Detection and Segmentation of Manufacturing Defects with Convolutional Neural Networks and Transfer Learning

Max K. Ferguson[1], Ronay Ak[2], Yung-Tsun Tina Lee[2], and Kincho. H. Law[1]

ABSTRACT

Quality control is a fundamental component of many manufacturing processes, especially those involving casting or welding. However, manual quality control procedures are often time-consuming and error-prone. In order to meet the growing demand for high-quality products, the use of intelligent visual inspection systems is becoming essential in production lines. Recently, Convolutional Neural Networks (CNNs) have shown outstanding performance in both image classification and localization tasks. In this article, a system is proposed for the identification of casting defects in X-ray images, based on the Mask Region-based CNN architecture. The proposed defect detection system simultaneously performs defect detection and segmentation on input images, making it suitable for a range of defect detection tasks. It is shown that training the network to simultaneously perform defect detection and defect instance segmentation, results in a higher defect detection accuracy than training on defect detection alone. Transfer learning is leveraged to reduce the training data demands and increase the prediction accuracy of the trained model. More specifically, the model is first trained with two large openly-available image datasets before finetuning on a relatively small metal casting X-ray dataset. The accuracy of the trained model exceeds state-of-the art performance on the GRIMA database of X-ray images (GDXray) Castings dataset and is fast enough to be used in a production setting. The system also performs well on the GDXray Welds dataset. A number of in-depth studies are conducted to explore how transfer

[1] Stanford University, Civil and Environmental Engineering, Stanford, CA, USA
[2] National Institute of Standards and Technology, Systems Integration Division, Gaithersburg, MD, USA

learning, multi-task learning, and multi-class learning influence the performance of the trained system.

Keywords

Smart Manufacturing, Transfer Learning, Defect Detection, Casting Defect Detection, Weld Defect Detection, Automated Surface Inspection, Convolutional Neural Networks

Introduction

Quality management is a fundamental component of a manufacturing process [1]. To meet growth targets, manufacturers must increase their production rate while maintaining stringent quality control limits. In a recent report, the development of better quality management systems was described as the most important technology advancement for manufacturing business performance [2]. In order to meet the growing demand for high-quality products, the use of intelligent visual inspection systems is becoming essential in production lines.

Processes such as casting and welding can introduce defects in the product which are detrimental to the final product quality [3]. Common casting defects include air holes, foreign-particle inclusions, shrinkage cavities, cracks, wrinkles, and casting fins [4]. If undetected, these casting defects can lead to catastrophic failure of critical mechanical components, such as turbine blades, brake calipers, or vehicle driveshafts. Early detection of these defects can allow faulty products to be identified early in the manufacturing process, leading to time and cost savings [5]. Automated quality control can be used to facilitate consistent and cost-effective inspection. The primary drivers for automated inspection systems include faster inspection rates, higher quality demands, and the need for more quantitative product evaluation that is not hampered by the effects of human fatigue.

Nondestructive examination techniques allow a product to be tested during the manufacturing process without jeopardizing the quality of the product. There are a number of nondestructive examination techniques available for producing two-dimensional and three-dimensional images of an object. Real-time X-ray imaging technology is widely used in defect detection systems in industry, such as on-line weld defect inspection [5]. Ultrasonic inspection and magnetic particle inspection can also be used to measure the size and position of casting defects in cast components [6,7]. X-ray Computed Tomography (CT) can be used to visualize the internal structure of materials. Recent developments in high resolution X-ray computed tomography have made it possible to gain a three-dimensional characterization of porosity [8]. However, automatically identifying casting defects in X-ray images still remains a challenging task in the automated inspection and computer vision domains.

The defect detection process can be framed as either an object detection task or an instance segmentation task. In the object detection approach, the goal is to place a tight-fitting bounding box around each defect in the image. In the image segmentation approach, the problem is essentially one of pixel classification, where the goal is to classify each image pixel as a defect or not. Instance segmentation is a more difficult variant of image segmentation, where each segmented pixel must be assigned to a particular casting defect. A comparison of these computer vision tasks is provided in Figure 1. In general, object detection and instance segmentation are difficult tasks, as each object can cast an infinite number of different 2-D images onto the retina [9]. Additionally, the number of instances in a particular image is unknown and often unbounded. Variations of the object's position, pose, lighting, and background represent additional challenges to this task.

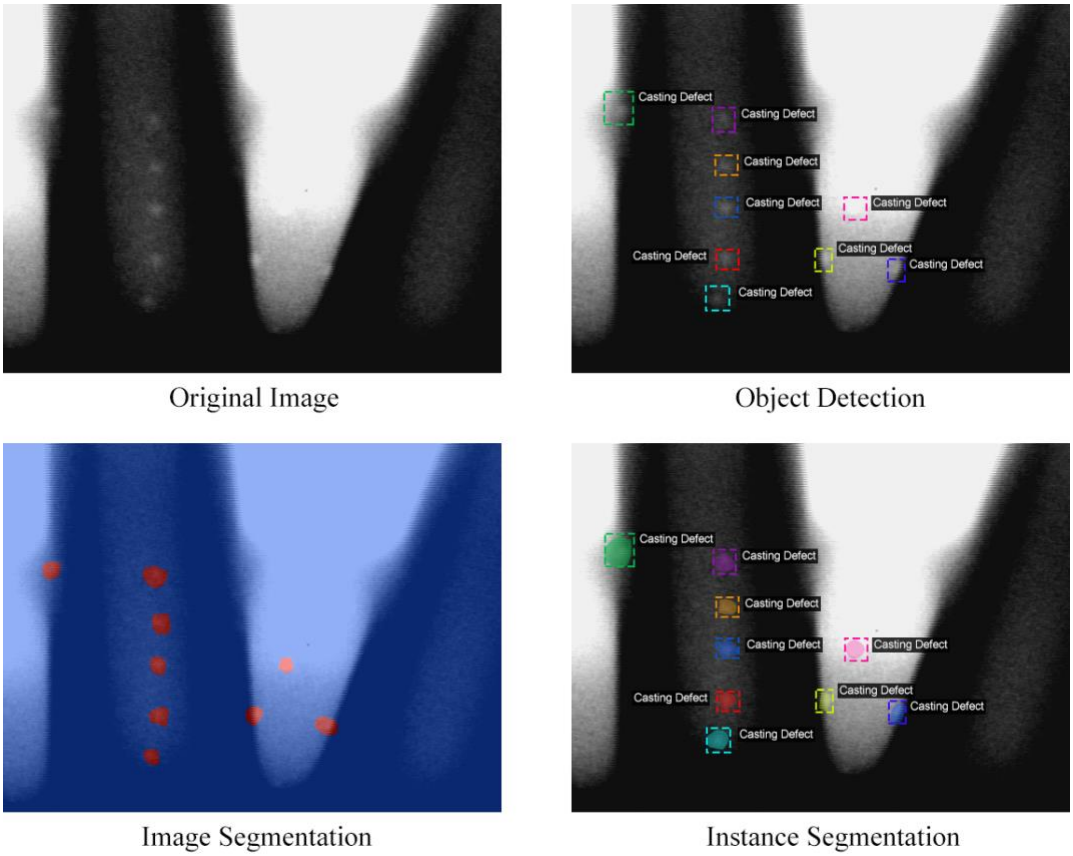


Figure 1. Examples of different computer vision tasks for casting defect detection

Many state-of-the-art object detection systems have been developed using the region-based convolutional neural network (R-CNN) architecture [10]. R-CNN creates bounding boxes, or region proposals, using a process called selective search. At a high level, selective search looks at the image through windows of different sizes and, for each size, tries to group together adjacent pixels by texture, color, or intensity to identify objects. Once the proposals are created, R-CNN warps the region to a standard square size and passes it through a feature extractor. A support vector machine (SVM) classifier is then used to predict what object is present in the image, if any. In more recent object detection architectures, such as region-based fully convolutional networks (R-FCN), each component of the object detection network is replaced by a deep neural network [11].

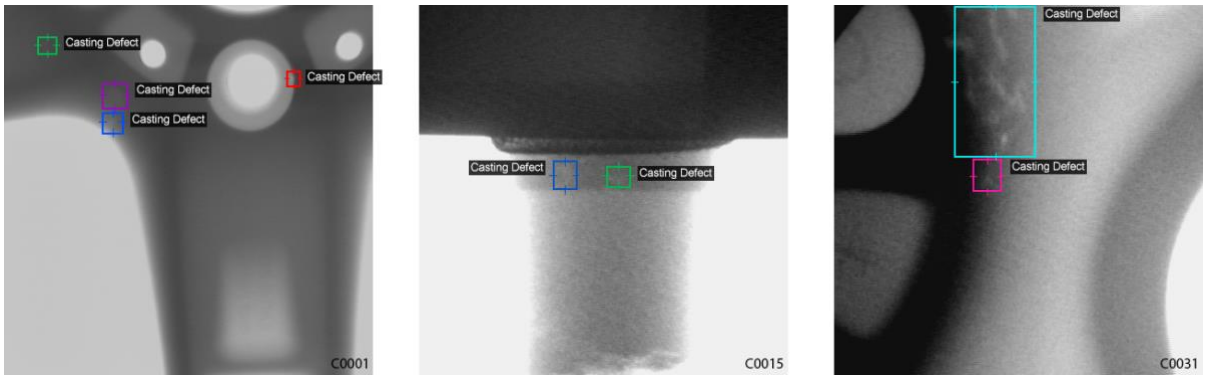


Figure 2. Examples of X-ray images in the GDXray Castings dataset. The colored boxes show the ground-truth labels for casting defects

In this work, a fast and accurate defect detection system is developed by leveraging recent advances in computer vision. The proposed defect detection system is based on the mask region-based CNN (Mask R-CNN) architecture [12]. This architecture simultaneously performs object detection and instance segmentation, making it useful for a range of automated inspection tasks. The proposed system is trained and evaluated on the GRIMA database of X-ray images (GDXray) dataset, published by Grupo de Inteligencia de Máquina (GRIMA) [13]. Some examples from the GDXray dataset are shown in Figure 2.

The remainder of this article is organized as follows: The first section provides an overview of related works, and the second section provides a brief introduction to CNNs. A detailed description of the proposed defect detection system is provided in the "Defect Detection System" section. The "Implementation Details and Experimental Results" section explains how the system is trained to detect casting defects, and provides the main experimental results, as well as a comparison with similar systems in the literature. The article is concluded with a number of in-depth studies, a thorough discussion of the results, and a brief conclusion.

Related Works

The detection and segmentation of casting defects using traditional computer vision techniques has been relatively well-studied. One popular method is background subtraction, where an estimated background image (which does not contain the defects) is subtracted from the preprocessed image to leave a residual image containing the defects and random noise [14,15]. Background subtraction has also been applied to the welding defect detection task, with varying levels of success [16–18]. However, background subtraction tends to be very sensitive to the positioning of the image, as well as random image noise [14]. A range of matched filtering techniques have also been proposed, with modified median (MODAN) filtering being a popular choice [19]. The MODAN–Filter is a median filter with adapted filter masks, that is designed to differentiate structural contours of the casting piece from casting defects [20]. A number of researchers have proposed wavelet-based techniques with varying levels of success [4,21]. In wavelet-based and frequency-based approaches, defects are commonly identified as high-frequency regions of the image, when compared to the comparatively lower frequency background [22]. Many of these approaches fail to combine local and global information from the image when classifying defects, making them unable to separate design features like holes and edges from casting defects.

