

Surface Defect Detection Based on Deep Learning Approach

Mykola Robotyshyn, Marianna Sharkadi and Mykola Malyar

Uzhhorod National University, Narodna Square 3, Uzhhorod, 88000, Ukraine

Abstract

Quality inspection is one of the most essential parts of any manufacturing process that helps businesses to ensure the quality of their product by detecting defects which is extremely important due to high market competition. Deep-learning methods have become the most promising approaches to solve this problem. The advantage of using deep learning is that it can detect the defects that cannot be detected by traditional machine vision algorithms. This paper presents a segmentation-based deep learning quality inspection system that is designed for detecting defects both outside and inside an object's surface. The system was tested on specific domain (reeds/bamboo straws) but can be generalized to any domain. The design of the system's architecture allows to iterate over typical machine learning cycle (collect data, model training, error analysis) in a fast way due to real-time collection of images on which the system makes mistakes. In this paper we gave equal attention to all crucial parts of the system: segmentation neural networks, decision algorithms, dataset and monitoring system. In our case we demonstrate some advantages of using an ensemble of binary segmentation models over one multiclass model, especially when it comes to data labeling. We discovered that applying post-processing rules after segmentation can significantly improve accuracy of your model. Experiments are performed on newly created dataset with real-world images from reeds straw factory and system's errors were compared with human level performance. The dataset is available on request for other to develop and test new models for surface defect detection problem.

Keywords ¹

Surface defect detection, deep learning, segmentation networks, quality and visual inspection, computer vision, data labeling, reed straws surface defect detection

1. Introduction

In industrial processes, one of the most important tasks when it comes to ensuring the proper quality of the finished product is inspection of the product's surfaces. Often, surface quality control is carried out manually and workers are trained to identify complex surface defects. Such control is, however, very time consuming, inefficient, and can contribute to a serious limitation of the production capacity [1]. To overcome these factors many giant companies started to implement their built-in solutions, so called automated visual inspection systems. These systems have a big level of generalization, the same solution works equally on different factories, totally different light conditions and even sometimes with different objects.


Despite tremendous achievements in automated visual inspection solutions there are still many areas where quality checking is performed by humans. Main reason for that is impossibility to apply built-in visual inspection solutions due to non typical object view, non typical object surface and variety of defects that determine whether a product is suitable for sale. For this case there is a need to build a customized inspection system from scratch. Building such systems from scratch is not

II International Scientific Symposium «Intelligent Solutions» IntSol-2021, September 28–30, 2021, Kyiv-Uzhhorod, Ukraine
EMAIL: mykolarobotyshyn@gmail.com (M. Robotyshyn); marianna.sharkadi@uzhnu.edu.ua (M. Sharkadi); mykola.malyar@uzhnu.edu.ua (M. Malyar)

ORCID: 0000-0001-6567-6974 (M. Robotyshyn); 0000-0002-1850-996X (M. Sharkadi); 0000-0002-2544-1959 (M. Malyar)

© 2021 Copyright for this paper by its authors.

Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

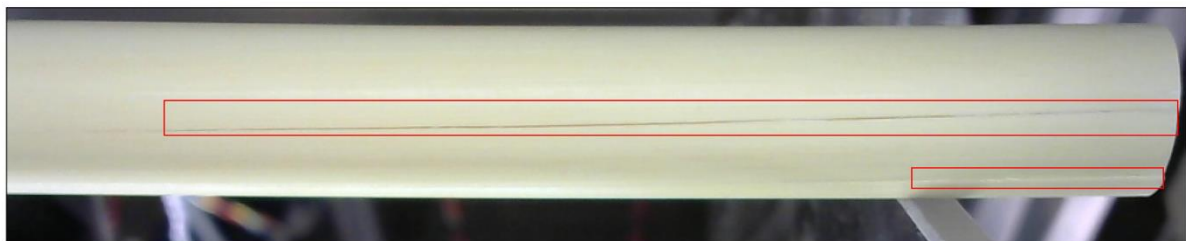
straightforward and includes many iterative steps: camera settings, data collection pipeline, model training, error analysis, deploying and real-time monitoring system.

This paper focuses on providing a detailed guide on how to build a surface defect detection system from scratch using modern machine learning approaches. Deep learning approaches have become the first choice methods when it comes to work with images. More and more new methods have been invented during the last few years that show huge improvement both in terms of accuracy and required computational resources. Compared to classical machine vision methods deep learning can directly learn low-level features, and have a higher capacity to represent complex structures, thus completely replacing hand engineering of features with automated learning process[1]. The algorithm that solves surface defect detection problems should be capable of finding the defect area on an object's surface, thus it means not just classify whether an object is defectable, but to understand exactly which pixels are defectable. As for any other supervised approaches to teach a system to detect defects we need to provide labeled data to the algorithm. The first question while working with deep learning supervised methods is how many annotated images we need to provide to achieve an acceptable level of accuracy? This is a crucial question because usually it is impossible to collect hundreds of thousands labeled images thus there is a need for either a neural network that can learn from small amounts of samples or very quality and diverse dataset. The one part the paper focuses on is the iterative process of how to collect appropriate dataset to train a model. We achieved accuracy results close to human level performance with approximate dataset's size within a range of 1.5k to 3k labeled images which is possible to collect for any domain.

Partly, this paper explores suitable deep learning methods to solve surface defect detection problems. In particular, the paper studies state-of-the-art semantic segmentation neural networks applied to a variety of different surface defects like cracks, spots, small circles holes, hammered stuff inside (see Fig. 1, Fig. 2). While choosing suitable neural network architecture we take into account four characteristic.

- speed of the network
- computational requirements
- accuracy on benchmark ImageNet dataset
- annotation requirements

Because it is possible to have more than one defect on object we experimented with multi-class neural network and ensemble of binary neural networks. Using separate networks for each class has its pros and cons. Obviously, the main disadvantage is significantly lower speed performance of the system. But there are a few advantages that are extremely important in practical cases. Firstly, easier and faster way to label images because each image contains only 2 classes: defect and no-defect. Secondly, it requires less amount of labeled images. And thirdly, total accuracy is usually higher.



