An Assistive Learning Workflow on Annotating Images for Object Detection

Vivian Wen Hui Wong
Engineering Informatics Group
Civil and Environmental Engineering
Stanford University
Stanford, United States
vwwong3@stanford.edu

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

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

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

***Abstract*—We present an end-to-end workflow to generate annotated image datasets for object detection semi-automatically, thereby reducing manual annotation need. With this workflow, which we call assistive learning, we are able to reduce manual annotation time on two experimental datasets by approximately 80%. The experimental results of this work show three contributions of the assistive learning workflow: (1) Savings on human annotation time; (2) generalizability to variable dataset sizes, domains and convolutional neural network (CNN) models; and (3) faster CNN training with limited amount of labeled data using a novel contextual sampling method, thereby a reduction in human workload early on in the assistive learning process. In addition, we wrap the workflow in an interactive annotation interface, allowing annotators without any machine learning experience to speed up the annotation process for training the CNN models.**

Keywords—data annotation workflow, computer vision, object detection, smart manufacturing, vehicle detection

I. INTRODUCTION

Object detection is one of the fundamental tasks in computer vision. This task is often useful for industrial applications in a variety of domains, such as medical imaging [1], agriculture [2] and robotics [3]. State-of-the-art object detectors are built with convolutional neural networks (CNNs). With well-built CNN architectures, object detectors can achieve excellent accuracies [4]. However, to obtain a high performance, object detectors must be trained with a vast number of training images. The images are typically labelled with bounding boxes; the process is laborious and time consuming to draw [5]. Given a diverse set of tasks that we would like to use CNNs for, being able to quickly build domain-specific datasets and models is important. There is therefore a need for a faster and more automatic annotation process.

Many existing works have been reported on methods that can reduce data-labeling efforts, including transfer learning [6], semi-supervised learning [7], and weakly supervised learning [8]. However, most of these methods still require a certain amount of training data specific to the problem being solved. Data scientists usually still have to build their own dataset to fine-tune their models. Obtaining ground truth labels for these custom datasets remain crucial and time-consuming.

Another notable technique to reduce training time for building a CNN model is active learning, where the model selects the most informative data points based on certain criteria, then inquires a human annotator to obtain labels of those points [9]. However, existing active learning algorithms face limitations. For instance, most criteria that determine informativeness can only be computed with classification problems and require extra CNN training and inference time to obtain. These limitations are further discussed in Section II.A. With the biggest drawback of active learning being the cost of computation, there is currently no practical, end-to-end workflow that uses active learning algorithms to sample image datasets for object detection.

On the other hand, works have been done to accelerate the obtainment of annotated data. Crowd-sourcing has been proposed to combine the efforts of many annotators [5]. This is, however, still costly when building large, problem-specific datasets that require expert labelers. Self-training, a technique for semi-supervised learning, has been suggested, where a model trained with a small amount of labeled data is employed to generate annotations for the remaining unlabeled data [10]. Most existing self-training works, as discussed later in Section II.B, suffer from the tradeoff between the workload of manual labeling and the accuracy of the predicted labels. The tradeoff is that it would cost more labor in the first place to get a more accurate model, but if we reduce the number of expensive manual annotations, the model trained can be inaccurate, and the generated annotations are of low quality.

In this work, we have developed an assistive learning workflow, which builds upon techniques of active learning and self-training. In active learning, the model samples the most informative data points and sends them to the annotator. In self-training, the model is first trained with partially labeled data, then used to annotate the remaining data. In the proposed assistive learning workflow, the model not only combines the behavior of learners in both techniques, but can also learn from the labeled instances and therefore gradually improve the

accuracy of predicted labels. This process is iterated through a feedback loop involving the annotator and the model.

This paper describes an end-to-end workflow called assistive learning that consists of a human annotator and a machine annotator cooperating in a feedback loop to annotate images in an object detection dataset. Our work shows that the assistive learning workflow is able to generalize well across object detection models and domains, and to overcome the limitation of labor-accuracy tradeoff. In addition, users can access the workflow using an intuitive user interface (UI), which allows visualization and generation of annotations. To demonstrate the robustness of assistive learning, we conduct experiments on two datasets with two CNN architectures, namely YOLOv3 [11] and Mask R-CNN [12]. The end-to-end workflow reduces the time of annotation by 79.4% and 83.1% in the two experimental datasets. Furthermore, the system is designed such that assistive learning can be utilized without the annotators having to concern themselves with the technical aspects of machine learning.

The rest of the paper is organized as follows. Section II provides an overview of related works. Section III presents a brief introduction to object detection and the CNN architectures deployed in our experiments. Section IV gives the details of the assistive learning workflow. Section V presents the experimental results. Section VI discusses the implications of the results, and, finally, Section VII briefly concludes our work.

II. Related Works

This study relates closely to the existing works on active learning and self-training for object detection. This section discusses how assistive learning leverages these works and the points of departure.

A. Active Learning

The key idea of active learning is the sampling of the most informative data point and querying it to an annotator. Active learning has been shown to reduce labeling costs on image classification tasks [13,14]. Several strategies have been formulated to evaluate the informativeness of a data point.

The most popular active learning sampling strategy for object detection is to use uncertainty indicators as measures of informativeness. The key to uncertainty-based sampling is to first calculate a degree of uncertainty of classification labels using, for example, entropy measure, and then sample the instance with the highest uncertainty. For example, uncertainty-based sampling strategies have been used for object detection tasks with shallow machine learning models [15,16,17]. Other works have applied uncertainty-based sampling to deep neural network architectures [18,19]. The works mentioned above, though achieve good results in many scenarios, sample instances solely based on the classification labels of one or two of the most uncertain detections. More information can be captured by aggregating the scores from each detection to calculate the total uncertainty of one image [20]. However, the method on deep CNN architectures still suffers from two drawbacks: First, the method does not account for the representation of the entire dataset [21]. Second, it has been shown that deep CNN models can fit even random noise labeling with high confidence, thereby reducing the effectiveness of uncertainty calculations [22]. In addition to uncertainty-based sampling, details of other sampling strategies in active learning, such as Query-By-Committee, can be found in [9].