In many traditional computer vision approaches, it is common to manually identify a number of features which can be used to classify individual pixels. Each image pixel is classified as a defect or treated as not being a defect, depending on the features that are computed from a local neighborhood around the pixel. Common features include statistical descriptors (mean, standard deviation, skewness, kurtosis) and localized wavelet decomposition [4]. Several fuzzy

logic approaches have also been proposed, but these techniques have been largely superseded by modern CNN-based computer vision techniques [23].

The related task of automated surface inspection (ASI) is also well-documented in the literature. In ASI, surface defects are generally described as local anomalies in homogeneous textures. Depending on the properties of surface texture, ASI methods can be divided into four approaches [24]. One approach is structural methods that model the texture primitives and displacements. Popular structural approaches include primitive measurement [25], edge features [26], and morphological operations [27]. The second approach is the statistical methods which measure the distribution of pixel values. The statistical approach is efficient for stochastic textures, such as ceramic tiles, castings, and wood. Popular statistical methods include histogram-based method [28], local binary pattern (LBP) [29], and co-occurrence matrix [30]. The third approach is filter-based methods that apply filter banks on texture images. The filter-based methods can be divided into spatial-domain [31], frequency-domain [32], and spatial-frequency domain [33]. Finally, model-based approaches construct representations of images by modeling multiple properties of defects [34].

The research community, including this work, is greatly benefited from well-archived experimental datasets, such as the GDXray dataset [13]. The performance of several simple methods for defect segmentation are compared in [37] using the GDXray Welds series, but each method is only evaluated qualitatively. A comprehensive study of casting defect detection using various computer vision techniques is provided in [38], where patches of size 32 x 32 pixels are cropped from GDXray Castings series and used to train and test a number of different classifiers. The best performance is achieved by a simple LBP descriptor with a linear SVM classifier [38]. Several deep learning approaches are also evaluated, obtaining up to 86.4 % patch classification

accuracy. When applying the deep learning techniques, the authors resize the $32 \times 32 \times 3$ pixel patches to a size of $244 \times 244 \times 3$ pixels so that they can be feed into pretrained neural networks [38,39]. A deep CNN is used for weld defect segmentation in [40] obtaining 90.5 % accuracy on the binary classification of 25×25 pixel patches.

Convolutional Neural Networks

There has been significant progress in the field of computer vision, particularly in image classification, object detection and image segmentation. The development of deep CNNs has led to vast improvements in many image processing tasks. This section provides a brief overview of CNNs. For a more comprehensive description, the reader is referred to [41].

In a CNN, pixels from each image are converted to a featurized representation through series of mathematical operations. Images can be represented as an order 3 tensor $\boldsymbol{I} \in \mathbb{R}^{H \times W \times D}$ with height H, width W, and D color channels [41]. The input sequentially goes through a number of processing steps, commonly referred to as layers. Each layer i, can be viewed as an arbitrary transformation $\boldsymbol{x}_{i+1} = f(\boldsymbol{x}_i; \boldsymbol{\theta}_i)$ with inputs $\boldsymbol{x}_i$, outputs $\boldsymbol{x}_{i+1}$, and parameters $\boldsymbol{\theta}_i$. The outputs of a layer are often referred to as a feature map. By combining multiple layers it is possible to develop a complex nonlinear function which can map high-dimensional data (such as images) to useful outputs (such as classification labels) [41]. More formally, a CNN can be thought of as the composition of number of functions:

$$f(\boldsymbol{x}) = f_N(\dots f_2(f_1(\boldsymbol{x}_1; \boldsymbol{\theta}_1); \boldsymbol{\theta}_2) \dots); \boldsymbol{\theta}_N), \tag{1}$$

where $\boldsymbol{x}_1$ is the input to the CNN and $f(\boldsymbol{x})$ is the output. There are several layer types which are common to most modern CNNs, including convolution layers, pooling layers and batch

normalization layers. A convolution layer is a function $f_i(\boldsymbol{x}_i; \boldsymbol{\theta}_i)$ that convolves one or more parameterized kernels with the input tensor, $\boldsymbol{x}_i$. Suppose the input $\boldsymbol{x}_i$ is an order 3 tensor with size $H_i \times W_i \times D_i$. A convolution kernel is also an order 3 tensor with size $H \times W \times D_i$. The kernel is convolved with the input by taking the dot product of the kernel with the input at each spatial location in the input. The convolution of a $H \times W \times 1$ kernel with an image is shown diagrammatically in Figure 3. By convolving certain types of kernels with the input image, it is possible to obtain meaningful outputs, such as the image gradients. In most modern CNN architectures, the first few convolutional layers extract features like edges and textures. Convolutional layers deeper in the network can extract features that span a greater spatial area of the image, such as object shapes.

Deep neural networks are, by design, parameterized nonlinear functions [41]. An activation function is applied to the output of a neural network layer to introduce this nonlinearity. Traditionally, the sigmoid function was used as the nonlinear activation function in neural networks. In modern architectures, the Rectified Linear Unit (ReLU) is more commonly used as the neuron activation function, as it performs best with respect to runtime and generalization

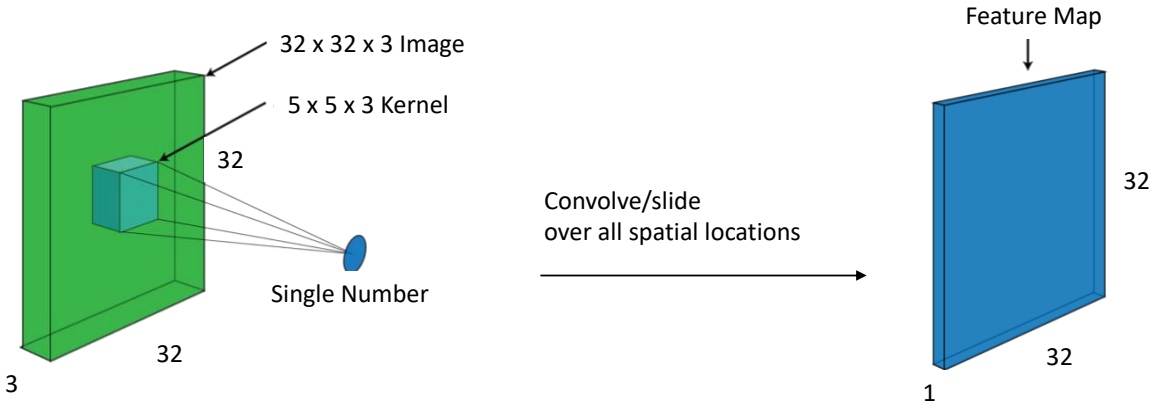


Figure 3. Convolution of an image with a kernel to produce a feature map. Zero-padding is used to ensure that the spatial dimensions of the input layer are preserved [42].

error [43]. The nonlinear ReLU function follows the formulation $f(z) = \max(0, z)$ for each value, z, in the input tensor $\boldsymbol{x}_i$. Unless otherwise specified, the ReLU activation function is used as the activation function in the defect detection system described in this article.

Pooling layers are also common in most modern CNN architectures [41,44]. The primary function of pooling layers is to progressively reduce the spatial size of the representation to reduce the number of parameters in the network, and hence control overfitting. Pooling layers typically apply a max or average operation over the spatial dimensions of the input tensor. The pooling operation is typically performed over a 2×2 or 3×3 area of the input tensor. By stacking pooling and convolutional layers, it is possible to build a network that allows a hierarchical and evolutionary development of raw pixel data towards powerful feature representations.

Training a neural network is performed by minimizing a loss function [41]. The loss function is normally a measure of the difference between the output of the neural network and the ground truth. As long as each layer of the neural network is differentiable, it is possible to calculate the gradient of the loss function with respect to the parameters. The backpropagation algorithm allows the numerical gradients to be calculated efficiently [45]. A gradient-based optimization algorithm such as stochastic gradient descent (SGD) can be used to find the parameters that minimize the loss function.

Recently, a number of techniques based on CNNs have been successfully applied to the object detection task. Two notable neural network approaches are Faster Region-Based CNN (Faster R-CNN) [46] and Single Shot Multibox Detector (SSD) [47]. These approaches share many

similarities, but the latter is designed to prioritize evaluation speed over accuracy. Both approaches are often used with a Visual Geometry Group (VGG) or Residual Network (ResNet) backbone for feature extraction. A comparison of different object detection networks is provided in [48]. Mask R-CNN is an extension of Faster R-CNN that simultaneously performs object detection and instance segmentation [12]. In previous research, it has been demonstrated that Faster R-CNN can be used as the basis for a fast and accurate defect detection system [49]. This work builds on that progress by developing a defect detection system that simultaneously performs object detection and instance segmentation.

RESIDUAL NETWORKS

The properties of a neural network are characterized by choice and arrangement of the layers, often referred to as the architecture. Deeper networks generally allow more complex features to be computed from the input image. However, increasing the depth of a neural network often makes it more difficult to train, due to the vanishing gradient problem [44]. The residual network (ResNet) architecture was designed to avoid many of the issues that plagued very deep neural networks. Most predominately, the use of residual connections helps to overcome the vanishing gradient problem [44]. A cell from the ResNet architecture is shown in Figure 4. There are a number of standard variants of the ResNet architecture, containing between 18 and 152 layers. In this work, the relatively large ResNet-101 variant with 101 trainable layers is used as the neural network backbone [44].

While ResNet was designed primarily to solve the image classification problem, it can also be used for a wider range of image processing tasks. More specifically, the outputs from the intermediate layers can be used as high-level representations of the image. When used this way, ResNet is referred to as a feature extractor, rather than a classification network.

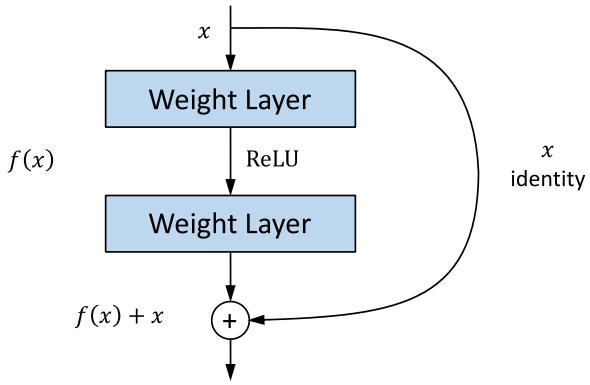


Figure 4. A cell from the Residual Network architecture.

Defect Detection System

In this section, a defect detection system is proposed to identify casting defects in X-ray images. The proposed system simultaneously performs defect detection and defect segmentation, making it useful for a range of automated inspection applications. The design of the defect detection system is based on the Mask R-CNN architecture [12]. As depicted in Figure 5, the defect detection system is composed of four modules. The first module is a feature extraction module that generates a high-level featurized representation of the input image. The second module is a CNN that proposes regions of interest (RoIs) in the image, based on the featurized image. The third module is a CNN that attempts to classify the objects in each RoI [46]. The fourth module performs image segmentation, with the goal of generating a binary mask for each region. Each module is described in detail throughout the remainder of this section.