(a) Defect class: cracks



(b) Defect class: circle holes

Figure 1: Defect classes where red color highlights defectable area

We evaluate our system on the dataset termed Reeds Surface Defect Dataset (ReedsSDD) that we created ourselves from reeds factory. Partly, the paper provides essential information regarding collecting images and labeling them. Capability to collect quality images is a crucial step to build a highly accurate system. Specifically, for surface defect detection problem quality images mean two things. Firstly, a list of images that covers the full surface of object's thus there is a need for more than one camera and their correct placement. Secondly, the defect area on image should be recognizable by the human eye thus there should be twice more attention to image quality because very often the defect area is small, like in crack defects. The paper covers information about camera model and settings we used, solution to cameras placements and tips on how to adjust camera settings to produce more quality images than with default settings.

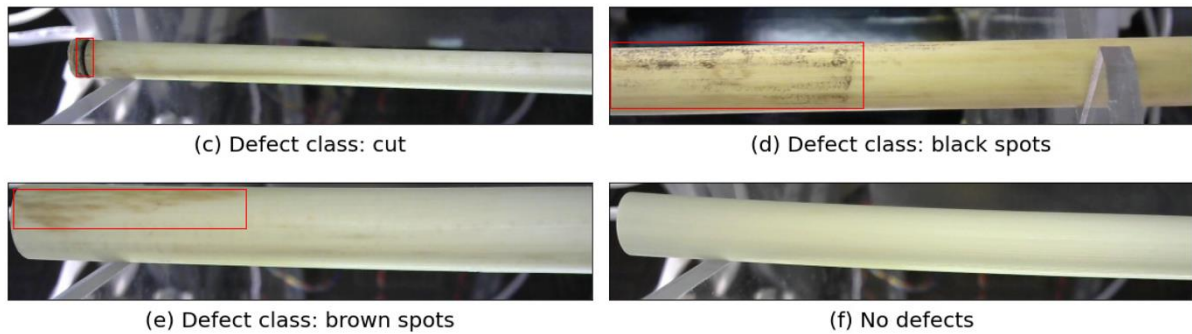


Figure 2: Defect classes where red line highlights defectable area

The remainder of the paper is organized as follows. The related work is presented in “Related work” section, with proposed approach and details of all important steps to build system from scratch in “Proposed approach”, important information regarding dataset in “Dataset” and system’s results comparison with human level performance together with implementation details in section “Evaluation”. The paper concludes with a discussion in “Discussion and conclusion” section.

2. Related work

After the breakthrough paper[2] deep learning approaches became more popular in any domain and manufacturing is not an exception. There are many practical cases where convolutional neural networks outperform classic machine learning algorithms on problems where input data are images. Main advantage of deep learning approaches is that they do not require any changes in algorithms when you apply them to another domain. On the other hand, in classic computer vision approaches, you very likely need to play with algorithm’s hyperparameters and even with image preprocessing operations.

Much research work has been done in the field of designing neural networks architectures so that acceptable accuracy can be achieved with a minimum amount of labeled images which is extremely important in practical cases when data labeling and collection is a time consuming process. Research work by [1] presents architectures that require only 25-30 labeled images to achieve enough level of accuracy. Most state-of-the-art semantic segmentation models use encoder-decoder architecture firstly described in [3] that proved results on biomedical image segmentation tasks. One well known problem when dealing with deep learning is the model’s speed thus more and more new approaches focused on designing neural networks with smaller amounts of parameters[4,5].

There are many works on applying supervised neural networks for surface defect detection problems. In 2012 the work of Masci et al. [6] was among the first who applied shallow neural networks to surface defect detection. Later papers [7,8,9,10] described the use of deep neural networks for detection of rail surface defects, brain tumor segmentation, inspection of laser welding defects, steel pipe defect detection respectively. During last years tremendous success were achieved in semi-supervised, weakly-supervised learning - approaches that partly eliminated the problem of labeled data. Latest publications[11,12] from top-notch companies like Facebook, Google showed compatible results of self-supervised methods compared to supervised on benchmarks datasets. Latest

work on surface-defect detection problems very often related to self-supervised learning. Combining small amounts of pixel-level labeled images with weakly labeled images can outperform most state-of-the-art supervised results in defect detection problems[13]. One-shot learning method was described in [14] to detect defects on steel surface. [15] contains a detailed overview of modern efficient approaches to solve problem.

Top giant companies like Omron provide built-in solutions called automatic visual inspection systems. [16] provides an overview of automatic visual inspection systems, their capabilities and limitations, areas where they can be used. To speed up building defect detection system from scratch Landing.AI company provides end-to-end visual inspection platform LandingLens [17]. It is designed to manage data, to perform automatic error-analysis and allows to train neural networks without programming knowledge.

Surface defect detection problem regarding reeds/bamboo straws domain hasn't been explored yet mainly because of lack of publicly available datasets. There are a limited number of papers, the classic computer vision approaches described in [18, 19] used for bamboo straw defect inspection. And for reeds straws defect detection there we did not find published papers yet.

Compared with methods mentioned above, the approach proposed in this paper is fully supervised and consists of two main steps. The first one is an ensemble of binary U-Net semantic segmentation models where each model is trained for one type of defect. Each model is trained to detect corresponding defect's pixels and runs consecutively one after one. The second step is a rule-based system that based on presence of defects, their type and area classify object to one of predefined classes. Models were trained on Reeds SDD dataset that we collected on reeds factory. The used dataset consists of a relatively small amount of images within a range of 1.5k to 3k for each defect type.

