Automated Defect Detection From Ultrasonic Images Using Deep Learning

Duje Medak, Luka Posilović, Marko Subašić, *Member, IEEE*, Marko Budimir, *Member, IEEE*, and Sven Lončarić, *Member, IEEE*

***Abstract*— Nondestructive evaluation (NDE) is a set of techniques used for material inspection and defect detection without causing damage to the inspected component. One of the commonly used nondestructive techniques is called ultrasonic inspection. The acquisition of ultrasonic data was mostly automated in recent years, but the analysis of the collected data is still performed manually. This process is thus very expensive, inconsistent, and prone to human errors. An automated system would significantly increase the efficiency of analysis, but the methods presented so far fail to generalize well on new cases and are not used in real-life inspection. Many of the similar data analysis problems were recently tackled by deep learning methods. This approach outperforms classical methods but requires lots of training data, which is difficult to obtain in the NDE domain. In this work, we train a deep learning architecture EfficientDet to automatically detect defects from ultrasonic images. We showed how some of the hyperparameters can be tweaked in order to improve the detection of defects with extreme aspect ratios that are common in ultrasonic images. The proposed object detector was trained on the largest dataset of ultrasonic images that was so far seen in the literature. In order to collect the dataset, six steel blocks containing 68 defects were scanned with a phased-array probe. More than 4000 VC-B-scans were acquired and used for training and evaluation of EfficientDet. The proposed model achieved 89.6% of mean average precision (mAP) during fivefold cross validation, which is a significant improvement compared to some similar methods that were previously used for this task. A detailed performance overview for each of the folds revealed that EfficientDet-D0 successfully detects all of the defects present in the inspected material.**

***Index Terms*— Automated defect detection, deep learning, flaw detection, ultrasonic image analysis, ultrasonic testing (UT).**

I. Introduction

NONDESTRUCTIVE evaluation (NDE) is a set of techniques used for material evaluation and defect detection in industry and science [1]. These methods do not damage the inspected material, which makes them perfect for continuous monitoring of critical components of some systems. NDE methods are used in aeronautics, oil and gas industry, various power plants, and other industries where it is crucial to detect material flaws in time in order to prevent further damages and disasters. A variety of NDE methods are used: ultrasonic, eddy current, thermography, and X-radiography, to name a few. Some of the advantages when using ultrasonic testing (UT) include simple usage, precise extraction of the defect location [2], and the ability to evaluate the structure of alloys of components with different acoustic properties [3]. UT employs a diverse set of methods based on the generation and detection of mechanical vibrations or waves within test objects [4]. One of the commonly used types of probes is called a phased-array probe. A phased-array probe is a multichannel ultrasonic system, which uses the principle of a time-delayed triggering of the transmitting transducer elements, combined with a time corrected receiving of detected signals [5]. Using a phased-array probe increases the reliability of inspections since the material is inspected from various angles. During an inspection, a probe is moved along the surface of the inspected component. At each position, the probe transmits and receives ultrasound energy. The amount of received energy is usually shown as a function of time in the representation called A-scan. Multiple A-scans are obtained when the probe is moved along one axis. The sequence of A-scans can then be visualized as an image called B-scan. Each column from the B-scan is obtained from the A-scan by converting the amplitude at a specific point in time into pixel intensity. The dimension of the inspected component and the resolution of the inspection determine the width of the B-scan. It is common to see B-scans that were created from several hundreds of A-scans. Other representations of UT data, such as volume corrected B-scans (VC-B-scans), C-scans, and D-scans, are also often used during the analysis. Some of the mentioned representations are shown in Fig. 1.

The acquisition of the UT data was mostly automated in recent years, but the analysis of the acquired data is still performed manually by trained experts. The amount of data that needs to be analyzed is immense, especially when a phased-array probe is used. The success of the analysis depends solely on the analyzer's knowledge and experience making this process prone to errors. Many efforts were made in order to develop methods that could assist the analyzers in the defect detection process.

The most popular approach for the automated analysis of ultrasonic data in NDE is based on the wavelet transform

Manuscript received January 20, 2021; accepted May 17, 2021. Date of publication May 19, 2021; date of current version September 27, 2021. This work was supported by the European Union through the European Regional Development Fund under Grant (Smart UTX) KK.01.2.1.01.0151. *(Corresponding author: Duje Medak.)*

Duje Medak, Luka Posilović, Marko Subašić, and Sven Lončarić are with the Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia (e-mail: duje.medak@fer.hr).

Marko Budimir is with the INETEC Ltd., 10000 Zagreb, Croatia.

Digital Object Identifier 10.1109/TUFFC.2021.3081750

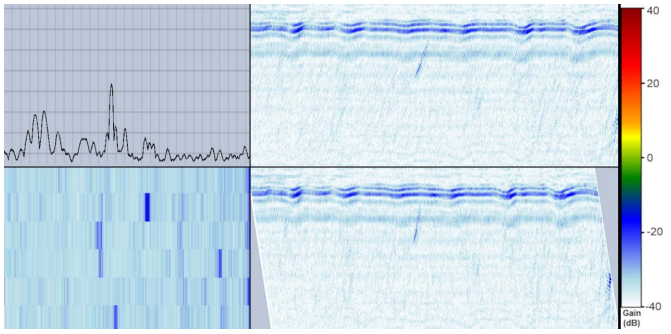

Fig. 1. Examples of different ultrasonic data representations. Top left: A-scan. Beneath it is the C-scan representation. Right: B-scan (top) and volume corrected B-scan (bottom).

of A-scans. Coefficients from the transformation can then be used as an input to some classifiers, such as artificial neural networks (ANNs) [6], [7] or support vector machines (SVMs) [8]–[11]. This approach works well when the dataset is limited since the feature extraction is predefined and the data samples are used solely for classifier training. Different kinds of transformations, such as Fourier transformation or Cosine transformation, can also be used in the feature extraction step as shown in [12]. Usage of a special type of neural network called convolutional neural network (CNN) is becoming more popular in recent years for the analysis of sequences and grid-like representations of the data. Some works [13], [14] showed that methods based on CNN achieve good results when applied for UT data analysis. While many authors achieved good results, datasets that were used for evaluation contained only a few thousand or even a few hundreds of A-scans. When an inspection is performed in a real-life situation, millions of A-scans are usually acquired. A small dataset containing only a fraction of that amount can hardly capture all the possible appearance variations of the signal. The main drawback of A-scan analysis is the lack of context from the surrounding area. Distinguishing between geometry, noise, and defect signals would be considerably easier if the information from the surrounding A-scans would be available.