The methods mentioned above mostly evaluate their trained model on every unlabeled instance, then calculate their measure of informativeness based on the outputted classification scores. This is impractical in the annotation scenario, because having a deep CNN model to evaluate every unlabeled image would be costly in both time and computational resources with many unlabeled images. Furthermore, with the most labor-intensive part of annotation being the accurate placement of bounding boxes, a sampling criterion computed with classification scores would not seem to be a natural and practical approach, and do not generalize to single-class object detection datasets.

Besides the above-mentioned active learning methods that solely use classification scores to determine informativeness, it has been proposed to use an intermediate CNN layer as a descriptor function to represent all images in the dataset [23]. The unlabeled image whose representation has the highest average Euclidean distance to all labeled image representations is sampled. This method, although theoretically feasible for CNN architectures and single-class datasets, is computationally inefficient, since every sampling instance requires a new round of training and evaluation with the CNN model. Our sampling method considers not only the average Euclidean distances of unlabeled images, but also some contextual criteria. The contextual sampling strategy is more computationally efficient as it does not require running machine learning models against every image. Furthermore, our method does not rely on the prediction of classification scores to sample the unlabeled images.

B. Self-Training

Many works have used a trained model to complete parts of the annotation process. A model trained with a limited number of labeled images is used to predict labels for a set of either unlabeled or weakly labeled images [24,25].

Some works, in addition to self-training, have involved humans in the process of correcting predicted labels. A Polygon-RNN tool has been introduced to allow the users to correct the polygon segmentation masks being generated by a deep CNN model [26,27]. Hand correction of masks can also be incorporated, and the model is iteratively retrained with corrected labels for medical image analysis [28].

The drawbacks of self-training works mentioned above are that they either do not cover the full end-to-end workflow or focus only on specific datasets. Our work leverages the level of human-machine interaction in existing self-training works by feeding the corrected data back to the model to continuously improve its accuracy throughout the annotation process.

III. Object Detection

Object detection is the task of generating a bounding box around the object of interest [29]. Different object detectors have been reported using machine learning methods [30,31,32]. Nowadays, CNNs (e.g. AlexNet [33], SqueezeNet [34]) trained using deep learning technologies are the predominant approach. There are two main types of CNN architectures for object detectors: one-stage and two-stage.

In this work, we have experimented both a one-stage and a two-stage detector, each with a different dataset, in order to demonstrate that the annotation tool is able to generalize across domains. We implement the one-stage YOLOv3 [11] architecture on a vehicle detection dataset [35], and the two-stage Mask R-CNN architecture on a manufacturing casting defect dataset named GDXray [36]. These two cases were chosen due to the high model scoring accuracies achieved in [35] and [37], as well as the fact that the two datasets have very different types of images. Vehicles on an image are much larger in size than defects on metal castings. Therefore a one-stage detector's grid-based prediction approach can achieve a high accuracy for the vehicle detection problem while dispensing less computational times compared to two-stage detectors. Manufacturing defects, however, are much smaller and can be easily mistaken as noise. Therefore a region proposal stage will increase the detection accuracy. Figs. 1 and 2 illustrate the images of each dataset.

A. *One-stage Object Detectors*

CNN architectures, such as RetinaNet [38], SSD [39] and YOLO [40], are one-stage object detectors whose single task is to directly predict confidence scores for classifying and constructing bounding boxes that surround the objects of interest. One-stage detectors have been shown to train faster than two-stage detectors [11,39].

A popular one-stage object detector is YOLOv3, which is one of the two refined models of YOLO, the first one-stage deep learning object detector published in 2015 [11,40,41]. YOLOv3 has been demonstrated to give a good balance of speed and accuracy in [35] and is adopted as an illustrative implementation of the assistive learning workflow for the vehicle detection dataset.

B. *Two-stage Object Detectors*

Other CNN architectures such as R-CNN [42], Fast R-CNN [43] and Mask R-CNN [12] are two-stage detectors, which consist of an intermediate stage that generates region proposals. Two-stage detectors are able to perform with very high accuracies even for images with fine details but require more computing resources to train [11].

Although training speed differs for different CNN architectures, in general, training good object detectors require a massive amount of training images with ground truth labels.

IV. Assistive Learning

We propose an end-to-end assistive learning workflow. This section is organized into two subsections. In the first subsection, we introduce the underlying framework of assistive learning, and we describe the role of the human annotator and the machine annotator in detail. In the second subsection, we describe the functionalities of an interactive UI, which allow any annotator to engage in the workflow.

A. *End-to-end Workflow*

We introduce a framework that involves the collaboration between a human annotator and a machine annotator, which *assists* and *learns* from the human. Given an unlabeled dataset, the framework consists of the feedback loop shown in Fig. 3.

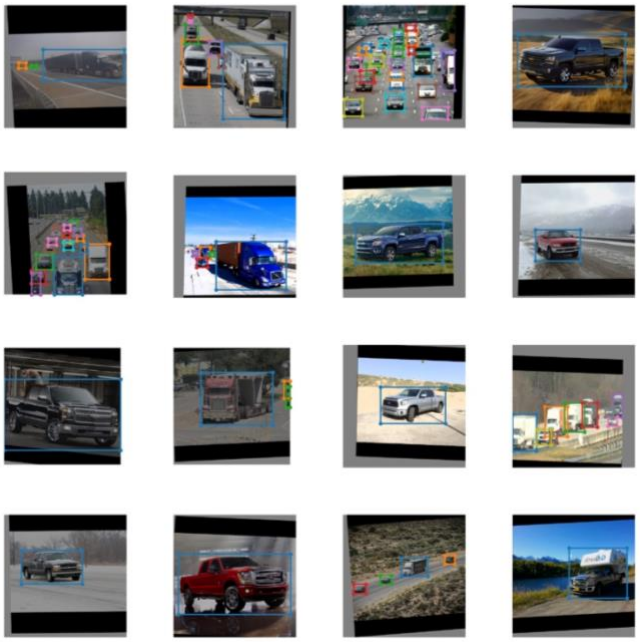

Fig. 1. Examples of labeled vehicle detection images. Vehicles are generally large and easily distinguishable visually.