3. Proposed approach

We addressed the surface defect detection problem as a segmentation-image problem. There are two main reasons why the segmentation approach is more suitable for this problem rather than classification. Firstly, the visual inspection system is required to highlight the exact defectable area on the object thus making predictions on pixel-wise level. Secondly, the classification approach has a low level of interpretability by making predictions on image-wise level and it leads to significant difficulties on error analysis stage to understand why model makes one or another type of mistakes and what changes need to be done to improve model. The process of object's defect detection consists of 2 main steps. The first one is an ensemble network - an ensemble of binary segmentation models that has been chosen over one multi-class model due to easier labeling process, less amount of training images and better total accuracy. The second step, where we classify the object to one of predefined classes, includes a rule-based process that is built on top of the ensemble network and uses ensemble network's output about presence of defects and area size to classify the object. The first step is referred to as an ensemble network, while the second stage, as rule-based process. Real time monitoring system has been built to provide an option to see highlighted defect area and respective size thus it provides an explanation why a system takes one or another decision. Three main components: ensemble network, rule-based process, monitoring system together with data collection pipeline that will be described later create a real-time defect detection system that paper is focused on[fig 3].

3.1. Ensemble network

Binary segmentation model returns probability for each pixel being defecting. Ensemble network consists of seven binary models, where each model is trained to detect one type of defect. Binary approach significantly speeds up the labeling process and moreover has more precise accuracy because the defects are very different in terms of area size, appearance, possible quantity per image and pixel-wise importance. Each model in ensemble networks runs consecutively one by one, the process can be stopped immediately depending on defect presence and the model's output goes as input to a rule-based process.

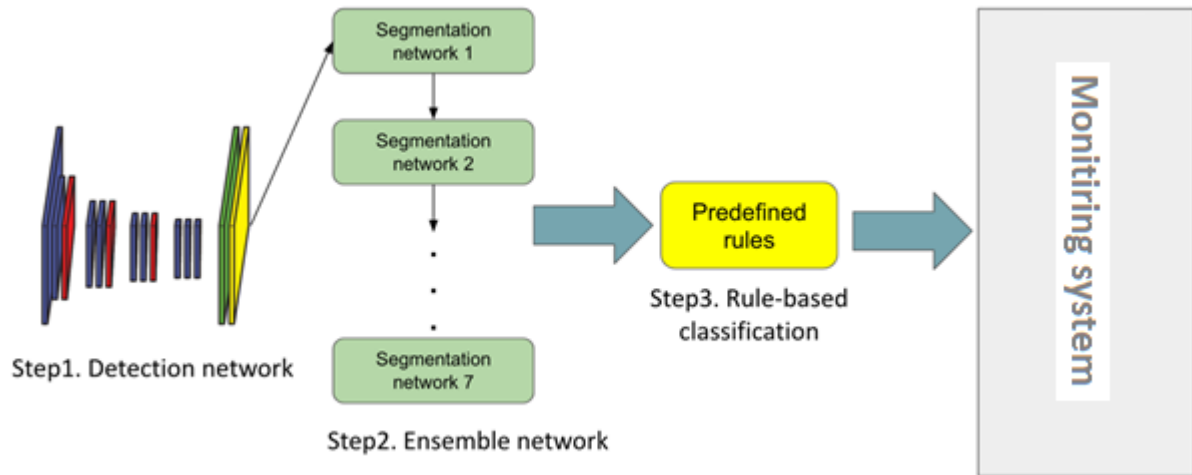


Figure 3: Main components of the system

Nowadays encoder-decoder architecture is the go-to choice when dealing with deep segmentation neural networks. Often the encoder is a classification neural network with pre-trained weights on ImageNet dataset. The aim of the encoder is to learn low-level representational features called feature map and usually it consists of convolutional filters followed by batch-normalization, non-linear activation function and max-pooling. Decoder takes feature map as input and up-samples it to get a dense classification. Ensemble network consists of models which are equal in terms of architecture and designed as U-Net model with MobileNetV2 pretrained encoder. Each model contains 6.5ml parameters. U-Net architecture with MobileNetV2 is a good tradeoff between the amount of parameters and benchmark accuracy on the ImageNet dataset. At the time of the study, pretrained MobileNetV3 encoder was not publicly available to use.

Binary segmentation model returns a two-dimension matrix termed as mask where each value corresponds to a pixel and has probability in range $[0,1]$ for being defected. As a closer value to 1 is more likely the pixel is defected. Each binary model has two hyperparameters: probability threshold and min area size. These values are used in post-processing operations. The first one, *probability threshold* is used to convert probabilities into either 1 or 0 by applying threshold operation which can be formulated as follows: if pixel value is less than threshold then assign 0 else 1. Modifying the *probability threshold* parameter allows to make the model more or less strict. For example, if probability threshold = 0.9, that means we want the model to predict that pixel is defected only in cases when the model is very confident thus the pixel probability is close to 1. The second hyperparameter, that applied after threshold operations, is termed as *min_area size* - minimal amount of defected pixels on the image to consider an object to be defected. What if an image contains 10 defective pixels? Does it mean that the object is defected? In practice, a defected object means that some area of surface is defected. Depending on the problem, type of defect, type of object's surface, the size of defected area to consider the object as defected is different. That's why we have a hyperparameter *min_area size* that controls the minimal size of the defected area. The main logic is - if the amount of defected pixels is less than the hyperparameter's value then the object is not defected and corresponding pixels will be assigned a value of 0. To sum up, adjusting two hyperparameters mentioned above for the specific type of defect allows to improve accuracy of each binary segmentation model thus to improve accuracy for ensemble network.

Very common problem when applying machine learning algorithms in production is false positive errors especially when it comes to using deep neural networks[20]. Amount of false positive errors increases when we slightly change the distribution of real-time data compared to training data. It happens, for example, when light conditions or image background are changing. Moreover these errors are sometimes very unreasonable like predicting defected areas outside the object's surface. To remove these unreasonable types of mistakes we propose one more post-processing operation that helps to make the system more stable. Before running an ensemble network we use another segmentation model to determine the area where the object is located. This is an easy problem and accuracy of this model is very close to optimal. Having this detection network allows us to remove

false predictions from ensemble networks that are outside the object's surface area. Detection network is the initial component of the system and also it allows to calculate the diameter of the object.