FEATURE EXTRACTION

The first module in the proposed defect detection system transforms the image pixels into a high-level featurized representation. Many CNN-based object detection systems use the VGG-16 architecture to extract features from the input image [10,46,50]. However, recent work has

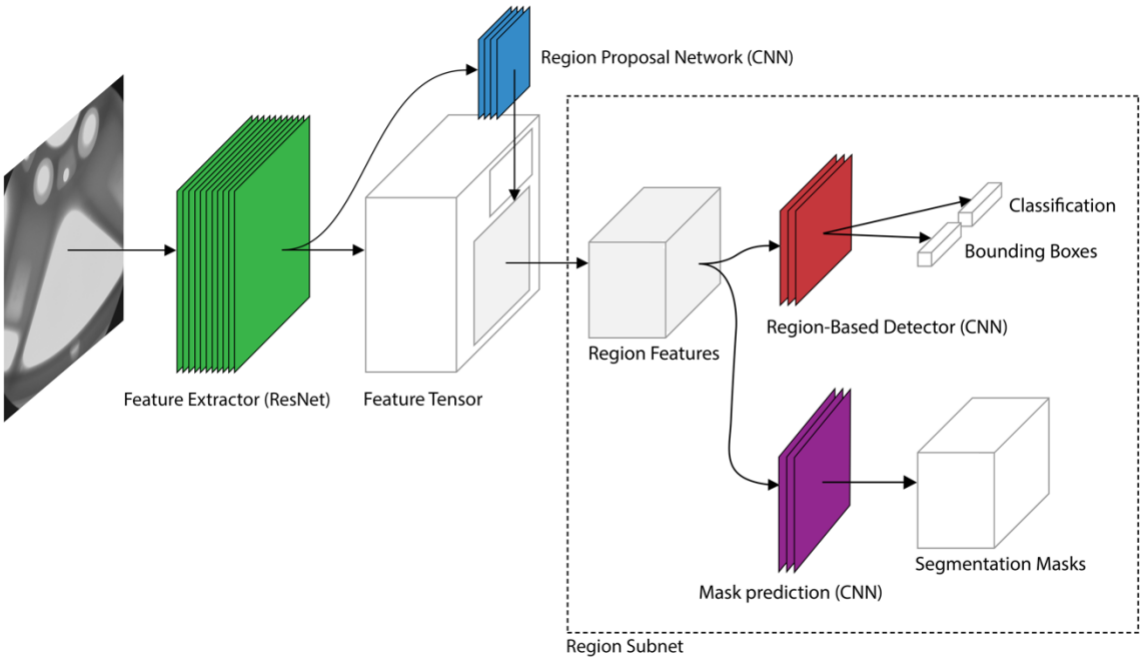


Figure 5. The neural network architecture of the proposed defect detection system. The system consists of four convolutional neural networks, namely the ResNet-101 feature extractor, region proposal network, region-based detector and the mask prediction network.

demonstrated that better results can be obtained with more modern feature extractors [48]. In a related work, we have shown that an object detection network with the ResNet-101 feature extractor results in a higher bounding-box prediction accuracy on the GDXray Castings dataset, than the same object detection network with a VGG-16 feature extractor [49]. Therefore, the ResNet-101 architecture is chosen as the backbone for the feature extraction module. The neural-network architecture of the feature extraction module is shown in Figure 5. Some feature maps from the feature extraction module are shown in Figure 6.

The ResNet-101 feature extractor is a very deep convolutional neural network with 101 trainable layers and approximately 27 million parameters. Hence, it is unlikely that the network can be trained to extract meaningful features from input images, using the relatively small GDXray dataset. One interesting property of CNN-based feature extractors is that the features they generate

often transfer well across different image processing tasks. This property is leveraged when training the proposed casting defect detection system, by first training the feature extractor on the large ImageNet dataset [51]. Throughout the training process the feature extractor learns to extract many different types of features, only some of which are useful on the comparatively simpler casting defect detection task. When training the object detection network on the GDXray Castings dataset, the system learns which features correlate well with casting defects and discards unneeded features. This process tends to work well, as it is much easier to discard unneeded features than it is to learn entirely new features.

Table 1. The neural network architecture used for feature extraction. The architecture is based on the ResNet-101 architecture, but excludes the "conv5_x" block which is primarily designed for image classification [44]. The term stride refers to the step size of the convolution operation.

Layer name	**Filter Size (width × height, number filters)**	**Output Size (width × height × depth)**
conv1	$7 \times 7, 64$, stride 2	$112 \times 112 \times 64$
conv2_x	3×3, max pool, stride 2	$56 \times 56 \times 256$
	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	
conv3_x	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$28 \times 28 \times 512$
conv4_x	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$14 \times 14 \times 1024$

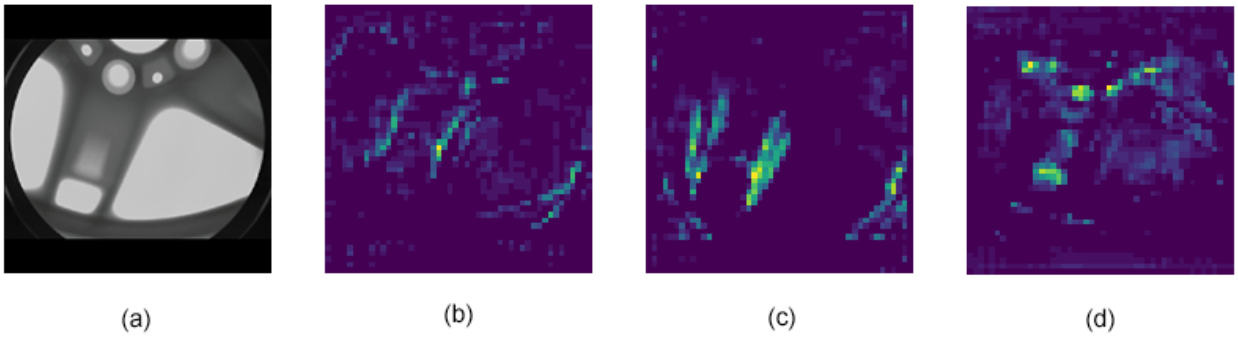


Figure 6. Feature maps from the last layer of the "conv 4" ResNet feature extractor. Clockwise from the top left image: (a) the resized and padded X-ray image, (b) a feature map which appears to capture horizontal gradients (c) a feature map which appears to capture long straight vertical edges, (d) a feature map which appears to capture hole-shaped objects.

REGION PROPOSAL NETWORK

The second module in the proposed defect detection system is the region proposal network (RPN). The RPN takes a feature map of any size as input and outputs a set of rectangular object proposals, each with a score describing the likelihood that the region contains an object. To generate region proposals, a small CNN is convolved with the output of the ResNet-101 feature extractor. The input to this small CNN is an $n \times n$ spatial window of the ResNet-101 feature map. At a high-level, the output of the RPN is a vector describing the bounding box coordinates and likeliness of objects at the current sliding position. An example output containing 50 region proposals is shown in Figure 7.

Anchor Boxes

Casting defects come in a range of different scales and aspect ratios. To accurately identify casting defects, it is necessary to evaluate the proposed bounding boxes with a range of box shapes, at every location in the image. These boxes are commonly referred to as anchor boxes. Anchors vary in aspect-ratio and scale, so as to contain any potential object in the image. At each sliding location, the RPN

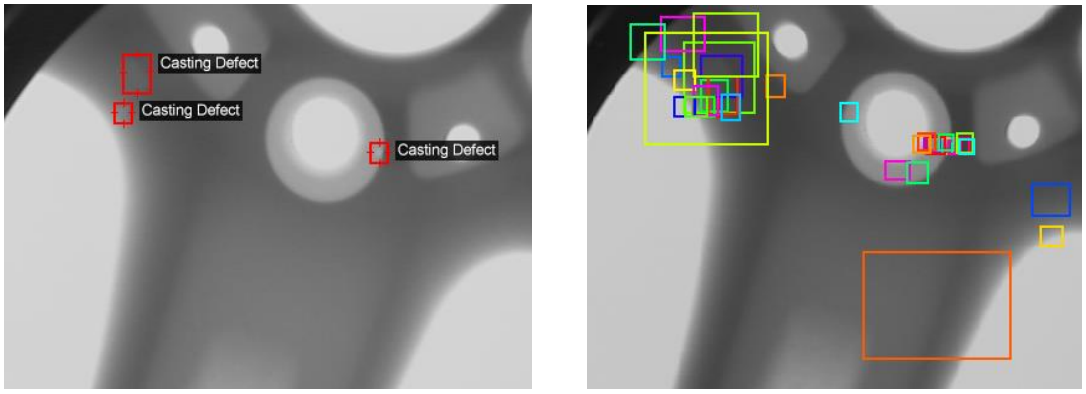


Figure 7. Ground truth casting defect locations (left). The top 50 region proposals from the RPN for the same X-Ray image (right).

estimates the likelihood that each anchor box contains an object. The anchor boxes for one position in the feature map are shown in Figure 8. In this work, anchor boxes with 3 scales and 5 aspect ratios are used, yielding 15 anchors at each sliding position. The total number of anchors in each image depends on the size of the image. For a convolutional feature map of a size $W \times H$ (typically ~2,400), there are $15WH$ anchors in total.

The size and scale of the anchor boxes are chosen to match the size and scale of objects in the dataset. It is common to use anchor boxes with areas of 128^2 , 256^2, and 512^2 pixels and aspect ratios of 1:1, 1:2, and 2:1, for detection of common objects like people and cars [46]. However, many of the casting defects in the GDXray dataset are on the scale of 20 x 20 pixels. Therefore, the smallest anchor box is chosen to be 16 x 16 pixels. Aspect ratios 1:1, 1:2, and 2:1 are used. Scale factors of 1, 2, 4, 8, and 16 are used. Most defects in the dataset are smaller than 64 x 64 pixels, so using scales 1, 2, and 4 could be considered sufficient for the defect detection task. However, the object detection network is pretrained on a dataset with many large objects, so the larger scales are included to avoid restricting the system during the pretraining phase.

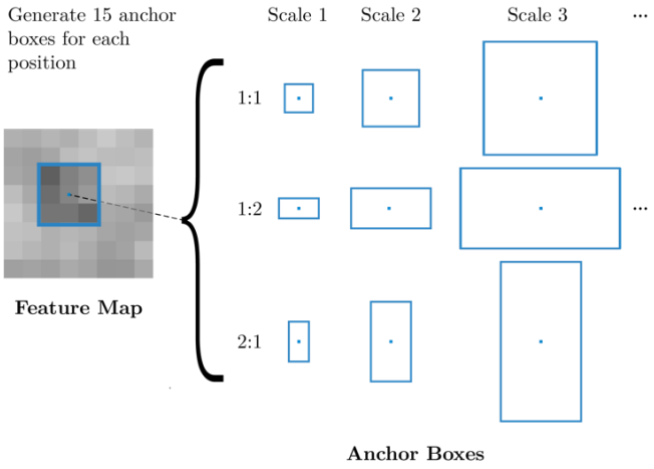


Figure 8. Anchor Boxes at a certain position in the feature map.

Region Proposal Network Architecture

The RPN predicts the bounding box coordinates and probability that the box contains an object, for all k anchor boxes at each sliding position. The $n \times n$ input from the feature extractor is first mapped to a lower-dimensional feature vector (512-d) using a fully connected neural network layer. This feature vector is fed into two sibling fully-connected layers: a box-regression layer (*loc*) and a box-classification layer (*cls*). The *class* layer outputs $2k$ scores that estimate the probability of *object* and *not object* for each anchor box. The *loc* layer has $4k$ outputs, which encode the coordinate adjustments for each of the k boxes. The reader is referred to [46] for a detailed description of the neural network architecture. The probability that an anchor box contains an object is referred to as the objectness score of the anchor box. This objectness score can be thought of as a way to distinguish objects in the image from the background. At the end of the region proposal stage, the top n anchor boxes are selected by objectness score as the region proposals.

