

Article

Automatic Detection of Ballast Unevenness Using Deep Neural Network

Piotr Bojarczak ^{1,*}, Piotr Lesiak ² and Waldemar Nowakowski ¹ 
¹ Faculty of Transport, Electrical Engineering and Computer Science, Casimir Pulaski Radom University, Malczewskiego 29, 26-600 Radom, Poland; w.nowakowski@uthrad.pl

² Faculty of Transportation and Computer Science, University of Economics and Innovation in Lublin, Projektowa 4, 20-209 Lublin, Poland; piotr.lesiak@wsei.lublin.pl

* Correspondence: p.bojarczak@uthrad.pl

Abstract: The amount of freight transported by rail and the number of passengers are increasing year by year. Any disruption to the passenger or freight transport stream can generate both financial and human losses. Such a disruption can be caused by the rail infrastructure being in poor condition. For this reason, the state of the infrastructure should be monitored periodically. One of the important elements of railroad infrastructure is the ballast. Its condition has a significant impact on the safety of rail traffic. The unevenness of the ballast surface is one of the indicators of its condition. For this reason, a regulation was introduced by Polish railway lines specifying the maximum threshold of ballast unevenness. This article presents an algorithm that allows for the detection of irregularities in the ballast. These irregularities are determined relative to the surface of the sleepers. The images used by the algorithm were captured by a laser triangulation system placed on a rail inspection vehicle managed by the Polish railway lines. The proposed solution has the following elements of novelty: (a) it presents a simple criterion for evaluating the condition of the ballast based on the measurement of its unevenness in relation to the level of the sleeper; (b) it treats ballast irregularity detection as an instance segmentation process and it compares two segmentation algorithms, Mask R-CNN and YOLACT, in terms of their application to ballast irregularity detection; and (c) it uses segmentation-related metrics—mAP (Mean Average Precision), IoU (Intersection over Union) and Pixel Accuracy—to evaluate the quality of the detection of ballast irregularity.

Keywords: deep learning; object detection; image segmentation; railway track diagnostics



Citation: Bojarczak, P.; Lesiak, P.; Nowakowski, W. Automatic Detection of Ballast Unevenness Using Deep Neural Network. *Appl. Sci.* **2024**, *14*, 2811. <https://doi.org/10.3390/app14072811>

Academic Editor: Suchao Xie

Received: 5 February 2024

Revised: 17 March 2024

Accepted: 25 March 2024

Published: 27 March 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Railway ballast condition has a significant impact on the safety of rail traffic, the lifecycle of railway track elements and the state of the rolling stock. According to [1], the ballast transmits the load from the sleeper/ballast interface to the sub-ballast and the subgrade, provides adequate permeability for drainage and keeps the sleepers dry, absorbs noise, vibration and energy, and provides stability to the track by withstanding vertical, longitudinal and lateral forces. Therefore, the good condition of track ballast is key for the safe and smooth running of a train [2]. For this reason, issues relating to ballast diagnostics have become of interest to researchers.

One of the devices used to determine the condition of the ballast is the Ground Penetrating Radar (GPR) [3,4]. It uses polarized high-frequency radio waves, usually in the range of several hundred MHz to several GHz. An extensive reference sampling and laboratory analysis were performed in [5] to aid in developing the GPR-based classification method for qualifying ballast fouling. The classification is made using a fouling index, which is calculated from the frequency contents of the GPR signal. Another article [6] also presents a system based on GPR. Algorithms such as Fast Fourier Transform (FFT) and Artificial Intelligence (AI), i.e., Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN), were applied to monitor ballast fouling. Other articles [7,8] used

the Dielectric Constant Method and 2 GHz GPR to evaluate ballast. In [9], the ballast state is assessed based on the GPR signal waveform and wavelet transform. Since the conversion of electromagnetic signal characteristics of GPR to ballast fouling is performed indirectly, the main drawback of methods based on GPR is the susceptibility to electromagnetic interference. It causes problems if the railway runs through areas that contain special structures such as turnouts, bridges with rail guards, etc., as the signal cannot be correctly detected because of iron interference.

Other methods that were used in the assessment of ballast are based on vibration measurements. Paper [10] demonstrates the detection of the damage status of ballast under a sleeper through the monitoring of the vibrations of the corresponding sleeper. The vibration is generated through a simple impact hammer test. Another paper [11] explored the feasibility of using particle acceleration responses to diagnose mud-pumping ballast. It used innovative wireless sensors with 3D-printed shells resembling the real shape of ballast particles. These sensors are placed in the ballast and measure acceleration. According to [11], it seems promising to use particle-scale acceleration underneath tie plates as readily implementable indicators for smart in-service track health monitoring. The main drawback of these methods is that the measurement is made at fixed points in the track.

Another paper [12] attempted to use infrared thermography to detect fouling ballast. It investigated the impact of the intensity of solar radiation and rainfall on the surface temperature of the ballast. The experimental results showed that noticeable temperature differences between clean and fouled ballast occur on sunny days and after rainfall. In contrast, the differences are imperceptible at nighttime.

Another type of device that can be used in ballast diagnostics is LIDAR (Light Detection and Ranging). It consists of a laser illuminating the track and a camera recording the image of the reflection from the ballast (surface map) [13,14]. Article [15] compared the performance of a low-cost solid-state LiDAR system with a traditional static LiDAR system. Both systems were mounted on a mobile inspection trolley and used to measure ballast geometry. The test was carried out on a section of rail track comprising a steel rail, concrete sleepers and ballast as structural support. The experiment showed that a low-cost LiDAR is able to determine ballast geometry but with slightly less accuracy than a traditional static LiDAR. Ref. [16] used a laser scanner together with classification methods to detect elements of railway tunnels based on the point cloud. Paper [17] used close-range photogrammetry to evaluate ballast particle degradation. The authors took into consideration the morphology of the ballast (e.g., shape, angularity and surface texture). Another article [18] used an algorithm for the extraction of vegetation and local muddy areas from ballast images.

The rapid development of both hardware and image-processing algorithms has led to the development of visual methods for railroad track diagnostics. In particular, the appearance of the paper [19] on the application of a deep convolutional network for object detection was a breakthrough in the field of machine learning. Since then, many papers on deep neural networks have been published [20–26]. The authors used deep learning networks in an algorithm that allows for the detection of irregularities in the ballast. These irregularities are determined relative to the surface of the sleepers. The detection of sleepers is performed using deep learning networks such as Mask R-CNN and YOLACT, which are used for image segmentation. The images used by the algorithm were captured by a laser triangulation system placed on a rail inspection vehicle managed by Polish railway lines. It allows for the continuous inspection of the track both during the day and the night and it provides immunity to electromagnetic interference, which eliminates the aforementioned limitations.

Our contributions are:

- We propose a simple criterion for the evaluation of the condition of the ballast based on the measurement of its unevenness in relation to the level of the sleeper;

- We treat ballast irregularity detection as an instance segmentation process. We examine two instance segmentation algorithms, Mask R-CNN and YOLACT, in terms of their application to ballast irregularity detection;
- The use of instance segmentation makes it possible to determine the unevenness of the ballast individually relative to each sleeper;
- We use segmentation-related metrics—mAP (Mean Average Precision), IoU (Intersection over Union) and Pixel Accuracy—to evaluate the quality of the detection of ballast irregularities.

2. Algorithm for Automatic Detection of Unevenness in Ballast

The poor condition of ballast has a significant impact on both traffic safety and travel comfort. The unevenness of the ballast surface is one measure of the condition of the ballast. The algorithm presented in this paper allows us to determine the unevenness of the ballast. According to Polish railway line standards, when ballast unevenness is within $\pm$ threshold = 35 mm, it does not pose a threat to traffic safety. These levels are measured relative to the top surface of the sleepers. Figure 1 presents a cross-section of the track with marked thresholds.

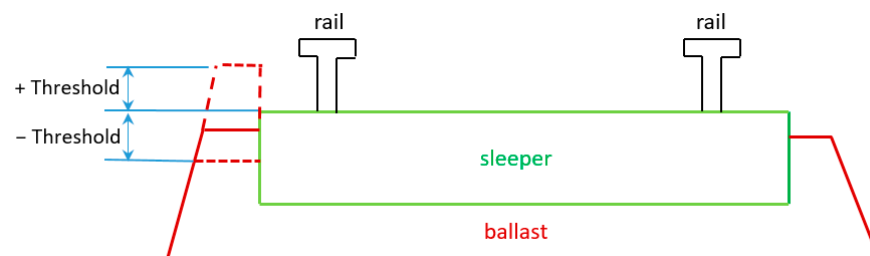


Figure 1. Cross-section of the track with marked threshold.