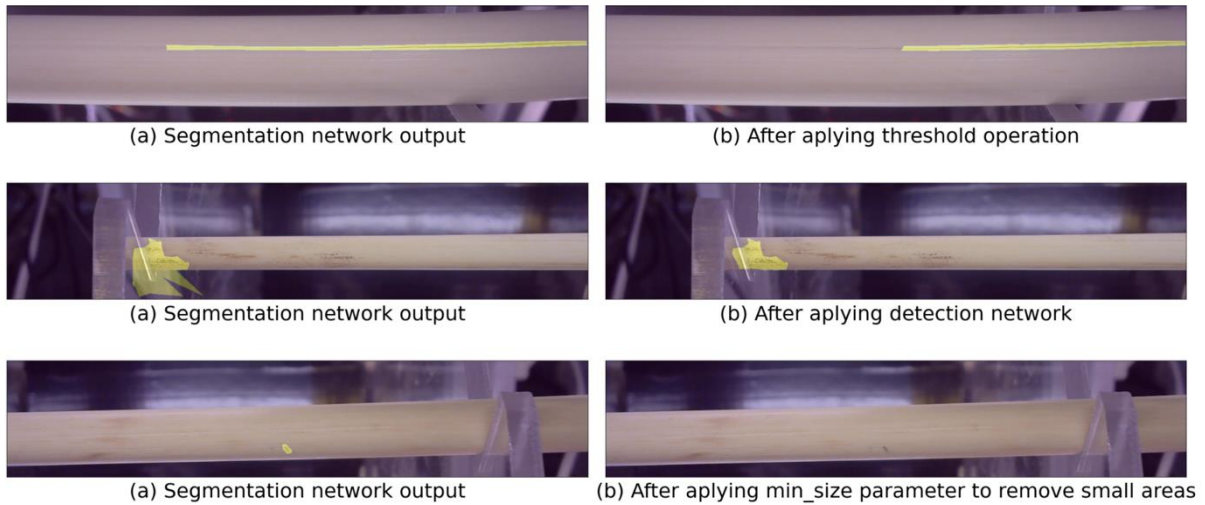


Figure 4: Post-processing operations

To sum up, an ensemble network together with an initial detection network are responsible for detecting different types of defects and size of corresponding defectable area. Later this information goes as input to rule-based classification. To give system better capabilities to generalization thus to be more confident in accuracy we apply post-processing operations [Fig 4].

3.2. Rule-based classification

Rule-based classification is a second main step in the system pipeline that takes as input information about what type of defects are present in an object, what corresponding defect area size is and what is the diameter of the object. In manufacturing processes some defects can be eliminated from the object, some objects can be divided by their size or diameter, some objects are more expensive than others thus there is a need to implement further classification of objects based on information from ensemble and detection networks.

In our case, in the factory objects could be classified into 11 different types. The classification procedure is based on predefined rules thus we termed this stage as rule-based classification. Simple rules look like:

- If the object contains no defects and its diameter is less than 10mm then classify the object as class7
- If the object contains defect type “cracks” or defect type “holes” then classify object as class0
- If the object contains defect type “spots” and defectable area is less than X pixels then classify object as class4

These rules should be able to classify any objects to one of predefined classes. The output of rule-based classification is final information about an object: object’s class and additional information like diameter of the object, defectable area size. Later output goes as input to the monitoring system thus it is important to send not only the object’s class but also relevant information regarding previous outputs of the system to provide as much as possible information to the monitoring system to better understand why the whole system takes one or another decision.

3.3. Monitoring system

Lack of interpretability is one of the main issues why businesses doubt the use of machine learning systems in production. Much research work has been done to understand the deep neural network’s behaviour, to understand why a model makes one or another decision and the best method to achieve it is visualization of feature maps on each layer[21]. Big advantage of using the segmentation approach is that it works on pixel-level thus we know the model's prediction not only for the whole

image but also for each pixel on the image. It allows us to stack input image with a predicted mask and create stacked image.

There are 3 main goals of building a monitoring system: interpretability, control the quality of images and real-time data collection. Firstly, as mentioned above, interpretability is crucial for integrating machine learning solutions to production. To guarantee it we plot a stack image with highlighted defectable areas thus it is possible in a moment to understand whether prediction is correct and why the system classified the object to corresponding class. Secondly, to ensure the correctness of the system, we must make sure that the images that the system takes as an input are high quality. For each object, the monitoring system plots the images taken from cameras. If something goes wrong, like blind images or moving a camera we can detect it and moreover if there are multiple cameras we know exactly which one needs to be fixed. The third goal is about creating opportunities to collect more data, especially to collect mistakes that the system makes. To improve a model's accuracy it is important to understand what mistakes the model makes and fill the train dataset with more images on which model failed.

Monitoring system is the last component of the defect detection system and as valuable as other components. In simple words it helps to “understand” why the system makes one or another decision. For reeds straw case monitoring system describes the following information:

- Diameter of the straw
- Defectable area
- Type of defect
- Straw class

3.4. Iteration cycle

In machine learning the process of creating a solution or model is always iterative and consists of three main steps: data preparation, model training and error analysis. According to Andrew Ng data-centric approach[22] researchers need to spend more time to prepare data and perform correct error analysis rather than focus only on model's architecture research. The goal of this chapter is to present how we can iterate over a typical machine learning cycle faster using a monitoring system for manufacturing cases.

Let's assume that we trained a model on the first version of the dataset. Now we need to determine accuracy thus we decide to test a model on new objects. Monitoring system allows us to analyze model performance by “watching” the model's prediction, defectable areas. Very often it is not enough just to increase the dataset size to improve accuracy. It is important to increase the dataset by adding images on which model makes mistakes. With the help of a monitoring system we are able to analyze predictions thus to identify types of images on which model makes mistakes. How to collect these images on production? Here it comes again to a monitoring system that allows real-time collection of all images taken from cameras.

To sum up, the monitoring system is a tool that helps to do error analysis and data collection processes much faster than without it. And moreover it allows us to collect not just a big dataset but good. We encourage everyone to build this system and perform error analysis and data collection steps using it.

