.105303, URL https://linkinghub.elsevier.com/retrieve/pii/S0926580524000396.

[22] Y. Kardovskyi, S. Moon, Artificial intelligence quality inspection of steel bars installation by integrating mask R-CNN and stereo vision, Autom. Constr. 130 (2021) 103850, http://dx.doi.org/10.1016/j.autcon.2021.103850, URL https://linkinghub.elsevier.com/retrieve/pii/S0926580521003010.

[23] W. Chen, X. Fu, W. Chen, Z. Peng, Intelligent detection of rebar size and position using improved DeeplabV3+, Appl. Sci. 13 (19) (2023) 11094, http://dx.doi.org/10.3390/app131911094, URL https://www.mdpi.com/2076-3417/13/19/11094.

[24] Q. Wang, J.C.P. Cheng, H. Sohn, Automated estimation of reinforced precast concrete rebar positions using colored laser scan data, Comput.-Aided Civ. Infrastruct. Eng. 32 (9) (2017) 787–802, http://dx.doi.org/10.1111/mice.12293, URL https://onlinelibrary.wiley.com/doi/10.1111/mice.12293.

[25] H. Liu, C. Lin, J. Cui, L. Fan, X. Xie, B.F. Spencer, Detection and localization of rebar in concrete by deep learning using ground penetrating radar, Autom. Constr. 118 (2020) 103279, http://dx.doi.org/10.1016/j.autcon.2020.103279, URL https://linkinghub.elsevier.com/retrieve/pii/S0926580519315882.

[26] V. Krause, I. Abdel-Qader, O. Abudayyeh, S. Yehia, An image segmentation algorithm for the detection of rebar in bridge decks from GPR scans, in: 2007 IEEE International Conference on Electro/Information Technology, IEEE, Chicago, IL, USA, 2007, pp. 114–119, http://dx.doi.org/10.1109/EIT.2007.4374476, URL https://ieeexplore-ieee-org.proxy.findit.cvt.dk/stamp/stamp.jsp?tp=&arnumber=4374476.

[27] P. Liu, Z. Ding, W. Zhang, Z. Ren, X. Yang, Using ground-penetrating radar and deep learning to rapidly detect voids and rebar defects in linings, Sustainability 15 (15) (2023) 11855, http://dx.doi.org/10.3390/su151511855, URL https://www.mdpi.com/2071-1050/15/15/11855.

[28] R.L. Johns, M. Wermelinger, R. Mascaro, D. Jud, F. Gramazio, M. Kohler, M. Chli, M. Hutter, Autonomous dry stone: On-site planning and assembly of stone walls with a robotic excavator, Constr. Robot. 4 (3–4) (2020) 127–140, http://dx.doi.org/10.1007/s41693-020-00037-6, URL https://link.springer.com/10.1007/s41693-020-00037-6.

[29] D. Jud, High-Accuracy Autonomous Excavation of Free-Form Shapes (Ph.D. thesis), ETH Zurich, 2021, http://dx.doi.org/10.3929/ethz-b-000503062, in press, 2021-08-31T06:06:37Z, URL https://www.research-collection.ethz.ch/handle/20.500.11850/503062.

[30] S. Cao, H. Duan, S. Guo, J. Wu, T. Ai, H. Jiang, BIM-based task planning method for wheeled-legged rebar binding robot, Archit. Eng. Des. Manag. (2024) 1–17, http://dx.doi.org/10.1080/17452007.2024.2328134, URL https://www.tandfonline.com/doi/full/10.1080/17452007.2024.2328134.

[31] X. Zhang, X. Zhang, H. Jiang, B. Ma, Y. Si, Efficient rebar bundling: vision robotics innovations, J. Chin. Inst. Eng. 47 (3) (2024) 312–324, http://dx.doi.org/10.1080/02533839.2024.2308242, URL https://www.tandfonline.com/doi/full/10.1080/02533839.2024.2308242.

[32] S. Du, F. Teng, Z. Zhuang, D. Zhang, M. Li, H. Li, Y. Weng, A BIM-enabled robot control system for automated integration between rebar reinforcement and 3D concrete printing, Virtual Phys. Prototyp. 19 (1) (2024) e2332423, http://dx.doi.org/10.1080/17452759.2024.2332423, URL https://www.tandfonline.com/doi/full/10.1080/17452759.2024.2332423.

[33] D. Katsatos, P. Charalampous, P. Schmidt, I. Kostavelis, D. Giakoumis, L. Nalpantidis, D. Tzovaras, Semantic 3D reconstruction for volumetric modeling of defects in construction sites, Robotics 13 (7) (2024) http://dx.doi.org/10.3390/robotics13070102, URL https://www.mdpi.com/2218-6581/13/7/102.

[34] N. Taha, A.N. Walzer, J. Ruangjun, T. Bürgin, K. Dörfler, E. Lloret-Fritschi, F. Gramazio, M. Kohler, Robotic AeroCrete – A novel robotic spraying and surface treatment technology for the production of slender reinforced concrete elements, 2019, p. 10, http://dx.doi.org/10.3929/ETHZ-B-000387276, Artwork Size: 10 p. Medium: application/pdf Publisher: ETH Zurich, URL http://hdl.handle.net/20.500.11850/387276.

[35] V. Frangez, E. Lloret-Fritschi, N. Taha, F. Gramazio, M. Kohler, A. Wieser, Depth-camera-based rebar detection and digital reconstruction for robotic concrete spraying, Constr. Robot. 5 (3–4) (2021) 191–202, http://dx.doi.org/10.1007/s41693-021-00063-y, URL https://link.springer.com/10.1007/s41693-021-00063-y.

