Deep learning computer vision for robotic disassembly and servicing applications

Daniel P. Brogan
University of Rhode Island

Nicholas M. DiFilippo

Musa Jouaneh
University of Rhode Island, jouaneh@uri.edu

Citation/Publisher Attribution

Brogan, D. P., DiFilippino, N. M., & Jouaneh, M. K. (2021). Deep learning computer vision for robotic disassembly and servicing applications. *Array, 12*, 100094. https://doi.org/10.1016/j.array.2021.100094
Available at: https://doi.org/10.1016/j.array.2021.100094

Array 12 (2021) 100094

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

Deep learning computer vision for robotic disassembly and servicing applications

Daniel P. Brogan [a], Nicholas M. DiFilippo [b,*], Musa K. Jouaneh [a]

[a] *University of Rhode Island, Kingston RI, USA*
[b] *Johnson & Wales University, Providence RI, USA*

ARTICLE INFO

Keywords:
Screw detection
YOLO
Robotic disassembly
Robotic servicing
Deep learning

ABSTRACT

Fastener detection is a necessary step for computer vision (CV) based robotic disassembly and servicing applications. Deep learning (DL) provides a robust approach for creating CV models capable of generalizing to diverse visual environments. Such DL CV systems rely on tuning input resolution and mini-batch size parameters to fit the needs of the detection application. This paper provides a method for determining the optimal compromise between input resolution and mini-batch size to determine the highest performance for cross-recessed screw (CRS) detection while utilizing maximum graphics processing unit resources. The Tiny-You Only Look Once v2 (Tiny-YOLO v2) DL object detection system was chosen to evaluate this method. Tiny-YOLO v2 was employed to solve the specialized task of detecting CRS which are highly common in electronic devices. The method used in this paper for CRS detection is meant to lay the ground-work for multi-class fastener detection, as the method is not dependent on the type or number of object classes. An original dataset of 900 images of 12.3 MPx resolution was manually collected and annotated for training. Three additional distinct datasets of 90 images each were manually collected and annotated for testing. It was found an input resolution of 1664 x 1664 pixels paired with a mini-batch size of 16 yielded the highest average precision (AP) among the seven models tested for all three testing datasets. This model scored an AP of 92.60% on the first testing dataset, 99.20% on the second testing dataset, and 98.39% on the third testing dataset.

1. Introduction

Electronic waste (e-waste) is a rapidly growing problem. In 2019 alone, 53.6 million metric tons (Mt) of e-waste was generated globally; however, only approximately 17.4% was formally recycled. It is estimated that the annual e-waste generation will increase to 74 Mt by 2030 , increasing at almost 2 Mt per year [1]. Over half of the e-waste gathered for recycling in developed countries is sent to developing countries for processing, where health and safety regulations are not enforced, and dangerous methods for recycling e-waste are used [2]. Methods to dispose of e-waste include destructive, semi-destructive, and non-destructive disassembly methods. Destructive disassembly methods involve destroying the product through shredding or metallurgical processes (hydro or pyro) to recover valuable resources. Non-destructive methods are more useful when the disassembly goal is to recycle or reuse parts of the product. However, non-destructive disassembly typically needs to be performed by trained workers, which can be expensive because of pay and safety [3]. Robotic disassembly offers an efficient non-destructive method for disassembling e-waste. This method can be used in situations where the goal is to reuse parts or disassemble parts that contain hazardous materials in a safe manner [4, 5]. Many products, such as laptops, cellphones, and electric vehicle batteries, have outer cases held together with fasteners which must be unfastened during non-destructive disassembly. To fully realize the potential of automated disassembly, it becomes necessary to implement a computer vision (CV) system capable of automatically recognizing and locating these screws on these outer cases [6].

