Automatic Localization of Casting Defects with Convolutional Neural Networks

Max Ferguson
Engineering Informatics Group
Civil and Environmental Engineering
Stanford University
Stanford, United States
maxferg@stanford.edu

Ronay Ak
Systems Integration Division
National Institute of Standards and Technology (NIST)
Gaithersburg, United States
ronay.ak@nist.gov

Yung-Tsun Tina Lee
Systems Integration Division
National Institute of Standards and Technology (NIST)
Gaithersburg, United States
yung-tsun.lee@nist.gov

Kincho H. Law
Engineering Informatics Group
Civil and Environmental Engineering
Stanford University
Stanford, United States
law@stanford.edu

***Abstract*—Automatic localization of defects in metal castings is a challenging task, owing to the rare occurrence and variation in appearance of defects. Convolutional neural networks (CNN) have recently shown outstanding performance in both image classification and localization tasks. We examine how several different CNN architectures can be used to localize casting defects in X-ray images. We take advantage of transfer learning to allow state-of-the-art CNN localization models to be trained on a relatively small dataset. In an alternative approach, we train a defect classification model on a series of defect images and then use a sliding classifier method to develop a simple localization model. We compare the localization accuracy and computational performance of each technique. We show promising results for defect localization on the GRIMA database of X-ray images (GDXray) dataset and establish a benchmark for future studies on this dataset.**

Keywords — Casting Defect Detection, Defect Localization, Convolutional Neural Network, Computer Vision

I. INTRODUCTION

Quality control is an important aspect of modern manufacturing processes [1]. With an increased level of competition in the manufacturing market, manufacturers must increase their production rate while maintaining stringent quality control limits. In order to meet the growing demand for high-quality products, the use of intelligent visual inspection systems is becoming essential in production lines.

The casting process can often introduce defects in the product which are detrimental to the final product quality [2]. The early detection of these defects can allow faulty products to be identified early in the manufacturing process, leading to time and cost savings [3]. It is beneficial to automate quality control where possible to ensure consistent and cost-effective inspection, especially if defect detection is to be carried out at many points along the production line. 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.

There are a number of nondestructive examination (NDE) 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 [3]. Ultrasonic inspection and magnetic particle inspection can also be used to measure the size and position of casting defects in cast components [4, 5]. An alternative method is three-dimensional X-ray computed tomography, that 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 [6, 7].

The defect detection process can be framed as either an image segmentation problem or an object localization problem. 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. In the object localization approach, the goal is to place a tight-fitting bounding box around each defect in the image. As a related task, defect classification can then be performed to classify different types of defects. Common casting defects include air holes, foreign-particle inclusions, shrinkage cavities, cracks, wrinkles and casting fins [8]. Some defect detection systems attempt to locate and classify defects, but, in this study, we choose to focus purely on defect localization.

In this work, we develop a fast and accurate casting defect localization system, utilizing recent advances in computer vision. More specifically, we develop a model that can localize casting defects in two-dimensional X-ray images. We train and evaluate the proposed casting defect localization models using the GRIMA database of X-ray images (GDXray) dataset, published by Grupo de Inteligencia de Máquina (GRIMA) [9]. The GDXray Castings subset contains 2727 X-ray images of metal parts, many of which contain casting defects.

Object localization is a widely studied but difficult problem in computer vision [10]. In many tasks, including defect detection, the aspect ratio, size, location and the number of objects in each image are typically unknown variables. There is not a large amount of literature describing the application of modern computer vision techniques for casting defect localization [11, 12]. It has been suggested that many advances in defect detection remain proprietary [3].

Most object detectors contain two important components: a feature extractor and an object classifier. In traditional methods, the feature extractor is usually a hand-engineered module, such as histogram of oriented gradients (HOG) or local binary patterns (LBP). The classifier is commonly a support vector machine (SVM) or a non-linear boosted classifier [3]. In more modern object detectors, the feature extractor and object classifier are often replaced by neural networks (NNs).

Recently, deep neural networks (DNNs) have attained impressive performance in many fields such as image classification, object detection, and semantic segmentation [13–16]. The performance of these deep networks has begun to exceed human performance in many tasks. For example, the human top-5 classification error rate on the large scale ImageNet dataset has been reported to be 5.1 %, whereas a state-of-the-art NN achieves a top-5 error rate of 3.57 % [13, 17]. The top-5 error rate indicates the fraction of the test images for which the correct label is not among the five labels considered most probable by the classification model. Large performance improvements have also been realized by training deep convolutional neural networks (CNN) for object detection [14, 15, 18]. While many NN architectures have been proposed for the object localization task, there is still no consensus on the best method. Therefore, we choose to train and compare a number of different architectures on the defect detection task. To summarize, our main contributions are as follows:

- We provide a concise summary of modern casting defect detection algorithms.
- We adapt several modern object detection networks to the casting defect task. We use these networks to conduct several experiments that explore the speed/accuracy tradeoff.
- We demonstrate that transfer learning can be used to improve defect localization on the relatively small GDXray dataset.

To the best of our knowledge, this study presents the first benchmark results for defect localization using the publicly available GDXray dataset.