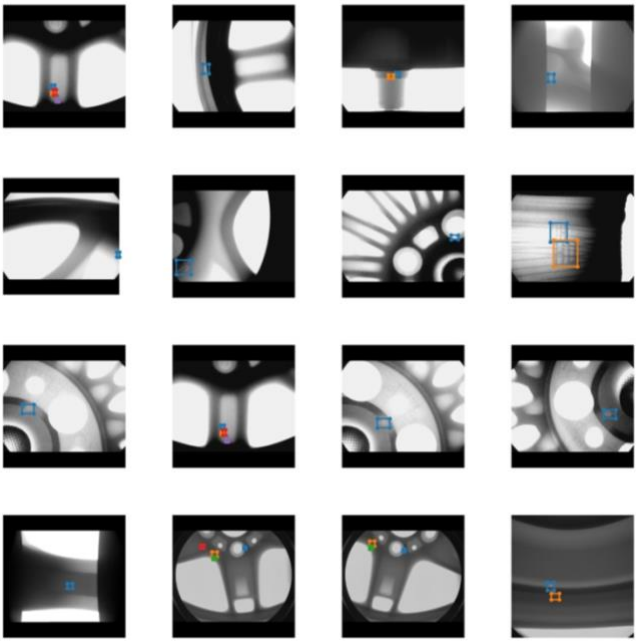

Fig. 2. Examples of labeled GDXray Casting images. Defects are small and very similar to surrounding design features.

The *human annotator* observes and annotates images suggested by the machine annotator. If the machine annotator has already annotated the image, the human annotator reviews the machine-generated labels and correct them if necessary. We consider human-generated labels as ground truth labels. The *machine annotator* consists of two modules: the object detection module and the contextual sampling module. The two modules work together to suggest and label a subsequent image, which is then sent to the human annotator for review.

1. The object detection module is a CNN model that is able to train on the ground truth labels that the human annotator has provided. It can therefore learn from more and more training

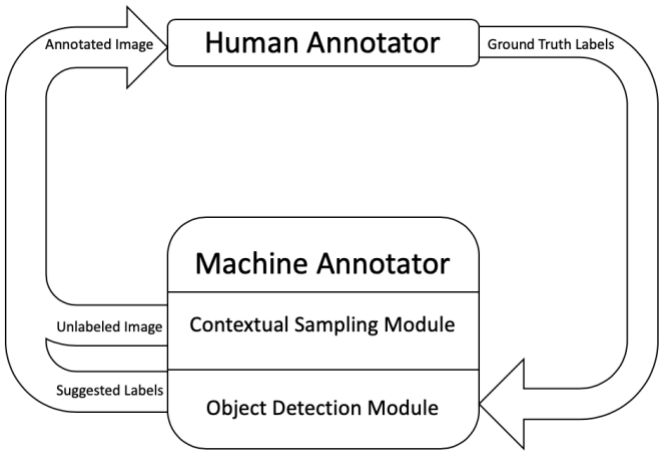

Fig. 3. Assistive learning feedback loop.

samples as we iterate through the loop as shown in Fig. 3. The object detection module annotates the sampled image outputted from the contextual sampling module and suggests the annotation to the human annotator.

2. The contextual sampling module ensures that the next unlabeled image sent to the human annotator satisfies the contextual criteria. The contextual criteria consist of two parts: uniqueness and average Euclidean distance. The criteria are applied on the raw images.

In industry applications, datasets often contain images captured from the same source or specimen. These images are therefore very similar. An example is the GDXray dataset [36], where several images are captured using the same metal casting. In contextual sampling, we avoid repeatedly sampling from the same specimen by examining an image's uniqueness. Uniqueness is defined as follows: Let's define a set of all images in a dataset, $\mathcal{X}$, consisting of labeled images $\mathcal{M}$ and unlabeled images $\mathcal{N}$, i.e. $\mathcal{X} = \mathcal{M} \cup \mathcal{N}$. Suppose there are K ($K \geq 1$) pre-defined specimens in the dataset $\mathcal{X}$, the dataset $\mathcal{X}$ can also be expressed as $\mathcal{X} = \bigcup_{i=1}^{K} X_i$, where X_i is the set of all images in specimen i. An unlabeled image x in specimen k (i.e. $x \in \mathcal{N} \cap X_k$, where $1 \leq k \leq K$) is considered unique if specimen k has the fewest number of images in the set of labeled images $\mathcal{M}$. In other words, if an unlabeled image x in specimen k ($x \in N \cap X_k$) is considered unique, then the number of images in $X_k \cap \mathcal{M}$ is fewer than or equal to that in all other $X_1, \dots, X_i, \dots, X_K \cap \mathcal{M}$, for $i \neq k$.

The other contextual sampling criterion considers the average Euclidean distance of an image. An image $x \in \mathcal{N}$ is sampled if of all the images that satisfy the uniqueness criterion, x has the highest average Euclidean distance to all images in the labeled set $\mathcal{M}$. We denote the average Euclidean distance between an image x and all images in $\mathcal{M}$ as $d_x = d(x, \mathcal{M})$.

Following the definition of the two contextual criteria, we now present the procedure of contextual sampling. We calculate the number of images in $X_i \cap \mathcal{M}$ for all $X_i \in \{X_1, \dots, X_K\}$ and select specimen k that has the fewest number of images among $X_i \cap \mathcal{M}$. For each unlabeled image $x \in \mathcal{N} \cap X_k$, we compute $d_x = d(x, \mathcal{M})$, the average Euclidean distance between x and all images in $\mathcal{M}$ and select the sampled image (denoted by $\hat{x}$) with maximum d_x among all unlabeled images x in specimen k. The pseudocode of this procedure is presented as shown in Algorithm 1. Note that for images that do not have pre-defined specimens (for example, the vehicle detection dataset), we set the number of specimens $K = 1$. Since there is only one specimen, the average Euclidean distance is the only contextual criterion.