Robotic servicing is another critical application of CV fastener detection. The amount of space debris reached 3 million kilograms in 2013 and continues to increase. This debris poses a serious threat to the safety of future space missions [7]. NASA suggests that retired satellites should either lower their orbits and reenter or raise their orbit to a graveyard region within 25 years of mission completion to mitigate space debris buildup, but this procedure has not been globally accepted because of the significant technical challenges and

* Corresponding author.
E-mail addresses: danbrogan7@my.uri.edu (D.P. Brogan), Nicholas.DiFilippo@jwu.edu (N.M. DiFilippo), jouaneh@uri.edu (M.K. Jouaneh).
URL: https://github.com/Dan-Brogan/Cross-Recessed-Screw_Deep-Learning-Datasets (D.P. Brogan).

https://doi.org/10.1016/j.array.2021.100094
Received 21 June 2021; Received in revised form 20 August 2021; Accepted 16 September 2021
Available online 23 September 2021

cost associated with it [8]. Robotic servicing can extend the life of current satellites without the need for sending more as replacements. Therefore, there is a need to develop on-orbit satellite servicing robots to increase the longevity of artificial satellites [9]. CV can be used to aid robotic servicing missions in the detection of important mechanical features such as fasteners and docking rings. The lighting and camera orientations are highly variable in this application and deep learning provides a possible solution for making generalized predictions in this variable environment. Robots are especially useful in satellite servicing missions, where sending humans can be much more costly [10].

Both automated disassembly and servicing robots require a CV system that detects (classifies and locates) fasteners and other objects so they can be engaged by the proper tool. This paper evaluates the detection of cross-recessed screws (CRS), a common fastener used in electronics.

The goal of many object detection systems is to detect large objects such as vehicles and people [11]. Having a high input resolution is not critical when detecting large objects because they typically occupy a large portion of the frame. In this case, it is usually desirable to use lower input layer resolutions as they can allow for faster detection speed at the expense of some average precision (AP) [12]. The challenge with detecting CRS and other small objects is they usually occupy a relatively small portion of the frame due to their size. Significant visual information about the screws' appearances is lost when processed by low resolution input layers.

The training and testing of deep learning (DL) object detection systems are usually highly dependent on available graphics processing unit (GPU) resources. The number of hidden layers in a neural network (NN), the input layer resolution, and the mini-batch size are all dependent on available GPU resources. Finding the optimal balance of these three parameters for a given GPU can be challenging, especially for detecting small objects. The Tiny-You Only Look Once v2 (Tiny-YOLO v2) DL object detection system was chosen for evaluation because YOLO v2 is highly documented in literature as a widely used state-of-the-art object detection system [12–17] and its Tiny configuration allowed for more GPU resources to be allocated for higher input resolutions. Tiny-YOLO v2 was set up using Darkflow [13], a Tensorflow translation of Darknet [18]. YOLO v2 has 32 hidden layers while Tiny-YOLO v2 has 16 hidden layers. Tiny-YOLO v2 is used so higher definition input layers at reasonable mini-batch sizes can be evaluated within the constraints of one NVidia Tesla V100 GPU with 32 Gigabytes (GB) of RAM. This paper provides a method for determining the optimal compromise between input resolution and mini-batch size to determine the highest performance for CRS detection while utilizing maximum GPU resources. The method used in this paper is defined in the numbered list below and was shown to work using the aforementioned GPU for the application of CRS detection using Tiny-YOLO v2.

1. Identify the highest input resolution the given GPU can support at the default mini-batch size.
    - NOTE: If the GPU is unable to support at least high definition (1280 x 720 pixels) at the default mini-batch size, a more capable GPU may be needed.
2. Select several evenly spaced mini-batch sizes above and below the default value and identify the maximum corresponding input resolution for each.
3. Obtain a training and testing dataset of images with a resolution equal to or greater than the highest input resolution value determined in the previous step.
4. Train one model at each input resolution/mini-batch size configuration using the discrete learning rate decay method discussed in this paper.
5. Evaluate the performance of each trained model on test datasets of images indicative of the desired operating regime.
6. Choose the highest performing model for use in the field.

1.1. Related work

1.1.1. Object recognition tasks

Recently, deep learning approaches have been applied to all CV application areas such as image classification [19,20], object recognition [21–23], semantic segmentation [24], depth estimation [25,26], and human detection [27,28]. The goal of object detection is to correctly classify the object as well as predict the object's location in an image [29]. Object detection research has primarily used deep convolutional neural network (DCNN), a feed-forward type of neural network which works by trying to match features across an image using convolution functions [30]. Wei et al. [31] compared the effectiveness of image processing and deep learning techniques on the detection of railway track fastener defects for missing or broken links. Four methods were compared: classical image processing, classification based on Dense-Scale Invariant Feature Transform (SIFT), classification based on the VGG16 DCNN, and classification based on Faster Region Based Convolutional Neural Network (R-CNN). The Dense-SIFT method scored the highest mean AP (mAP) of 99.26% but had the slowest image processing time of 2.21s per image. Faster R-CNN scored the second highest mAP of 97.90% with the fastest image processing time of 0.23s per image.