This problem is solved if B-scans are used for the automated analysis. In this case, the spatial information from the surrounding area is available. This extra information can be used to improve defect detection while reducing the number of false positives (FPs). The wavelet transform that was commonly used for A-scan analysis also proved to be a useful tool when dealing with images. Cygan *et al.* [15] first denoised images using the wavelet transform and then performed defect detection using the Radon transform. In [16], the wavelet transform was used for feature extraction. The authors also tried Gabor filters but determined that the wavelet transform achieves better results. The authors of that work used Fuzzy C-Mean clustering to classify extracted features. With the development of deep learning models, new approaches for the image analysis of the UT data based on CNNs appeared. Ye *et al.* [2] compared a CNN with the traditional approaches that use handcrafted descriptors in combination with SVM. The authors demonstrated that the CNN-based approach yields superior results. Some recent works [17]–[19] also showed that CNN architectures can successfully be applied to analyze the UT data. Virkkunen *et al.* [17] showed that a custom CNN can be trained on artificially generated data. The performance of this approach was compared with the human expert's performance of detection and the authors concluded that automated analysis using the deep learning approach works better and has a higher probability of detection (POD). However, this approach was not tested on a real (nongenerated) dataset of B-scans. In [18], it was demonstrated how the data needed for the training of a deep learning architecture can be simulated. The authors used simulated data to train a network for crack characterization. The proposed deep learning approach was compared to the 6-dB drop method. The deep learning model was able to size 97% of the tested defects of lengths 1–5 mm within ± 1 mm, while the 6 dB method could only size 48% of the defects. In [19], real B-scan data were used to train popular object detectors. To deal with a limited amount of data, the authors used the pretrained architectures YOLOv3 [20] and SSD [21] and performed heavy data augmentation during the training. Even though the achieved results were good, the amount of testing data was very small (only 98 images).

In this work, we perform defect detection with EfficientDet architecture, a state-of-the-art object detection algorithm that was not used for this problem before. EfficientDet belongs to the one-stage family of detectors, meaning that the objects are searched in the predefined rectangles called anchors or default boxes [21]. Anchors are rough guesses about the objects' dimensions and positions in the image. The shapes and positions of the anchors are determined from the hyperparameters provided during the training. We propose a novel procedure for calculating anchors' hyperparameters values in order to improve the detection of defects with extreme aspect ratios that are common in UT images. To the best of our knowledge, detection algorithms with the ability of defect localization from UT B-scan were only previously shown in [19]. We compared EfficientDet with the best performing method from that work, YOLOv3. In addition, we also made a comparison with the RetinaNet, an improved version of the other method used in that work, SSD. We showed that some previous works used CNNs for defect detection, but the approach proposed in this work has the following merits.

1) We are the first to employ a state-of-the-art object detector EfficientDet on this task. We proposed a novel procedure for calculating anchors' hyperparameters and demonstrated that using the calculated values improves the model's average precision by a significant amount.
2) We used the largest dataset of real UT B-scans for training and evaluation that was used so far in the literature (over 4000 images). The collected database displays 68 unique defects that were created using various methods. This ensures that the obtained results represent a realistic performance of the proposed defect detection method.
3) We divide our dataset into five disjunctive subsets (folds) and perform fivefold cross validation [22]. We then conduct a detailed analysis of model performance for each of the folds. This is used to prove that the proposed architecture is reliable enough to be used for automated

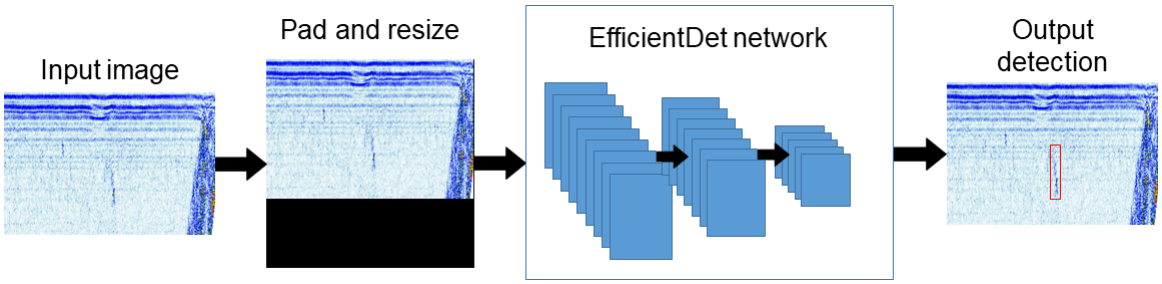

Fig. 2. Illustration of the proposed approach. The input image is first padded to get a squared image and then resized to the network's input size. The preprocessed image is fed to trained EfficientDet [23] architecture. Finally, nonmaximum suppression and confidence thresholding are applied to the model's output to ensure that only relevant detections are kept.

defect detection. Performing such rigorous testing is the most thorough evaluation that was so far done to test the performance of a deep learning object detector for defect detection from ultrasonic images.