The state of the railway track is monitored using a track inspection vehicle managed by the Polish railway lines—Figure 2. All available data used in our research were collected using this vehicle. One of the several measurement systems installed on the board of the vehicle is a laser triangulation system. It comprises pulsed high-power laser line projectors and synchronized cameras. This system captures a high-resolution intensity image and 3D range profile of the railway track. Laser light is used to illuminate railway surfaces and high-speed cameras are used to capture images of the projected light including its intensity. Figure 3 shows a block diagram of the laser triangulation system. Such a system captures both Intensity and Range images of the railway track simultaneously. Intensity images are produced by mapping the intensity of the reflected laser light, and Range images (depth map), marked in this figure as D_r , are produced by mapping the elevation of each measurement point. The pixels of these two maps correspond one-to-one, that is, the corresponding image regions in the intensity image appear at the same position in the depth map. If a certain target is detected in the intensity map, the corresponding target can be found at the same position in the depth map (Range image).

Figure 4 shows a block diagram of the ballast unevenness detection algorithm. It is composed of two components. The first component presented in this figure as the “detection of sleepers” box is responsible for the detection of sleepers in the track and is described in Section 2.1. It uses the intensity image coming from the laser triangulation system as an input. Deep neural networks for instance segmentation are used for sleeper detection (both wooden and concrete). The second component presented in this figure as the “detection of ballast out of range $\pm$ Threshold” box specifies the areas of ballast for which its height is outside the defined range of the $\pm$ threshold = 35 mm and is described in Section 2.2. It uses the range image (map) coming from the laser triangulation system as an input and sleeper masks generated by the first component. The width and the height of both the intensity and the range image are equal to 288 and 1024, respectively.



Figure 2. Track inspection vehicle managed by Polish railway lines.

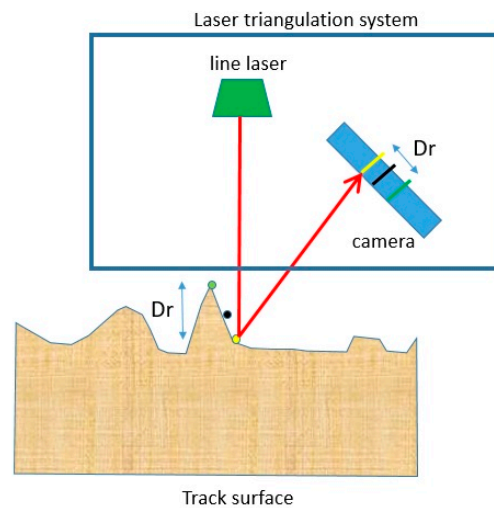


Figure 3. Block diagram for a laser triangulation system.

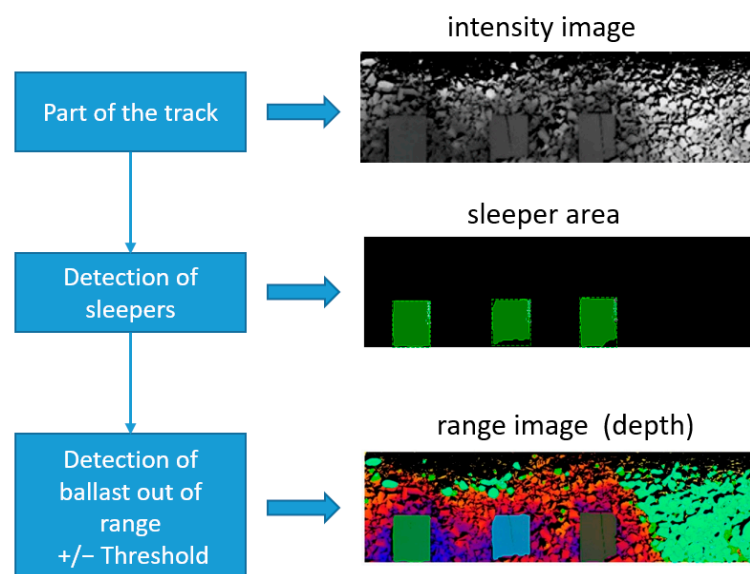


Figure 4. Block diagram for the proposed algorithm.

2.1. Detection of Sleepers

We used the intensity image generated by the laser triangulation system as an input. The unevenness of the ballast is measured relative to the surface of each sleeper. On this basis, the algorithm should detect each sleeper as well as its mask. Therefore, it provides the classes of the image objects and the location of the image objects in the form of bounding boxes. Additionally, it should provide labels for every pixel belonging to the objects. Each pixel is labelled according to the object class within which it is enclosed. This process is called instance segmentation and simultaneously solves the problem of object detection as well as semantic segmentation (pixel labelling). According to [27,28], instance segmentation algorithms can be divided into four groups:

- Classification of mask proposals (modest segmentation accuracy, slow and difficult to optimize training, slow testing, not suitable for real-time applications);
- Detection followed by segmentation (relatively simple to train, better generalization, relatively faster, good segmentation accuracy);
- Labelling pixels followed by clustering (relatively simpler techniques, lesser segmentation accuracy, intense computation necessitating high computational power, not suited for real-time applications);
- Dense sliding window methods (modest segmentation accuracy, use complex algorithms, difficult to train and optimize, not suitable for real-time applications).

Due to its advantages, detection followed by segmentation was selected. Two networks—Mask R-CNN and YOLACT—belonging to this group were tested in the article. Mask R-CNN and YOLACT are a two-stage and a one-stage instance segmentation algorithm, respectively. Mask R-CNN as a two-stage algorithm is more accurate and slower than YOLACT.

2.1.1. Mask R-CNN Network

The Mask R-CNN is the network that performs instance segmentation. Figure 5 shows the structure of the Mask R-CNN network. This network uses a convolutional neural network (CNN) to extract feature maps that are used to describe the objects being detected (sleepers). The CNN is often called the “backbone”, which is usually a pre-trained neural network. At present, three types of convolutional neural networks—AlexNet, VGG-Net and ResNet—can be used as a backbone. AlexNet, as the oldest network type, which produces the largest classification error on the ImageNet database (16.5%) [19], was discarded. VGG-Net reduced the classification error to 7.6% while increasing the number of network layers to 16–19 [29]. Thanks to the adoption of the core idea of Identity Shortcut Connection [30], ResNet further reduces this error to 3.6%. Using classification error on the ImageNet test database as a criterion, the authors decided to test Resnet51 and Resnet101 as a backbone. In order to narrow down the search for an object in an image, the network can be divided into two parts. In the first part, the region of interest (ROI) is determined using the Region Proposal Network (RPN)—marked by red rectangles on the feature map. To generate the ROI, the input of an $n \times n$ spatial window is slid over the convolutional feature map output, and the features covered by the spatial window are fed into two sibling fully connected layers—a box classification layer and a box regression layer. A box classification layer determines the probability of occurrence of the objects in the ROI area. On the other hand, a box regression layer generates the coordinates of the ROI center and its width and height.

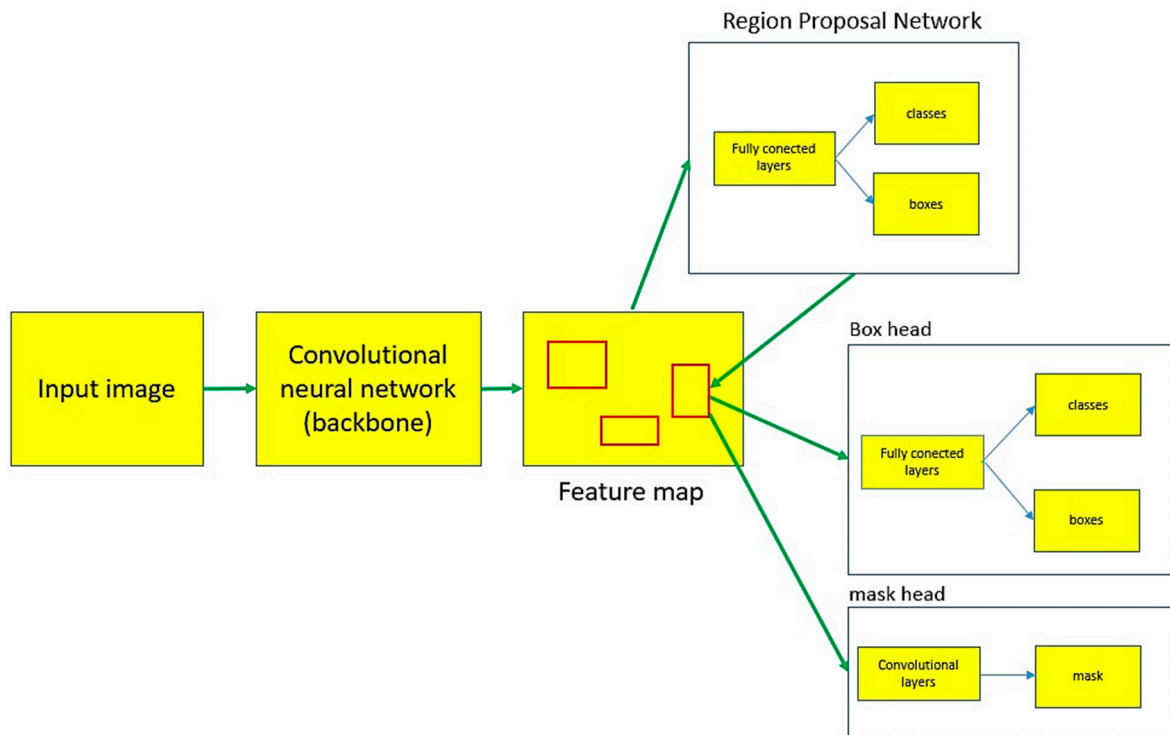


Figure 5. Structure of the Mask R-CNN algorithm.