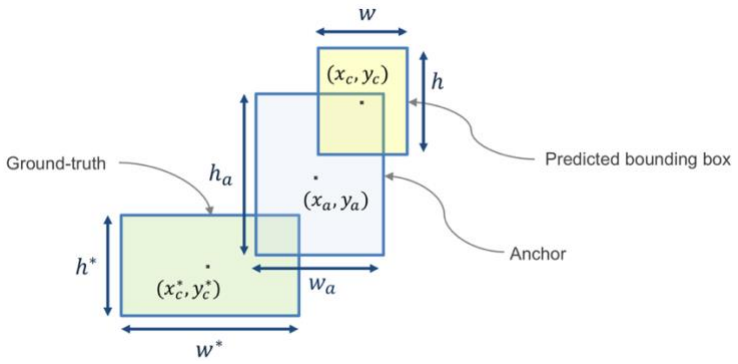


Figure 9. The geometry of an anchor, a predicted bounding box, and a ground truth box

Training

Training the RPN involves minimizing a combined classification and regression loss, that is now described. For each anchor, a, the best matching defect bounding box b is selected using the intersection over union (IoU) metric. If such a match is found, then a is assumed to contain a defect and it is assigned a ground-truth class label $p_a^* = 1$. In this case, a vector encoding of box b with respect to anchor a is created, and denoted $\phi(b; a)$. If no match is found, then a does not contain a defect and the class label is set $p_a^* = 0$. At training time, the location loss function L_{loc} captures the distance between the true location of a bounding box and the location of the region proposal [46]. The location-based loss for a is expressed as a function of the predicted box encoding $f_{loc}(\boldsymbol{I}; a, \boldsymbol{\theta})$ and ground truth $\phi(b_a; a)$:

$$L_{loc}(a, \boldsymbol{I}; \boldsymbol{\theta}) = p_a^* \cdot L_{\text{SmoothL1}}\big(\phi(b_a; a) - f_{loc}(\boldsymbol{I}; a, \boldsymbol{\theta})\big), \tag{2}$$

where $\boldsymbol{I}$ is the image, $\boldsymbol{\theta}$ is the model parameters, and L_{SmoothL1} is the smooth L1 loss function, as defined in [50]. The box encoding of box b with respect to a is a vector:

$$\phi(b; a) = \left[\frac{x_c}{w_a}, \frac{y_c}{h_a}, \log w, \log h\right]^T, \tag{3}$$

where x_c and y_c are the center coordinates of the box, w is the box width, and h is the box height. w_a and h_a are the width and height of the anchor a. The geometry of an anchor, a predicted bounding box, and a ground truth box is shown diagrammatically in Figure 9. The classification loss is expressed as a function of the predicted class $f_{cls}(\boldsymbol{I}; a, \boldsymbol{\theta})$ and p_a^*:

$$L_{cls}(a, \boldsymbol{I}; \boldsymbol{\theta}) = L_{CE}\big(p_a^*,\ f_{cls}(\boldsymbol{I}; a, \boldsymbol{\theta})\big), \tag{4}$$

where L_{CE} is the cross-entropy loss function. The total loss for a is expressed as the weighted sum of the location-based loss and the classification loss [48]:

$$L_{RPN}(a, \boldsymbol{I}; \boldsymbol{\theta}) = \alpha \cdot L_{loc}(a, \boldsymbol{I}; \boldsymbol{\theta}) \ + \ \beta \cdot L_{cls}(a, \boldsymbol{I}; \boldsymbol{\theta}), \tag{5}$$

where α, β are weights chosen to balance localization and classification losses [48]. To train the object detection model, (5) is averaged over the set of anchors and minimized with respect to parameters $\boldsymbol{\theta}$.

Region Proposal Network Transfer Learning

The RPN is an ideal candidate for the application of transfer learning, as it identifies regions of interest (RoIs) in images, rather than identifying particular types of objects. Transfer learning is a machine learning technique where information that is learned in one setting is exploited to improve generalization in another setting. It has been shown that transfer learning is particularly applicable for domain-specific tasks with limited training data [52,53]. When training an object detection network on a large dataset with many classes, the RPN learns to identify subsections of the image that likely contain an object, without discriminating by object class. This property is leveraged by first pretraining the object detection system on a large dataset with many classes of objects, namely the Microsoft

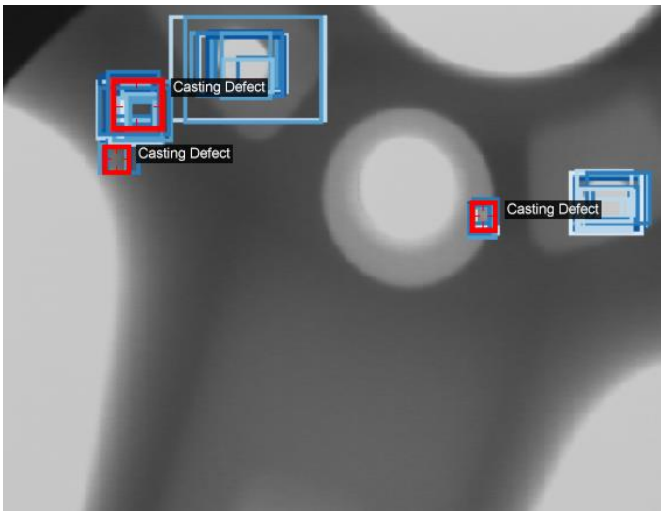


Figure 10. The top 50 regions of interest, as predicted by a region proposal network trained on the Microsoft Common Objects in Context dataset. The predicted regions of interest are shown in blue, and the ground-truth casting defect locations are shown in red.

Common Objects in Context (COCO) dataset [54]. Interestingly, when the RPN from the trained object detection system is applied to an X-ray image, it immediately identifies casting defects amongst other interesting regions of the image. The output of the RPN after training solely on the COCO dataset is shown in Figure 10.

REGION-BASED DETECTOR

Thus far the defect detection system is able to select a fixed number of region proposals from the original image. This section describes how a region-based detector (RBD) is used to classify the casting defects in each region, and fine-tune the bounding box coordinates. The RBD is based on the Faster R-CNN object detection network [46].

The input to the RBD is cropped from the output of ResNet-101 feature extractor, according to the shape of the regressed bounding box. Unfortunately, the size of the input is dependent on the size of the bounding box. To address this issue, an RoIAlign layer is used to convert the input to a fixed-length feature vector [12]. RoIAlign works by dividing the $h \times w$ RoI

window into an $H \times W$ grid of sub-windows of size $h/H \times w/W$. Bilinear interpolation [55] is used to compute the exact values of the input features at four regularly sampled locations in each sub-window. The reader is referred to [50] for a more detailed description of the RoIAlign layer. The resulting feature vector has spatial dimensions $H \times W$, regardless of the input size.

Each feature vector from the RoIAlign layer is fed into a sequence of convolutional and fully connected layers. In the proposed defect detection system, the RBD contains two convolutional layers and two fully connected layers. The last fully connected layer produces two output vectors: The first vector contains probability estimates for each of the K object classes plus a catch-all "background" class. The second vector encodes refined bounding-box positions for one of the K classes.

The RBD is trained by minimizing a joint regression and classification loss function, similar to the one used for the RPN. The reader is referred to [50] for a detailed description of the loss function and training process.

Defect Segmentation

Instance segmentation is performed by predicting a segmentation mask for each RoI. The prediction of segmentation masks is performed using another CNN, referred to as the instance segmentation network. The input to the segmentation network is a block of features cropped from the output of the feature extractor. The instance segmentation network has a $28 \times 28 \times K$ dimensional output for each RoI, which encodes K binary masks of resolution 28×28, one for each of the K classes. The instance segmentation network is shown alongside the RBD in Figure 12.

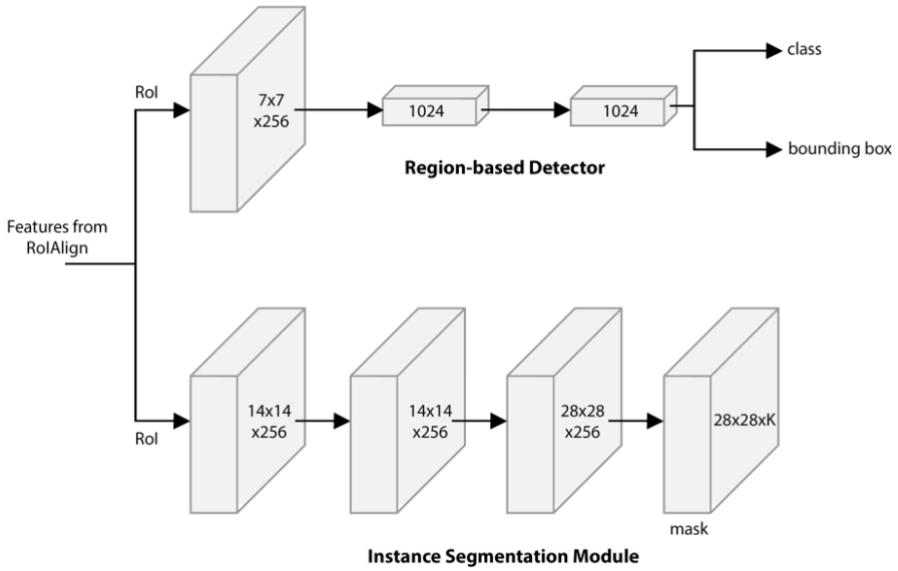


Figure 11. Head architecture of the proposed defect detection network. Numbers denote spatial resolution and channels. Arrows denote either convolution, deconvolution, or fully connected layers as can be inferred from context (convolution preserves spatial dimension while deconvolution increases it). All convolution layers are 3×3, except the output convolution layer which is 1×1. Deconvolution layers are 2×2 with stride 2.

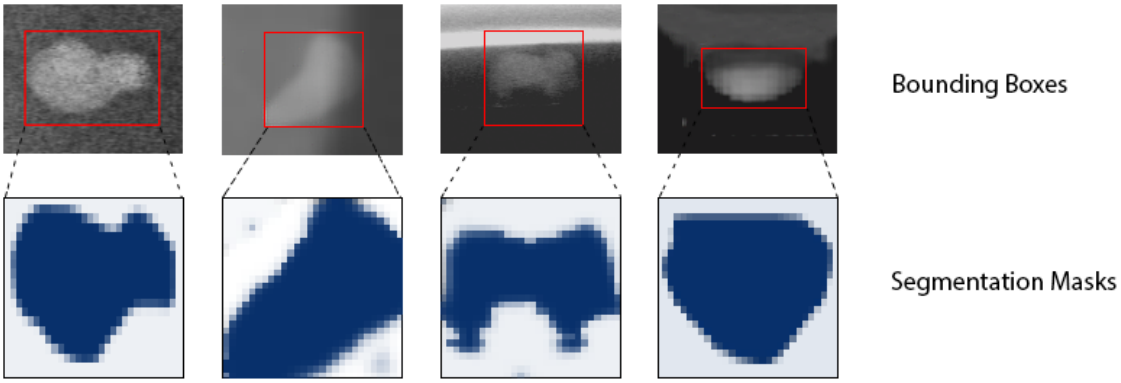


Figure 12. Examples of floating point masks. The top row shows predicted bounding boxes, and the bottom row shows the corresponding predicted segmentation masks. Masks are shown here at 28 x 28 pixel resolution, as predicted by the instance segmentation module.

During training, a per-pixel sigmoid function is applied to the output of the instance segmentation network. The loss function L_{mask} is defined as the average binary cross-entropy loss.

For an RoI associated with ground-truth class i, L_{mask} is only defined on the i-th mask (other mask outputs do not contribute to the loss). This definition of L_{mask} allows the network to generate masks for every class without competition among classes. It follows that the instance segmentation network can be trained by minimizing the joint RBD and mask loss. At test time, one mask is predicted for each class (K masks in total). However, only the i-th mask is used, where i is the predicted class by the classification branch of the RBD. The 28 × 28 floating-number mask output is then resized to the RoI size, and binarized at a threshold of 0.5. Some example masks are shown in Figure 13.