K. Zhang et al. [32] applied an attention mechanism, which made their model more sensitive to foreground pixels, to a custom CNN to improve the detection of foreign objects in coal processing. Their model correctly identified 97% of the foreign objects in their test set and resized images to 416 x 416 pixels with a batch size of 4 for training. The low resolution worked well for their application because the foreign objects of concern occupied a considerable portion of the frame. The small batch size seemed to work well because there is a high variation of possible foreign objects, so it is desirable to avoid over-normalizing the model to retain its sensitivity to such variation.

Y. Zhang et al. [33] examined how well a deep learning model could identify if a bolt was loose or tightened to monitor a structure's health (e.g., a bolt that loosens over time). For testing, bolts were loosened to various heights and the model was able to detect bolts that were loosened by just 0.5 cm. Overall, the model was able to achieve a mAP of 95.03%. Wang, Li, and Zhang [34] created a construction waste recycling robot capable of detecting loose nails and screws. Their vision system used the Faster R-CNN and their model achieved a mAP of 89.10% on their testing dataset of nails and screws. Li, Zhao, and Pan [35] used Fisher criteria in a four hidden-layer network to obtain the location and classification of defects in fabrics. Their model scored a detection rate (DR) of over 90% on their testing dataset, where DR is the ratio of correctly detected defective samples.

The YOLO framework in particular has led to the development of many promising applications [5,14,36–38]. Ding et al. [36] developed a novel Unmanned Aerial Vehicle (UAV) capable of semi-automated aerial drilling and screwing. Their design used the YOLO v3 CV system to detect targets and maintain alignment in real-time during drilling and screwing processes. A custom dataset of 600 images of targets at different angles and distances was used to train the YOLO v3 model and experiments successfully demonstrated high precision aerial drilling and screwing.

Zheng et al. [37] presented a dataset of 13,000 images of UAV flight scenarios and evaluated the performance of eight different DL CV systems on UAV detection. Their study evaluated RetinaNet, Single-Shot Detector, YOLO v3, Feature Pyramid Network, Faster R-CNN, RefineDet, Grid R-CNN, and Cascade R-CNN DL CV systems. Each system was trained using 70% of the dataset and the remaining 30% was used for testing. YOLO v3 achieved an AP of 72.3%, which is between the lowest performer, RefineDet, at 69.5%, and the highest performer, Grid R-CNN, at 82.4%. They reported that among all eight systems, Grid R-CNN had the slowest image processing time at 157 ms while YOLO v3 had the fastest image processing time at 32 ms.

Chen et al. [14] used a detection pipeline consisting of Super-Resolution CNN (SRCNN) and YOLO v3 to detect electrical components from UAV inspection images. They used SRCNN to enhance the resolution of blurry images before sending them to YOLO v3 for detection and were able to achieve a mAP of 93.60% with their detection pipeline.

Yildiz and Wörgötter [5] investigated several DL methods for screw detection in hard drives. The first method they evaluated used a Hough Transform to detect circles which acted as screw candidates. The screw candidates were sent to a classifier which predicted the class and location of those candidates. Their best model used a weighted decision of the predictions made by both the InceptionV3 and Xception classifier. This model scored an AP of 80.23% on their testing dataset. They compared these results to a model they trained using YOLO v3, which scored an AP of 66.47% on their testing dataset.

1.1.2. *Transfer learning with neural networks*

Transfer learning is the method of appending training to a pre-trained model to repurpose it for the needs of the desired application. A common issue that arises in many problems is limited training data because of the cost of obtaining and annotating new training data [39]. Various applications that have used the YOLO network such as object detection [40,41] and diagnosis of medical issues [42–44] have instituted transfer learning methods.

Li et al. [16] introduced a method based on transfer learning and sample enhancement with a small number of training samples that was able to classify 87.5% of objects. They first initialized training weights using unrelated sample data from the PASCAL Visual Object Classes (VOC) dataset with Tiny-YOLO v2 then used the Tiny-YOLO v2 network to further train the data.