The size of the ROI area generated by the RPN is variable. For this reason, in the next step (part) of the algorithm, the variable ROI size is converted into a fixed size. Here, fully connected layers also occur—box classification layers and box regression layers—which generate the class of the detected object and the coordinates of the bounding box, respectively. Additionally, the ROI is also fed to a fully convolutional network (FCN) [31] to generate the mask of an object being detected. According to [32], the algorithm uses the following objective function during training:

$$L = L_{CLS1} + L_{CLS2} + L_{box1} + L_{box2} + L_{mask} \quad (1)$$

where L_{CLS1} is a binary classifier loss corresponding to the classes block in RPN; L_{box1} is a regression loss corresponding to the boxes block in RPN; L_{CLS2} is a loss for the classifier corresponding to the classes block in the box head; L_{box2} is a regression loss corresponding to the boxes block in the box head; and L_{mask} is a mask binary cross-entropy loss corresponding to the mask block in the mask head.

2.1.2. YOLACT Network

The YOLACT network is a one-stage algorithm used for instance segmentation. The algorithm can be divided into two parallel subtasks, namely a prototype network block and a target detection block. Figure 6 shows a block diagram for the YOLACT algorithm. Both blocks use a convolutional neural network along with a Feature Pyramid Network (FPN) [33]. The FPN adds lateral connections between the P5 and P3 layers of the pyramid. The idea is to take top-down strong features from the C5 layer of the CNN and propagate them to the high-resolution feature maps in the C3 layer of the CNN, thus creating strong features across all levels. The prototype network block generates a set of image-sized “prototype masks”. These masks are independent of the instance. It predicts a set of k prototype masks for the entire image using the FCN. The P3 layer from the Feature Pyramid Network (FPN) is fed to an FCN.

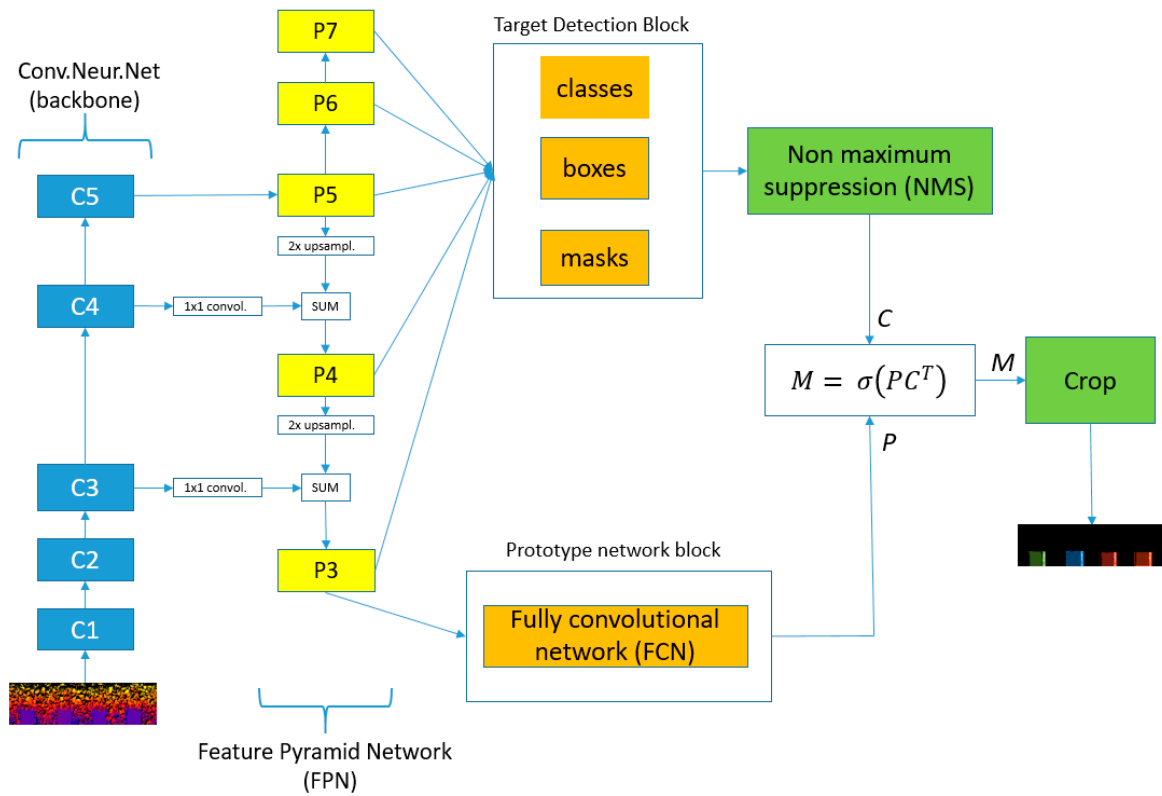


Figure 6. Structure of the YOLACT algorithm.

The target detection block consists of three parallel branches. The first branch predicts c class confidences and the second one predicts the four coordinates of the bounding box. The third branch predicts k mask coefficients, one corresponding to each prototype. After generating k mask coefficients and passing them through the Non-Maximum Suppression algorithm (NMS), they are combined with prototype masks using the following formula [34]:

$$M = \sigma(PC^T) \quad (2)$$

where P is an $h \times w \times k$ matrix of prototype masks and C is an $n \times k$ matrix of mask coefficients for n instances surviving NMS. The final mask M is cropped with the predicted bounding box generated by the target detection block.

YOLACT uses the following loss function while training [34]:

$$L = L_{CLS} + L_{box} + L_{mask} \quad (3)$$

The classification loss L_{CLS} and the box regression loss L_{box} are defined in the same way as in [35] and the mask loss L_{mask} is the pixel-wise binary cross entropy between the assembled masks and the ground truth masks.

2.2. Determination of the Area of Unevenness of the Ballast

This part of the algorithm is designed to determine areas of ballast unevenness in reference to sleeper levels. The laser triangulation system can simultaneously generate an intensity image (map) and a range image (depth). The pixels of these two images correspond one-to-one. Therefore, the coordinates for the sleepers generated by the deep neural network based on the intensity image can be directly mapped to the range image. The algorithm is presented in the form of pseudocode in Algorithm 1.

Algorithm 1. Algorithm to determine areas of ballast unevenness in reference to sleeper levels

```

1: for each sleeper detected by instance segmentation network in the intensity image do
2: Using range image calculate the mean level of the sleeper =  $ML$ 
3: calculate  $Upper\_Threshold := ML + Threshold$ 
4: calculate  $Bottom\_Threshold := ML - Threshold$ 
5:  $Mask := \emptyset$ 
6: for each  $m$  from 1 to  $image\_height$  do
7:   for each  $n$  from 1 to  $image\_width$  do
8:     if ( $D(m, n) < Bottom\_Threshold$ ) or ( $D(m, n) > Upper\_Threshold$ ) then
9:        $Mask := Mask + (m, n)$ 
10:    end if
11:  end for
12: end for
13: Mark points in the range image based on  $Mask$ 
14: end for

```

Where $D(m, n)$ denotes the depth corresponding to the coordinates (m, n) in the range image; threshold = 35 is the threshold value determined towards the sleeper level; and $Image_height$ and $image_width$ are the height and width of the range image, respectively.

According to this pseudocode, the unevenness of the area is calculated separately for each sleeper detected by the instance segmentation algorithm. These areas are defined relative to the mean level of the sleeper surface.

3. Experimental Results

To train both the Mask R-CNN and the YOLACT networks, we collected 1600 intensity images of the track. All images came from a track inspection vehicle managed by Polish railway lines. The intensity images are captured by a fast camera installed under the floor of the inspection vehicle. This camera is equipped with a Gigabit Ethernet (1 GigE) interface. The captured images are sent via 1 GigE to a server installed on the board of this vehicle. The group of selected images with wooden and concrete sleepers included images recorded in good weather (900) as well as during rainfall (700). All images were pre-processed. In this process, each image pixel was normalized so that the average pixel intensity value in the image was zero. This pre-processing significantly improves the convergence of the gradient optimization used to train both the Mask R-CNN and the YOLACT networks. All the available images were randomly divided into three groups in the proportions 60%, 20% and 20%: training data hereinafter referred to as *train_data*, validation data hereinafter referred to as *valid_data* and testing data hereinafter referred to as *test_data*.

We did not train both networks from scratch. To improve the performance of both networks and speed up their training we decided to use transfer learning [36,37]. This means that a model developed for one task is reused as the starting point for a model on a second task.