Implementation Details and Experimental Results

This section describes the implementation of the casting defect detection system described in the previous section. The model is primarily trained and evaluated using images from the GDXray dataset. [13]. The Castings series of this dataset contains 2727 X-ray images mainly from automotive parts, including aluminum wheels and knuckles. The casting defects in each image are labelled with tight fitting bounding-boxes. The size of the images in the dataset ranges from 256 × 256 pixels to 768 × 572 pixels. To ensure the results are consistent with previous work, the training and testing data is divided in the same way as described in [49].

TRAINING

The model is trained in a manner similar to many other modern object detection networks, such as Faster R-CNN and Mask R-CNN [12,46]. However, several adjustments are made to account for the small size of casting defects, and the limited number of images in the GDXray dataset. Images are scaled so that the longest edge is no larger than 768 pixels. Images are then padded with black pixels to a size of 768 × 768 pixels. Additionally, the images are randomly

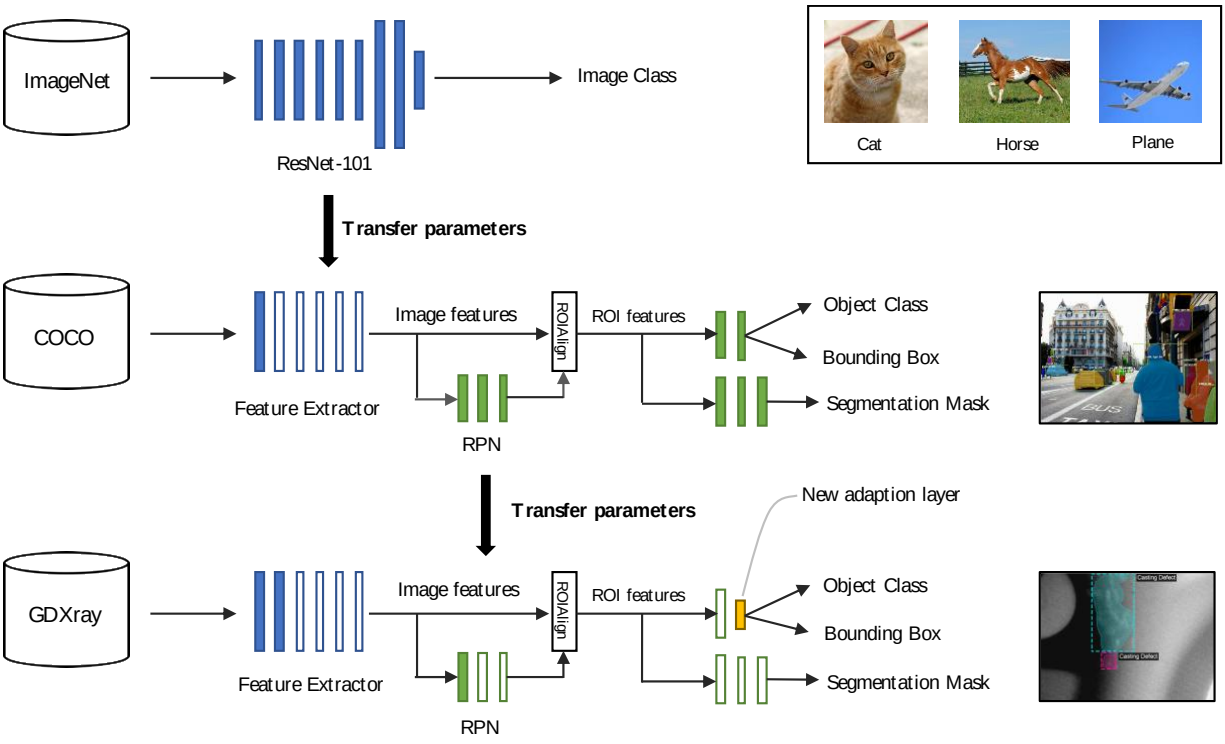


Figure 13. Training the proposed defect detection system with GDXray and transfer learning.

flipped horizontally and vertically at training time. No other form of preprocessing is applied to the images at training or testing time.

Transfer learning is used to reduce the total training time and improve the accuracy of the trained models, as depicted in Figure 13. The ResNet-101 feature extractor is initialized using weights from a ResNet-101 network that was trained on the ImageNet dataset. The defect detection system is then pre-trained on the COCO dataset [54]. When pretraining the model, the learning rates are adjusted to the schedule outlined in [48]. Training on the relatively large COCO dataset ensures that each model is initialized to localize common objects before it is trained to localize defects. Training on the COCO dataset is conducted using 8 NVIDIA K80 GPUs. Each mini-batch has 2 images per GPU and each image has 100 sampled RoIs, with a ratio of 1:3 of positive to

negatives. As in Faster R-CNN, an RoI is considered positive if it has IoU with a ground-truth box of at least 0.5 and negative otherwise.

The defect detection system is further fine-tuned on the GDXray dataset as follows: The output layers of the RBD and instance segmentation layers are resized, as they return predictions for the 80 object classes in the COCO dataset. More specifically, the output shape of these layers is resized to accommodate for two output classes, namely “Casting Defect” and “Background”. The weights of the resized layers are initialized randomly using a Gaussian distribution with zero mean and a 0.01 standard deviation. The defect detection system is trained on the GDXray dataset for 80 epochs, holding all parameters fixed except those of the output layers. The defect detection system is then trained further for an additional 80 epochs, without holding any weights fixed.

INFERENCE

The defect detection system is evaluated on a 3.6 GHz Intel Xeon E5 desktop computer machine with 8 CPU cores, 32 GB RAM, and a single NVIDIA GTX 1080 Ti Graphics Processing Unit (GPU). The models are evaluated with the GPU being enabled and disabled. For each image, the top 600 region proposals are selected by objectness score from the RPN and evaluated using the RBD. Masks are only predicted for the top 100 bounding boxes from the RBD. The proposed defect detection system is trained with and without the instance segmentation module, to investigate whether the inclusion of the instance segmentation module changes bounding box prediction accuracy. The accuracy of the system is evaluated using the GDXray Castings dataset. Every image in the testing data set is processed individually (no batching). The accuracy of each model is evaluated using the mean of average precision (mAP) as a metric [56]. The IoU metric is used to determine whether a bounding box prediction is to be considered correct. To be considered

a correct detection, the area of overlap a_o between the predicted bounding box B_p and ground truth bounding box B_{gt} must exceed 0.5 according to the formula:

$$a_o = \frac{\text{area}(\text{B}_\text{p} \cap \text{B}_\text{gt})}{\text{area}(\text{B}_\text{p} \cup \text{B}_\text{gt})}, \tag{6}$$

where $\text{B}_\text{p} \cap \text{B}_\text{gt}$ denotes the intersection of the predicted and ground truth bounding boxes and $\text{B}_\text{p} \cup \text{B}_\text{gt}$ denotes their union. The average precision is reported for both the bounding box prediction (mAP_bbox) and segmentation mask prediction (mAP_mask).

MAIN RESULTS

As shown in Table 2, the speed and accuracy of the defect detection system is compared to similar systems from previous research [49]. The proposed defect detection system exceeds the previous state-of-the-art performance on casting defect detection reaching an mAP_bbox of 0.957. Some example outputs from the trained defect detection system are shown in Figure 14. The proposed defect detection system exceeds the Faster R-CNN model from [49] in terms of accuracy and evaluation time. The improvement in accuracy is thought to be largely due to benefits arising from joint prediction of bounding boxes and segmentation masks. Both systems take a similar amount of time to evaluate on the CPU, but the proposed system is faster than the Faster R-CNN system when evaluated on a GPU. This difference arises probably because our implementation of Mask R-CNN is more efficient at leveraging the parallel processing capabilities of the GPU than the Faster R-CNN implementation used in [49]. It should be noted that single stage detection systems such as the SSD ResNet-101 system proposed in [49] have a significantly faster evaluation time than the defect detection system proposed in this article.

When the proposed defect detection system is trained without the segmentation module, the system only reaches an mAP_{bbox} of 0.931. That is, the bounding-box prediction accuracy of the proposed defect detection system is higher when the system is trained simultaneously on casting defect detection and casting defect instance segmentation tasks. This is a common benefit of multi-task learning which is well-documented in the literature [12,46,50]. The accuracy is improved when both tasks are learned in parallel, as the bounding box and segmentation modules use a shared representation of the input image (from the feature extractor) [57]. However, it should be noted that the proposed system is approximately 12% slower when simultaneously performing object detection and image segmentation. The memory requirements at training and testing time are also higher, when object detection and instance segmentation are performed simultaneously compared to pure object detection. For inference, the GPU memory requirement for simultaneous object detection and instance segmentation is 9.72 Gigabytes, which is 9 % higher than that for object detection alone.

Table 2. Comparison of the accuracy and performance of each model on the defect detection task. Results are compared to the previous state-of-the-art results, presented in [49].

Method	**Evaluation time / image using CPU [s]**	**Evaluation time / image using GPU [s]**	$\boldsymbol{mAP}_{\boldsymbol{bbox}}$	$\boldsymbol{mAP}_{\boldsymbol{mask}}$
Defect detection system (Object detection only)	5.340	0.145	0.931	-
Defect detection system (Detection & segmentation)	6.240	0.165	**0.957**	**0.930**
Faster R-CNN [49]	6.319	0.512	0.921	-
SSD ResNet-101 [49]	**0.141**	**0.051**	0.762	-

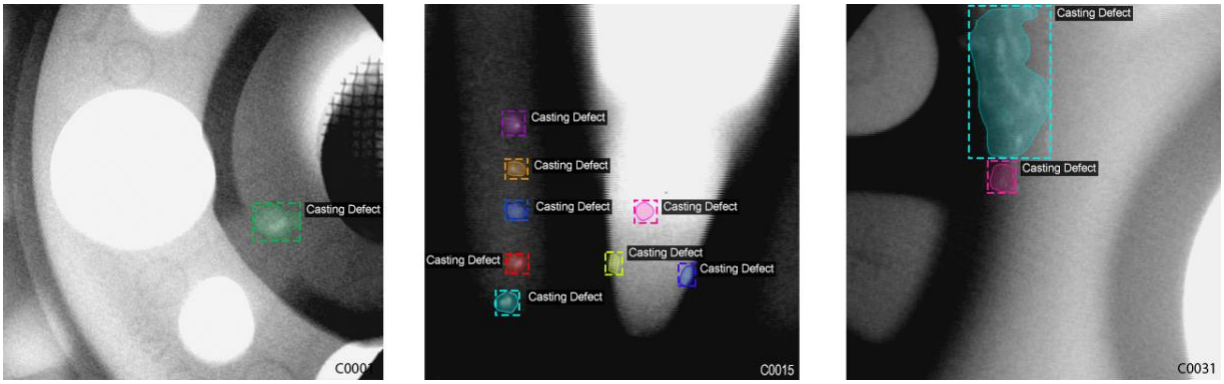


Figure 14. Example detections of casting defects from the proposed defect detection system

ERROR ANALYSIS

The proposed system makes very few misclassifications on GDXray Castings test dataset. In this section two example misclassifications are presented and discussed. Figure 15 provides an example where the defect detection system produces a false positive detection. In this case, the proposed defect detection system identifies a region of the X-ray image which appears to be a defect in the X-ray machine itself. This defect is not included GDXray castings dataset, and hence is labelled as a misclassification. Similar errors could be avoided in future systems by removing bounding box predictions which lie outside the object being imaged. Figure 16 provides an example where the bounding box coordinates are incorrectly predicted, resulting in a misclassification according to the IoU metric. However, it should be noted that the label in this case is particularly subjective; the ground truth could alternatively be labelled as two small defects rather than one large one.