Transfer learning can be used to improve detection results of models. Raza and Hong [41] designed a computer vision model using YOLO v3 to monitor for fish in a marine ecosystem. They used a transfer learning method that was pre-trained on 1.2 million samples of the ImageNet dataset. By incorporating the transfer learning method as well as some other improvement techniques, they were able to increase the mAP by 4.13%. Montalbo et al. [42] developed a model that could detect three types of brain tumors and used Tiny-YOLO v4 and pre-trained weights from the COCO dataset. They achieved a mAP of 93.14% which outperformed other studies that had tried to detect brain tumors using different deep learning networks.

1.1.3. *Automatic screw detection for disassembly*

In applications such as robotic disassembly, automated screw unfastening is an important task robots can execute. Robots already perform screw fastening for assembly operations [45,46], and there have been many studies detailing the designs of robotic systems [47,48] and end-effectors [49–51] for fastening applications. In these assembly applications, when screw locations are known in advance, fixtures and compliance devices can be used to achieve proper screw alignment. When screw locations are not known in advance, as is typically the case with disassembly operations [38,52], vision systems may be used to determine screw positions [45].

Gil et al. [53] used various computer vision techniques such as Douglas–Peucker's algorithm, adaptive thresholding, Canny edge detection, and region detection with template matching to identify features such as screws and other components (covers, wires, batteries, etc.) on electronic equipment to create a robotic system to perform disassembly tasks. Bdwidi et al. also designed a workstation to automatically disassemble electric vehicle motors. They used a Microsoft Kinect sensor capable of providing depth data, feature point detectors such as the Harris detector, and then multiple optimization steps to identify screws and remove false positives. A drawback of using these types of classifiers is that they can be heavily dependent on lighting and require controlled lighting environments. Vongbonyung et al. [54,55] designed a robotic system that could learn actions and revise them to make cuts to disassemble monitors. This system was also able to deal with uncertainties that can arise during automated disassembly. The system used computer vision to automatically determine the location of screws and was able to find over 80% of them however, the authors reported a high number of false positive (82.83%) and false negative (35.78%) detections. A false positive detection would lead to redundant cutting operations and would require human intervention for disassembly to proceed.

Wegener et al. [6] proposed a concept for a human-assisted robot workstation for the disassembly of electric vehicle batteries where fastener detection was a primary task for the robot. They investigated three methods of fastener detection: using a computer-aided design (CAD) database, physically demonstrating the location of the screws, and a CV algorithm. They determined that detailed CAD databases are usually not accessible by the recycler and physical demonstrations are too time-consuming, thus making these methods impractical. The final option of using a CV algorithm was investigated using a Haar-Cascade classifier trained on positive and negative images to create a model for detecting the desired object classes. Their model was only able to correctly detect 50% of the screws in their testing dataset.

DiFilippo and Jouaneh [4] developed an automated robotic disassembly system that combined CV and force sensing to remove screws from the back of laptops. The system comprised two webcams, a Microsoft Kinect sensor, and a 3-axis cartesian robot with an actuated sensor-equipped (SE) screwdriver. Once a laptop was placed on the workspace, the overhead webcam identified circles as screw candidates using a Hough Circle Transform. The robot would then move to the locations of these circles and, using a webcam attached to the robot's end-effector, perform classical computer vision techniques to center the screw. The SE screwdriver would then test if the circle was a screw by attempting to remove it. If a screw was detected, the robot removed the screw, and if no screw was detected, the robot would move to the following circle location. This process proved to be time-consuming. By using the Soar cognitive architecture [56], screw locations could be stored in semantic memory after the first pass, thus reducing screw removal time on subsequent passes. Even so, the fastest CV time per circle was 6.5s. This paper builds upon their previous work by proposing an optimized DL Tiny-YOLO v2 based CV system that can process high-resolution images at over 3 frames per second (FPS). The image processing speed of this method is not dependent on the number of screws/screw-like objects in each image.