4. Dataset

The proposed system is evaluated on a Reeds Surface Defect Detection (Reeds SDD) dataset that we created ourselves in an industrial factory. This section provides details regarding dataset creation from scratch, camera's settings and placements, labeling images. To get access to the dataset, please contact authors. We guarantee to provide quick access.

There are many options for what camera type to use and as in machine learning choosing a camera is also an iterative process. We experimented with different cameras starting from industrial one like Basler and simple web camera Logitech C270. The main criterion while taking the decision whether a camera is suitable or not is quality of image taken from camera. Taking into consideration that we are dealing with surface defect detection problem, one should pay twice more attention to assure good image quality. How to understand without collecting a lot of images and not training neural networks

whether image quality is acceptable and neural networks will detect defects well? There is one simple rule: defectable areas should be recognizable by the human eye when looking at the image. If humans do not see these areas in the image then very likely neural networks also will fail. And if humans easily recognize these areas in a moment then image quality is acceptable for further data collection. After repeating this process we selected the Logitech C270 web camera due to the acceptable level of image quality and much lower price compared to industrial cameras.

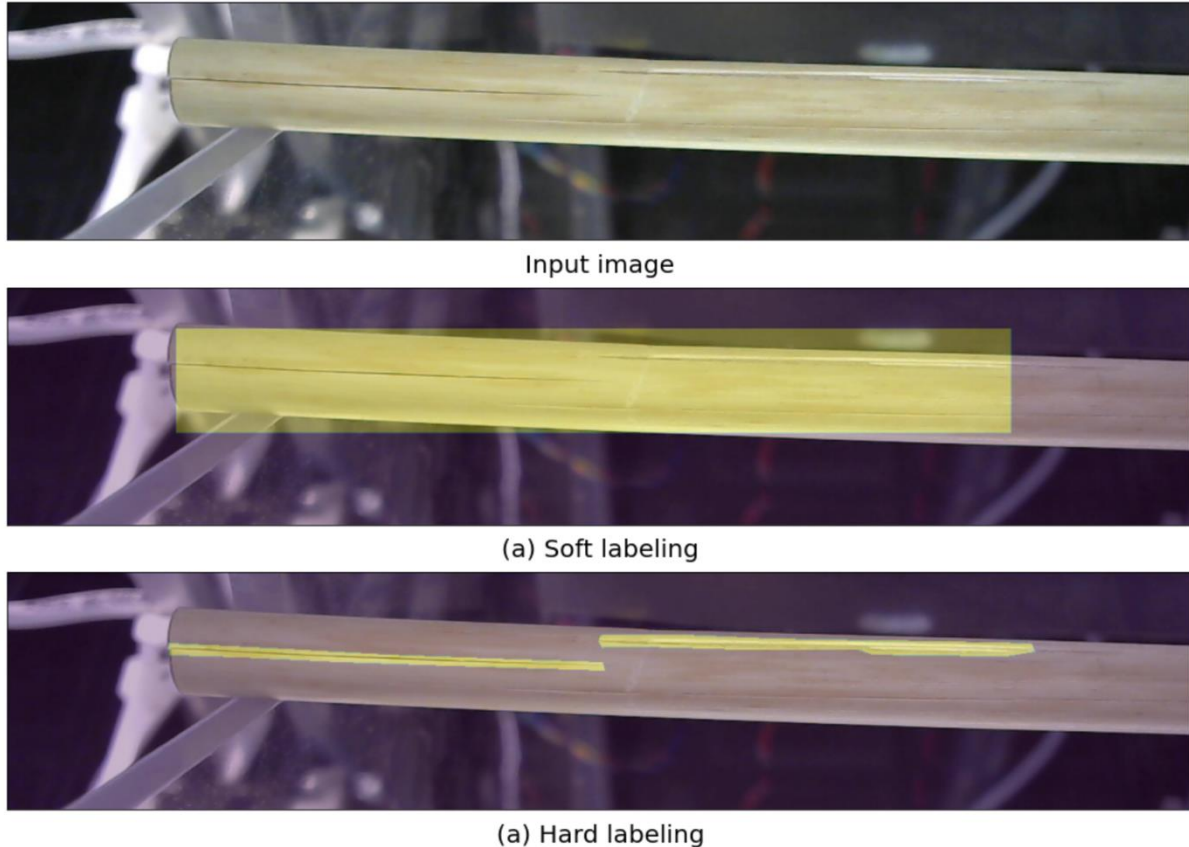


Figure 5: Comparison of “hard” and “soft” labeling

In this paragraph we focus on important camera’s settings and how to adjust them to improve image quality compared to default settings. Main settings are the next:

- Focus distance
- Auto-exposure

In surface defect detection problem there is a need to place cameras close to the object to see the object’s surface with high quality. According to our experiments approximately 4.7 inches is enough distance to see all defectable areas on reeds straws objects. After moving the camera closer to the object focus distance should be changed to prevent image from blurring.

Reeds straw’s surface has different shades of brown and dark colors thus we need to assure light conditions to make these shadows easily distinguishable. In practice the system should be designed in a way that small changes in light condition don’t affect the system accuracy. Adjusting auto-exposure allows to make images more stable to light changes and moreover allows to change the camera’s exposure to make images brighter or darker without changing outside light. We noticed that best settings values for auto exposure are within a range of 40-60.

To detect defects on more complex objects like reed straws it is not enough to make images only from one camera. Images should cover all surface area thus there is a need to understand what optimal amount of cameras we need and what is the most suitable placement for them. Based on our experiments we determine that a web camera is capable of covering 120 degrees. In order to clearly see defects like crack defects we need to place cameras not far than 4.7 - 5 inches from the object but it allows us to cover only half of the straw’s length. So, two cameras were placed to take images of 120 degrees of straw’s surface. To cover the entire surface area (360 degree), four more cameras were

placed so that the area they were looking at did not intersect with other camera's view areas. Having determined the optimal way to place the cameras, we began to make images to create a dataset.