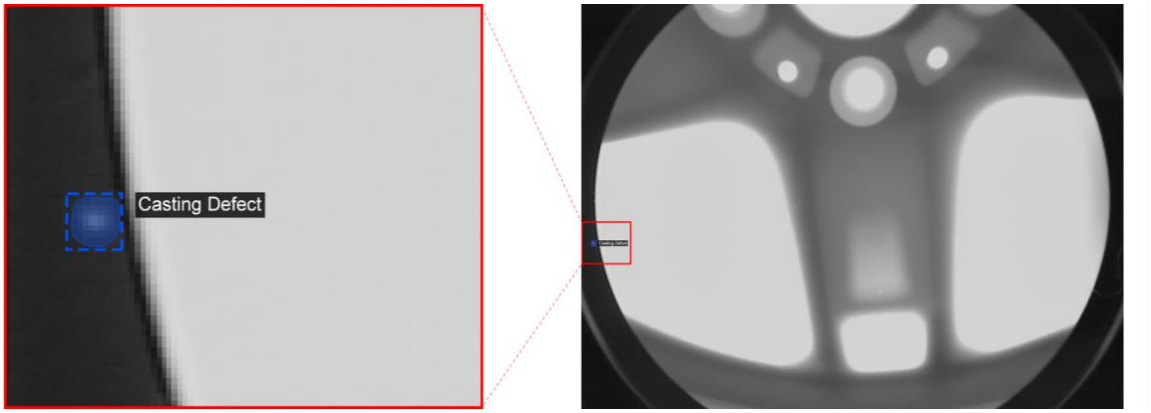


Figure 15. An example of a false positive casting defect label, where a casting defect is incorrectly detected in the X-ray machine itself. This label is considered a false positive as ground-truth defects should only be labeled on the object being scanned.

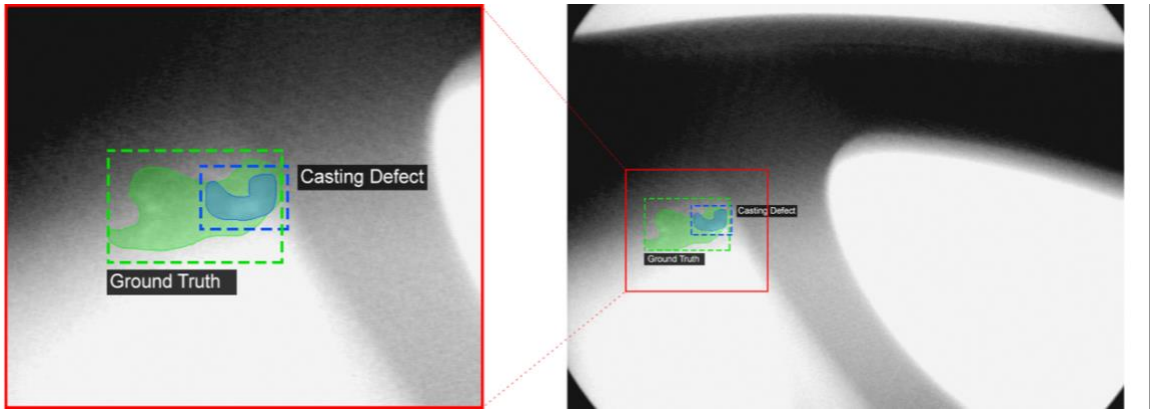


Figure 16. Casting defect misclassification due to bounding box regression error. In this instance, the defect detection system failed to regress the correct bounding box coordinates resulting in a misclassification according to the IoU metric.

Discussion

During the development of the proposed casting defect detection system, a number of experiments were conducted to better understand the system. This section presents the results of these experiments, and discusses the properties of the proposed system.

SPEED / ACCURACY TRADEOFF

There is an inherent tradeoff between speed and accuracy in most modern object detection systems [48]. The number of region proposals selected for the RBD is known to affect the speed and accuracy of object detection networks based on the Faster R-CNN framework [12,46,50]. Increasing the number of region proposals decreases the chance that an object will be missed, but it increases the computational demand when evaluating the network. Researchers typically achieve good results on complex object detection tasks using 3000 region proposals. A number of tests were conducted to find a suitable number of region proposals for the defect detection task. Figure 17 shows the relationship between accuracy, evaluation time and the number of region proposals. Based on these results, the use of 600 region proposals is considered to provide a good balance between speed and accuracy.

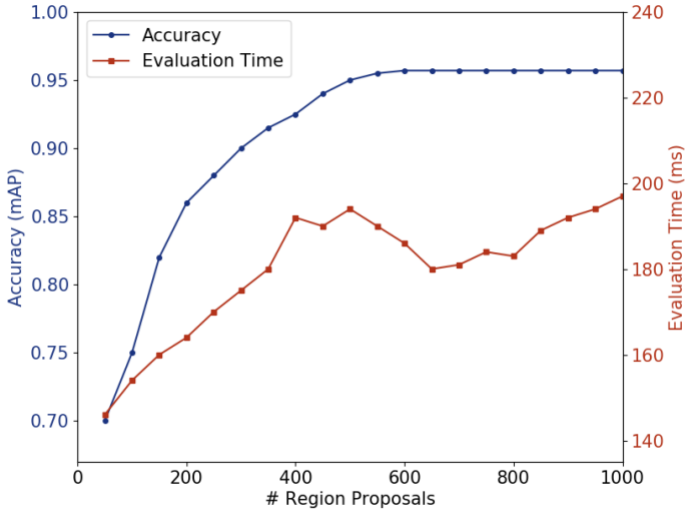


Figure 17. Relationship between casting defect detection accuracy, evaluation speed, and the number of region proposals.

DATA REQUIREMENTS

As with many deep learning tasks, it takes a large amount of labelled data to train an accurate classifier. To evaluate how the size of the training dataset influences the model accuracy, the defect detection system is trained several times, each time with a different amount of training data. The mAP_{bbox} and mAP_{mask} performance of each trained system is observed. Figure 18 shows how the amount of training data affects the accuracy of the trained defect detection system. The object detection accuracy (mAP_{bbox}) and segmentation accuracy (mAP_{mask}) improve significantly when the size of the training dataset is increased from ~1100 to 2308 images. It also appears that a large amount of training data is required to obtain satisfactory instance segmentation performance compared to defect detection performance. Extrapolating from Figure 18 suggests that a higher mAP could be achieved with a larger training dataset.

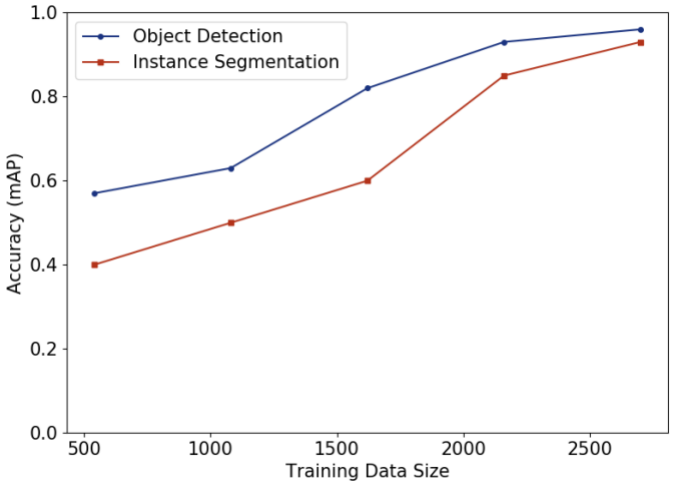


Figure 18. Mean average precision (mAP) on the test set, given different sized training sets. The object detection accuracy (mAP_{bbox}) and segmentation accuracy (mAP_{mask}) are both shown.

TRAINING SET AUGMENTATION

It is well-documented that training data augmentation can be used to artificially increase the size of training datasets with the purpose of increasing the test prediction accuracy [12,50]. The effect of several common image augmentation techniques on testing accuracy is evaluated in this section. Randomly horizontally flipping images is a technique where images are horizontally flipped at training time. This technique tends to be beneficial when training CNNs, as the label of an object is agnostic to horizontal flipping. On the other hand, vertical flipping is less common as many objects, such as cars and trains, seldomly appear upside-down. Gaussian blur is a common technique in image processing as it helps to reduce random noise that may have been introduced by the camera or image compression algorithm [58]. In this study, the Gaussian blur augmentation technique involved convolving each training image with a Gaussian kernel using a standard deviation of 1.0 pixels. Adding Gaussian noise to the training images is also a common technique for improving the robustness of the trained model to noise in the input images [59]. In this study, zero-mean Gaussian noise with a standard deviation equal to 0.05 of the image dynamic range, is added to each image. In this context, the dynamic range of the image is defined as the range between the darkest pixel and the lightest pixel in the image. The augmentation techniques are applied during the training phase only, with the original images being used at test time.

As shown in Table 3, data augmentation techniques significantly impact the accuracy of the trained system. The best accuracy is obtained using a combination of horizontal and vertical flipping. Additional techniques, such as Gaussian blur, Gaussian noise, and random cropping negatively impact accuracy. Due to the small size of most casting defects in the GDXray Castings dataset, blurring or adding noise to the images likely makes defect detection more difficult, counteracting the benefits of these data augmentation techniques.

Table 3. Influence of data augmentation techniques on test accuracy. The bounding box prediction accuracy (mAP_{bbox}) and instance segmentation accuracy (mAP_{mask}) are reported on the GDXray Castings test set.

Horizontal flip	**Vertical flip**	**Gaussian Blur**	**Gaussian Noise**	**Random Cropping**	mAP_{bbox}	mAP_{mask}
-	-	-	-	-	0.907	0.889
Yes	-	-	-	-	0.936	0.920
Yes	Yes	-	-	-	**0.957**	**0.930**
Yes	Yes	Yes	-	-	0.854	0.832
Yes	Yes	-	Yes	-	0.897	0.883
Yes	Yes	-	-	Yes	0.950	0.931

TRANSFER LEARNING

This study hypothesized that transfer learning is largely responsible for the high prediction accuracy obtained by the proposed defect detection system. The system is able to generate meaningful image features and good region proposals for GDXray casting images, before it is trained on the GDXray Casting dataset. This is made possible by initializing the ResNet feature extractor using weights pretrained on the ImageNet dataset and subsequently training the defect detection system on the COCO dataset. To test the influence of transfer learning, three training schemes are tested: In training scheme (a) the proposed defect detection system is trained on the GDXray Castings dataset without pretraining on the ImageNet or COCO datasets. Xavier initialization [60] is used to randomly assign the initial weights to the feature extraction layers. In training scheme (b) the same training process is repeated but the feature extractor weights are initialized using weights pretrained on the ImageNet dataset. Training scheme (c) uses pretrained ImageNet weights COCO pretraining, as described in the “Defect Detection System” section.

Table 4. Quantitative results indicating the influence of transfer learning on the accuracy of the trained defect detection system. The bounding box prediction accuracy (mAP_{bbox}) and instance segmentation accuracy (mAP_{mask}) are reported on the GDXray Castings training dataset and GDXray Castings test dataset.