3.1. Training and Testing Mask R-CNN Network

The Mask R-CNN network was implemented using the Keras package in Python. Using the idea of transfer learning, two pre-trained networks were tested. The first uses Resnet51 as the backbone, while the second uses Resnet101. Each of the networks was trained for 400 epochs on a computer equipped with an Intel I5 CPU, NVIDIA GeForce GTX 1080 Ti GPU and 48 GB RAM. We used AdamOptimizer with a learning rate $Lr = 0.0001$, weight decay $w_dec = 0.0005$ and momentum = 0.9 to minimize the loss function (loss error) defined by Formula (1). While training, two groups of data were used: *train_data* for training and *valid_data* for validation. Figure 7 shows the loss error for the Mask R-CNN with the ResNet101 backbone during the learning process for the *train_data* set (Figure 7a) and the *valid_data* set (Figure 7b).

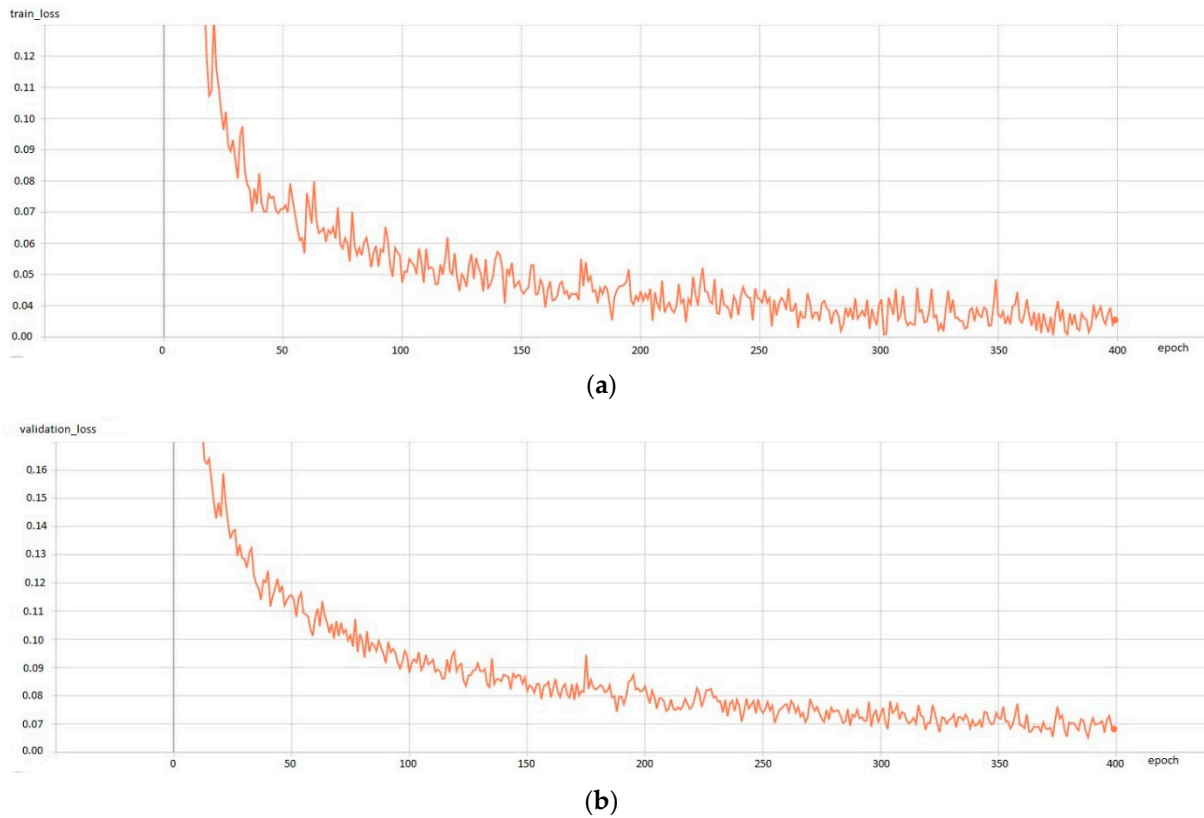


Figure 7. The loss error for Mask R-CNN with ResNet101 backbone defined by Formula (1) during the learning process for the *train_data* set (a) and the *valid_data* set (b).

After training, the performance of each network was verified on a *test_data* set. The network performs both the detection of individual sleepers and their segmentation. For this reason, the Average Precision (AP) metric was used to assess the quality of sleeper detection, while Intersection over Union (IoU), Dice coefficient and Pixel Accuracy metrics were used to assess the quality of the segmentation.

Pixel Accuracy calculates the percentage of pixels in the image that were correctly classified. It is commonly reported for each class separately. Pixel Accuracy can be defined as:

$$\text{Pixel accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

where TP true positive represents pixels that are correctly classified to belong to the sleeper class (according to the target (truth) mask); TN true negative represents pixels that are correctly classified as not belonging to the sleeper class; FP false positive represents pixels that are classified as belonging to the sleeper class though they do not belong to sleeper class; and FN false negative denotes pixels that are classified as not belonging to the sleeper class though they belong to the sleeper class.

Intersection over Union (IoU) quantifies the percentage overlap between the target (truth) mask and our prediction output (predicted mask). The IoU is defined as:

$$\text{IoU} = \frac{T_A \cap P_B}{T_A \cup P_B} = \frac{TP}{TP + FP + FN} \quad (5)$$

where T_A denotes pixels belonging to the target mask and P_B represents pixels that are classified as belonging to the target mask.

The dice coefficient can be defined as:

$$Dice = \frac{2 * (T_A \cap P_B)}{T_A + P_B} = \frac{2 * TP}{TP + TN + FP + FN} \quad (6)$$

AP is calculated based on the PASCAL VOC 2017 challenge. *AP* is used to evaluate sleeper detection. For this reason, in the IoU metric in (5), a target bounding box is used instead of a target mask. Additionally, true positives (*TP1*), false positives (*FP1*) and false negatives (*FN1*) were redefined. *TP1* is calculated when the IoU for the detected object is higher than the assumed threshold, whereas *FP1* is calculated when the IoU is lower than the assumed threshold. *FN1* is determined when the model did not predict a bounding box at a certain position although the target bounding box exists at that position.

The precision–recall curve is used while defining the *AP* metric. Precision is the ratio between the correct detections and the total number of detections:

$$\text{precision} = \frac{\text{correct detections}}{\text{total number of detections}} = \frac{TP1}{TP1 + FN1} \quad (7)$$

Recall is the ratio between the correct detections and all available objects (target bounding boxes):

$$\text{recall} = \frac{\text{correct detections}}{\text{all target bounding boxes}} \quad (8)$$

A precision–recall curve plots the value of precision against the recall for different confidence threshold values. The *AP* metric is defined as:

$$AP = \int_{r=0}^{r=1} p(r) dr \quad (9)$$

where *p* and *r* denote precision and recall, respectively. The precision–recall curve is usually approximated with rectangles and the integral in (9) is calculated as a sum of rectangles [38].

Table 1 shows the *AP*, IoU, Dice coefficient and Pixel Accuracy calculated for the networks with Resnet51 and Resnet101 backbones. Additionally, the mean processing time is also shown. An IoU threshold of 0.5 was used for all *AP* metric calculations.

Table 1. Detection and segmentation metrics for Mask R-CNN with two different backbones.

	Mean Average Precision	IoU	Dice Coefficient	Pixel ACCURACY	Mean Processing Time [ms]
Resnet51	0.8571	0.7602	0.8046	0.7845	125
Resnet101	0.8901	0.78004	0.8304	0.8063	200

As can be seen from Table 1, the highest values of the metrics were obtained for the Mask R-CNN network with Resnet101, while the shortest processing time was obtained for the network with Resnet51. This is due to the difference in complexity of the Resnet101 and the Resnet51 network structures. The complexity of the Mask R-CNN network with Resnet101 is higher, which results in its higher accuracy (higher values of metrics) at the expense of increased processing time.

Figure 8 shows examples of detection and segmentation of the sleepers from the intensity images of the track obtained for Mask R-CNN with the Resnet101 backbone. The first column shows the image of the track, the second the image with the predicted bounding box and mask overlaid on it, the third the predicted bounding box and mask and the last the target (truth) bounding box and mask. If we compare the predicted bounding boxes (dashed lines) and predicted masks (green areas) in column three with the corresponding truth bounding boxes (dashed lines) and truth masks (green areas) in column four, we can see negligible differences between them. The exception is the last lower sleeper found in the image from row three. Due to its very small size, it was not detected by the Mask R-CNN network.