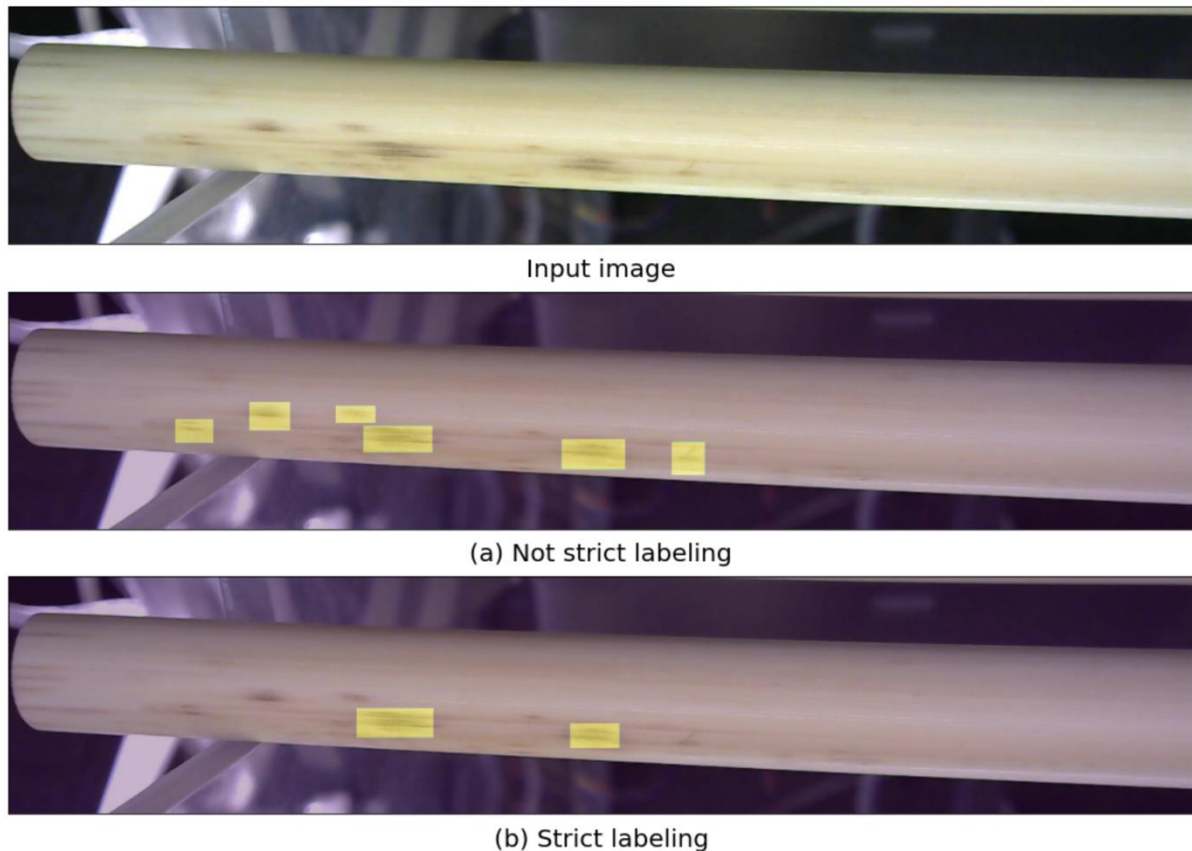


Figure 6: Comparison of “strict” and “not strict” labeling

The most time-consuming process of creating a dataset is not taking images but labeling them. Segmentation approach requires to label each defectable area on the image. To simplify and speed up the labeling process we created a separate dataset for each type of defect thus it required to label only one corresponding type of defect on the image. From the speed perspective we proposed two ways of labeling termed “hard labeling” and “soft labeling”. The difference between them is how accurate labels are [see Fig 5]. In “hard labeling” we labeled each defectable area very accurately, pixel by pixel using a polygon figure so it takes significantly more time compared to “soft labeling” that requires to label defects nearly accurately, for example using rectangles and covering areas without defects also. The reason we proposed two approaches is because some types of defects like brown or dark spots are easy to detect thus it requires not super accurate labels, but for defects like crack lines, small holes accurate labels are crucial. From the defect definition perspective we proposed two other ways of labeling termed “strict” and “not strict”. Very often because of the object's surface specialities and ambiguous defect definition it is not possible to say that this area is 100% defectable and another area is 100% not defectable. We called these ambiguous areas as arguable areas. “Strict” way of labeling means that most arguable areas were labeled as defectable while “not strict” means that these areas we did not label [see Fig 6]. Again it depends on the type of defects and defect's importance. “Strict” labeling is more sensitive to a bigger amount of false positive mistakes. Labeling is a very important process so choosing the best tradeoff between “hard” and “soft”, between “strict” and “not strict” strategies can be vital to assure good quality dataset thus good model's accuracy. For the labeling process we used a labeling tool program VIA that is publicly accessible.

Reeds SDD dataset consists of 7 separate datasets for each type of defect. Approximate size of each dataset is within a range of 1-3 thousand of images. All images have the same resolution 1280x960. Some datasets were labeled by “strict” labeling and some by “not strict”. Based on our experiments, it was not possible to achieve an acceptable level of accuracy with lower image

resolution. One should be careful to apply different augmentation functions during training to assure that defectable areas are still recognizable by human eyes.

5. Evaluation

This chapter is about three details regarding the system. Firstly, we propose a way to make error analysis and evaluated the system's accuracy with human level performance. Secondly, training details chapter has some interesting exploration regarding training binary segmentation model, loss function and speed of the model on inference. Thirdly, we described the technology stack we use to build all components of the system.