			GDXRay Castings Training Set		**GDXRay Castings Test Set**	
Training Scheme	**Feature Extractor Initialization**	**Pretraining on MS COCO Dataset**	mAP_{bbox}	mAP_{mask}	mAP_{bbox}	mAP_{mask}
a	Xavier Initialization [63] (Random)	No	0.970	0.960	0.651	0.420
b	Pretrained ImageNet Weights	No	1.00	0.981	0.874	0.721
c	Pretrained ImageNet Weights	Yes	1.00	0.991	**0.957**	**0.930**

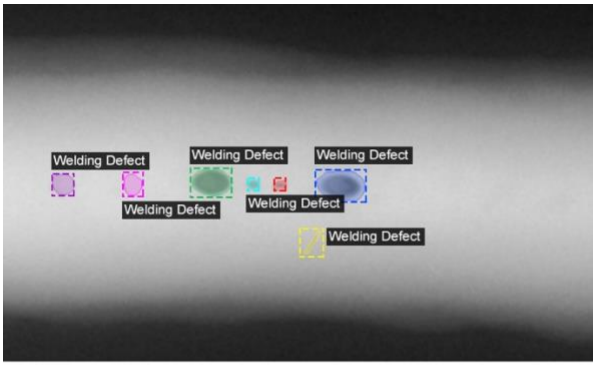
In Table 4, each trained system is evaluated on the GDXray Castings test dataset. Training scheme (a) does not leverage transfer learning, and hence the resulting system obtains a low mAP_{bbox} of 0.651 on the GDXray Castings test dataset. In training scheme (b), the feature extractor is initialized using pretrained ImageNet, and hence the system obtains a higher mAP_{bbox} of 0.874 on the same dataset. By fully leveraging transfer learning, training scheme (c) leads to a system that obtains a mAP_{bbox} of 0.957, as described earlier. In Table 4, the mAP of the trained systems is also reported on the GDXray Castings training dataset. In all cases, the model fits the training data closely, demonstrating that transfer learning affects the system's ability to generalize predictions to unseen images rather than its ability to fit to the training dataset.

WELD DEFECT SEGMENTATION WITH MULTI-CLASS LEARNING

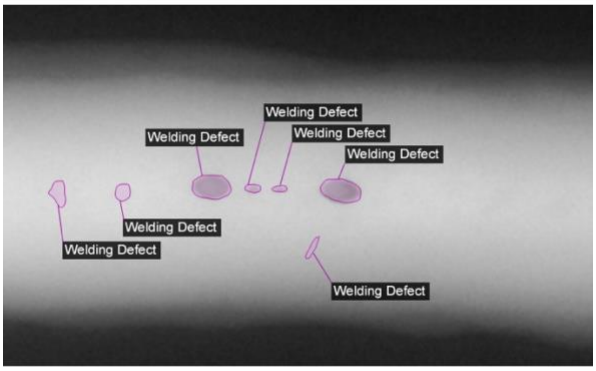
The ability to generalize a model to multiple tasks is highly beneficial in a number of applications. The proposed defect detection system was retrained on both the GDXray Castings dataset and the GDXray Welds dataset. The GDXray Welds dataset contains 88 annotated high-resolution X-ray images of welds, ranging from 3176 to 4998 pixels wide. Each high-resolution image is divided horizontally into 8 smaller images for testing and training, yielding a total of 704

images. 80 % of the images are randomly assigned to the training set, with the remaining 20 % assigned to the testing set. Unlike the GDXray Castings dataset, the GDXray Welds dataset is only annotated with segmentation masks. Bounding boxes are fitted to the segmentation masks by identifying closed shapes in the mask using a binary border following algorithm [61], and wrapping each shape in a tightly fitting bounding box. The defect detection system is simultaneously trained on images from the Castings and Welds training sets. The defect detection system is able to simultaneously identify casting defects and welding defects, reaching a segmentation accuracy ($\mathrm{mAP}_{\mathrm{mask}}$) of 0.850 on the GDXray Welds test dataset.

Some example predictions are shown in Figure 19. The detection and segmentation of welding defects can be considered very accurate, especially given the small size of the GDXray Welds dataset with only 88 high-resolution images. Unfortunately, there is no measurable improvement on the accuracy of casting defect detection when jointly training on both datasets.



Defect Detection System (Ours)



Ground Truth

Figure 19. Comparison of weld defect detections to ground truth data, using one image from the GDXray Welds series. The task is primarily an instance segmentation task, so the ground truth bounding boxes are not shown.

DEFECT DETECTION ON OTHER DATASETS USING ZERO-SHOT TRANSFER

A good defect detection system should be able to classify defects for a wide range of different objects. The defect detection system can be said to generalize well if it is able to detect defects in objects that do not appear in the training dataset. In the field of machine learning, zero-shot transfer is the process of taking a trained model, and using it, without retraining, to make predictions on an entirely different dataset. To test the generalization properties of the proposed defect detection system, the trained system is tested on a range of X-ray images from other sources.

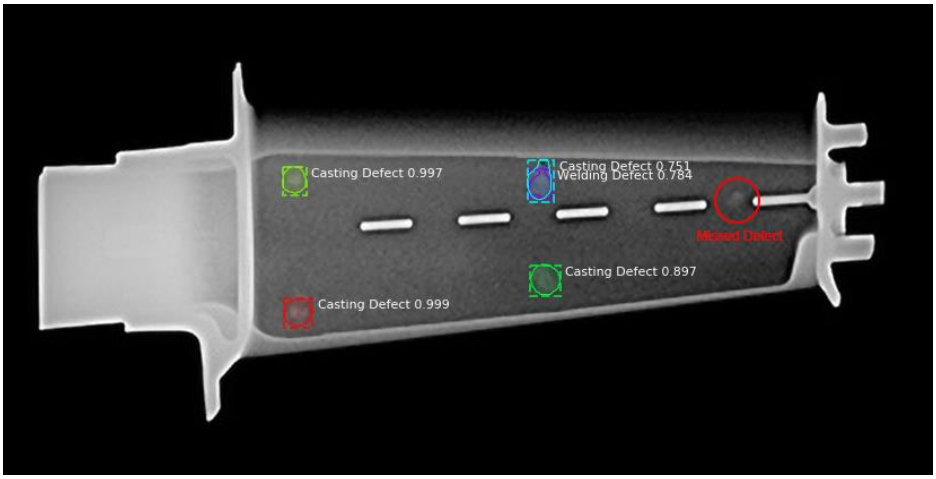


Figure 20. Defect detection and segmentation results on an X-ray image of a jet turbine blade. The defect detection system correctly identifies four out of the 5 five defects in the image. The top right defect is incorrectly classified as both a "Casting Defect" and a "Welding Defect".

The system correctly identifies a number of defects in a previously unseen X-ray image of a jet turbine blade, as shown in Figure 20. The jet turbine blade contains five casting defects, of which four are identified correctly. It is unsurprising that the system fails to identify one of the casting defects in the image, as there are no jet engine turbine blades in the GDXray dataset. Nonetheless, the fact that the system can identify defects in images from different datasets demonstrates its potential for generalizability and robustness.

Summary and Conclusion

This work presents a defect detection system for simultaneous detection and segmentation of defects in metal castings. This ability to simultaneously perform defect detection and segmentation makes the proposed system suitable for a range of automated quality control applications. The proposed defect detection system exceeds state-of-the-art performance for defect detection on the GDXray Castings dataset obtaining a mean average precision (mAP_{bbox}) of 0.957, and establishes a new benchmark for instance segmentation on the same dataset. This high-accuracy system is developed by leveraging a number of powerful paradigms in machine learning,

including transfer learning, dataset augmentation, and multi-task learning. The benefit of the application of each of these paradigms was evaluated quantitively through extensive ablation testing.

The defect detection system described in this work is able to detect casting and welding defects with very high accuracy. Future work could involve training the same network to detect defects in other materials such as wood or glass. The proposed defect detection system was designed for multi-class detection, so the system could naturally be extended detect a range of different defect types in multiple materials. The defect detection system described in this work could also be trained to detect defects in additive manufacturing applications.

The proposed defect detection system is accurate and performant enough to be useful in a real manufacturing setting. However, the training process for the system is complex and computationally expensive. Future work could focus on developing a standardized method of representing these models, making it easier to distribute the trained models.

ACKNOWLEDGMENTS

The authors acknowledge the support by the Smart Manufacturing Systems Design and Analysis Program at the National Institute of Standards and Technology (NIST), US Department of Commerce. This work was performed under the financial assistance award (NIST Cooperative Agreement 70NANB17H031) to Stanford University. Certain commercial systems are identified in this article. Such identification does not imply recommendation or endorsement by NIST; nor does it imply that the products identified are necessarily the best available for the purpose. Further, any opinions, findings, conclusions, or recommendations expressed in this material are those of

the authors and do not necessarily reflect the views of NIST or any other supporting U.S. government or corporate organizations.

References

[1] Rao, T. R., 2007, *Metal Casting: Principles and Practice*, New Age International.

[2] Kronos Incorporated, 2016, "The Future of Manufacturing: 2020 and Beyond," IndustryWeek Special Research Report, p. 12.

[3] Rajkolhe, R., and Khan, J., 2014, "Defects, Causes and Their Remedies in Casting Process: A Review," International Journal of Research in Advent Technology, **2**(3), pp. 375–383.

[4] Li, X., Tso, S. K., Guan, X.-P., and Huang, Q., 2006, "Improving Automatic Detection of Defects in Castings by Applying Wavelet Technique," IEEE Transactions on Industrial Electronics, **53**(6), pp. 1927–1934.

[5] Ghorai, S., Mukherjee, A., Gangadaran, M., and Dutta, P. K., 2013, "Automatic Defect Detection on Hot-Rolled Flat Steel Products," IEEE Transactions on Instrumentation and Measurement, **62**(3), pp. 612–621.

[6] Baillie, I., Griffith, P., Jian, X., and Dixon, S., 2007, "Implementing an Ultrasonic Inspection System to Find Surface and Internal Defects in Hot, Moving Steel Using EMATs," Insight-Non-Destructive Testing and Condition Monitoring, **49**(2), pp. 87–92.

[7] Lovejoy, M., 2012, *Magnetic Particle Inspection: A Practical Guide*, Springer Science & Business Media.

[8] Masad, E., Jandhyala, V., Dasgupta, N., Somadevan, N., and Shashidhar, N., 2002, "Characterization of Air Void Distribution in Asphalt Mixes Using X-Ray Computed Tomography," Journal of Materials in Civil Engineering, **14**(2), pp. 122–129.

[9] Pinto, N., Cox, D. D., and DiCarlo, J. J., 2008, "Why Is Real-World Visual Object Recognition Hard?," PLoS Computational Biology, **4**(1), p. e27.

[10] Girshick, R., Donahue, J., Darrell, T., and Malik, J., 2014, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, Ohio, pp. 580–587.

[11] Dai, J., Li, Y., He, K., and Sun, J., 2016, "R-FCN: Object Detection via Region-Based Fully Convolutional Networks," *Advances in Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain, pp. 379–387.

[12] He, K., Gkioxari, G., Dollár, P., and Girshick, R., 2017, "Mask R-CNN," *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, Venice, Italy, pp. 2980–2988.

[13] Mery, D., Riffo, V., Zscherpel, U., Mondragón, G., Lillo, I., Zuccar, I., Lobel, H., and Carrasco, M., 2015, "GDXray: The Database of X-Ray Images for Nondestructive Testing," Journal of Nondestructive Evaluation, **34**(4), p. 42.

[14] Piccardi, M., 2004, "Background Subtraction Techniques: A Review," *2004 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, IEEE, The Hague, Netherlands, pp. 3099–3104.

[15] Rebuffel, V., Sood, S., and Blakeley, B., 2006, "Defect Detection Method in Digital Radiography for Porosity in Magnesium Castings," Materials Evaluation, ECNDT.