ALGORITHM 1: DETAILED IMPLEMENTATION OF CONTEXTUAL SAMPLING TO SAMPLE THE NEXT IMAGE FROM UNLABELED IMAGES.

Input
  $\mathcal{X}$ Set of all images in the dataset
  $\mathcal{M}$ Set of all labeled images in the dataset
  $\mathcal{N}$ Set of all unlabeled images in the dataset
  K Number of specimens in the dataset
  X_i Set of all images from specimen i
Set $k \Leftarrow$ specimen with fewest images in $\mathcal{M}$
Set $d_{\hat{x}} \Leftarrow 0$
for x in $\mathcal{N} \cap X_k$ **do**
  Compute $d_x = d(x, \mathcal{M})$ ▷ average Euclidean distance between x and all images in $\mathcal{M}$
  if $d_x > d_{\hat{x}}$ **then**
    Set $d_{\hat{x}} \Leftarrow d_x$ and $\hat{x} \Leftarrow x$
  end if
end for
Output
  $\hat{x}$ Sampled image to be added to $\mathcal{M}$

B. *Functionality of User Interface*

A user interface (UI) is necessary for the level of human-machine interaction needed in our workflow. To that end, we develop an interface for the workflow and implement features to visualize, create, edit and remove annotations, to sample images, and to train the object detector.

1) Visualization and Annotation

Fig. 4 illustrates an example of using the interface to visualize bounding boxes generated by the machine annotator. Since the workflow is flexible to the object detector deployed, it can output any box predicted by the object detectors.

Using the library Fabric.js [44], the UI is able to render images and boxes. Furthermore, it is intuitive for a user to rescale bounding boxes with anchor points on the four edges and corners. Deletion of a box is done simply with the click of a key. An additional box can be drawn by pressing down the mouse on the location of one corner and releasing the mouse at the box's diagonal corner. An example of a human-reviewed annotation is illustrated in Fig. 5.

In addition to the intuitive annotating features, a visualization feature of the UI is to allow the annotator to detect with different models. In Fig. 4, we demonstrate that prediction can be done with YOLOv3 and Mask R-CNN on the same image. This is a convenient feature for debugging when switching models, or for the annotator to compare their accuracies.

2) Training

After the human annotator has reviewed and labeled a batch of images, the training function can take the batch of ground truth data and use it to train the object detector. Training will occur whenever the human annotator decides to do so. Since the

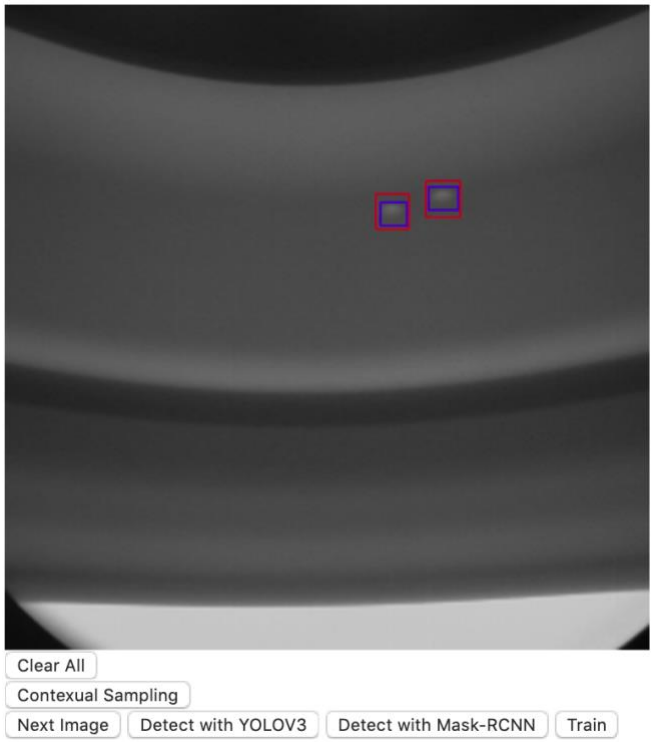

Fig. 4. A screenshot of the annotating UI. Image labeled is from the GDXray Castings dataset. Red bounding boxes are from YOLOv3 object detector. Blue bounding boxes are from Mask R-CNN object detector.

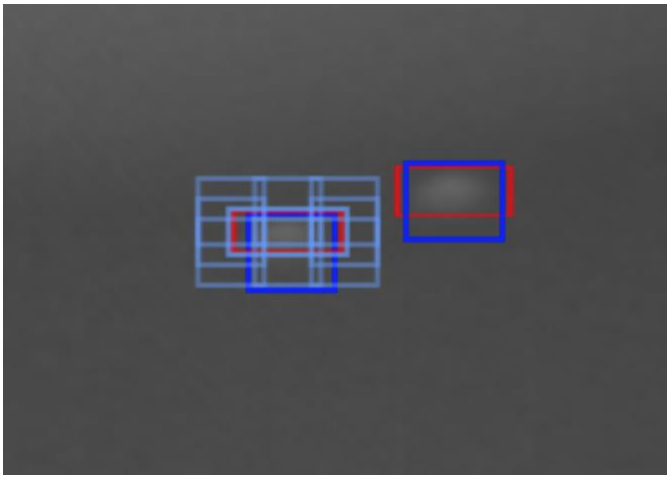

Fig. 5. An example of editing bounding boxes using anchors. With the same image in Fig. 4, the human annotator can manually edit the red bounding boxes to fit them tighter to the defects. The blue bounding boxes recommended by the object detector are kept in place as reference.

model can be trained in the background, the human annotator has the option to do manual annotations while the model trains itself. We specifically allow this parallel process because annotators are usually paid, for example, on an hourly basis, so any unnecessary waiting would be uneconomical.

The training function must be called by the human annotator by pressing the "Train" button. It will not automatically commence whenever an image has been annotated. This is a designed feature, since usually, training after every image is undesirable. Although training time differs for each model, the improved accuracy of adding only one image to the training set does not usually compensate for the cost in training time.