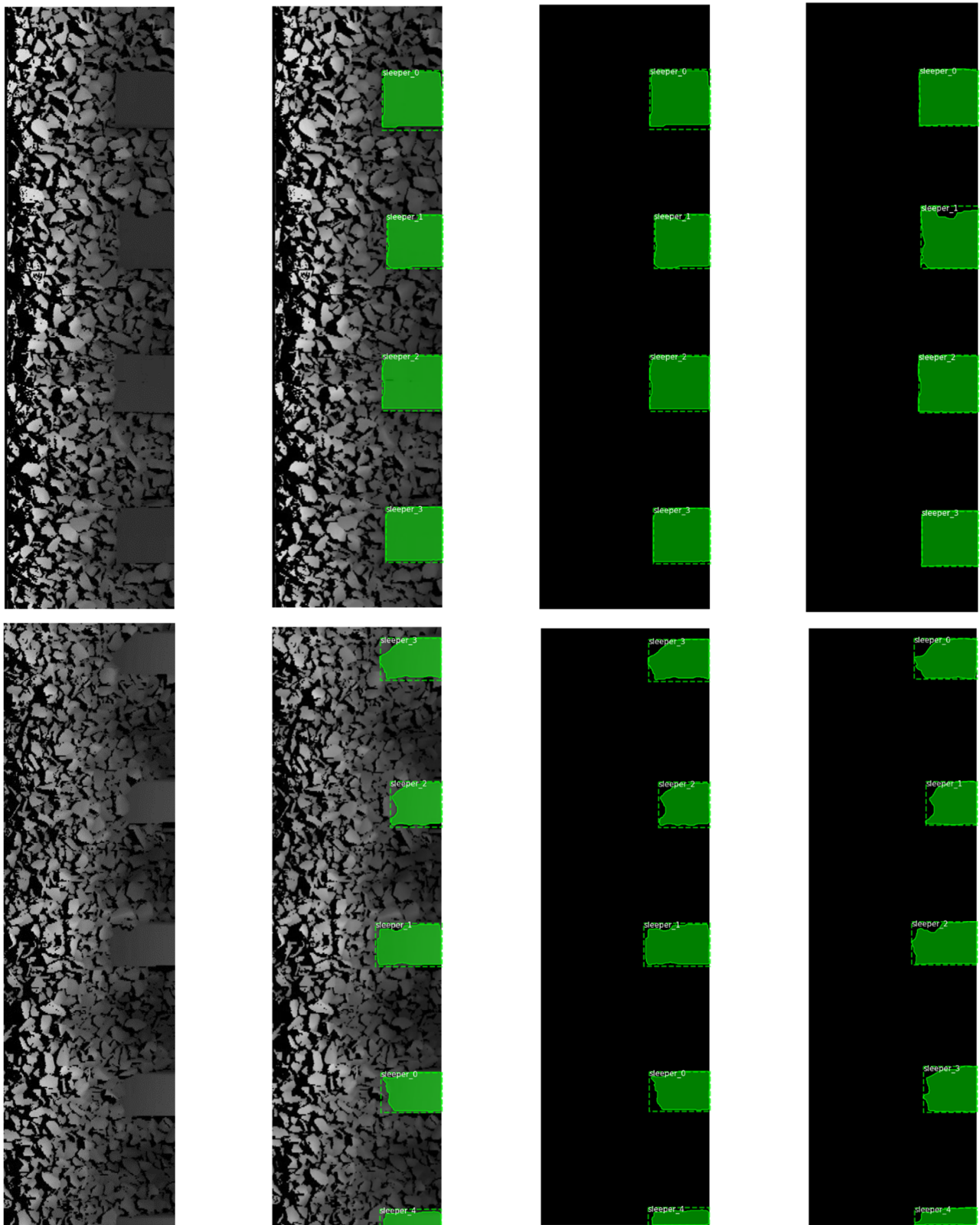


Figure 8. Cont.

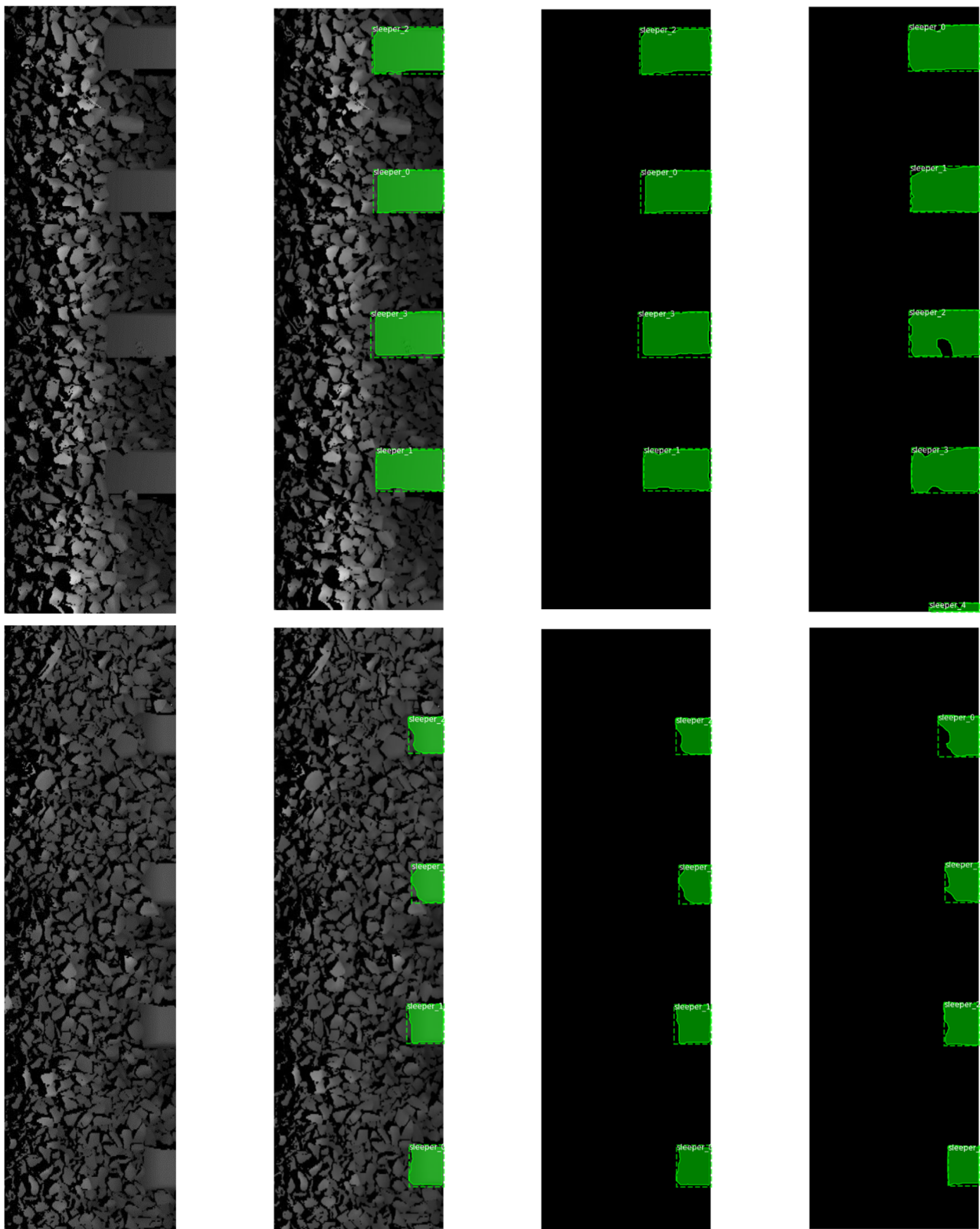


Figure 8. Examples of detection and segmentation of sleepers from intensity images of the track obtained for Mask R-CNN with Resnet101 backbone. The first column corresponds to the image of the track, the second the image with the predicted bounding box and mask overlaid on it, the third the predicted bounding box and mask and the fourth the target (truth) bounding box and mask.

3.2. Training and Testing the YOLACT Network

The YOLACT network was implemented using the Pytorch package in Python. Using the idea of transfer learning, two pre-trained networks were tested. The first uses Resnet51 as the backbone, while the second uses Resnet101. Each of the networks was trained for 400 epochs on a computer equipped with an Intel I5 CPU, NVIDIA GeForce GTX 1080 Ti GPU and 48 GB RAM. We used AdamOptimizer with a learning rate $Lr = 0.0001$, weight decay $w_dec = 0.0005$ and momentum $= 0.9$ to minimize the loss function (loss error) defined by Formula (3). While training, two groups of data were used: *train_data* for training and *valid_data* for validation.

After training, the performance of each network was verified on a *test_data* set. The performance of the network was checked based on the same metrics as the Mask R-CNN network. Table 2 shows the obtained results. As for Mask R-CNN, the highest accuracy was obtained with the Resnet101 backbone, whereas the lowest processing time was obtained using Resnet51.

Table 2. Detection and segmentation metrics for YOLACT with two different backbones.

	Mean Average Precision	IoU	Dice Coefficient	Pixel ACCURACY	Mean Processing Time [ms]
Resnet51	0.8214	0.7226	0.7668	0.7737	44
Resnet101	0.8571	0.7380	0.7912	0.7956	61

Figure 9 shows examples of the detection and segmentation of sleepers from the intensity images of the track obtained using YOLACT with a Resnet101 backbone. The first column shows the image of the track, the second the image with the predicted bounding box and mask overlaid on it, the third the predicted bounding box and mask and the last, the target (truth) bounding box and mask. In order to compare the results of the Mask R-CNN network with those of the YOLACT network, Figure 9 contains the same test intensity images (column one) as Figure 8. The differences between the predicted bounding boxes (solid lines) and the predicted masks (colored areas) in column three and the corresponding truth bounding boxes (solid lines) and truth masks (colored areas) in column four are slightly larger than for the Mask R-CNN network. In addition, the YOLACT network did not detect the last lower sleeper found in the image from row two. It should be noted that this sleeper was detected by the Mask R-CNN network (Figure 8, row two). The YOLACT network is a one-stage network. It is less accurate than the two-stage Mask R-CNN network. This is particularly evident when detecting objects of a small size.