II. Materials and Methods

Fig. 2 shows a high-level illustration of the proposed method. A VC-B-scan is padded to get an image of equal width and height. The image is resized to 512 × 512 pixels and fed into the EfficientDet object detection algorithm. The output of the network is a list of bounding boxes and associated confidences. Nonmaximum suppression is performed to ensure that duplicate boxes are removed. Finally, by applying the confidence threshold, we only keep the boxes with higher probabilities.

A. Used Deep Learning Architecture

Deep learning object detectors can be divided into two categories: one-stage detectors, such as YOLOv1 [24], SSD [21], YOLOv2 [25], RetinaNet [26], and YOLOv3 [20], and two-stage detectors, such as R-CNN [27], Fast-RCNN [28], and Faster-RCNN [29]. One-stage detectors search for the object's presence at predefined positions. This is usually implemented as a dense grid of rectangular-shaped areas where the model decides for each area whether it contains some particular object or not. Two-stage detectors first run a region proposal algorithm and then classify only the proposed areas. The number of areas that need to be classified is decreased, but the overall complexity is increased because an extra step for region proposal is needed. Two-stage detectors are usually slower but more accurate compared to one-stage detectors, but this accuracy gap was recently reduced. Reliability of defect detection should always be the primary criterion for choosing the proper model, but considering the amount of data that needs to be analyzed, it would be beneficial if the used model was fast. A good tradeoff between accuracy and speed is offered by the EfficientDet model [23], which belongs to the one-stage detector family.

This architecture was developed with computational efficiency in mind. The authors created a base model (EfficientDet-D0) that can be scaled up depending on the available resources. The family of EfficientDet models includes a total of eight models (D0–D7). However, having a more complex network does not always lead to an improvement, especially if the objects are simple like it is in the case with defects from the UT data. Experimental results that we presented in Section IV show that the performance of the smaller and faster EfficientDet-D0 model is better than those of EfficientDet-D1 and EfficientDet-D2. Like other one-stage detectors, EfficientDet searches for object presence at predefined areas called anchors (default boxes). This dense grid of rectangles covers a variety of different shapes. Every anchor-based object detector needs a list of aspect ratios and scales as an input to calculate the shape and size of anchors. Object detectors predict the locations of the objects with respect to these anchors. Having proper anchors hyperparameters can speed up model training and improve accuracy. There are several approaches [30]–[32] that can be used to estimate good anchors hyperparameters, but it is often needed to make some assumptions about the objects' shapes so that the calculation would be possible. When working with natural images, most of the researchers simply use default values of aspect ratios and scales for RetinaNet or EfficientDet. These values are shown in Table I. Looking at Fig. 3, one can notice that the aspect ratios of our bounding boxes are much more extreme. We decided to calculate new values using K-means with the Jaccard distance, as proposed in YOLOv2 [25]. Values obtained from this procedure can be used directly for training the YOLOv3 model, but using them to train RetinaNet or EfficientDet is not straightforward. Aspect ratios and scales for RetinaNet and EfficientDet were calculated in the following way.

TABLE I
Comparison Between Commonly Used Default Values of Aspect Ratios and Scales With the Values We Used in This Work

	default	ours
aspect ratios	[0.5, 1, 2]	[3.77 5.73 6.94 10.05 13.56]
scales	$[2^{0/3}, 2^{1/3}, 2^{2/3}]$	[0.88 1.34 1.46] *
*scales before merging [1.34, 1.49, 1.43, 0.88, 0.89]		

TABLE II
DATASET OVERVIEW

	number of defects	number of images	number of annotations
fold 1	14	1006	1317
fold 2	15	915	1439
fold 3	16	872	1437
fold 4	12	298	1316
fold 5	11	1083	1128
total	68	4174	6637

1) First, we used K-means with the Jaccard distance to calculate five default shapes. This procedure calculates which shapes of bounding boxes will on average fit the best to the samples (after they are resized to the input image size) from the dataset. The obtained widths and heights are expressed as absolute values (in pixels), so they need to be converted into a list of aspect ratios and scales.
2) Aspect ratios can simply be calculated by dividing the height of each shape by its width.
3) Determining scale requires knowledge about the detection process of EfficientDet. EfficientDet performs object detection on five different scales. In order for that to be possible, five feature maps (called P3–P7 in the original publication [23]) of different resolutions are used. Each of these feature maps has an assigned template anchor size. Sizes of template anchors range from 32×32 for the P3 feature map to 512×512 for the P7 feature map. We calculated the scales by finding which of the template anchors is the most similar to our shape. As a similarity measure, we used the absolute difference between template anchor size and the bigger side of our calculated shape. Once we know which template anchor is the most similar, we can calculate the scale factor by dividing the maximum size of our shape by the anchor size. The described procedure for scales calculation can be written mathematically as shown in 1a and 1b.
4) To decrease the total number of anchors, we merged the values of scales that were similar.

The final values are shown in Table I. It can be seen that the calculated values greatly differ from the commonly used default values. In Section IV, we proved that using these values improves the performance of the EffcientDet model by a significant amount

$$s_i = \frac{\max(\text{width}(BB_i), \text{height}(BB_i))}{BTA_i} \tag{1a}$$

$$BTA_i = \arg\min_{T_j} |\max(\text{width}(BB_i), \text{height}(BB_i)) - T_j)|$$
$$T_j \in \{32, 64, 128, 256, 512\} \tag{1b}$$

where

BB_i ith shape calculated using K-means.
T_j template anchor size.