[16] Nacereddine, N., Zelmat, M., Belaifa, S. S., and Tridi, M., 2005, "Weld Defect Detection in Industrial Radiography Based Digital Image Processing," Transactions on Engineering Computing and Technology, **2**, pp. 145–148.

[17] Wang, G., and Liao, T. W., 2002, "Automatic Identification of Different Types of Welding Defects in Radiographic Images," NDT & E International, **35**(8), pp. 519–528.

[18] Kaftandjian, V., Joly, A., Odievre, T., Courbiere, C, and Hantrais, C, 1998, "Automatic Detection and Characterization of Aluminium Weld Defects: Comparison between Radiography, Radioscopy and Human Interpretation," Society of Manufacturing Engineers, Copenhagen, Denmark, pp. 1179–1186.

[19] Mery, D., Jaeger, T., and Filbert, D., 2002, "A Review of Methods for Automated Recognition of Casting Defects," Insight, **44**(7), pp. 428–436.

[20] MacKenzie, D. S., and Totten, G. E., 2005, *Analytical Characterization of Aluminum, Steel, and Superalloys*, CRC press.

[21] Tang, Y., Zhang, X., Li, X., and Guan, X., 2009, "Application of a New Image Segmentation Method to Detection of Defects in Castings," The International Journal of Advanced Manufacturing Technology, **43**(5–6), pp. 431–439.

[22] Zang, X.-W., Ding, Y.-Q., Lv, Y.-Y., Shi, A.-Y., and Liang, R.-Y., 2011, "A Vision Inspection System for the Surface Defects of Strongly Reflected Metal Based on Multi-Class SVM," Expert Systems with Applications, **38**(5), pp. 5930–5939.

[23] Lashkia, V., 2001, "Defect Detection in X-Ray Images Using Fuzzy Reasoning," Image and Vision Computing, **19**(5), pp. 261–269.

[24] Xie, X., 2008, "A Review of Recent Advances in Surface Defect Detection Using Texture Analysis Techniques," Electronic Letters on Computer Vision and Image Analysis (ELCVIA), **7**(3).

[25] Kittler, J., Marik, R., Mirmehdi, M., Petrou, M., and Song, J., 1994, "Detection of Defects in Colour Texture Surfaces.," *IAPR Workshop on Machine Vision Applications (MVA)*, Kawasaki, Japan, pp. 558–567.

[26] Wen, W., and Xia, A., 1999, "Verifying Edges for Visual Inspection Purposes," Pattern Recognition Letters, **20**(3), pp. 315–328.

[27] Mallik-Goswami, B., and Datta, A. K., 2000, "Detecting Defects in Fabric with Laser-Based Morphological Image Processing," Textile Research Journal, **70**(9), pp. 758–762.

[28] Kim, C.-W., and Koivo, A. J., 1994, "Hierarchical Classification of Surface Defects on Dusty Wood Boards," Pattern Recognition Letters, **15**(7), pp. 713–721.

[29] Niskanen, M., Silvén, O., and Kauppinen, H., 2001, "Color and Texture Based Wood Inspection with Non-Supervised Clustering," *2001 Scandinavian Conference on Image Analysis (SCIA 2001)*, Bergen, Norway, pp. 336–342.

[30] Conners, R. W., Mcmillin, C. W., Lin, K., and Vasquez-Espinosa, R. E., 1983, "Identifying and Locating Surface Defects in Wood: Part of an Automated Lumber Processing System," IEEE Transactions on Pattern Analysis and Machine Intelligence, (6), pp. 573–583.

[31] Ade, F., Lins, N., and Unser, M., 1984, "Comparison of Various Filter Sets for Defect Detection in Textiles," *14th International Conference on Pattern Recognition (ICPR)*, Montreal, Canada, pp. 428–431.

[32] Hosseini Ravandi, S. A., and Toriumi, K., 1995, "Fourier Transform Analysis of Plain Weave Fabric Appearance," Textile Research Journal, **65**(11), pp. 676–683.

[33] Hu, J., Tang, H., Tan, K. C., and Li, H., 2016, "How the Brain Formulates Memory: A Spatio-Temporal Model Research Frontier," IEEE Computational Intelligence Magazine, **11**(2), pp. 56–68.

[34] Conci, A., and Proença, C. B., 1998, "A Fractal Image Analysis System for Fabric Inspection Based on a Box-Counting Method," Computer Networks and ISDN Systems, **30**(20–21), pp. 1887–1895.

[35] Xie, X., and Mirmehdi, M., 2007, "TEXEMS: Texture Exemplars for Defect Detection on Random Textured Surfaces," IEEE Transactions on Pattern Analysis and Machine Intelligence, **29**(8), pp. 1454–1464.

[36] Cohen, F. S., Fan, Z., and Attali, S., 1991, "Automated Inspection of Textile Fabrics Using Textural Models," IEEE Transactions on Pattern Analysis and Machine Intelligence, **13**(8), pp. 803–808.

[37] Mirzaei, F., Faridafshin, M., Movafeghi, A., and Faghihi, R., 2017, "Automated Defect Detection of Weldments and Castings Using Canny, Sobel and Gaussian Filter Edge Detectors: A Comparison Study," Tehran, Iran.

[38] Mery, D., and Arteta, C., 2017, "Automatic Defect Recognition in X-Ray Testing Using Computer Vision," *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, Santa Rosa, California, pp. 1026–1035.

[39] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A., 2015, "Going Deeper with Convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015)*, Boston, USA.

[40] Ren, R., Hung, T., and Tan, K. C., 2018, "A Generic Deep-Learning-Based Approach for Automated Surface Inspection," IEEE Transactions on Cybernetics, **48**(3), pp. 929–940.

[41] Wu, J., 2017, *Convolutional Neural Networks*, Published online at https://cs.nju.edu.cn/wujx/teaching/15_CNN.pdf.

[42] Kim, J., Kwon Lee, J., and Mu Lee, K., 2016, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, Las Vegas, United States, pp. 1646–1654.

[43] Nair, V., and Hinton, G. E., 2010, "Rectified Linear Units Improve Restricted Boltzmann Machines," *27th International Conference on Machine Learning (ICML)*, Haifa, Israel, pp. 807–814.

[44] He, K., Zhang, X., Ren, S., and Sun, J., 2016, "Deep Residual Learning for Image Recognition," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pp. 770–778.

[45] Werbos, P. J., 1990, "Backpropagation through Time: What It Does and How to Do It," Proceedings of IEEE, **78**, pp. 1550–1560.

[46] Ren, S., He, K., Girshick, R., and Sun, J., 2015, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Advances in Neural Information Processing Systems (NIPS 2015)*, pp. 91–99.

[47] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C., 2016, "SSD: Single Shot Multibox Detector," *14th European Conference on Computer Vision (ECCV 2016)*, pp. 21--37.

[48] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., and others, 2017, "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*.

[49] Ferguson, M., Ak, R., Lee, Y.-T. T., and Law, K. H., 2017, "Automatic Localization of Casting Defects with Convolutional Neural Networks," *2017 IEEE International Conference on Big Data (Big Data 2017)*, IEEE, Boston, USA, pp. 1726–1735.

[50] Girshick, R., 2015, "Fast R-CNN," *2015 IEEE International Conference on Computer Vision (ICCV)*, IEEE, pp. 1440–1448.

[51] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., and others, 2015, "Imagenet Large Scale Visual Recognition Challenge," International Journal of Computer Vision, **115**(3), pp. 211–252.

[52] Kolar, Z., Chen, H., and Luo, X., 2018, "Transfer Learning and Deep Convolutional Neural Networks for Safety Guardrail Detection in 2D Images," Automation in Construction, **89**, pp. 58–70.

[53] Gao, Y., and Mosalam, K. M., "Deep Transfer Learning for Image-Based Structural Damage Recognition," Computer-Aided Civil and Infrastructure Engineering, **33**, pp. 748–768.

[54] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L., 2014, "Microsoft COCO: Common Objects in Context," *European Conference on Computer Vision (ECCV 2014)*, Springer, Zurich, Switzerland, pp. 740–755.

[55] Jaderberg, M., Simonyan, K., Zisserman, A., and others, 2015, "Spatial Transformer Networks," *Advances in Neural Information Processing Systems*, Montréal, Canada, pp. 2017–2025.

[56] Manning, C., Raghavan, P., and Schütze, H., 2008, *Introduction to Information Retrieval*, Cambridge University Press.

[57] Caruana, R., 1997, "Multitask Learning," Machine learning, **28**(1), pp. 41--75.

[58] Takeda, H., Farsiu, S., and Milanfar, P., 2007, "Kernel Regression for Image Processing and Reconstruction," IEEE Transactions on Image Processing, **16**(2), pp. 349–366.

[59] Zheng, S., Song, Y., Leung, T., and Goodfellow, I., 2016, "Improving the Robustness of Deep Neural Networks via Stability Training," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, Las Vegas, United States, pp. 4480–4488.

[60] Glorot, X., and Bengio, Y., 2010, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," *Thirteenth International Conference on Artificial Intelligence and Statistics*, Sardinia, Italy, pp. 249–256.

[61] Suzuki, S., and Abe, K., 1985, "Topological Structural Analysis of Digitized Binary Images by Border Following," Computer Vision, Graphics, and Image Processing, **30**(1), pp. 32–46.

List of Table Headings

List of Figure Captions

Figure 8. Anchor Boxes at a certain position in the feature map. There are 15 anchor boxes defined at each sliding window position in the feature map.

Figure 9. The geometry of an anchor, a predicted bounding box, and a ground truth box.

Figure 10. Output from the region-based detector (RBD) for an image with three casting defects. The original proposals are shown as dotted rectangles, and the corrected bounding boxes are shown as solid rectangles. The bounding boxes are all placed correctly but the green and yellow bounding boxes are placed on the same defect.

Figure 11. Head architecture of the proposed defect detection network. Numbers denote spatial resolution and channels. Arrows denote either convolution, deconvolution, or fully connected layers as can be inferred from context (convolution preserves spatial dimension while deconvolution increases it). All convolution layers are 3×3, except the output convolution layer which is 1×1. Deconvolution layers are 2×2 with stride 2.

Figure 12. Examples of floating point masks. The top row shows predicted bounding boxes, and the bottom row shows the corresponding predicted segmentation masks. Masks are shown here at 28 × 28 pixel resolution, as predicted by the instance segmentation module.

Figure 13. Training the proposed defect detection system with GDXray and transfer learning

Figure 14. Example detections of casting defects from the proposed defect detection system.

Figure 15. An example of a false positive casting defect label, where a casting defect is incorrectly detected in the X-ray machine itself. This label is considered a false positive as ground-truth defects should only be labeled on the object being scanned.

Figure 16. Casting defect misclassification due to bounding box regression error. In this instance, the defect detection system failed to regress the correct bounding box coordinates resulting in a misclassification according to the IoU metric.

Figure 17. Relationship between casting defect detection accuracy, evaluation speed, and the number of region proposals.

Figure 18. Mean average precision (mAP) on the test set, given different sized training sets. The object detection accuracy (mAP_{bbox}) and segmentation accuracy (mAP_{mask}) are both shown.

Figure 19. Comparison of weld defect detections to ground truth data, using one image from the GDXray Welds series. The task is primarily an instance segmentation task, so the ground truth bounding boxes are not shown.

Figure 20. Defect detection and segmentation results on an X-ray image of a jet turbine

blade. The defect detection system correctly identifies four out of the 5 five defects in the image. The top right defect is incorrectly classified as both a “Casting Defect” and a “Welding Defect”.