2. Background on tiny-YOLO v2 object detection system

Tiny-YOLO v2 is a lightweight version of YOLO v2, which is derived from the original YOLO object detection system [12]. YOLO, YOLO v2, and Tiny-YOLO v2 are high performance DL object detection systems that can be applied to real-time applications. YOLO is unique from other object detection systems in that it simultaneously predicts bounding boxes and object classes with a single NN.

Tiny-YOLO v2 has 1 input layer, 9 convolutional layers and 6 maximum pooling layers. The unmodified Tiny-YOLO v2 input layer resizes images to 416 x 416 pixels, where the output is passed to convolutional layers for feature extraction. Max pooling is used to reduce the dimensionality of the convolutional layer outputs.

YOLO divides an image into an S x S grid, where each grid cell predicts *B* bounding boxes with corresponding confidence scores. The confidence, *C*, represents the probability that an object is encompassed in a bounding box and is represented as:

$$C = Pr(Object) * IOU^{truth}_{pred} \tag{1}$$

where intersection over union (IOU) is defined as the ratio of the area of overlap of the detection and ground truth bounding boxes divided by the area formed by their union. A graphical representation of IOU is shown in Fig. 1.

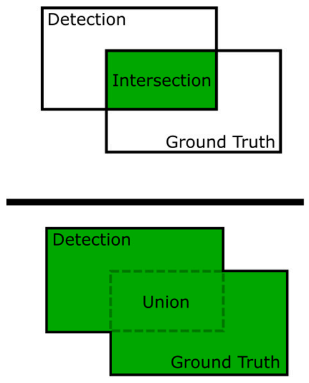


Fig. 1. Graphical representation of intersection over union metric.

Table 1
Tiny-YOLO v2 model configurations.

Model	Mini-batch size	Input resolution
A	8	2368 x 2368
B	12	1920 x 1920
C	16	1664 x 1664
D	20	1472 x 1472
E	24	1344 x 1344
F	28	1248 x 1248
G	32	1184 x 1184

3. Training method

Input resolution and mini-batch size are parameters that directly influence the performance of a deep NN. The input resolution is defined in the first layer of the Tiny-YOLO v2 network. In this layer, images of any size are accepted into the network and resized to the input resolution specified by the network's configuration. In this paper, high definition (pixel density equal to or greater than 1280 x 720) input resolutions are specified. The original aspect ratio is maintained during this process. A higher input resolution allows for the network to process higher resolution images.

At every step (gradient update), the model updates its weights based on the normalized training loss results from one mini-batch of images. The training loss reports the error between the model's predictions and the ground truth of the training dataset at the end of each step. A higher mini-batch size allows for more normalized learning whereas a smaller mini-batch size may evoke noise in the reported training loss.

Both input resolution and mini-batch size depend on the availability of GPU memory allocations. Therefore, it is desirable to determine the optimal compromise between these two parameters. A total of seven models were trained using different combinations of input resolutions and mini-batch sizes to determine the optimal training conditions. These combinations were chosen to represent a wide spread of reasonable combinations of mini-batch size and resolution from which a trend in reported AP should emerge. Table 1 shows the input resolution/mini-batch pairing for each model. The default mini-batch size for Tiny-YOLO v2 is 16 and mini-batch sizes above and below this default value were explored in evenly-spaced steps of four mini-batches from half of the default mini-batch value (8) to twice the default mini-batch value (32). Each model's mini-batch size was paired with a respective input resolution that fully utilizes the resources of the GPU. These input resolution values were identified by a trial-and-error process which finds the highest input resolution that does not give a memory error for a given batch size. The input resolution was chosen to be square to remain unbiased to one image orientation as input images may be landscape or portrait orientation. It should be noted the input resolution of Tiny-YOLO v2 can have rectangular dimensions.

All datasets were manually collected using a Google Pixel 12.3 MPx (3036 x 4048 pixels) camera to maintain uniform high resolution. Tiny-YOLO v2 automatically resizes these images down to specified input resolutions during training and testing. Thus, it is desirable to start with an image resolution that is the same or higher than the specified input resolution so that maximum visual information can be maintained.

Ground truth files containing the location and classification of screws were manually generated for each image. Tiny-YOLO v2 uses these ground truth files to train a model by associating the location and classification of screws with the respective images. The testing process also requires these ground truth images, as they are compared with detection results to determine AP.