B. Dataset

For the development and evaluation of the proposed method, we used an in-house dataset. Creating test specimens with artificial defects inside is a costly process. Companies have to invest a fair amount of money for the acquisition of those blocks as well as the equipment that is needed to perform UT. It is only logical that they would like to keep that data private in order to maintain their competitiveness compared to other companies. Besides the dataset provided in [17], which was artificially generated, there are no publicly available datasets that could be used for the development and evaluation of methods for UT analysis. Having a large dataset ensures the credibility of the obtained results. It also makes training of a large deep learning model (with tens of millions of parameters) possible. The data used in this work was obtained by scanning six stainless steel blocks. Blocks contained between six and 34 defects. Defects were artificially created using various methods leading to different types of defects, such as side-drilled holes, flat bottom holes, thermal fatigue cracks, mechanical fatigue cracks, electric discharge machined notches, solidification cracks, and incomplete penetration of the weld. The scanning was done using the INETEC Dolphin scanner in combination with INETEC dual-phased-array probe with 2×16 elements, element dimensions 1.45 mm (pitch) $\times$ 1.3 mm (width), longitudinal wave, the central frequency of 2.25 MHz, and the frequency average bandwidth $\geq$70% at $-$6-dB gain. The collected data include only the shallow parts of the blocks (up to 200 mm). After all of the data were collected, multiple human experts analyzed it and determined the positions of the defects. The location of each defect was annotated by a bounding box. Even though all of the positions of the defects were known from the blocks' blueprints, manual annotation is needed to ensure that only the visible defects will be labeled. One of the most important questions about the experimental setup is how to split the data. Using a hold-out method is the most common way to test the performance of the model, but it is not the most reliable. Even if unique images are contained in the test set, they often represent some defects that already appeared in the train set (but on the image from a different angle for example). Having similar images in the train and test subset would lead to an unrealistically good model's performance. To provide a fair evaluation, we decided to split all of our data into five subsets (folds) where each fold contains unique defects, as shown in Table II. This ensures that all of the images used for testing as well as the defects that are displayed in those images are unique and will not be used for training. Each fold was made to contain approximately 20% of all available annotations. The width of most of the original images is a lot larger than their height. This can cause problems since some

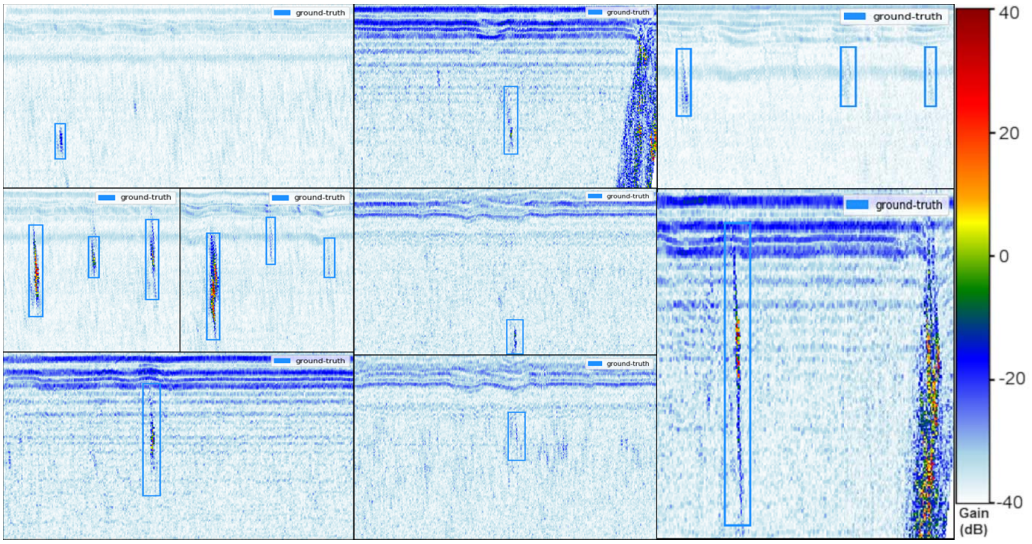

Fig. 3. Examples of the used VC-B-scans with ground-truth labels.

defects would not be easily seen after padding and resizing the image. To avoid this, we split some of the images into multiple patches. The resulting images have an aspect ratio closer to 1, so the amount of needed padding is minimized. The height of the images varies between 200 and 375 pixels, while their width varies between 300 and 400 pixels. Defects displayed in a B-scan usually appear slanted, so the bounding boxes do not fit perfectly around them. Since each of the acquired A-scans was taken at some angle and projected into exactly one image column, B-scans do not display the internal structure of the material realistically. If VC-B-scan is used, each A-scan is transferred onto the image at the same angle that the ultrasonic waves were propagated through the material. This skews VC-B-scans as shown in Fig. 1, but the orientations of the displayed defects are more similar to the physical orientation of the defects inside of the material. Even though image representation of UT data is naturally in the grayscale colormap, B-scans are often colored for easier manual inspection. We also used pseudo-colored images that were exported using the INETEC SignyOne ultrasound data acquisition and analysis software. A few example images from the dataset are shown in Fig. 3.

III. Experimental Setup

A. Model Training

We trained three representatives from the EfficientDet family (EfficientDet-D0, EfficientDet-D1, and EfficientDet-D2). We tried two approaches for weight initialization as follows:

1) randomly initialized weights;
2) weights from a model pretrained on COCO [33] dataset.

Using cross validation to evaluate the performance of the model means that every model is trained five times. Each time a different fold is left out as a test set, while the four remaining folds are used to train the model. In addition, we also left out 15% of the training subset for validation. The validation subset was used to decrease the learning rate on plateaus and early stopping of the training. The training subset was augmented during the training, which is commonly done to improve the generalization of the model and increase precision. Following transformations were used: horizontal flip, random crop, translation, and visual effects (contrast, brightness, and color enhancement). We trained EfficientDet-D0 with batch size 8 and 500 steps per epoch. EfficientDet-D1 and EfficientDet-D2 were trained with batch size 4 and 1000 steps per epoch. All of the models were trained using the Adam optimizer with an initial learning rate of $1e^{-3}$. The training was performed on a single NVIDIA RTX 2080 Ti GPU on a machine with AMD Threadripper 1920X and 128 GB of RAM. We compared the performance of the EfficientDet model with two popular object detectors YOLOv3 and RetinaNet (with ResNet [34] backbone). The same hyperparameters (optimizer, batch size, number of steps, and callback hyperparameters) as the ones used for the EfficientDet-D0 model were used when training these models. To have a fair comparison, these models were also pretrained on the COCO dataset. We calculated anchors for YOLO using K-means as described in [25]. For RetinaNet, we used the same values as for EfficientDet (described in Section II).