II. RELATED WORK

The detection and localization of casting defects with traditional computer vision techniques is a well-studied task. 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 [19–21]. However, background subtraction tends to be very sensitive to the positioning of the image, as well as random image noise. A range of matched filter techniques have also been proposed, with modified median (MODAN) filtering being a popular choice [22]. 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 [23]. A number of other authors have proposed wavelet-based techniques with varying levels of success [8, 24]. Interestingly, the convolutional feature extraction technique proposed in this work can be seen as a generalization of many traditional filter-based techniques, where instead of specifying the filter type a priori, we attempt to learn the optimum filters [25].

Feature-based detection is another useful technique, where each image pixel is classified as a defect or not depending on 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 [8]. Selecting a threshold value to determine if a pixel is part of a defect region remains a large challenge of this method, although adaptive threshold methods have been proposed [26]. Several fuzzy logic approaches have also been proposed, but these techniques have been largely superseded by modern NN-based computer vision techniques [27].

Many state-of-the-art object detection systems developed from the region-based convolutional neural network (R-CNN) architecture [28]. 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 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 DNN [18].

III. OBJECT DETECTION ARCHITECTURES

Our main approach draws on recent successes of CNN for image feature extraction and object localization [13–15, 17]. We focus primarily on applying three recent object detection architectures to the casting defect detection task, namely: Faster R-CNN [14], R-FCN (Region-based Fully Convolutional Networks [18]) and SSD (Single Shot Multibox Detector [15]). While these architectures were originally presented with a particular feature extractor, we review each method using both the Visual Geometry Group (VGG) and Residual Network (ResNet) type feature extractors. This allows us to make a fair comparison between the performance of object detection networks on the casting defect localization task.

At a high level, all three architectures consist of a single convolutional network, trained with a mixed regression and classification objective. Each architecture defines a preset list of anchor boxes at different spatial locations in the image. These anchors often vary in aspect-ratio and scale, such as to contain any potential object in the image. The model is trained to make two predictions for each anchor: the offset between the anchor and the ground truth bounding box, and the class of the object inside the anchor, if any.

Each of the object detectors described in this work minimize a combined classification and regression loss, that we now describe. For each anchor, a, the best matching defect bounding box b is selected. The precise method for selecting the bounding box depends on the object detection architecture [14, 15, 18]. If such a match is found, then we assume a contains a defect and we assign it a ground-truth class label $y_a = 1$. In this case, we also assign a vector encoding of box b with respect to anchor a, denoted $\phi(b_a; a)$. If no match is found, we assume that a does not contain a defect and we set the class label $y_a = 0$. The location-based loss for a is expressed as a function of the predicted box encoding $f_{loc}(\boldsymbol{I}; a, \boldsymbol{\theta})$ and $\phi(b_a; a)$, where $\boldsymbol{I}$ is the image and $\boldsymbol{\theta}$ is the model parameters. The classification loss is expressed as a function of the predicted class $f_{class}(\boldsymbol{I}; a, \boldsymbol{\theta})$ and y_a. The total loss for a is expressed as the weighted sum of the location-based loss and the classification loss [29]:

$$L(a, \boldsymbol{I}; \boldsymbol{\theta}) = \alpha \cdot y_a \cdot l_{loc}\big(\phi(b_a; a) - f_{loc}(\boldsymbol{I}; a, \boldsymbol{\theta})\big) + \beta \cdot l_{class}\big(y_a, f_{class}(\boldsymbol{I}; a, \boldsymbol{\theta})\big), \quad (1)$$

where α, β are weights chosen to balance localization and classification losses [30]. The location loss function l_{loc} captures the distance between the true location and the estimated one. We use the Smooth L_1 loss function as l_{loc} with all architectures, even though the L_2 loss function was originally proposed in [15, 30]. The classification loss l_{class} measures the difference between the predicted class and the true class, for each anchor. We use the cross-entropy loss function for l_{class} [17]. Table I summarizes the box encoding and loss functions used with each architecture. To train the object detection model, (1) is averaged over the set of anchors and minimized with respect to parameters $\boldsymbol{\theta}$.

A. *Faster R-CNN*

Faster R-CNN is a state-of-art object detection system composed of two modules. The first module is a deep fully-convolutional neural network that proposes regions, and the second module is a feed-forward neural network that attempts to classify the objects in each region [14]. We will refer to the first module as the region proposal network (RPN) and the second as the region-based detector (RBD). The RBD makes two predictions for each region: a discrete class prediction for each region, and a continuous prediction of an offset by which the region needs to be shifted to best fit the ground truth bounding box.

Unlike [14], we choose to decouple the feature extractor from the RPN. This allows us to review the performance of R-CNN with two different feature extractors. This also makes it much easier to initialize the feature extractor with pretrained weights, especially when using standardized feature extractors such as VGG-16 or ResNet-101 [17, 31].

We now describe the configuration of the Faster R-CNN object detector used in this work. We configure the RPN to generate 300 region proposals for each image, as in [14].

TABLE I. LOSS AND BOX ENCODING FUNCTIONS FOR EACH ARCHITECTURE, AS USED IN THIS WORK.