In order to confirm the difference in the performance of the two models (Mask R-CNN with Resnet101 versus YOLACT with Resnet101), a statistical hypothesis test was performed. Because instance segmentation (Mask R-CNN and YOLACT) is composed of an object classification part (binary classification—sleepers and no sleepers) and a pixel classification part (binary classification—pixel belongs to the sleeper or pixel belongs to background), the test was conducted on both parts. The statistical test was selected based on the following criteria:

- Models are deep neural networks that are very large and their training is time-consuming. We prefer a statistical hypothesis test that enables us to compare the models on a single test data set instead of performing multiple training (k-fold cross-validation with a modified paired Student's *t*-test);
- Models are binary classifiers;
- There are no assumptions about the type of data distribution;
- The test data for models are the same.

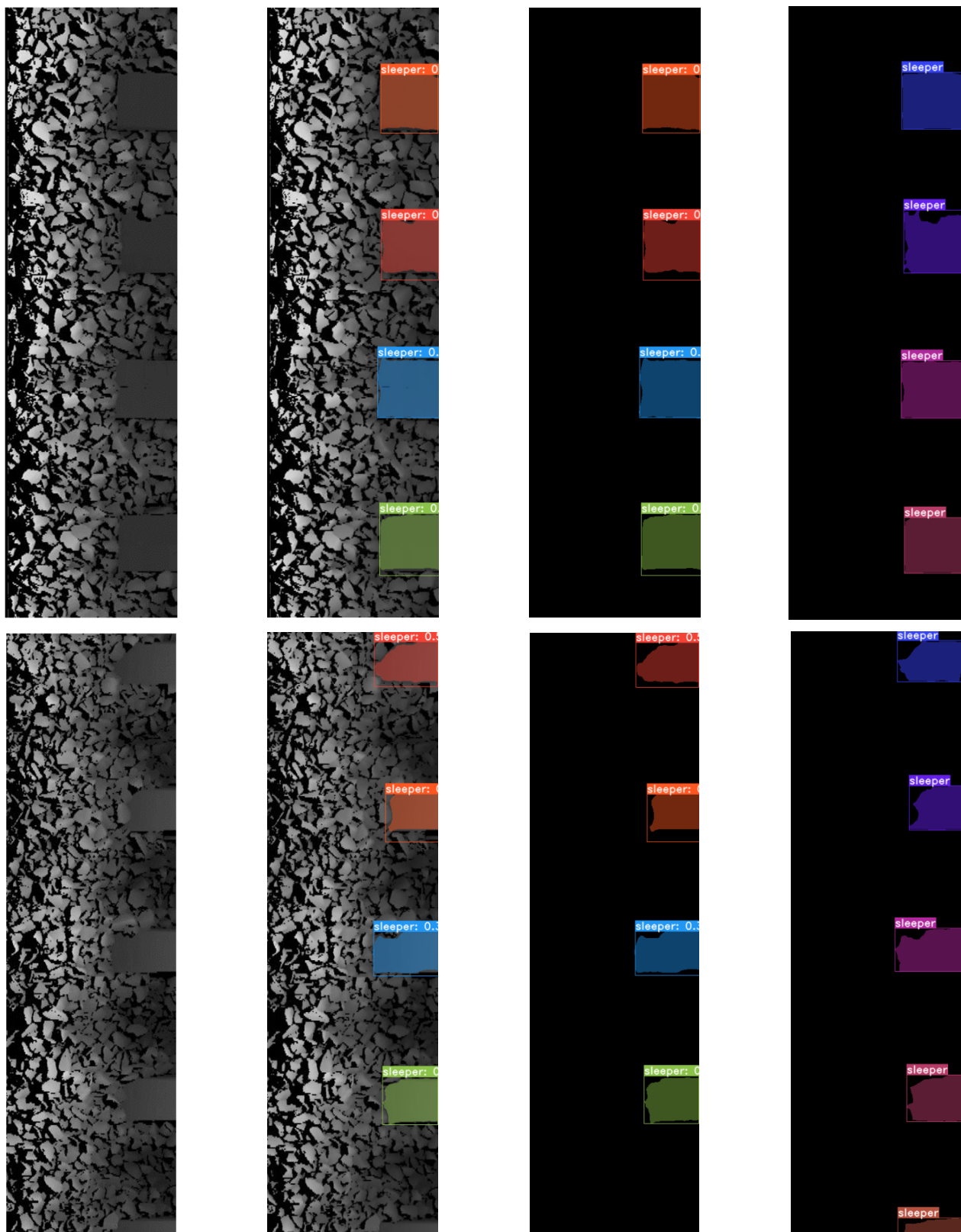


Figure 9. Cont.

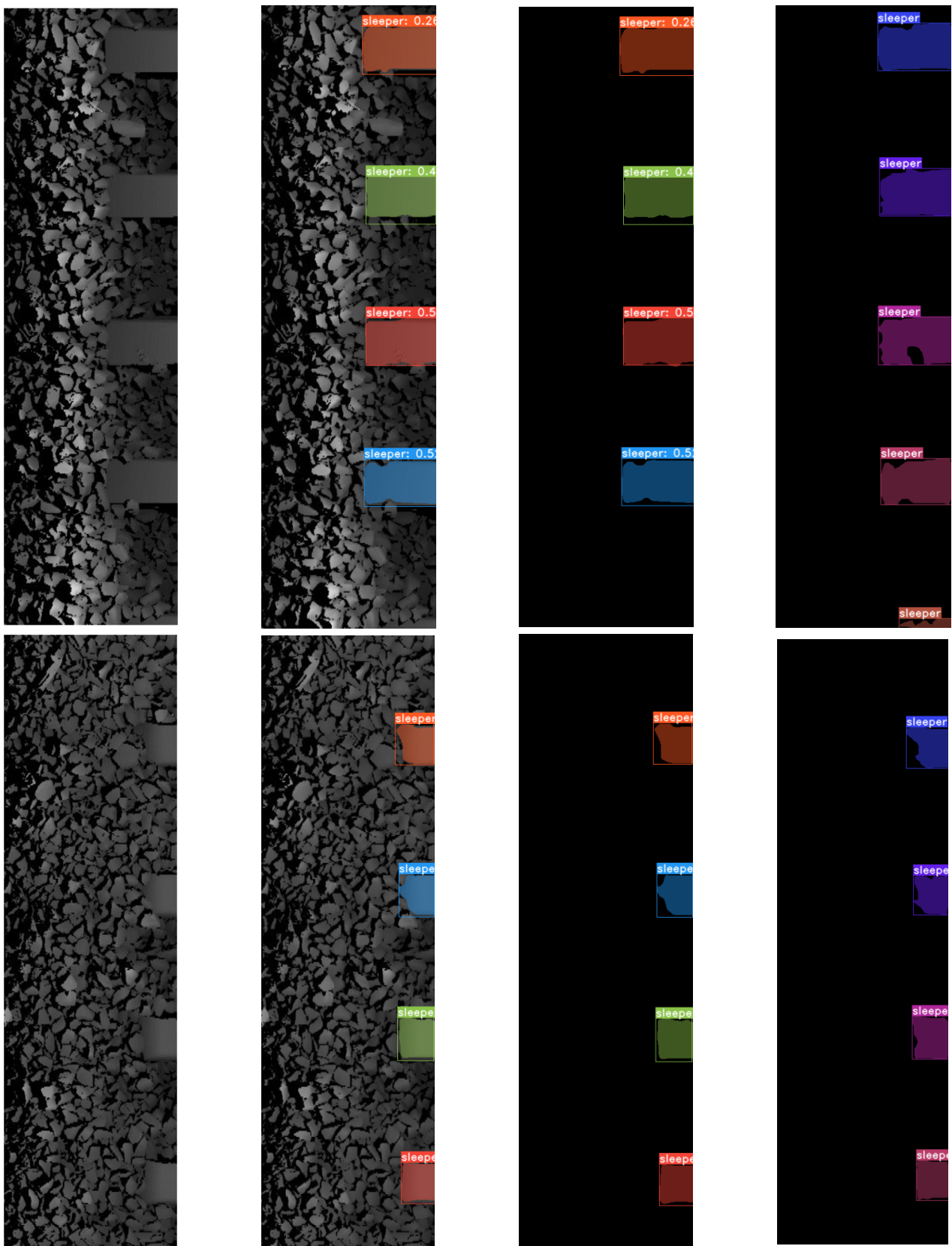


Figure 9. Examples of detection and segmentation of sleepers from intensity images of the track obtained for YOLACT with Resnet101 backbone. The first column represents the image of the track, the second the image with the predicted bounding box and mask overlaid on it, the third the predicted bounding box and mask and the fourth the target (truth) bounding box and mask.