The training dataset consists of 900 images of general electronics and hardware with embedded CRS. The objects in this dataset include laptops, computer towers, hard drives, oscilloscopes, power supplies, and other assorted hardware. Due to the diverse assortment of objects, this dataset contains many variations of CRS. These 900 images were taken in highly variable environments with various lighting conditions and distances to the object (ranging from approximately 4 to 8 inches from the surface of the objects). This dataset is intended to be highly variable as it is hypothesized this variability will improve the generalizing ability of the models. Fig. 2 shows a sample of the images included in this set.

Due to the relatively small training dataset of 900 images, a transfer learning approach similar to the one discussed in [16] is employed to avoid overfitting. Each model initializes training from the Tiny-YOLO v2 Visual Objects Classes (VOC) weights file from [18] which has been pre-trained on the VOC [11] dataset.

The generalizing ability of the model at a given training iteration is evaluated by a validation set. The validation set will be referred to as Test Set A, which contains 90 images of hardware with embedded CRS. The images in Test Set A are not present in the training dataset and are used to gauge the performance of the model throughout training. After training was completed, the final performance results from each model on Test Set A were recorded in Section 4.

The training method for each model is as follows. A discrete learning rate decay method was used to achieve the optimal AP on Test Set A. The learning rate is a parameter that dictates the amount of change applied to the model's weights after each training iteration in response to the reported error between the model prediction and the ground truth. Fig. 3 shows the learning rate progression method for each trained model. This method entails first training a model at a high learning rate of 5e-5 until a maximum attainable AP is reached on Test Set A for this learning rate. The model then continues training at a reduced learning rate of 2e-6 until a maximum attainable AP is reached on Test Set A for this reduced learning rate. To determine the maximum AP for both the high learning rate and the reduced learning rate, validation tests are performed where the loss convergence occurs that determine the AP where further training will cause the model to overfit the data. The final trained model is the result of this procedure.

Fig. 4(a) shows the overall view of training loss curves for all seven models. All loss curves closely follow the same trend, but models with higher input resolutions tend to initialize with a greater loss value. This higher loss is likely associated with there being more to learn from higher resolution images. Higher resolution images inherently contain more information, so it follows the initial training loss increases with input resolution. Models with higher resolution also generally take more steps to train; however, this trend is not followed exactly.

Fig. 4(b) shows a zoomed-in view of where the training loss for all seven models begins to converge. As mentioned earlier, models with lower batch sizes tend to evoke more noise in the reported training loss. This is represented clearly in Fig. 4(b) since Model A has the lowest batch size and shows the most noise, while Model G has the highest batch size and shows a smooth curve. All seven models consistently converge in the order of increasing initial loss values. While this appears negligible in the overall view, it is helpful to confirm this behavior in the zoomed-in view as it is expected that higher initial loss values should take longer to converge.

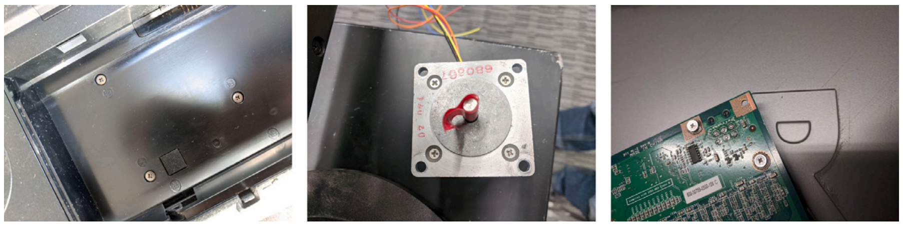


Fig. 2. Sample of images used for training each model.

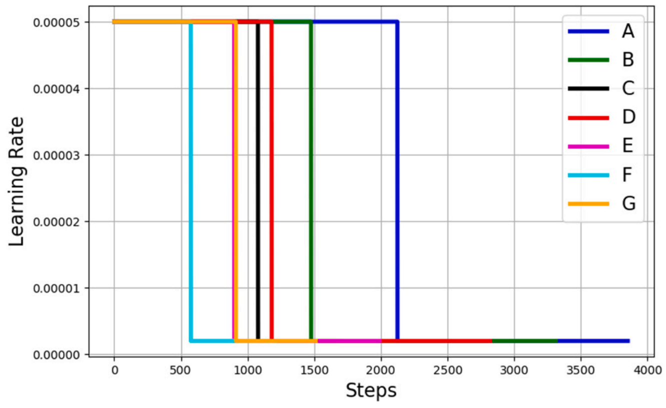


Fig. 3. Learning rate progression of each model throughout the training process.