Architecture	*Box Encoding*[a] $\boldsymbol{\phi(b_a; a)}$	*Class Loss* $\boldsymbol{l_{class}}$	*Loc. Loss* $\boldsymbol{l_{loc}}$
Faster R-CNN	$[\frac{x_c}{w_a}, \frac{y_c}{h_a}, \log w, \log h]$	Cross Entropy	Smooth L_1
R-FCN	$[\frac{x_c}{w_a}, \frac{y_c}{h_a}, \log w, \log h]$	Cross Entropy	Smooth L_1
SSD	$[x_0, y_0, x_1, y_1]$	Cross Entropy	Smooth L_1

[a] Boxes are encoded with respect to a matching anchor a via a function ϕ, where $[x_0, y_0, x_1, y_1]$ are the min/max coordinates of the box, x_c , y_c are its center coordinates, and w, h are its width and height. The terms w_a and h_a denote the width and height of the matching anchor, and log is used to denote the natural logarithm.

Anchor boxes are generated using 1:1, 1:2, and 2:1 aspect ratios at three different scales. As with many object detection networks, we find that faster R-CNN produces duplicate object proposals. To reduce redundancy, we apply the non-maximum suppression (NMS) algorithm on the proposed outputs [33].

We train and test the Faster R-CNN model using the images from the GDXray Castings dataset. Images are scaled to 600 pixels on the shorter edge. Additionally, we randomly flip the images horizontally at training time. This data augmentation technique is applied to artificially increase the size of the training dataset. We do not apply any other form of preprocessing to the images at training or testing time.

B. *R-FCN*

In contrast to the Faster R-CNN architecture, the R-FCN region-based detector is fully convolutional with almost all computation shared on the entire image. In Faster R-CNN, region-specific RBD must be applied several hundred times per image, greatly reducing performance [18]. In R-FCN, the feature map is cropped at the last layer of features prior to prediction, instead of cropping it at the layer where region proposals are generated [18]. This approach of pushing cropping to the last layer minimizes the amount of per-region computation that must be done.

We now describe the configuration of the R-FCN object detector used in this work. Similar to the Faster R-CNN network, we train and test R-FCN with the original unprocessed images from the GDXray dataset. We use the same anchor scales and aspect ratios as for Faster R-CNN. Again, we generate 300 region proposals for each image. Images are scaled to 600 pixels on the shorter side. Random horizontally flipping of the images is applied as a data augmentation technique. Again, we use NMS to reduce redundancy in the predicted output.

C. *Single Shot MultiBox Detector (SSD)*

Recently, a new family of object detection networks were proposed, in which the region proposal step is completely eliminated [15, 33]. One of the popular architectures, SSD, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales [15]. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. The SSD architecture uses feature maps from multiple different layers of the feature extractor to handle objects of various sizes. SSD is conceptually simpler than other methods as it eliminates the proposal generation, and subsequent feature resampling stage. The simplified architecture of the SSD network makes it faster than Faster R-CNN and R-FCN at test time [15].

We now describe the configuration of the SSD object detector used in this work. Unlike the Faster R-CNN and R-FCN architectures, the input size of the SSD architectures is fixed. We choose to use an input size of 300×300 pixels, as is common in many SSD implementations [15, 29]. During the training process, we randomly crop the input images to 300×300 pixels. If the cropped image does not contain any ground-truth-labelled defects, it is discarded. Cropped images are randomly flipped, both horizontally and vertically at training time. The amount of benefit obtained from horizontal and vertical flipping the training images often depends on the type of images in the dataset. For example, horizontally flipping an image of a boat will give an equally probable image, while vertically flipping the same image will yield an improbable image. Flipping an image from the GDXray Castings dataset either horizontally or vertically will yield an equally probable image. We chose to vertically flip input images when training the SSD model, as the SSD architecture is believed to be less robust to image transformation than methods such as Faster R-CNN [15]. However, future work could explore the benefits of each data augmentation technique in more detail.

When evaluating a test image with an SSD model it is common to scale the image to the model input size (in this case 300×300 pixels). The aspect ratio is often preserved by padding the image with black pixels. However, we found that down-scaling the GDXray images made the defects very difficult to detect. Therefore, at test time, we slide the SSD classifier over the original image with a stride of 150 pixels. We post-process the resulting predictions using non-maximum suppression (NMS) to remove duplicate boundary boxes.

IV. SLIDING CLASSIFIER

A simple approach to the object localization task is to slide a classifier over the image, generating a heat-map of potential object locations. The heat-map can then be post-processed, to generate object location predictions. Although this method has been largely superseded by modern computer vision techniques, it can provide useful insight into the object localization task.

We develop a convolutional defect classifier based on the Xnet architecture proposed for a similar defect classification task [35]. The Xnet classifier is modified to reduce overfitting, leading to the architecture shown in Table II. We refer to this modified architecture as XnetV2. We add L_2 regularization to the last four convolutional layers of the neural network. Conceptually, the lower convolutional layers use the output from the upper layers to classify each image; applying regularization to the lower layers can prevent the classifier overfitting. This is done by adding a complexity penalty, sum of squares of weights, to the loss function. For detailed information regarding regularization and overfitting, we refer the reader to [36]. The max-pooling layers in Xnet are replaced with convolutional layers with stride 2. A dropout probability of 0.25 is used during training. The proposed network architecture is "fully convolutional" in that it only contains convolutional and ReLU layers [37]. This allows the network to take an input of arbitrary size and produce correspondingly-sized output with efficient inference and learning. However, we deliberately design the network so that a 32×32 pixel image is mapped to a $1 \times 1 \times 2$ sized tensor.