C. Implementation Details

The annotation interface developed is an application on the web browser built using the model-view-controller architecture [45]. As shown in Fig. 6, the frontend development of the tool is implemented with Javascript and HTML. The Javascript HTML5 canvas library Fabric.js [44] is used to render the images and bounding boxes on an HTML canvas, as well as to draw, edit or remove bounding boxes. In backend development, the tool uses Python 3.7 for contextual sampling, model training and detection. The CNN implementation is based on the PyTorch framework [46]. A controller module is written to connect the frontend to the backend features.

V. Experimental Results

This section discusses the two case studies to estimate the amount of manual annotation time reduced by assistive learning, and to show that our sampling method is able to reduce the amount of data required for a CNN model to achieve a certain level of accuracy.

A. Vehicle Detection with YOLOv3

The YOLOv3 model is trained and evaluated using a total of 3500 vehicle images from [35]. The training set consists of 2600 images, including a combination of daytime camera photos on a highway and online vehicle images. The training set has 4391 vehicle instances (2947 trucks, 344 pick-up trucks, and 1100 cars). The testing set has 900 daytime camera images captured on the highway, and consists of 1226 vehicles (566 trucks, 178 pick-up trucks, and 482 cars). Camera images under various weather and lighting conditions as well as online images are included to ensure that the machine annotator (when both detecting and sampling) can generalize, and that future camera images captured under various conditions can receive an accurate predicted label. We also augment the images by flipping them vertically and horizontally. A weight pre-trained on ImageNet data is used to initialize parameters of the model [29].

1) Annotation Time

To estimate the time taken to annotate, we use the results presented in [5], which states that the average time it takes a person to draw a bounding box is 88.0 seconds [5]. We make the assumption that editing or removing an incorrect bounding box takes half the time, or 44.0 seconds. This is in fact, quite a conservative estimation, since deleting a bounding box takes only a couple of seconds, and the adjustment of an existing box is easy when there is already a suggestion. Using the common standards in the object detection field, an intersection-over-

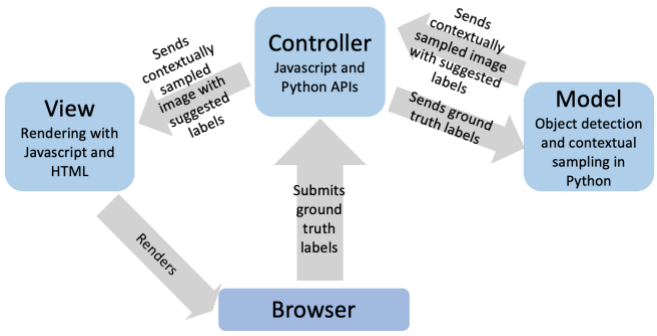

Fig. 6. Model-view-controller design of annotation interface.

union (IOU) of 0.5 or above is considered a correct prediction [47]. We therefore assume that a bounding box will need to be either edited or removed when IOU is under 0.5. Note that we do not count the training time into the calculation of manual annotation time. This is because during training, the human annotator is not required to do any work. Using our system, the human annotator is free to either take a break or annotate while the model trains in the background.

To simulate annotating the vehicle detection dataset with YOLOv3, we split the 2600 images in the training set into 10 smaller datasets as shown in Table I, with dataset sizes ranging from the smallest to the largest. The order of images is obtained using the contextual sampling method described earlier in Section IV. Since the vehicle detection dataset does not contain specimens, we use average Euclidean distance as the only contextual criterion. The first dataset is considered fully annotated by a human annotator. For the first dataset, we train it for 5 epochs with weights frozen for intermediate layers of the CNN, and another 35 epochs without holding weights fixed. We test the newly trained model on the second dataset, and compare the evaluation result to ground truth labels to compute the amount of human annotation time needed to correct inaccurate labels. The second dataset, after the simulated correction process, is now considered fully annotated. The model then trains on all fully annotated images, which now consist of the first and the second dataset, for 35 epochs. The process is repeated for all remaining datasets: For each dataset after the first, we evaluate it with the model trained with previously added fully annotated images. We then add the evaluated images to the fully annotated images, simulating the hand-correction of a human annotator, then train for 35 epochs.

Since the goal of this study is to demonstrate the effectiveness of the techniques proposed, no hyperparameter optimization is conducted. The training schedule is also not aiming to fully minimize training loss. As shown in Table I, when trained on the same training set and tested on the testing set as [35], the model achieves an mean average precision (mAP) of 0.831, which is 88.7% of the mAP of 0.937 in [35]. Our accuracy is sufficiently high for the purpose of this study, but to achieve accuracies as high as [35], one will need to tune all hyperparameters and train for more epochs.

TABLE I. TOTAL ANNOTATION TIME TO LABEL ALL IMAGES USING YOLOV3 WITH VEHICLE DETECTION DATASET. RESULTS COMPARED WITH AND WITHOUT THE USE OF MACHINE ANNOTATOR.

# of images trained	**# of images annotated with machine annotator**	**Time with human annotator alone (hours)**	**Time with assistive learning (hours)**	**% Time reduced**	**mAP**
30	5	0.54	0.48	11.4%	0.042
35	10	0.37	0.32	13.3%	0.197
45	25	1.10	0.88	20.0%	0.097
70	30	1.54	0.83	46.0%	0.343
100	100	6.43	2.86	55.5%	0.227
200	200	9.34	2.59	72.3%	0.535
400	400	15.11	3.37	77.7%	0.674
800	800	29.02	5.15	82.3%	0.691
1600	1000	37.74	6.28	83.4%	0.767
2600	900	32.24	4.68	85.5%	0.831
Total hours	-	133.42	27.44	**79.4%**	

Table I shows that when annotating the entire vehicle detection dataset, assistive learning significantly reduces manual annotation time compared to a human annotator working alone. As more labeled images are added to the training set, the machine annotator saves a higher percentage of time for the human annotator, due to improved performance of the machine annotator with more training.