According to [39], McNemar’s test satisfies all the aforementioned criteria. This test uses the contingency table to calculate the McNemar’s test statistic [39]:

$$statistic = \frac{(a - b)^2}{a + b} \quad (10)$$

Let *model_1* denote the first model and *model_2* denote the second model. Then, *a* denotes the number of samples correctly classified by *model_1* and simultaneously misclassified by *model_2*; *b* denotes the number of samples misclassified by *model_1* and simultaneously correctly classified by *model_2*; *a* and *b* are obtained from the contingency table.

If *a* and *b* are similar, both models make errors in much the same proportion, just on different instances of the test set. In this case, the result of the test would not be significant and the null hypothesis would not be rejected. On the other hand, if *a* and *b* are not similar, not only do both models make different errors, but in fact have a different relative proportion of errors on the test set. In this case, the result of the test would be significant and we would reject the null hypothesis. Therefore, the rejection of the null hypothesis means that the two models have different performances when trained on the particular training. The null hypothesis is rejected when *p* value ≤ alpha and is not rejected when *p* value > alpha. Alpha denotes a significance level and *p* value is probability and measures how likely it is that any observed difference between the models is due to chance. A *p* value close to 0 indicates that the observed difference is unlikely to be due to chance.

McNemar’s test was performed separately for the object classification part and the pixel classification part of the Mask R-CNN and YOLACT models. The test was conducted on 320 images (*test_data* set) that contained 1287 sleepers and 10,194,844 pixels belonging to sleepers. During testing, the significance level alpha was set to 0.05. In the test of the null hypothesis, “the object classification part of Mask R-CNN with Resnet101 and the object classification part of YOLACT with Resnet101 are the same model”, the McNemar’s test generated *p* value = 0.00464. Therefore, this hypothesis can be rejected and it means that these two models are different for the object classification parts. In the testing of the pixel classification part, the null hypothesis was formulated as “the pixel classification part of Mask R-CNN with Resnet101 and the pixel classification part of YOLACT with Resnet101 are the same model”. The McNemar’s test generated a *p* value = 0.00138. Therefore, this hypothesis can be rejected as well and it means that these two models are different for the pixel classification parts.

Based on Tables 1 and 2 and McNemar’s tests, the authors decided to choose the Mask R-CNN network with Resnet101 for the detection and segmentation of the sleepers. We emphasized the accuracy of the detection and segmentation of the sleepers at the expense of an increased processing time.

3.3. Detection of the Unevenness of the Ballast

After determining the position of the sleepers, the algorithm in Algorithm 1 detects areas of ballast irregularity whose levels exceed the assumed threshold = 35 mm relative to the sleeper level. Because the assessment of unevenness of the ballast boils down to an instance segmentation process, we checked the quality of the proposed algorithm using AP, IoU and Pixel Accuracy metrics for images from the *test_data* set. These three metrics were obtained for the areas where the height is out of the range of +/− 35 mm. Table 3 shows these metrics.

Table 3. Detection and segmentation metrics for ballast irregularity detection algorithm.

Average Precision AP	IoU	Pixel Accuracy
0.91	0.93	0.91

Figure 10 shows the results of the proposed algorithm for the automatic detection of ballast irregularities. Figure 10a,c,e,g show the range images (depth map) of the track section

and Figure 10b,d,f,h show the image with the detected sleepers and area of the ballast with the height out of range $\pm$ threshold overlaid on it (the area marked in light green).

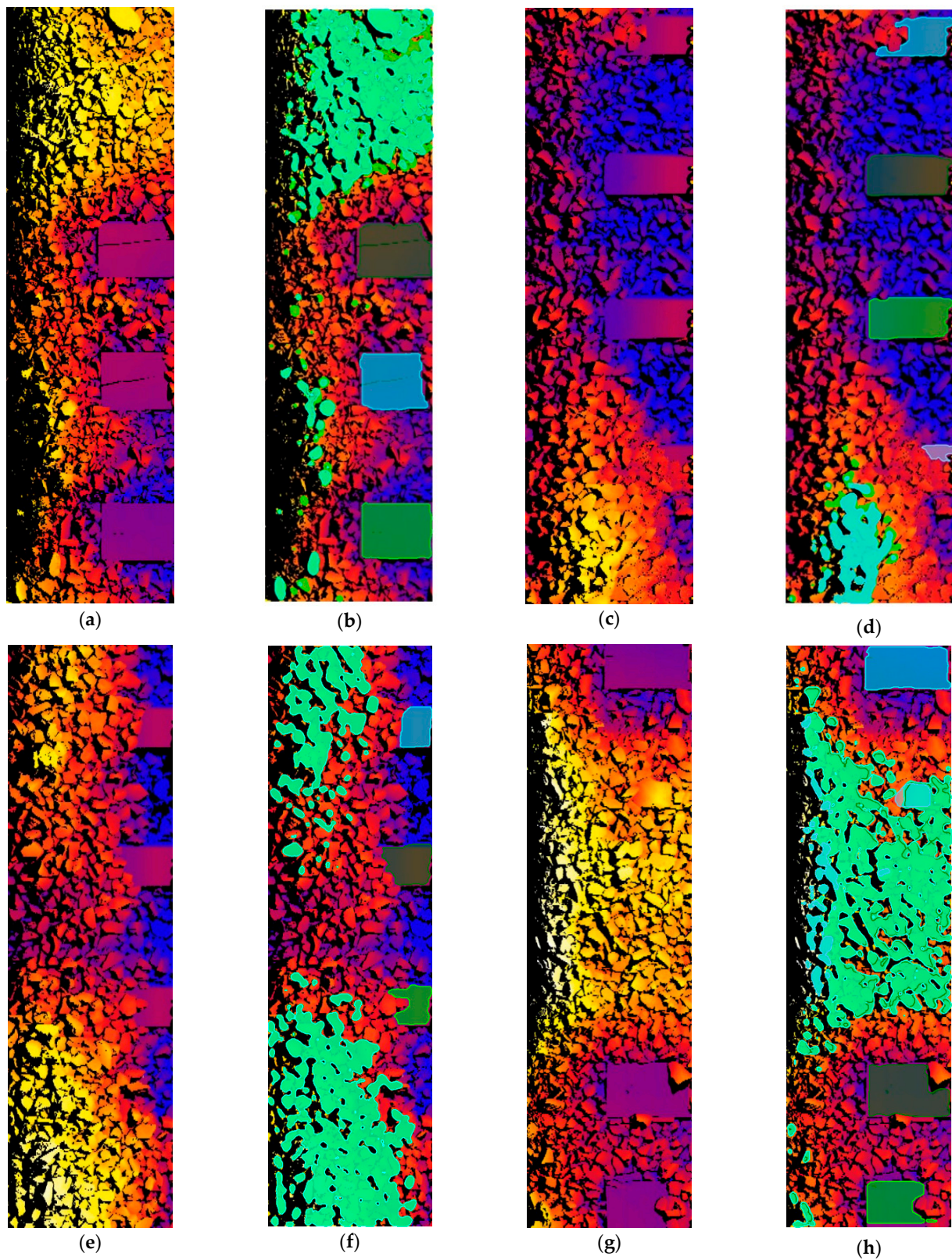


Figure 10. Example results of the proposed algorithm for automatic detection of ballast irregularities, (a,c,e,g) range image (depth map) of track section, (b,d,f,h) image with the detected sleepers and area of the ballast (marked in light green) with the height out of range $\pm$ 35 mm overlaid on it.

4. Conclusions

The algorithm presented in this article, based on deep neural networks, allows for the detection of ballast unevenness in compliance with a regulation introduced by the Polish railway lines. The unevenness of the ballast surface is one of the indicators of its condition. For this reason, a regulation was introduced by the Polish Railways specifying the maximum threshold of ballast unevenness. The algorithm presented uses track images which are scanned by a LASER triangulation system installed on a track inspection vehicle managed by the Polish railway line authorities. One of the key elements of the algorithm is the deep neural network used to detect and segment the sleepers from the intensity image. Two networks were tested: Mask R-CNN and YOLACT with two types of backbones: Resnet101 and Resnet51. Due to its higher accuracy ($AP = 0.8901$, $IoU = 0.78004$ and Pixel Accuracy = 0.8063), the Mask R-CNN network with Resnet101 was selected. Because the determination of unevenness of the ballast based on the track image boils down to an instance segmentation process (areas of unevenness of the ballast undergo detection and segmentation), the quality of the detection of ballast irregularity was verified with the help of the $AP = 0.901$ measure, while the quality of the segmentation of the ballast was checked using the $IoU = 0.93$ and Pixel Accuracy = 0.91 measures. The algorithm presented in the article, using Mask R-CNN, makes it possible to effectively assess the level of unevenness of the ballast, which is correlated with the state of the ballast. The proposed system allowed for the effective implementation of the regulations introduced by the Polish Railways regarding ballast evenness. It can be an alternative to other existing systems.