TABLE II. PROPOSED XNETV2 ARCHITECTURE

l	*Layer type*	*Filter Size* (w×h, n)	*Stride*	*Output Size* $w \times h \times d$
0	Input	-	-	$32 \times 32 \times 1$
1	Convolution	$7 \times 7, 32$	1	$32 \times 32 \times 32$
2	Convolution	$3 \times 3, 32$	2	$16 \times 16 \times 32$
3	ReLU	-	-	$16 \times 16 \times 32$
4	Convolution	$3 \times 3, 64$	1	$16 \times 16 \times 64$
5	Convolution	$3 \times 3, 64$	2	$8 \times 8 \times 64$
6	ReLU	-	-	$8 \times 8 \times 64$
7	Dropout	-	-	$8 \times 8 \times 64$
8	Convolution+L_2	$3 \times 3, 128$	1	$6 \times 6 \times 128$
9	Convolution+L_2	$3 \times 3, 128$	2	$2 \times 2 \times 128$
10	ReLU	-	-	$2 \times 2 \times 128$
11	Convolution+L_2	$2 \times 2, 64$	1	$1 \times 1 \times 64$
12	Convolution+L_2	$1 \times 1, 2$	1	$1 \times 1 \times 2$
13	Softmax	-	-	$1 \times 1 \times 2$

Our key insight is to train the classification model on a large dataset of defect patches. We obtain a public dataset containing 47,520 cropped X-ray images of size 32×32 pixels along with the corresponding labels [35]. The dataset contains 23,760 patches with label *defect* and 23,760 patches with label *no-defect*. Example patches from this dataset are shown in Fig. 1. The dataset was originally synthesized by cropping defect images from the GDXray Castings series. When generating the dataset, the original cropped images were rotated at 6 different angles (0°, 60°, 120° ... 300°) and flipped both horizontally and vertically. This data augmentation technique was applied to artificially increase the size of the dataset, and encourage rotation invariance in classifiers that are subsequently trained on the dataset.

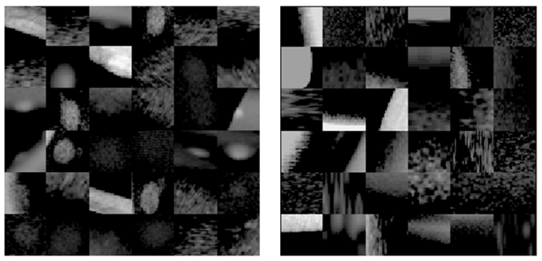

Fig 1. Examples of patches containing defects (left) and no-defects (right).

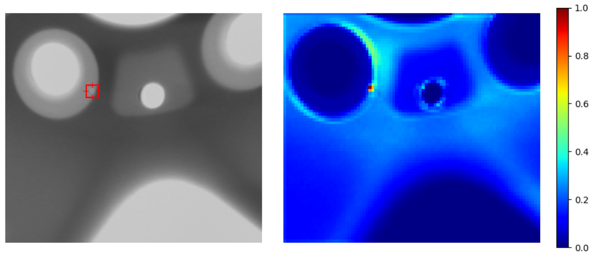

Fig 2. Original X-ray image containing a single defect (left), and the corresponding defect heat-map produced by the sliding classifier (right).

The XnetV2 architecture returns a vector of scores $\hat{y} \in \mathbb{R}^2$, with each component corresponding to the likelihood of a particular class. For the casting defect detection task, we only consider two classes: *defect* and *no-defect*.

We train the XnetV2 architecture on the patch dataset using the binary cross-entropy loss function [29]:

$$L(w) = -\frac{1}{N}\sum_{n=1}^{N} y_n \log \hat{y}_n + (1 - y_n)\log(1 - \hat{y}_n) \ , \qquad (2)$$

where N is the number of training examples and $y_n \in \mathbb{R}^2$ is a vector containing the true class labels.

Applying the trained classifier to the images in the GDXray dataset produces a defect heat-map, as shown in Fig. 2. The heat-map can be seen as an estimate of the probability that a defect exists at the corresponding spatial location in the original image. The selective search algorithm is used to generate a set of bounding box proposals from the heat-map [38]. Bounding boxes with an average heat-map value greater than 0.5 are labelled as defects.

V. Feature Extractors

It has been shown that the choice of feature extractor can strongly influence the output of the object detection network. For casting defect detection, the choice of feature extractor is likely very important:

- Casting defects such as air bubbles tend to be very small, and can easily be mistaken for random image noise.
- Casting can generate large oddly shaped voids as shown in Fig. 3. The feature extractor must operate over a large enough space to capture these formations.
- Most state-of-the-art feature extractors are designed and trained to detect well-defined objects such as and cars. We suspect that the features used to detect these objects may differ from the features used to detect casting defects.

We choose to base our feature extractors on the VGG and ResNet architectures [21, 26]. We initialize the feature extractors with pretrained weights created using the ImageNet dataset. This initialization strategy represents a form of transfer learning, where knowledge from the ImageNet dataset is transferred to the casting defect detection task.