5.1. Error-analysis

To apply a surface defect detection system to production it is crucial to understand how accurate the system is. On the other hand, it is difficult to evaluate the accuracy of the system with just a one number due to the large number of different defects, the different proportion of these defects and the impossibility in some cases to say unambiguously to which defect class the object belongs. The following evaluation metrics were used to determine the accuracy of the system for each defect class [see Table 1]:

- Recall - what percentage of straws with a defect the system recognizes
- False positive error (FP) - what percentage of straws the system erroneously defines as straws with a defect, even though they have no defect

Table 1

Evaluation metrics for each defect class

Defect class	Recall (%)	FP (%)
Hammered stuff inside	90 % - 96 %	2 % - 5 %
Cut	97 %	< 1 %
Brown spots	97 %	2 % - 3 %
Circle holes	90 % - 95 %	5 % - 10 %
Dark spots	90 % +	1 % - 3 %
Cracks	60 % - 70 %	1 % - 3 %

These metrics were empirically determined during system testing on industrial reeds straws factory. The main reason why for some defect classes metrics are not a single value but interval is ambiguous defect definition thus sometimes it is not possible to classify for sure. As mentioned above, evaluating system accuracy with one value is difficult and requires to customize calculation to business needs.

Taking into consideration the real-time proportion of defects, we calculated the error rate of the system as one value and it is approximately 10 %. So, the system makes a mistake for each one of ten straws and this result is very close to human level.

How to understand whether system accuracy is acceptable for deploying it to production? What type of defect is a system's accuracy bottleneck?

One way to do it is to compare accuracy metrics of each type of defect to human-level accuracy. Firstly, it will give us a picture of how the quality inspection process will change in terms of previous level of accuracy after deploying the system to production. Of course, the system accuracy should be close to human-level not to harm business. Secondly, comparing accuracy to human-level allows us to find types of defects where we are worse than humans thus this is the room for improvements. But how to make this comparison correct and equal?

Humans have a main advantage compared to machines: they analyze objects by eyes not based on images from web-camera like deep neural networks.

Therefore there is a need to evaluate human performance both on images and on real objects as workers do [Table2].

Table 2
System and human-level evaluation metric (only Recall)

Defect class	System (%)	Human-level on images (%)	Human-level on real objects (%)
Hammered stuff inside	90 % - 96 %	90+ %	95%
Cut	97 %	99 %	99%
Brown spots	97 %	95+ %	95+%
Circle holes	90 % - 95 %	90+ %	95+ %
Dark spots	90 % +	95+ %	95+ %
Cracks	60 % - 70 %	65 % - 73 %	85 %-90 %

From the table above it is clear that the system's metric values for all defect classes except one are very close or even equal to human-level performance. To achieve even greater accuracy for these defect classes much more annotated images are required so it is a very time consuming process and does not guarantee that the results will be better as they are now approaching to human level. Instead the Table2 shows that the system's accuracy for defect type class6 is close to “human-level on images” and extremely worse compared to “human-level on objects”. It means that the problem is not in model but in image quality because there is significant difference in both humans’ metric values.

Approach to compare the system's accuracy with human-level performance has two main objectives: firstly it gives an understanding how accurate the system is comparing to humans and secondly what needs to be changed model or dataset to improve results.

5.2. Implementation details

Crucial part of source code for training binary segmentation models was taken from github repository `segmentation_models.pytorch`. The repository provides different segmentation architectures and a variety of backbones including the combination that we used U-Net with MobilenetV2 backbone. Joint loss function of cross-entropy and dice loss was used as a learning function with a fixed *learning rate* value of 0,0001. Proportion of positive and negative samples in the training dataset was not balanced and to fix this problem when it necessary we adjusted the *class weight* parameter in the cross-entropy loss function. To test the accuracy of the model on validation dataset and to choose best model among others IoU score was used. After the best model was chosen we tested it in real-time, evaluated accuracy and collected main mistakes to increase quality of data using a monitoring system. This process was repeated at least 4 times for each class of defects. Having a monitoring system allowed us to iterate much faster over a typical model development loop. Training deep neural networks requires gpu computational resources. At the beginning of research we used Kaggle platform to utilize their gpu resources and to store datasets. Later we switched to locally installed Nvidia GTX 1070. Regarding the speed of the system, it takes 1 second with Nvidia GTX 1070 to run all components of the system including: detection network, ensemble network and rule-based classification on one object. The object is represented by 6 outer surface images and 2 inner. Outer surface images have dimension 256x1248 while inner are 256x248. To process 6 images with binary segmentation model 100 ms is needed thus ~17 ms for one image. We experimented to set up the *batch size* parameter to 6 to make predictions faster but it didn't give a boost. To process 2 images with input size 248x248 takes less than 8 ms. To upload one binary model to RAM the server used approximately 600 mb and the whole ensemble network utilized up to 5gb.

Main programming language to design defect detection system was written in the Python programming language, while the PyTorch framework was used for machine learning tasks like training neural networks and making predictions by them. Flask was used as the primary framework to design monitoring system. Server with an installed inspection system used the Linux operating system and local network to send data on the central computer to provide a monitoring system with real-time data.

6. Discussion and conclusion

This paper explored a defect detection problem on industrial objects and a possible approach to solve it by building a system from scratch based on deep learning segmentation methods. Main components of the system were presented including: decision network, ensemble network, rule-based classification and monitoring system. The idea of using many binary segmentation models instead of one multi-class was described together with its main advantages. The system was evaluated on the newly created Reeds Surface Defect Detection (Reeds SDD) dataset, which is publicly available upon request to the authors, and key metrics to perform reasonable accuracy evaluation were determined. Post-processing techniques were described to reduce false positives mistakes and to improve the system performance.

The experiments on Reeds SDD dataset demonstrated that to achieve an acceptable level of accuracy there is a need to have a quality and diverse dataset rather than just “big”. A method to compare results with human level results can give two main conclusions: whether system accuracy is acceptable for deploying it to production and whether data or models need to be changed to improve results. Accuracy metrics close to human level performance were achieved on datasets with an amount of images within a range of 1.5k - 3k that are relatively small and can be collected in any domain. A few options to annotate images in terms of labeling speed and accuracy were described depending on the type of defect, its geometrical form, how unambiguous the defect definition is, the defect’s business importance and how recognizable defectable areas are. For the reed straw domain, we have proven that the standard web camera Logitech C270 along with specific camera placement are suitable to assure high quality images. Ambiguous defect definition leads to more than one possible option on how to annotate images thus to different datasets and respectively different results. Very likely this problem also exists in other domains. We are thinking about a methodology to label defectable areas not as binary values 1 or 0 but rather as probability values in range of [0,1]. In our opinion, if we design a loss function in a way to penalize it more for mistakes where the probability value is close to 1 then a network will predict non-disputed areas more correctly than now and errors in disputed areas will be reasonable due to defect ambiguous definition. Such an approach requires spending more time on data labeling but it looks more natural and closer to a human's way to solve the problem.