2) Sampling Methods

By experimenting with YOLOv3 on the vehicle detection dataset, we compare the contextual sampling method (average Euclidean distance as the only contextual criterion) proposed in Section IV with random sampling. To conduct the experiment, we train a YOLOv3 model on various numbers of images, then compare the results with those obtained from random sampling. Pre-trained ImageNet weights are used for training. Each time the model is trained on 5 epochs with weights of intermediate layers fixed, then 35 epochs not fixed. Five identical, independent experiments are run.

Fig. 7 shows that the contextual sampling method improves the performance of the object detector when there are little labeled data. With 50 labeled images, the mean mAP from 5 independent runs is 0.32 for random sampling and 0.40 for contextual sampling. Contextual sampling improves the mean mAP by 25.0%. With 100 labeled images, contextual sampling improves the mean mAP by 23.3%, from 0.43 with random sampling to 0.53 with contextual sampling. This is because with contextual sampling, the labeled data show less similarity. The object detector can therefore generalize and fit faster. As the number of labeled images grows larger, however, the performance of the model given the two sampling methods converge.

B. Castings Defect Detection with Mask R-CNN

The Mask R-CNN model is trained and evaluated using images from the GDXray Castings dataset [36]. The Casting Series dataset contains 2727 images. Many of the images in the

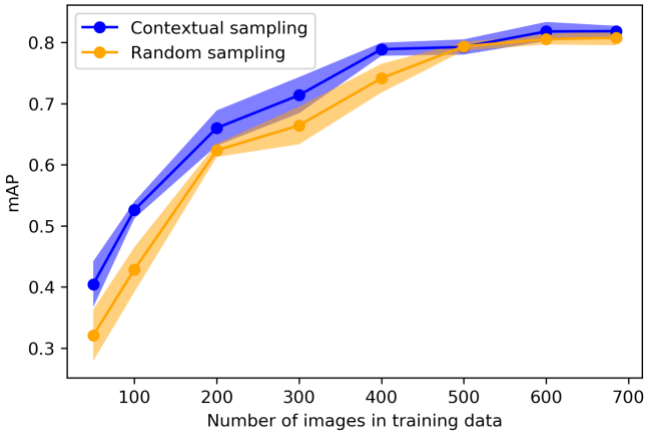

Fig. 7. Relationship between mean average precision (mAP) and number of training images, sampled randomly and contextually on the vehicle detection dataset. mAP are obtained by testing on the same test dataset. The shaded areas show the standard deviations from 5 independent runs.

TABLE II. TOTAL ANNOTATION TIME TO LABEL ALL IMAGES USING MASK R-CNN WITH GDXRAY CASTINGS DATASET. RESULTS COMPARED WITH AND WITHOUT THE USE OF MACHINE ANNOTATOR.

# of images trained	# of images annotated with machine annotator	Time with human annotator alone (hours)	Time with assistive learning (hours)	% Time reduced	mAP
30	5	0.22	0.09	61.1%	0.483
35	10	0.46	0.28	39.5%	0.150
45	25	2.47	0.31	87.6%	0.616
70	30	2.98	0.55	81.6%	0.750
100	100	8.75	1.14	87.0%	0.656
200	200	16.57	3.24	80.5%	0.793
400	285	23.44	4.11	82.5%	0.768
685	171	12.27	1.64	86.7%	0.872
Total hours	-	67.17	11.34	**83.1%**	

dataset were unlabeled or had ambiguous labels, so we chose to train on the 685 training images specified in [37,48]. Testing was conducted on the 171 images specified in [37,48]. This dataset contains a single casting defects class, but the labeled X-ray images come from 33 specimens. Images coming from the same specimen are very similar. The model parameters are initialized with weights pre-trained on COCO data [49].

1) Annotation Time

Similar to the vehicle detection problem, to simulate annotating the GDXray Castings dataset with Mask R-CNN, we split the 685 images in the training set into 8 smaller datasets, with dataset sizes ranging from the smallest to the largest. Contextual sampling is applied (with average Euclidean distance and specimen uniqueness as the contextual criteria). Due to high computing time of Mask R-CNN, we only train 5 epochs for each dataset. The first dataset includes an additional epoch for fixing the weights on all except the output layer. We evaluate every dataset after the first dataset with the model trained with previously evaluated datasets.

Table II shows the reduction of manual annotation time using a Mask R-CNN object detector to assist the annotation of GDXray Castings images. In this experiment, the models are trained on 1 epoch with fixed weights on intermediate layers followed by 5 epochs without holding weights fixed, the mAP of 0.872 for constructing the bounding boxes surrounding the defects is obtained using the same 685 training images and 171 testing images as discussed in [37,48]. The result is also compatible with the CNN model training under similar setting [37]. Although it is not conducted in this study because of extensive computational time and hyperparameter tuning required, as reported in [37], an mAP of 0.957 could be obtained when trained with 80 epochs of fixed weights on intermediate layers and another 80 epochs without holding weights fixed. That is, the trained model in this experiment achieves 91.1% of the best mAP as reported in [37]. This result is reasonable, considering that the model parameters are not optimally updated at each incremental training step. In short, the results show the workflow has the capability to greatly reduce manual annotation time.

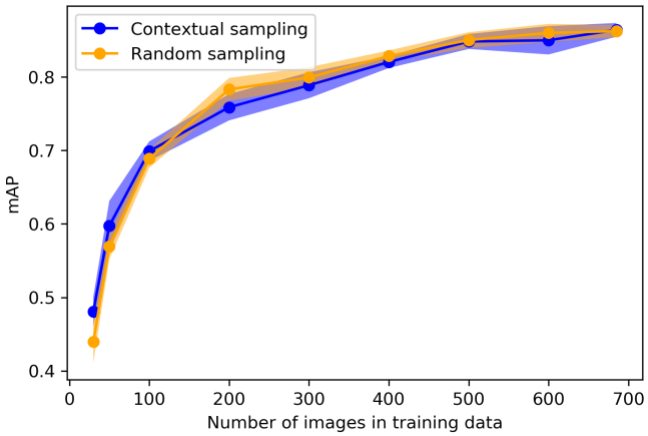

Fig 8. Relationship between mean average precision (mAP) and number of training images, sampled randomly and contextually on the GDXrays castings dataset. mAP are obtained by testing on the same test dataset. The shaded areas show the standard deviations from 5 independent runs.