A. VGG-16

The VGG architecture has been widely used in computer vision over the last few years. It consists of stacked convolutional and max pooling layers. We choose to use the smaller, and hence faster, 16-layer architecture known as VGG-16 [32]. The VGG-16 architecture is detailed in Fig. A1 in the Appendix. The outputs of the upper convolutional layers are used as feature maps in the object detection networks:

- **Faster R-CNN:** We extract features from the "conv5" layer whose stride size is 16 pixels.
- **SSD:** Following [15], we extract feature maps from the "conv4_3", and "fc7" layers. The "fc6" and "fc7" layers from the VGG-16 architecture are converted to convolutional layers as described in [15]. Five additional convolutional layers with decaying spatial resolution are appended to the VGG network, as described in [30].

Presently, we have not obtained the results from the R-FCN network with VGG-16 feature extractor, so we do not discuss this model in further detail here.

B. ResNet-101

The 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 [17]. We choose to use the relatively large ResNet-101 variant which has 101 trainable layers [17]. The ResNet-101 architecture is detailed in Table A1 in the Appendix. We freeze the batch normalization parameters to be those estimated during ImageNet pretraining. Again, the outputs of the upper convolutional layers are used as feature maps in the object detection networks:

- **Faster R-CNN:** We extract features from the last layer of the "conv4" block with stride size 16 pixels. Feature maps are cropped and resized to 14×14 then max-pooled to 7×7.
- **R-FCN:** We extract features from "block3" layer whose stride size is 16 pixels.
- **SSD:** We use the feature map from the last layer of the "conv4" block, which has a stride size of 16 pixels. Five additional convolutional layers with decaying spatial resolution are appended, with depths 512, 512, 256, 256, 128, as described in [30].

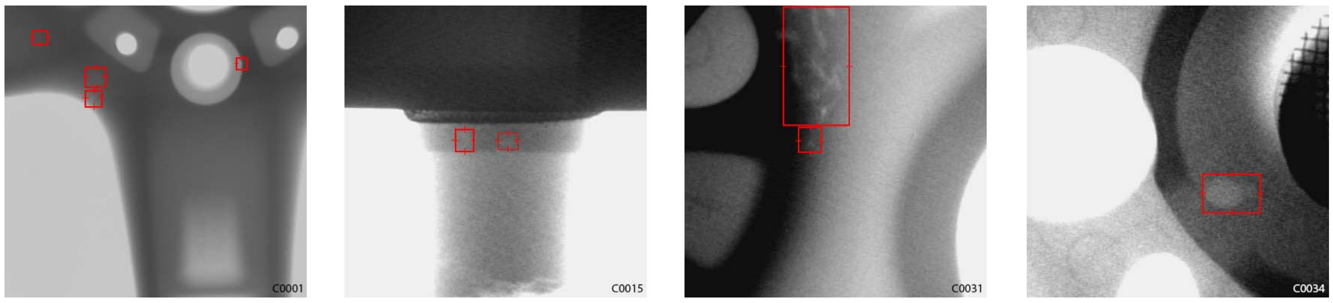

Fig. 3. Examples of X-ray images in the GDXray Castings dataset. The red boxes show the ground-truth labels for casting defects

VI. Implementation Details

Transfer learning is used to reduce the total training time and improve the accuracy of the trained models. In our early experiments, we found it was difficult to train the object localization networks solely on the GDXray dataset. The main issue was that the localization models tended to overfit the training dataset, without learning any meaningful features. As a remedy, we initialize the localization model feature extractors using weights from feature extractors that were trained on the ImageNet dataset. In this way, the top layers of the localization network are already initialized to extract image features before we start training on the GDXray dataset. We then pretrain each localization model on the Microsoft Common Objects in Context (COCO) dataset [39]. When pretraining the model, we adjust the learning rates according to the schedule outlined in [30]. Training on the relatively large COCO dataset ensures that each model is initialized to localize common objects before it is trained to localize defects. Finally, we fine-tune the localization models on the GDXray Castings dataset.

The GDXray Casting 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. Fig. 3 shows a random collection of images from the GDXray Casting dataset. The images in the dataset were randomly split into a training and testing set using an 80/20 split. We publicly release the list of images in the training and test set to facilitate fair comparison in future work [40].

When training the models, we choose to exclude images without defects, as we found this makes the training process much more stable. With non-defective samples excluded, the training set contains a total of 2308 images with 2334 annotated defects.

Training is completed on the Google Cloud Machine Learning Engine using a cluster with 5 worker nodes and 3 parameter servers [41]. Asynchronous stochastic gradient descent with momentum is used to train each model on the distributed hardware. Each worker node performs graph calculations on a NVIDIA Tesla Kepler K80 GPU.

Training the sliding classification architecture was completed in less than an hour. Fine-tuning the weights for the larger object localization architectures took up to 6 hours per model. The training loss for the fine-tuning process is shown in Fig. 4.