[36] B. Xiao, C. Chen, X. Yin, Recent advancements of robotics in construction, Autom. Constr. 144 (2022) 104591, http://dx.doi.org/10.1016/j.autcon.2022.104591, URL https://linkinghub.elsevier.com/retrieve/pii/S0926580522004617.

[37] R. Duan, H. Deng, M. Tian, Y. Deng, J. Lin, SODA: A large-scale open site object detection dataset for deep learning in construction, Autom. Constr. 142 (2022) 104499, http://dx.doi.org/10.1016/j.autcon.2022.104499, URL https://linkinghub.elsevier.com/retrieve/pii/S0926580522003727.

[38] A. Xuehui, Z. Li, L. Zuguang, W. Chengzhi, L. Pengfei, L. Zhiwei, Dataset and benchmark for detecting moving objects in construction sites, Autom. Constr. 122 (2021) 103482, http://dx.doi.org/10.1016/j.autcon.2020.103482, URL https://www.sciencedirect.com/science/article/pii/S0926580520310621.

[39] B. Xiao, S.-C. Kang, Development of an image data set of construction machines for deep learning object detection, J. Comput. Civ. Eng. 35 (2) (2021) 05020005, http://dx.doi.org/10.1061/(ASCE)CP.1943-5487.0000945, URL https://ascelibrary.org/doi/10.1061/%28ASCE%29CP.1943-5487.0000945.

[40] Z. Wang, Y. Wu, L. Yang, A. Thirunavukarasu, C. Evison, Y. Zhao, Fast personal protective equipment detection for real construction sites using deep learning approaches, Sensors 21 (10) (2021) 3478, http://dx.doi.org/10.3390/s21103478, URL https://www.mdpi.com/1424-8220/21/10/3478.

[41] H. Wang, Z. Ye, D. Wang, H. Jiang, P. Liu, Synthetic datasets for rebar instance segmentation using mask R-CNN, Buildings 13 (3) (2023) 585, http://dx.doi.org/10.3390/buildings13030585, URL https://www.mdpi.com/2075-5309/13/3/585.

[42] J. Jin, W. Zhang, F. Li, M. Li, Y. Shi, Z. Guo, Q. Huang, Robotic binding of rebar based on active perception and planning, Autom. Constr. 132 (2021) 103939, http://dx.doi.org/10.1016/j.autcon.2021.103939, URL https://linkinghub.elsevier.com/retrieve/pii/S0926580521003903.

[43] YouTube, Terms of service, URL https://www.youtube.com/static?template=terms.

[44] Directive (EU) 2019/790 of the European parliament and of the council of 17 April 2019 on copyright and related rights in the digital single market and amending directives 96/9/EC and 2001/29/EC (text with EEA relevance), 130, 2019, URL http://data.europa.eu/eli/dir/2019/790/oj/eng.

[45] eLov - Ophavsretsloven §11b, URL https://www.elov.dk/ophavsretsloven/paragraf/11b/.

[46] ORB-HD/deface, 2024, original-date: 2020-02-20T10:31:44Z, URL https://github.com/ORB-HD/deface.

[47] Y. Xu, W. Yan, G. Yang, J. Luo, T. Li, J. He, CenterFace: Joint face detection and alignment using face as point, Sci. Program. 2020 (2020) 1–8, http://dx.doi.org/10.1155/2020/7845384, URL https://www.hindawi.com/journals/sp/2020/7845384/.

[48] L. Chen, X. Chu, X. Zhang, J. Sun, Simple baselines for image restoration, in: S. Avidan, G. Brostow, M. Cissé, G.M. Farinella, T. Hassner (Eds.), Computer Vision – ECCV 2022, Springer Nature, Switzerland, Cham, 2022, pp. 17–33, http://dx.doi.org/10.1007/978-3-031-20071-7_2.

[49] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft COCO: Common objects in context, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision – ECCV 2014, Springer International Publishing, Cham, 2014, pp. 740–755, http://dx.doi.org/10.1007/978-3-319-10602-1_48.

[50] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The pascal visual object classes (VOC) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338, http://dx.doi.org/10.1007/s11263-009-0275-4.

[51] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, IEEE Computer Society, Los Alamitos, CA, USA, 2016, pp. 3213–3223, http://dx.doi.org/10.1109/CVPR.2016.350, URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.350.

[52] FiftyOne open source, URL https://voxel51.com/fiftyone/.

[53] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M.Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, S. Chintala, Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation, ASPLOS'24, in: 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, vol. 2, ACM, 2024, http://dx.doi.org/10.1145/3620665.3640366, URL https://pytorch.org/assets/pytorch2-2.pdf.

[54] D. Bolya, S. Foley, J. Hays, J. Hoffman, TIDE: A general toolbox for identifying object detection errors, in: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Eds.), Computer Vision – ECCV 2020, Springer International Publishing, Cham, 2020, pp. 558–573, http://dx.doi.org/10.1007/978-3-030-58580-8_33.

[55] S.A.A. Kohl, B. Romera-Paredes, C. Meyer, J.D. Fauw, J.R. Ledsam, K.H. Maier-Hein, S.M.A. Eslami, D.J. Rezende, O. Ronneberger, A probabilistic U-net for segmentation of ambiguous images, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS '18, Curran Associates Inc., Red Hook, NY, USA, 2018, pp. 6965–6975, http://dx.doi.org/10.5555/3327757.3327800, URL https://dl.acm.org/doi/10.5555/3327757.3327800.

[56] P. Schmidt, R.E. Andersen, J. Casas Lorenzo, C. Gascon Bononad, Self-Collected Sequences and Metadata of ConRebSeg, Technical University of Denmark, 2024, http://dx.doi.org/10.11583/DTU.26213762.