2) Sampling Methods

Since the GDXray Castings dataset consists of specimens that consist of similar images, the contextual sampling method considers uniqueness and average Euclidean distance when sampling. Similar to the sampling experiments conducted in Section A, with GDXray dataset and Mask R-CNN object detector, similar results are obtained as shown in Fig. 8. When the dataset size is small, contextual sampling slightly outperforms random sampling by 9.4% mAP when trained on 30 images and 5.0% mAP when trained on 50 images.

Even though the contextual sampling method improves performance with limited labeled images, the human annotator should still take into consideration the labor-accuracy tradeoff. More upfront labor investment leads to better predicted labels. However, with the assistive learning workflow, the tradeoff can be converted to a training time-labor tradeoff. The human annotator can choose to train more frequently to increase accuracy of predicted labels, thereby reducing data-labeling efforts.

VI. DISCUSSION

In this section, we evaluate our workflow based on the experimental results presented in the previous section. Besides having shown that the workflow efficiently reduces the workload of human annotators, this section discusses additional benefits and the costs to consider when using assistive learning.

A. Generalization on Datasets

By experimenting our tool with two datasets, we were able to show that the workflow can be applied to problems from different domains due to the flexibility of the object detector and sampling strategy. In this subsection, we outline why the two datasets differ a lot in object detection, in order to illustrate the importance of cross-domain capability.

The two datasets studied are both commonly observed industrial problems in engineering yet differ mainly in the following ways:

- **Size of objects** Vehicles usually take up a larger portion of an image compared to casting defects. YOLO is not ideal for small objects since it limits the number of nearby detections [50]. Two-stage architectures that use regional proposal are

shown to be better at locating small objects [40]. Alternatively, one-stage architectures can be finetuned to target small objects [51].

- **Noisiness of background** Vehicles, which have distinct colors and shape, are normally quite distinguishable from their backgrounds. On the other hand, casting defects often look very similar to design features like holes and edges [37]. CNN architectures with high quality feature maps would be better at distinguishing such defects.
- **Classes and specimens** The vehicle detection dataset has three classes of vehicles, whereas the GDXray Castings dataset has a single class but contains similar images from the same specimens. It is therefore important to have a contextual sampling method that is applicable to either scenarios.

In summary, it is important for the human annotator to take caution when choosing a suitable object detector (i.e. the CNN architecture) for annotation, as it greatly depends on the type of objects in the custom image dataset being built. Our workflow offers flexibility to customize CNN models and datasets, and introduces a computationally efficient contextual sampling method that is applicable to a variety of datasets.

B. Costs for Consideration

Although the tool is able to reduce the need of manual labor, there are additional costs to be considered. Firstly, the cost of training time can be quite significant with more complex CNN models. With these complex models, it might cost less time overall to annotate with human annotators. For example, with a one-stage object detector like YOLOv3, training time increases linearly with the number of training images. Although training the object detector more often will allow it to learn faster and reduce labeling efforts, one must evaluate whether the overall cost in time is more expensive than hand-annotation.

The second factor is the monetary cost of computational resources. CNN model training nowadays is mostly done on GPUs, which are powerful in computation. For the tool to train effectively, it therefore must connect to either a cloud GPU or a physical GPU, which can be costly. To find the most economical way to annotate a dataset, it may worth to compare the cost of hiring human annotators versus the available (and, possibly, purchasing) computing resources.

C. Future Work

The assistive learning workflow presented in this work is able to reduce manual annotation time and can be applied to annotate and build datasets. However, the training process of the machine annotator can be computationally expensive. Future work could explore training time and its tradeoff between prediction accuracy. It would also be interesting to further evaluate the workflow on object detection with other datasets and to apply the approach to other applications, such as image segmentation, natural language processing and multimodal deep learning.

VII. Conclusion

This paper has presented an assistive learning workflow to annotate bounding boxes on images for the task of object detection. By conducting two experiments, we are able to show that by connecting a human annotator and a machine annotator with a feedback loop, we can significantly reduce the amount of manual annotation time. We also propose a novel contextual sampling method, which improves the performance of YOLOv3 and Mask R-CNN object detector when trained on a very limited number of labeled images. Together, we deploy the workflow in the backend of an intuitive user interface. The experimental results show that our workflow is able to generalize to datasets in different domains and is flexible to various object detection architectures. We also note that there exists a tradeoff between training time and hand-annotation workload.

In summary, this work proposes an end-to-end workflow that uses assistive learning to annotate images for object detection. Our estimation of savings in hand-annotation efforts is conservative yet still exceptional. We hope that this workflow can be adopted to produce valuable labeled datasets across all domains, while minimizing manual labor.

VIII. Acknowledgment and Disclaimer

This research is partially supported by the Smart Manufacturing Systems Design and Analysis Program at the National Institute of Standards and Technology (NIST), US Department of Commerce, Grant Numbers 70NANB18H193 and 70NANB19H097 awarded 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.

IX. References

[1] Z. Li, M. Dong, S. Wen, X. Hu, P. Zhou, and Z. Zeng, "CLU-CNNs: Object detection for medical images," *Neurocomputing*, vol. 350, pp. 53–59, Jul. 2019.

[2] L. Saxena and L. Armstrong, "A survey of image processing techniques for agriculture," *Proceedings of Asian Federation for Information Technology in Agriculture*, pp. 401-413, 2014.

[3] D. Xu, Q. Huang and H. Liu, "Object detection on robot operation system," *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1155-1159, 2016,

[4] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: a survey," *arXiv preprint arXiv: 1905.05055*, 2019.

[5] H. Su, J. Deng, and F-F. Li, "Crowdsourcing annotations for visual object detection," *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[6] S. J. Pan and Q. Yang, "A survey on transfer learning," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, 2010.

[7] X. Zhu, "Semi-supervised learning literature survey," Technical Report TR-1530, Univ. of Wisconsin-Madson, 2005.

[8] Z-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, no. 1, pp. 44–53, 2018.