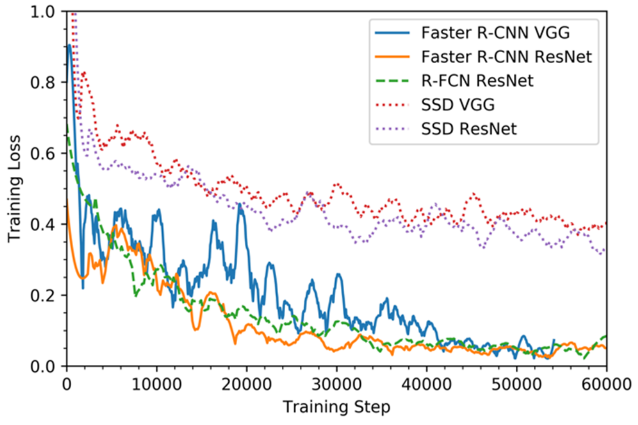

Fig 4. Training loss (smoothed) for the object localization networks, during the fine-tuning process. Note that the loss function for the SSD network varies from that of the R-FCN and Faster R-CNN networks, so the relative magnitudes of loss is not relevant.

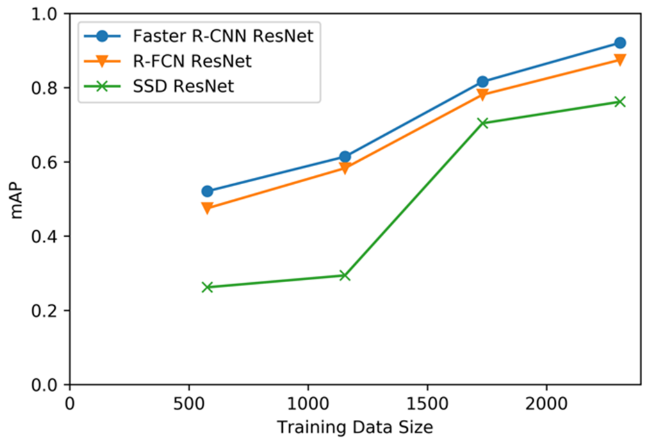

Fig. 5. Mean average precision (mAP) of each object detection network on the test set, given different sized training sets.

VII. Experiments

We conduct a number of experiments with the object localization models. The accuracy of each model on the test set is compared using the mean of average precision (mAP) [42]. We use the intersection over union metric (IoU) 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:

$$a_o = \frac{\mathrm{area}(\mathrm{B_p} \cap \mathrm{B_{gt}})}{\mathrm{area}(\mathrm{B_p} \cup \mathrm{B_{gt}})}\ , \tag{3}$$

where $\mathrm{B_p} \cap \mathrm{B_{gt}}$ denotes the intersection of the predicted and ground truth bounding boxes and $\mathrm{B_p} \cup \mathrm{B_{gt}}$ denotes their union.

The mAP for each architecture is shown in Table III. The Faster R-CNN architecture achieves a very high mAP of 0.921 with the ResNet feature extractor. The mAP of the SSD architecture is considerably lower than the Faster R-CNN and R-FCN architectures, regardless of the feature extractor. This is unsurprising, as the SSD architecture was designed to prioritize evaluation speed over classification accuracy [15].

In general, the defect localization models using the ResNet feature extractor performed better than those using the VGG feature extractor. This is not surprising, as the ResNet architecture has been shown to outperform the VGG architecture on a number of different tasks [17, 29]. However, most of these tasks involve identifying common objects such as cars, planes and bicycles. Additionally, the ResNet-101 feature extractor has a much larger number of layers than the VGG-16 feature extractor, which might not be beneficial in the defect localization task. For these reasons, we were unsure whether the defect localization models with the ResNet feature extraction architecture would yield better results than the models with the VGG feature extraction architecture.

The evaluation speed of each architecture is also measured. The models are evaluated on a 2.2 GHz Intel Xeon E5 virtual machine with 8 virtual CPU cores, 32 GB RAM, and a single NVIDIA Tesla Kepler K80 GPU. The models are evaluated with the GPU being enabled and disabled. Every image in the testing data set is processed individually (no batching). The average evaluation time per image in the testing set is reported. Both the image preprocessing and post-processing are included in the reported evaluation time.

Table III shows that the evaluation time varies significantly between the networks. The SSD VGG-16 network is the fastest with an evaluation time of 0.088 seconds/image using the CPU and 0.025 seconds/image when using the GPU. The Faster R-CNN ResNet-101 architecture is the slowest, requiring 0.512 seconds/image when using GPU. Interestingly, the evaluation time seems to be inversely correlated with the mAP, showing the speed/accuracy tradeoff.

The localization accuracy of the sliding window method is significantly lower than that of the other object detection architectures. At a high level, this demonstrates the significant amount of progress that has been made in object localization over the last few years. However, there are a number of practical reasons why the performance of the sliding window method is so low. Firstly, the receptive field of the sliding window method is 32 × 32 pixels; that is, the network must generate a class score based on a 32 × 32 pixel image tile. Therefore, it is unlikely that the sliding window method can classify defects that are larger than this receptive field. Secondly, we use a naïve thresholding method to convert the defect heat-map into bounding box predictions. We hypothesize that this method is suboptimal, especially when compared to the way that the state-of-the-art object detectors are trained.

As with many deep learning tasks, it takes a large amount of labelled data to train an accurate classifier. We train the best performing classifiers with different amounts of training data, and observe the performance of each classifier. Fig. 5 shows how the amount of training data affects the accuracy of the classifier. We notice that the accuracy improves significantly when the size of the training dataset is increased from ~1100 to 2308 images. Extrapolating from Fig. 5 suggests that a higher mAP could be achieved with a larger training dataset.