B. Evaluation Metric

The mean average precision (mAP) metric as given in the later versions of PASCAL VOC (2010–2012) [35] was used as an evaluation metric. This is a common metric to compare the performance of object detectors. The value of mAP is determined by the area under the precision–recall curve. In order to calculate the curve, the number of true positives (TPs), FPs, and false negatives needs to be calculated first. Each output detection of the model contains the coordinates of the bounding box and a probability of that box containing a defect. To determine which output predictions are TPs, the intersection over the union between the predicted bounding boxes and the ground-truth labels needs to be calculated. IOU

TABLE III
MEAN AVERAGE PRECISION FOR DIFFERENT CONFIGURATIONS OF EFFICIENTDET-D0 MODEL

model	mAP
EfficientDet-D0 (512x512), default anchors, pretrained on coco	0.837
EfficientDet-D0 (512x512), custom anchors, pretrained on coco	**0.896**
EfficientDet-D0 (512x512), custom anchors, random weights initialization	0.875
EfficientDet-D0 (384x384), custom anchors, pretrained on coco	0.881

TABLE IV
MEAN AVERAGE PRECISION FOR EACH OF THE FOLDS. BOLD TEXT INDICATES THE BEST PERFORMANCE FOR THAT FOLD

model	fold1	fold2	fold3	fold4	fold5	average
YOLOv3 (416x416)	0.846	0.787	0.756	0.901	0.742	0.806
RetinaNet (ResNet50)	0.856	**0.833**	0.826	0.924	0.832	0.854
RetinaNet (ResNet101)	0.829	0.831	0.818	0.920	0.794	0.838
RetinaNet (ResNet152)	0.872	0.821	0.830	0.901	0.850	0.855
EfficientDet-D0	**0.937**	0.829	**0.879**	**0.943**	0.893	**0.896**
EfficientDet-D1	0.927	0.793	0.869	0.917	**0.901**	0.881
EfficientDet-D2	0.936	0.780	0.826	0.920	0.895	0.871

is calculated as shown in the following equation:

$$\text{iou} = \frac{\text{area}(BB_{\text{pred}} \cap BB_{\text{gt}})}{\text{area}(BB_{\text{pred}} \cup BB_{\text{gt}})} \tag{2}$$

where

BB_{pred} predicted bounding box.
BB_{gt} ground-truth bounding box.

Predicted bounding boxes that have an intersection over union (IOU) with some ground-truth label higher than 0.5 are considered TPs. Predicted bounding boxes that do not have matching ground-truth boxes are considered FPs, and the ground-truth boxes that were not matched with any predicted bounding box are considered false negatives (FN). The numbers of TPs, FPs, and false negatives are then used to calculate precision and recall as shown in 3a and 3b

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3a}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3b}$$

where

TP number of true positive predictions.
FN number of false negative predictions.
FP number of false positive predictions.

By changing the confidence threshold, we can get precision values for different recall values and plot the precision–recall curve. The area under that curve is used to compare the performances of different models.

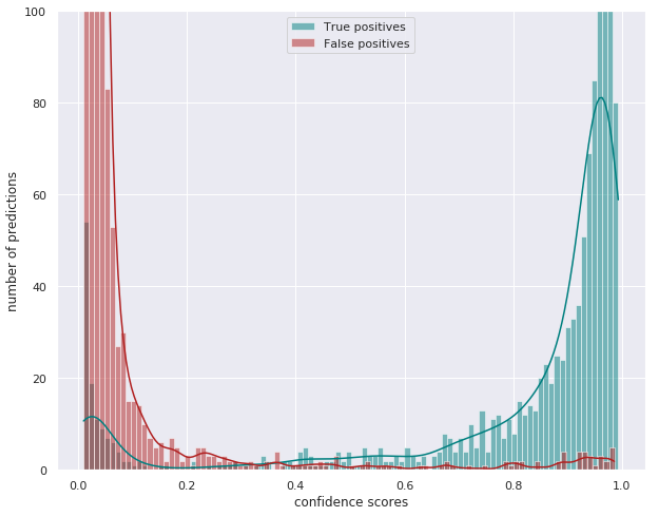

Fig. 4. Histogram of EfficientDet-D0 confidence scores and the density estimate lines using a Gaussian kernel. Nonmaximum suppression with a threshold of 0.3 was done before plotting.

IV. RESULTS AND DISCUSSION

In order to determine the best configuration for EfficientDet, we run a few experiments with different setups. Some of our findings can be seen in Table III. We showed that calculating aspect ratios and scales as proposed in this work improves the mAP by almost 6%. We also showed that using a smaller input image resolution (384 × 384 pixel) decreases the model's performance even though most images from our dataset are smaller than 384 × 384 pixel. We think that this has to do with the architecture of EfficientNet that downsamples the input image in an early stage, which leads to information loss. Comparison of EfficientDet with YOLOv3 and RetinaNet is shown in Table IV. We experimentally determined that RetinaNet performs better if the input images are only padded, so we did not resize the images as we did for EfficientDet and YOLOv3. Even the smallest baseline model EfficientDet-D0, which performs worse than RetinaNet on common benchmark datasets such as COCO [33] and PASCAL [35], outperformed the best version of RetinaNet by