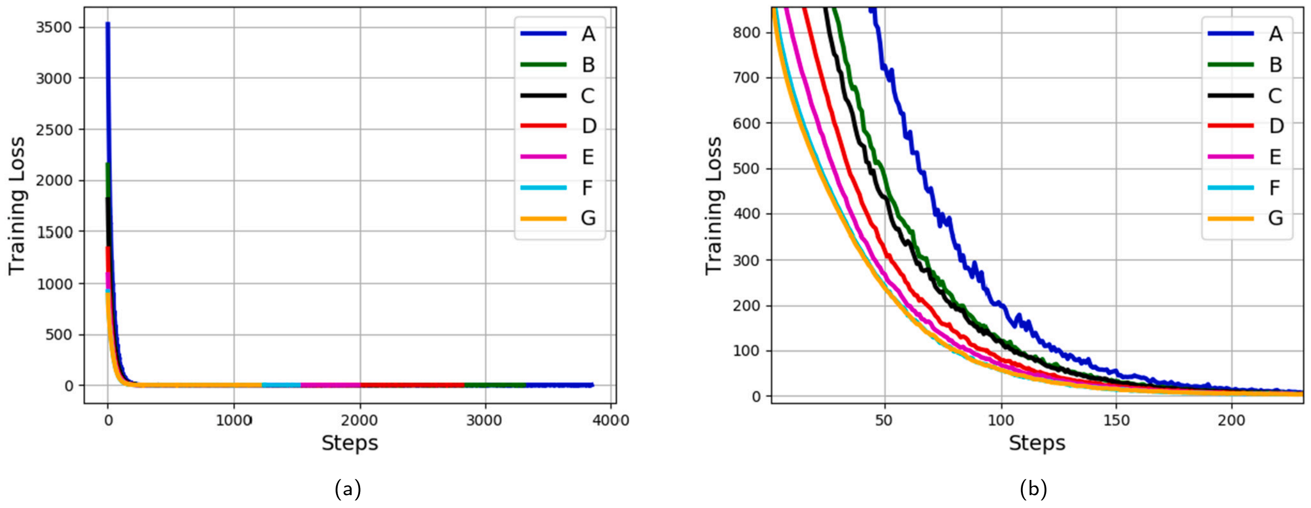


Fig. 4. Training loss reported over steps. (a) Overall view of each model throughout the training process. (b) Zoomed-in view of initial convergence for each model.

4. Testing results and discussion

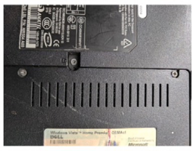
Each model was tested on three distinct datasets; Test Sets A, B, and C. The authors chose to make the visual environments (lighting, size of CRS relative to overall frame, distance, and camera angle) equally diverse between these test sets as to capture a broad representation of possible conditions that could be encountered by this vision system during nominal operation. As a result, the authors did not feel a need to do a systematic investigation of model performance as a function of the visual environment. Each testing dataset consists of 90 new images outside of the training dataset. Table 2 provides the total number of CRS, a description, and sample image for each test set.

Test Set A, which was also used for training validation, consists of images of general hardware with embedded CRS. This test set evaluates the ability of the models to make detections on new images of similar objects to those found in the training dataset.

It is useful to evaluate the models' performance when given specialized tasks they were not primarily trained to encounter. Test Sets B and C provide two different specialized tasks. Test Set B evaluates the ability of the models to perform CRS detection on laptops and Test Set C evaluates their ability to perform CRS detection on boxed electronics such as power supplies, power tools, and oscilloscopes.

The AP metric is used to evaluate the performance of each model. AP computes the area under a monotonically decreasing precision–recall curve for a single class as defined in [11]. For reference, Fig. 5 shows similar information to Table 3 but in a visual format for the precision–recall curve for Model C on Test Set A. Similar graphs can be constructed for all of the models (A–G) on all of the Test Sets (A,B,C). IOU is used to differentiate true positives (TP) from false positives (FP). A TP is defined as a prediction with the correct classification that has an IOU greater than 50%. A FP is defined as a detection with an IOU less than 50%. Python scripts developed by Cartucho, Ventura, and

Table 2
Test set descriptions.