TABLE III. Comparison of the accuracy and performance of each model on the casting defect localization task

Method	***Evaluation time / image using CPU*** **[s]**	***Evaluation time / image using GPU*** **[s]**	***mAP***
Sliding window method	2.231	0.231	0.461
Faster R-CNN VGG-16	7.291	0.438	0.865
Faster R-CNN ResNet-101	9.319	0.512	**0.921**
R-FCN ResNet-101	3.721	0.375	0.875
SSD VGG-16	**0.088**	**0.025**	0.697
SSD ResNet-101	0.141	0.051	0.762

VIII. Future Work

The object detection models described in this work are accurate enough and can be evaluated fast enough to be useful in a real manufacturing setting. However, the training process for these models 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. It would also be interesting to apply the object detection models from this work to other defect detection tasks, such as welding defect detection. Finally, the models developed in this work could be adapted to detect casting defects in three-dimensional X-ray computed tomography images.

IX. Conclusion

In this work, we have studied how several state-of-the-art object detectors can be used to localize casting defects in the GDXray casting dataset. By decoupling the feature extraction layer from the object detection architecture, we could evaluate each object detection architecture with different feature extractors. Using an adapted version of the Faster R-CNN architecture, we were able to achieve a mAP of 0.921 on the testing dataset. This represents an extremely strong result, especially given the inherit difficultly of defect localization. The results show that there is a tradeoff between localization accuracy and inference time, with the Faster R-CNN models taking much longer to generate predictions than the other models.

To the best of our knowledge, this work represents the first attempt at casting defect localization with the GDXray dataset. We hope that the results presented here will serve as a benchmark for future work in casting defect localization.

Acknowledgment

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.

Disclaimer

References

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

[2] R. Rajkolhe and J. Khan, “Defects, causes and their remedies in casting process: A review,” *International Journal of Research in Advent Technology*, vol. 2, no. 3, pp. 375–383, 2014.

[3] S. Ghorai, A. Mukherjee, M. Gangadaran, and P. K. Dutta, “Automatic defect detection on hot-rolled flat steel products,” *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 3, pp. 612–621, 2013.

[4] I. Baillie, P. Griffith, X. Jian, and S. Dixon, “Implementing an ultrasonic inspection system to find surface and internal defects in hot, moving steel using EMATs,” *Insight-Non-Destructive Testing and Condition Monitoring*, vol. 49, no. 2, pp. 87–92, 2007.

[5] M. Lovejoy, *Magnetic particle inspection: a practical guide*. Springer Science & Business Media, 2012.

[6] J.-Y. Buffiere, S. Savelli, P.-H. Jouneau, E. Maire, and R. Fougeres, “Experimental study of porosity and its relation to fatigue mechanisms of model Al–Si7–Mg0. 3 cast Al alloys,” *Materials Science and Engineering: A*, vol. 316, no. 1, pp. 115–126, 2001.

[7] E. Masad, V. Jandhyala, N. Dasgupta, N. Somadevan, and N. Shashidhar, “Characterization of air void distribution in asphalt mixes using X-ray computed tomography,” *Journal of materials in civil engineering*, vol. 14, no. 2, pp. 122–129, 2002.

[8] X. Li, S. K. Tso, X.-P. Guan, and Q. Huang, “Improving automatic detection of defects in castings by applying wavelet technique,” *IEEE Transactions on Industrial Electronics*, vol. 53, no. 6, pp. 1927–1934, 2006.

[9] D. Mery, V. Riffo, U. Zscherpel, G. Mondragón, I. Lillo, I. Zuccar, H. Lobel, and M. Carrasco, “GDXray: The database of X-ray images for nondestructive testing,” *Journal of Nondestructive Evaluation*, vol. 34, no. 4, p. 42, Nov. 2015.

[10] C. H. Lampert, M. B. Blaschko, and T. Hofmann, “Beyond sliding windows: Object localization by efficient subwindow search,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1–8.

[11] H. Rathod and P. Maniar, “Prediction and investigation of shrinkage porosity defect in sand casting process - A review,” 2016.

[12] A. Sata and B. Ravi, “Bayesian inference-based investment-casting defect analysis system for industrial application,” *The International Journal of Advanced Manufacturing Technology*, pp. 1–15, 2017.

[13] O. Russakovsky *et al.*, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[14] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” *European conference on computer vision*, pp. 21--37, 2016.

[16] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[17] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object detection via region-based fully convolutional networks,” in *Advances in neural information processing systems*, 2016, pp. 379–387.

[19] A. Gayer, A. Saya, and A. Shiloh, “Automatic recognition of welding defects in real-time radiography,” *NDT international*, vol. 23, no. 3, pp. 131–136, 1990.

[20] W. Daum, P. Rose, and H. Heidt, “Automatic recognition of weld defects in X-ray-inspection,” *Materialpruefung*, vol. 28, no. 6, pp. 177–180, 1986.

[21] J. Munro, R. McNulty, W. Nuding, H. Busse, H. Wiacker, R. Link, K. Sauerwein, and R. Grimm, “Weld inspection by real-time radioscopy,” *NDT and E International*, vol. 3, no. 29, p. 191, 1996.