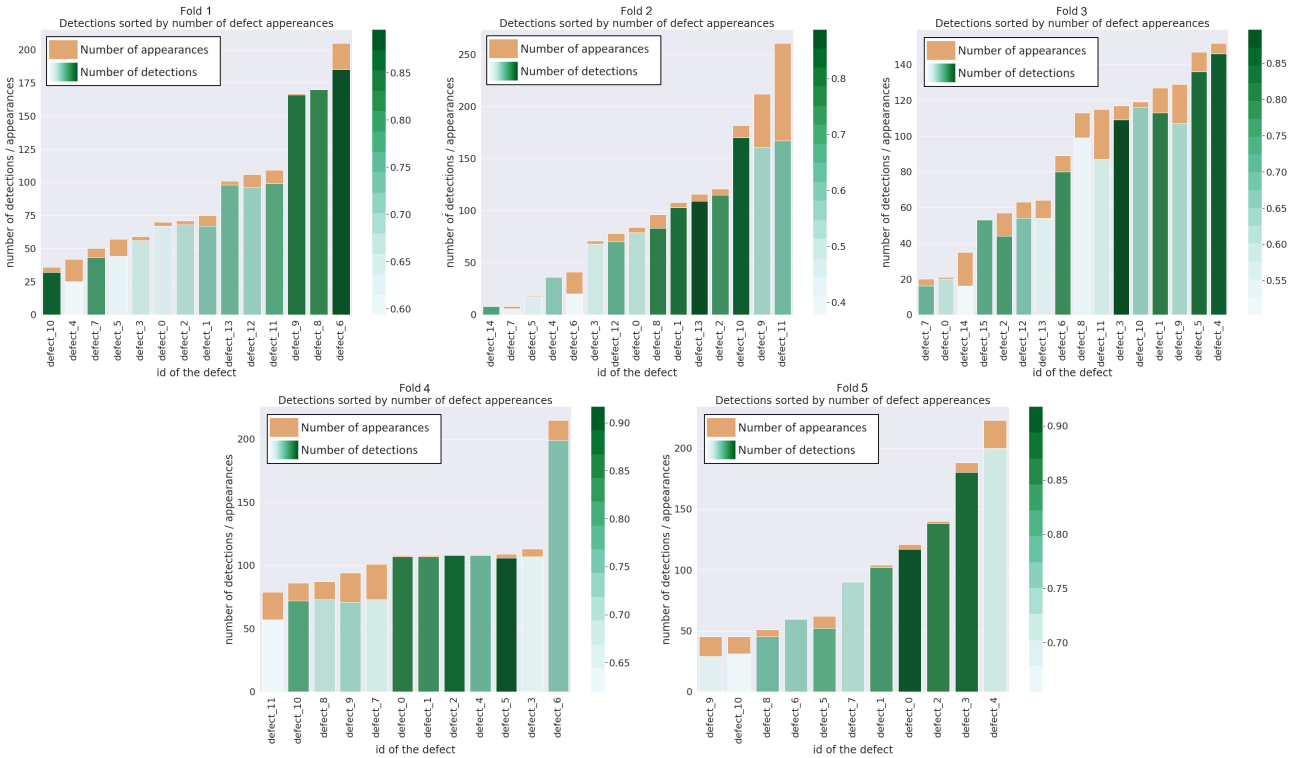

Fig. 5. Detection results obtained with EfficientDet-D0 with confidence threshold 0.3. Each of the bars shows how many appearances of the defect did the model detect. The color hue of the bar represents the maximum confidence from all of the detection for that defect. Best viewed in color.

more than 4%. EfficientDet-D1 and EfficientDet-D2 perform worse than the baseline EfficientDet-D0 model. Besides being the most accurate from all of the tested models, the smallest architecture EfficientDet-D0 is also the fastest. The average inference time of the used EfficientDet-D0 model on NVIDIA RTX 2080 Ti GPU is 26 ms. We determined from the obtained results that the EfficientDet-D0 is the most suitable choice for automated defect detection. We did not make any modifications to the EfficientDet-D0 architecture that is specific to the used hardware configuration, so we believe that this architecture can also be applied for similar tasks in the other NDE technologies. Due to the small number of parameters that have to be trained, this architecture is also very convenient for situations in which the number of available images is limited.

When the inference on the new data is performed, a confidence threshold needs to be set to limit the number of FPs. In Fig. 4, we showed how the confidences of the chosen EfficientDet-D0 model relate to the number of FPs and TPs. The Gaussian kernel density estimate lines are also shown in the figure. This plot was calculated for one specific fold, but similar distributions are obtained for other folds as well. Each prediction that has an IOU overlap with the ground-truth annotation greater than 0.5 was considered TPs. This definition causes a small increase of the TPs for the small confidence threshold values because several predictions are matched with a ground-truth label. The confidence threshold is usually set to 0.5, but looking at Fig. 4, we noticed that we could set the threshold to a lower value without significantly increasing the number of FPs. We set it to 0.3 because it is roughly the value for which the ratio of TPs and FPs becomes greater than one. Even if the confidence threshold of 0.3 is used, some of the TPs will be removed, so it is important to test whether the proposed model is able to detect all of the defects. To determine this, we performed a detailed analysis for each of the test folds. In Fig. 5, we showed exactly how many times some defect from the test fold can be seen (how many annotations of the same defect we have). We also showed the number of detections when using the EfficientDet-D0 model with a threshold of 0.3. The proposed model successfully detected 87.5% of the annotations. However, it is important to note that all of the defects have at least one detection, meaning that none of the defects will pass undetected. In fact, the EfficientDet-D0 detected on average 85.7% of appearances of some defect. Undetected annotations are usually some borderline cases for which the defect's signal becomes too weak and even the human operators would not annotate it if they did not confirm their decision by looking at the block's blueprints. The percentage of FPs when using a threshold of 0.3 is 16.7%. We think that this could be decreased by converting the predicted bounding boxes into real-life coordinates and performing some postprocessing/filtering. To compare the results with the previous state of the art, we performed the same detailed analysis for the YOLOv3 model. We set the object threshold to 0.3 even though this value is too low for YOLOv3 architecture and causes a large number of FPs (almost 50%). Even with such a low threshold, this model was not able to detect all of the defects. There were two defects from the fold3 (defect 7 and defect 2) and two defects from the fold5 (defects 9 and 10) for which the YOLOv3 did not manage to detect any annotations. Finally, we also tested the RetinaNet model with