The paper provides essential information about a possible approach to build a visual inspection system together with a monitoring system to control image quality in real-time and perform machine learning iteration cycle faster by real-time error analysis and collection of images on which the system makes mistakes. In our opinion the future work on surface defect detection problem should be focused on providing more publicly available datasets from different industrial domains and modifying methods to work well on as small as possible amount of annotated images.

7. References

- [1] Tabernik, D., Šela, S., Skvarč, J. et al. Segmentation-based deep-learning approach for surface-defect detection. *J Intell Manuf* 31, 759–776 (2020). <https://doi.org/10.1007/s10845-019-01476-x>.
- [2] Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*. 25. doi:10.1145/3065386.
- [3] Ronneberger O., Fischer P., Brox T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N., Hornegger J., Wells W., Frangi A. (eds) *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28.
- [4] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR* abs/1704.04861 (2017).
- [5] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen, MobileNetV2: Inverted Residuals and Linear Bottlenecks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510-4520. arXiv:1801.04381.

- [6] J. Masci, U. Meier, D. Ciresan, J. Schmidhuber and G. Fricout, "Steel defect classification with Max-Pooling Convolutional Neural Networks," The 2012 International Joint Conference on Neural Networks (IJCNN), 2012, pp. 1-6, doi:10.1109/IJCNN.2012.6252468.
- [7] S. Faghieh-Roohi, S. Hajizadeh, A. Núñez, R. Babuska and B. De Schutter, "Deep convolutional neural networks for detection of rail surface defects," 2016 International Joint Conference on Neural Networks (IJCNN), 2016, pp. 2584-2589, doi: 10.1109/IJCNN.2016.7727522.
- [8] Mlynarski, Pawel & Delingette, Hervé & Criminisi, Antonio & Ayache, Nicholas, Deep learning with mixed supervision for brain tumor segmentation. *Journal of Medical Imaging* 6(03):1, doi:10.1117/1.JMI.6.3.034002.
- [9] Yang, Yatao, Runze Yang, Longhui Pan, Junxian Ma, Y. Zhu, Tao Diao and L. Zhang. "A lightweight deep learning algorithm for inspection of laser welding defects on safety vent of power battery." *Comput. Ind.* 123 (2020), doi:103306.
- [10] Yang, Dingming, Yanrong Cui, Zeyu Yu, and Hongqiang Yuan. "Deep Learning Based Steel Pipe Weld Defect Detection." *arXiv preprint arXiv:2104.14907* (2021).
- [11] Caron, Mathilde, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. "Emerging properties in self-supervised vision transformers." *arXiv preprint arXiv:2104.14294* (2021).
- [12] Ting Chen, *Advancing Self-Supervised and Semi-Supervised Learning with SimCLR*, 2020. URL: <https://ai.googleblog.com/2020/04/advancing-self-supervised-and-semi.html>, [Accessed 1 June 2021].
- [13] Božič, Jakob, Domen Tabernik, and Danijel Skočaj. "Mixed supervision for surface-defect detection: from weakly to fully supervised learning." *Computers in Industry* 129 (2021), doi: 10.1016/j.compind.2021.103459.
- [14] Deshpande, Aditya M., Ali A. Minai, and Manish Kumar. "One-Shot Recognition of Manufacturing Defects in Steel Surfaces." *Procedia Manufacturing* 48 (2020): 1064-1071, doi: 10.1016/j.promfg.2020.05.146.
- [15] Andrei-Alexandru Tulbure, Adrian-Alexandru Tulbure, Eva-Henrietta Dulf, A review on modern defect detection models using DCNNs – Deep convolutional neural networks. *Journal of Advanced Research*, 2021, ISSN 2090-1232, <https://doi.org/10.1016/j.jare.2021.03.015>.
- [16] Karim Tout. *Automatic Vision System for Surface Inspection and Monitoring : Application to Wheel Inspection. Performance [cs.PF]*. Université de Technologie de Troyes, 2018. English. ffNNT : 2018TROY0008ff. fftel-02974251.
- [17] Landing.ai platform, 2021. URL: <https://landing.ai/platform/>, [Accessed: 1 March 2021].
- [18] H. Kuang, Y. Ding, R. Li and X. Liu, "Defect detection of bamboo strips based on LBP and GLCM features by using SVM classifier," 2018 Chinese Control And Decision Conference (CCDC), 2018, pp. 3341-3345, doi:10.1109/CCDC.2018.8407701.
- [19] Q. Xiansheng, H. Feng, L. Qiong and S. Xin, "Online defect inspection algorithm of bamboo strip based on computer vision," 2009 IEEE International Conference on Industrial Technology, 2009, pp. 1-5, doi:10.1109/ICIT.2009.4939598.
- [20] Giordano D., Kavasidis I., Palazzo S., Spampinato C. (2015) Rejecting False Positives in Video Object Segmentation. In: Azzopardi G., Petkov N. (eds) *Computer Analysis of Images and Patterns. CAIP 2015. Lecture Notes in Computer Science*, vol 9256. Springer, Cham. https://doi.org/10.1007/978-3-319-23192-1_9.
- [21] Fan, Fenglei, Jinjun Xiong and Ge Wang. "On Interpretability of Artificial Neural Networks." *ArXiv abs/2001.02522* (2020): n. pag.
- [22] A Chat with Andrew on MLOps: From Model-centric to Data-centric AI, video, 2021. URL: <https://www.youtube.com/watch?v=06-AZXmwHjo>, [Accessed: 31 May 2021].