[9] B. Settles, "Active learning literature survey", University of Wisconsin-Madison Department of Computer Sciences, Technical Report 1648, 2009.

[10] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods", *Proc. 33rd. Annual Meeting of the Association of Computational Linguistics*, pp. 189-196, 1995.

[11] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

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

[13] A. Kapoor, K. Grauman, R. Urtasun and T. Darrell, "Active learning with gaussian processes for object categorization," *2007 IEEE 11th International Conference on Computer Vision*, Rio de Janeiro, Brazil, pp. 1-8, 2007.

[14] A. J. Joshi, F. Porikli and N. Papanikolopoulos, "Multi-class active learning for image classification," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, pp. 2372-2379, 2009.

[15] Bietti, A. "Active learning for object detection on satellite images," Technical Report, Caltech, 2012. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.471.1233&rep=rep1&type=pdf (Accessed: August 1, 2019).

[16] Y. Abramson, Y. Freund, "Active learning for visual object detection," San Diego:Department of Computer Science and Engineering, University of California, 2006.

[17] S. Vijayanarasimhan and K. Grauman, "Large-scale live active learning: Training object detectors with crawled data and crowds," *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, CO, USA, pp. 1449-1456, 2011.

[18] D. Wang and Y. Shang, "A new active labeling method for deep learning," *2014 International Joint Conference on Neural Networks (IJCNN)*, Beijing, China, pp. 112-119, 2014.

[19] C. Feng, M.-Y. Liu, C.-C. Kao, T.-Y. Lee, "Deep active learning for civil infrastructure defect detection and classification", *ASCE International Workshop on Computing in Civil Engineering*, Seattle, WA, USA, pp. 298-306, 2017.

[20] C-A. Brust, C. Käding, and J. Denzler, "Active learning for deep object detection," *arXiv preprint arXiv:1809.09875*, 2019.

[21] T. He *et al*., "An active learning approach with uncertainty, representativeness, and diversity," *The Scientific World Journal*., vol. 2014, Article ID 827586, 6 pages, 2014.

[22] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *International Conference on Learning Representations*, vol. abs/1611.03530, 2017

[23] A. Smailagic *et al*., "MedAL: Deep active learning sampling method for medical image analysis," *arXiv preprint arXiv:1809.09287*, Sep. 2018.

[24] B. Adhikari, J. Peltomaki, J. Puura and H. Huttunen, "Faster bounding box annotation for object detection in indoor scenes," *2018 7th European Workshop on Visual Information Processing (EUVIP)*, Tampere, Finland, pp. 1-6, 2018.

[25] C. Rosenberg, M. Hebert and H. Schneiderman, "Semi-supervised self-training of object detection models," *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1*, Breckenridge, CO, USA, pp. 29-36, 2005.

[26] L. Castrejón, K. Kundu, R. Urtasun and S. Fidler, "Annotating object instances with a Polygon-RNN," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, pp. 4485-4493, 2017.

[27] D. Acuna, H. Ling, A. Kar and S. Fidler, "Efficient interactive annotation of segmentation datasets with Polygon-RNN++," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, pp. 859-868, 2018.

[28] B. Lutnick *et al*., "Iterative annotation to ease neural network training: Specialized machine learning in medical image analysis.," *Nature Machine Intelligence*, vol. 1, no. 2, pp. 112-119, 2019.

[29] O. Russakovsky *et al*., "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, pp. 1-42, 2015.

[30] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Kauai, HI, USA, pp. I-I, 2001.

[31] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, vol. 1, pp. 886-893, 2005.

[32] P. Felzenszwalb, D. McAllester and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," *2008 IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, pp. 1-8, 2008.

[33] M. Z. Alom *et al.*, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv preprint arXiv:1803.01164*, 2018.

[34] F. N. Iandola, *et al.*, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," *arXiv preprint arXiv:1602.07360*, 2016.

[35] R. Hou, S. Jeong, K. H. Law, and J. P. Lynch, "Reidentification of trucks in highway corridors using convolutional neural networks to link truck weights to bridge responses," *Smart Structures and Materials + Nondestructive Evaluation and Health Monitoring*, St. Louis, Missouri, USA, 2019.

[36] 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.

[37] M. Ferguson, R. Ak, Y.-T. T. Lee, and K. H. Law, "Detection and segmentation of manufacturing defects with convolutional neural networks and transfer learning," *Smart and Sustainable Manufacturing Systems*, vol. 2, no. 1, pp. 137-164, 2018.

[38] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2999-3007, 2017.

[39] 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.

[40] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp. 779-788, 2016.

[41] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, pp. 6517-6525, 2017.

[42] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, pp. 580-587, 2014.

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

[44] Printio.ru Lab, "Fabric.js." Available: http://fabricjs.com. [Accessed: 1-Jul-2019].

[45] J. Deacon, "Model-view-controller (MVC) architecture," *Computer Systems Development*, pp. 1-6, 2005.

[46] A. Paszke *et al*., "Automatic differentiation in PyTorch," in *31st Conference on Neural Information Processing Systems (NIPS)*, pp. 1-4, 2017, [online] Available: pytorch.org.

[47] A. Arnab and P. H. S. Torr, "Pixelwise instance segmentation with a dynamically instantiated network," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, pp. 879-888, 2017.

[48] M. Ferguson, R. Ak, Y. T. Lee and K. H. Law, "Automatic localization of casting defects with convolutional neural networks," *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA, pp. 1726-1735, 2017.

[49] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ra- manan, P. Dolla ́r, and C. L. Zitnick. Microsoft Coco: Common Objects in Context. *arXiv preprint arXiv:1405.0312*, 2014.

[50] Q. Peng *et al*., "Pedestrian detection for transformer substation based on gaussian mixture model and YOLO," *8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, Hangzhou, China, pp. 562-565, 2016.

[51] G. Cao *et al*., "Feature-fused SSD: fast detection for small objects," in *9th International Conference on Graphic and Image Processing*, Qingdao, China, volume 10615, page 106151E, International Society for Optics and Photonics, 2018.