a ResNet152 feature extractor. We again used a confidence threshold of 0.3, which resulted in 19% of false-positive predictions. This model was also unable to detect all of the defects. Defect 4 from the fold1 and defects 7 and 6 from the fold2 did not have any detections. The presented results show that using the proposed EfficientDet-D0 model not only improves the mAP but also enables the detection of all of the defects in the material.

V. Conclusion

Manual analysis of the UT data is a time-consuming and laborious process prone to human error. In order to automate this process and help human experts with the analysis, a reliable method must be developed. In this work, we demonstrated that the EfficientDet-D0 architecture can successfully be adapted to detect defects from images obtained with a phased-array probe. We proposed a novel procedure for calculating the anchors' hyperparameters and showed that this increases the performance of the network significantly. The proposed EfficientDet-D0 model achieved an mAP of 89.6%, which is an improvement of 9% compared to the previous state-of-the-art architecture YOLOv3. While the presented results prove that the EfficientDet-D0 successfully detects all of the defects from the material, it would be useful to compare its performance to the performance of human inspectors. This can be done by performing a POD study, but such study goes beyond the scope of this work. If proven to be equally reliable as the human inspectors, methods similar to the one presented in this work could soon be used in real-life situations in order to assist the human operators with the analysis of the UT data.

References

[1] L. Cartz, *Nondestructive Testing: Radiography, Ultrasonics, Liquid Penetrant, Magnetic Particle, Eddy Current*. Materials Park, OH, USA: ASM International, 1995. [Online]. Available: https://books.google.hr/books?id=0spRAAAAMAAJ

[2] J. Ye, S. Ito, and N. Toyama, "Computerized ultrasonic imaging inspection: From shallow to deep learning," *Sensors*, vol. 18, no. 11, p. 3820, Nov. 2018, doi: 10.3390/s18113820.

[3] S. Davì *et al.*, "Correction of B-scan distortion for optimum ultrasonic imaging of backwalls with complex geometries," *Insight, J. Brit. Inst. Non-Destructive Test.*, vol. 62, no. 4, pp. 184–191, Apr. 2020.

[4] D. Forsyth, "Nondestructive testing of corrosion in the aerospace industry," in *Corrosion Control in the Aerospace Industry* (Woodhead Publishing Series in Metals and Surface Engineering), S. Benavides, Ed. Cambridge, U.K.: Woodhead Publishing, 2009, ch. 5, pp. 111–130. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9781845693459500050

[5] L. von Bernus, A. Bulavinov, D. Joneit, M. Kröning, M. Dalichov, and K. M. Reddy, "Sampling phased array: A new technique for signal processing and ultrasonic imaging," in *Proc. Eur. Conf. Non-Destructive Test. (ECNDT)*, Berlin, Germany, 2006. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.218.3412&rep=rep1&type=pdf

[6] F. Bettayeb, T. Rachedi, and H. Benbartaoui, "An improved automated ultrasonic nde system by wavelet and neuron networks," *Ultrasonics*, vol. 42, no. 1, pp. 853–858, 2004, doi: 10.1016/j.ultras.2004.01.064.

[7] S. Sambath, P. Nagaraj, and N. Selvakumar, "Automatic defect classification in ultrasonic NDT using artificial intelligence," *J. Nondestruct. Eval.*, vol. 30, no. 1, pp. 20–28, Mar. 2011, doi: 10.1007/s10921-010-0086-0.

[8] M. Khelil, M. Boudraa, A. Kechida, and R. Drai, "Classification of defects by the SVM method and the principal component analysis (PCA)," *Int. J. Electr. Comput. Eng.*, vol. 1, no. 9, pp. 1–6, 2007, doi: 10.5281/zenodo.1060751.

[9] V. Matz, M. Kreidl, and R. Smid, "Classification of ultrasonic signals," *Int. J. Mater. Product Technol.*, vol. 27, no. 3/4, pp. 145–155, 2006, doi: 10.1504/IJMPT.2006.011267.

[10] A. Al-Ataby, W. Al-Nuaimy, C. R. Brett, and O. Zahran, "Automatic detection and classification of weld flaws in TOFD data using wavelet transform and support vector machines," *Insight, Non-Destructive Test. Condition Monitor.*, vol. 52, no. 11, pp. 597–602, Nov. 2010, doi: 10.1784/insi.2010.52.11.597.

[11] Y. Chen, H.-W. Ma, and G.-M. Zhang, "A support vector machine approach for classification of welding defects from ultrasonic signals," *Nondestruct. Test. Eval.*, vol. 29, no. 3, pp. 243–254, Jul. 2014, doi: 10.1080/10589759.2014.914210.

[12] F. C. Cruz, E. F. S. Filho, M. C. S. Albuquerque, I. C. Silva, C. T. T. Farias, and L. L. Gouvêa, "Efficient feature selection for neural network based detection of flaws in steel welded joints using ultrasound testing," *Ultrasonics*, vol. 73, pp. 1–8, Jan. 2017, doi: 10.1016/j.ultras.2016.08.017.

[13] M. Meng, Y. J. Chua, E. Wouterson, and C. P. K. Ong, "Ultrasonic signal classification and imaging system for composite materials via deep convolutional neural networks," *Neurocomputing*, vol. 257, pp. 128–135, Sep. 2017, doi: 10.1016/j.neucom.2016.11.066.