Author Contributions: Conceptualization, P.B. and P.L.; methodology, P.B. and W.N.; validation, W.N.; formal analysis, P.B. and P.L.; investigation, P.B.; resources, P.L.; data curation, P.B., P.L. and W.N.; writing—original draft preparation, P.B. and P.L.; writing—review and editing, W.N.; visualization, P.L. and W.N.; supervision, P.B.; project administration, P.B.; funding acquisition, W.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Acknowledgments: The authors would like to thank Ludwik Madej, head of the Diagnostic Measurements Department of the Office of the Diagnostic Centre of PKP Polskie Linie Kolejowe S.A., for providing the images from the ballast survey.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Remennikov, A.M.; Kaewunruen, S. Experimental load rating of aged railway concrete sleepers. *Eng. Struct.* **2014**, *76*, 147–162. [\[CrossRef\]](#)
2. Sadeghi, J.M.; Zakeri, J.A.; Najjar, M.E.M. Developing Track Ballast Characteristic Guideline in order to Evaluate its Performance. *IJR Int. J. Railw.* **2016**, *9*, 27–35. [\[CrossRef\]](#)
3. Scanlan, K.M. Evaluating Degraded Ballast and Track Geometry Variability along a Canadian Freight Railroad through Ballast Maintenance Records and Ground-Penetrating Radar. Ph.D. Thesis, Civil and Environmental Engineering, University of Alberta, Edmond, AB, Canada, 2018. [\[CrossRef\]](#)
4. Wang, S.; Liu, G.; Jing, G.; Feng, Q.; Liu, H.; Guo, Y. State-of-the-Art Review of Ground Penetrating Radar (GPR) Applications for Railway Ballast Inspection. *Sensors* **2022**, *22*, 2450. [\[CrossRef\]](#) [\[PubMed\]](#)
5. Silvast, M.; Nurmikolu, A.; Wiljanen, B.; Levomaki, M. An Inspection of Railway Ballast Quality Using Ground Penetrating Radar in Finland. *Proc. Inst. Mech. Eng. Part F J. Rail Rapid Transit.* **2010**, *224*, 345–351. [\[CrossRef\]](#)
6. Massaro, A.; Dipierro, G.; Selicato, S.; Cannella, E.; Galiano, A.; Saponaro, A. Intelligent Inspection of Railways Infrastructure and Risks Estimation by Artificial Intelligence Applied on Noninvasive Diagnostic Systems. In Proceedings of the 2021 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT), Rome, Italy, 7–9 June 2021; pp. 231–236. [\[CrossRef\]](#)
7. Benedetto, A.; Tosti, F.; Bianchini Ciampoli, L.; Calvi, A.; Brancadoro, M.G.; Alani, A.M. Railway ballast condition assessment using ground-penetrating radar—An experimental, numerical simulation and modelling development. *Constr. Build. Mater.* **2017**, *140*, 508–520. [\[CrossRef\]](#)

8. Artagan, S.S.; Borecky, V. Advances in the nondestructive condition assessment of railway ballast: A focus on GPR. *NDT E Int.* **2020**, *115*, 102290. [\[CrossRef\]](#)
9. Shangguan, P.; Al-Qadi, I.L.; Leng, Z. Ground-Penetrating Radar Data to Develop Wavelet Technique for Quantifying Railroad Ballast–Fouling Conditions. *Transp. Res. Rec. J. Transp. Res. Board* **2012**, *2289*, 95–102. [\[CrossRef\]](#)
10. Lama, H.F.; Wongb, M.T. Railway Ballast Diagnose through Impact Hammer Test. *Procedia Eng.* **2011**, *14*, 185–194. [\[CrossRef\]](#)
11. Wang, M.; Xiao, Y.; Li, W.; Zhao, H.; Hua, W.; Jiang, Y. Characterizing Particle-Scale Acceleration of Mud-Pumping Ballast Bed of Heavy-Haul Railway Subjected to Maintenance Operations. *Sensors* **2022**, *22*, 6177. [\[CrossRef\]](#) [\[PubMed\]](#)
12. Liang, X.; Niu, X.; Liu, P.; Lan, C.; Yang, R.; Zhou, Z. Test on fouling detection of ballast based on infrared thermography. *NDT E Int.* **2023**, *140*, 102956. [\[CrossRef\]](#)
13. Zarembski, A.M.; Grissom, G.T.; Euston, T.L. On the use of Ballast Inspection Technology for the Management of Track Substructure. *Transp. Infrastruct. Geotechnol.* **2014**, *1*, 83–109. [\[CrossRef\]](#)
14. Sadeghi, J.; Najar, M.E.M.; Zakeri, J.A.; Kuttelwascher, C. Development of railway ballast geometry index using automated measurement system. *Measurement* **2019**, *138*, 132–142. [\[CrossRef\]](#)
15. Aldao, E.; González-Jorge, H.; González-deSantos, L.M.; Fontenla-Carrera, G.; Martínez-Sánchez, J. Validation of Solid-State LiDAR Measurement System for Ballast Geometry Monitoring in Rail Tracks. *Infrastructures* **2023**, *8*, 63. [\[CrossRef\]](#)
16. Sánchez-Rodríguez, A.; Riveiro, B.; Soilán, M.; González-deSantos, L.M. Automated detection and decomposition of railway tunnels from mobile laser scanning datasets. *Autom. Constr.* **2018**, *96*, 171–179. [\[CrossRef\]](#)
17. Paixao, A.; Afonso, C.; Delgado, B.; Fortunato, E. Evaluation of ballast particle degradation under micro-deval testing using photogrammetry. *Advances in Transportation Geotechnics IV. Lect. Notes Civ. Eng.* **2022**, *165*, 113–124. [\[CrossRef\]](#)
18. Lesiak, P.; Bojarczak, P.; Sokolowski, A. Algorithm for the extraction of selected rail track balast degradation using machine vision. *Transp. Probl.* **2023**, *18*, 129–141. [\[CrossRef\]](#)
19. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with Deep Convolutional Neural Networks. *Commun. ACM* **2012**, *60*, 1097–1105. [\[CrossRef\]](#)
20. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaria, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *J. Big Data* **2021**, *8*, 1–74. [\[CrossRef\]](#) [\[PubMed\]](#)
21. Modi, A.S. Review Article on Deep Learning Approaches. In Proceedings of the 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 14–15 June 2018; pp. 1635–1639. [\[CrossRef\]](#)
22. Samek, W.; Montavon, G.; Lapuschkin, S.; Anders, C.J.; Müller, K.R. Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications. *Proc. IEEE* **2021**, *109*, 247–278. [\[CrossRef\]](#)
23. Sharma, P.; Singh, A. Era of deep neural networks: A review. *Proc. Int. Conf. Comput. Commun. Netw. Technol.* **2017**, 1–5. [\[CrossRef\]](#)
24. Shrestha, A.; Mahmood, A. Review of Deep Learning Algorithms and Architectures. *IEEE Access* **2019**, *7*, 53040–53065. [\[CrossRef\]](#)
25. Wu, H.; Liu, Q.; Liu, X. A review on deep learning approaches to image classification and object segmentation. *Comput. Mater. Contin.* **2019**, *60*, 575–597. [\[CrossRef\]](#)
26. Zhao, Z.; Zheng, P.; Xu, S.; Wu, X. Object Detection with Deep Learning: A Review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [\[CrossRef\]](#) [\[PubMed\]](#)
27. Hafiz, A.M.; Bhat, G.M. A survey on instance segmentation: State of the art. *Int. J. Multimed. Inf. Retr.* **2020**, *9*, 171–189. [\[CrossRef\]](#)
28. Gu, W.; Bai, S.; Kong, L. A review on 2D instance segmentation based on deep neural networks. *Image Vis. Comput.* **2022**, *120*, 104401. [\[CrossRef\]](#)
29. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. 3rd International Conference on Learning Representations (ICLR 2015). *arXiv* **2014**, arXiv:1409.1556. [\[CrossRef\]](#)
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778. [\[CrossRef\]](#)
31. Shelhamer, E.; Long, J.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 640–651. [\[CrossRef\]](#) [\[PubMed\]](#)
32. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988. [\[CrossRef\]](#)
33. Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 936–944. [\[CrossRef\]](#)
34. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y. YOLACT: Real-time Instance Segmentation. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9157–9166. [\[CrossRef\]](#)
35. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016*; Part I 14; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2016. [\[CrossRef\]](#)

36. Tan, C.; Sun, F.; Kong, T.; Zhang, W.; Yang, C.; Liu, C. A survey on Deep Transfer Learning. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, 4–7 October 2018; Part III* 27; Springer: Berlin/Heidelberg, Germany, 2016. [[CrossRef](#)]
37. Iman, M.; Arabnia, H.R.; Rasheed, K. A Review of Deep Transfer Learning and Recent Advancements. *Technologies* **2023**, *11*, 40. [[CrossRef](#)]
38. Padilla, R.; Netto, S.L.; da Silva, E.A.B. A Survey on Performance Metrics for Object-Detection Algorithms. In *Proceedings of the 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), Niteroi, Brazil, 1–3 July 2020*; pp. 237–242. [[CrossRef](#)]
39. Dietterich, T.G. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Comput.* **1998**, *10*, 1895–1923. [[CrossRef](#)] [[PubMed](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.