[22] D. Mery, T. Jaeger, and D. Filbert, “A review of methods for automated recognition of casting defects,” *INSIGHT-WIGSTON THEN NORTHAMPTON-*, vol. 44, no. 7, pp. 428–436, 2002.

[23] D. S. MacKenzie and G. E. Totten, *Analytical characterization of aluminum, steel, and superalloys*. CRC press, 2005.

[24] Y. Tang, X. Zhang, X. Li, and X. Guan, “Application of a new image segmentation method to detection of defects in castings,” *The International Journal of Advanced Manufacturing Technology*, vol. 43, no. 5–6, pp. 431–439, 2009.

[25] T. Wiatowski and H. Bölcskei, “A mathematical theory of deep convolutional neural networks for feature extraction,” *arXiv preprint arXiv:1512.06293*, 2015.

[26] A. Kehoe and G. Parker, “An intelligent knowledge based approach for the automated radiographic inspection of castings,” *NDT & E International*, vol. 25, no. 1, pp. 23–36, 1992.

[27] V. Lashkia, “Defect detection in X-ray images using fuzzy reasoning,” *Image and vision computing*, vol. 19, no. 5, pp. 261–269, 2001.

[28] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[29] A. Buja, W. Stuetzle, and Y. Shen, “Loss functions for binary class probability estimation and classification: Structure and applications,” *Working draft, November*, 2005.

[30] J. Huang *et al.*, “Speed/accuracy trade-offs for modern convolutional object detectors,” *arXiv preprint arXiv:1611.10012*, 2016.

[31] J. E. Gentle, *Numerical linear algebra for applications in statistics*. Springer Science & Business Media, 2012.

[32] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[33] A. Neubeck and L. Van Gool, “Efficient non-maximum suppression,” in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 2006, vol. 3, pp. 850–855.

[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[35] D. Mery and C. Arteta, “Automatic defect recognition in X-ray testing using computer vision,” in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, 2017, pp. 1026–1035.

[36] N. Buduma and N. Locascio, *Fundamentals of Deep Learning: Designing Next-generation Machine Intelligence Algorithms*. O’Reilly Media, Inc., 2017.

[37] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[38] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[39] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in *European conference on computer vision*, 2014, pp. 740–755.

[40] M. Ferguson, “Casting defect localization source code,” 12-Sep-2017. [Online]. Available: https://github.com/maxkferg/casting-defect-detection. [Accessed: 12-Sep-2017].

[41] Google Inc., “Google Cloud Machine Learning Engine,” 29-Sep-2016. [Online]. Available: https://cloud.google.com/ml-engine/. [Accessed: 19-Oct-2017].

[42] G. Salton and M. J. McGill, “Introduction to modern information retrieval,” 1986.

Appendix: Feature Extraction Network Architectures

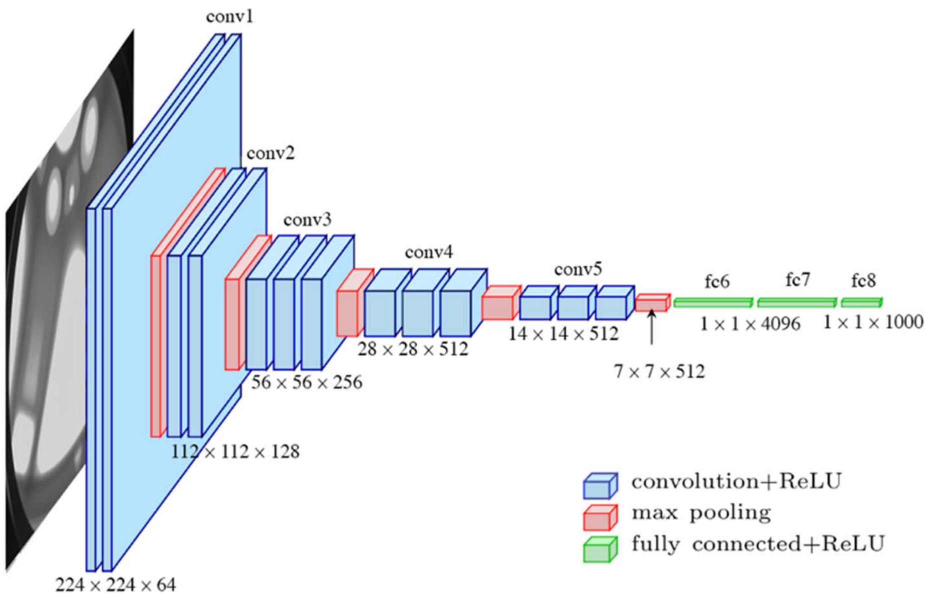

Fig. A1. The standard VGG-16 network architecture as proposed in [32]. Note that only layers "conv1" to "fc7" are used in the feature extractor.

TABLE A1. ResNet-101 Architecture as proposed in [17]. Only layers "conv1" to "conv4_x" are used in the feature extractor

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 $\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$56 \times 56 \times 256$
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	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$14 \times 14 \times 1024$
conv5_x	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$7 \times 7 \times 2048$
output	average pool, fully connected, softmax	$1 \times 1 \times 1000$