[14] N. Munir, H.-J. Kim, J. Park, S.-J. Song, and S.-S. Kang, "Convolutional neural network for ultrasonic weldment flaw classification in noisy conditions," *Ultrasonics*, vol. 94, pp. 74–81, Apr. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0041624X18305754

[15] H. Cygan, L. Girardi, P. Aknin, and P. Simard, "B-scan ultrasonic image analysis for internal rail defect detection," in *Proc. World Congr. Railway Res.*, Oct. 2003, pp. 1–6.

[16] A. Kechida, R. Drai, and A. Guessoum, "Texture analysis for flaw detection in ultrasonic images," *J. Nondestruct. Eval.*, vol. 31, no. 2, pp. 108–116, Jun. 2012, doi: 10.1007/s10921-011-0126-4.

[17] I. Virkkunen, T. Koskinen, O. Jessen-Juhler, and J. Rinta-Aho, "Augmented ultrasonic data for machine learning," *J. Nondestruct. Eval.*, vol. 40, no. 1, pp. 1–11, Mar. 2021.

[18] R. J. Pyle, R. L. T. Bevan, R. R. Hughes, R. K. Rachev, A. A. S. Ali, and P. D. Wilcox, "Deep learning for ultrasonic crack characterization in NDE," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 68, no. 5, pp. 1854–1865, May 2021.

[19] L. Posilović, D. Medak, M. Subašić, T. Petković, M. Budimir, and S. Lončarić, "Flaw detection from ultrasonic images using YOLO and SSD," in *Proc. 11th Int. Symp. Image Signal Process. Anal. (ISPA)*, Sep. 2019, pp. 163–168.

[20] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767

[21] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, vol. 9905, Oct. 2016, pp. 21–37.

[22] T. Hastie, J. Friedman, and R. Tisbshirani, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY, USA: Springer, 2017.

[23] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," 2019, *arXiv:1911.09070*. [Online]. Available: http://arxiv.org/abs/1911.09070

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/CVPR.2016.91

[25] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *CoRR*, vol. abs/1612.08242, pp. 1–9, Dec. 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[26] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, pp. 1–10, Aug. 2017. [Online]. Available: http://arxiv.org/abs/1708.02002

[27] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, pp. 1–21, Nov. 2013. [Online]. Available: http://arxiv.org/abs/1311.2524

[28] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448, doi: 10.1109/ICCV.2015.169.

[29] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2016.2577031

[30] M. Zlocha, Q. Dou, and B. Glocker, "Improving RetinaNet for CT lesion detection with dense masks from weak RECIST labels," 2019, *arXiv:1906.02283*. [Online]. Available: http://arxiv.org/abs/1906.02283

[31] M. Ahmad, M. Abdullah, and D. Han, "Small object detection in aerial imagery using RetinaNet with anchor optimization," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, Jan. 2020, pp. 1–3.

[32] Y. Zhong, J. Wang, J. Peng, and L. Zhang, "Anchor box optimization for object detection," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 1275–1283.
[33] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Computer Vision—ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014, pp. 740–755.
[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, pp. 1–12, Dec. 2015. [Online]. Available: http://arxiv.org/abs/1512.03385
[35] M. Everingham, L. van Gool, C. Williams, J. Winn, and A. Zisserman. (2012). *The PASCAL Object Recognition Database Collection*. Accessed: May 1, 2020. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/

Duje Medak received the M.Sc. degree from the Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia, in 2019, where he is currently pursuing the Ph.D. degree.

He is currently working as a Researcher with Image Processing Group, Department of Electronic Systems and Information Processing, University of Zagreb. His research interests include image processing, image analysis, machine learning, deep learning, and deep learning object detection methods and their application in the NDE domain.

Luka Posilović received the M.Sc. degree from the Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia, in 2019, where he is currently pursuing the Ph.D. degree.

He is also working as a Young Researcher with Image Processing Group, University of Zagreb. His research interests include visual quality control, object detection, and synthetic image generation.

Marko Subašić (Member, IEEE) received the Ph.D. degree from the Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia, in 2007.

Since 1999, he has been working with the Department for Electronic Systems and Information Processing, Faculty of Electrical Engineering and Computing, University of Zagreb, where he is currently working as an Associate Professor. He teaches several courses at the graduate and undergraduate levels. His research interests include image processing and analysis and neural networks, with a particular interest in image segmentation, detection techniques, and deep learning.

Dr. Subašić is a member of the Croatian Center for Computer Vision, the Croatian Society for Biomedical Engineering and Medical Physics, and the Centre of Research Excellence for Data Science and Advanced Cooperative Systems.

Marko Budimir (Member, IEEE) received the M.Sc. degree in physics from the Faculty of Science, University of Zagreb, Zagreb, Croatia, in 2000, and the Ph.D. degree from the École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, in 2006.

From 2006 to 2008, he worked at EPFL. Since 2008, he has been working with the Institute of Nuclear Technology (INETEC). He coordinated many key projects at INETEC. Although he is a key person in a company of industry sector, he is still working close to the field of science.

Sven Lončarić (Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Cincinnati, Cincinnati, OH, USA, in 1994, as a Fulbright Scholar.

He is currently a Full Professor of electrical engineering and computer science with the Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia. With his students and collaborators, he has coauthored more than 200 publications in scientific journals and conferences. He is the Founder of the Center for Computer Vision, University of Zagreb, where he is also the Head of the Image Processing Group. He has served as the Co-Director for the National Center of Research Excellence in Data Science and Cooperative Systems.

Prof. Lončarić is a member of the Croatian Academy of Technical Sciences. He received several awards for his scientific and professional work. He was the Chair of the IEEE Croatia Section.