Test Set	Total CRS	Description	Sample Image
A	164	General hardware	
B	131	Laptops	
C	200	Boxed Electronics	

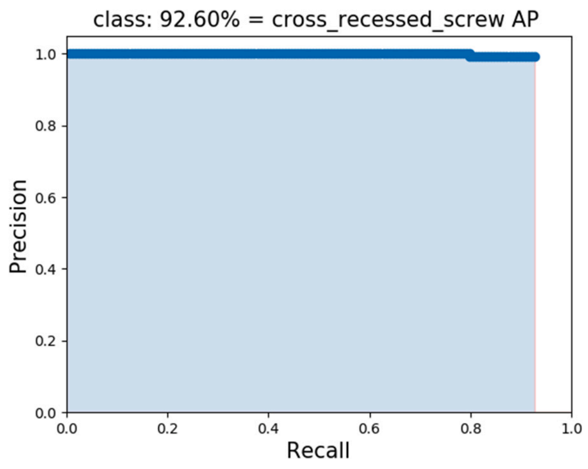


Fig. 5. Model C precision–recall curve for CRS detection on Test Set A.

Veloso [17] were used to plot the AP and generate visual overlays of the detections over ground truth bounding boxes.

Table 3 shows the AP, TP, FP, network initialization time, total prediction time, and FPS for each model on Test Sets A, B, and C. The network initialization time is the amount of time Tiny-YOLO v2 takes to set up its network for testing. The total prediction time is the amount of time spent passing the entire dataset through the network while generating detection output files for each image in sequence. FPS is defined as the number of images passed through the network divided by the total prediction time. The reported network initialization time and total prediction time are averaged results taken from five trials for each model. These results are averaged to account for the GPU's slight variation in computing times. The AP, TP, and FP will always remain constant for a given trained model as neither the model's weights nor the input image pixels change during testing. Model C scored the highest AP on all three test sets.

Table 4 shows a sample of images representative of Model C's performance on all three test sets. Detections are shown as green or red boxes labeled ''CRS", which stands for cross-recessed screw. Ground truth boxes associated with each detection are shown in blue. Green boxes represent TP and red boxes represent FP.

Test Set A contains a total of 164 screws. Model C correctly predicted 152 screws while only making one FP prediction. This shows Model C was more likely to miss a TP rather than assign a FP in Test Set A. As shown in Table 4, Model C performs exceedingly well when

Table 3
Results on Test Sets A, B, and C.

Model	Test Set A					
	AP	TP	FP	Network Init. time (s)	Total Prediction time (s)	Frames Per second
A	85.54%	142	2	10.053	49.576	1.815
B	92.44%	154	4	10.052	42.014	2.142
C	92.60%	152	1	10.047	35.507	2.535
D	90.41%	151	4	10.022	32.989	2.728
E	91.74%	153	8	10.057	30.193	2.981
F	86.97%	146	4	10.039	28.915	3.113
G	83.66%	141	10	10.047	28.720	3.134
Model	Test Set B					
	AP	TP	FP	Network Init. time (s)	Total Prediction time (s)	Frames Per second
A	90.44%	120	4	10.084	49.869	1.805
B	98.88%	130	4	10.042	41.722	2.157
C	99.20%	130	2	10.045	35.541	2.532
D	95.01%	125	2	10.066	31.892	2.822
E	93.54%	123	4	10.058	30.599	2.941
F	92.01%	121	2	10.072	28.610	3.146
G	93.50%	123	4	10.016	28.470	3.161
Model	Test Set C					
	AP	TP	FP	Network Init. time (s)	Total Prediction time (s)	Frames Per second
A	84.82%	170	1	10.090	48.901	1.840
B	90.50%	181	0	10.090	40.497	2.222
C	98.39%	197	1	10.081	34.244	2.628
D	94.42%	189	2	10.048	30.033	2.997
E	91.21%	183	7	10.048	29.393	3.062
F	90.24%	183	4	10.065	28.016	3.212
G	90.10%	181	10	10.053	27.577	3.264

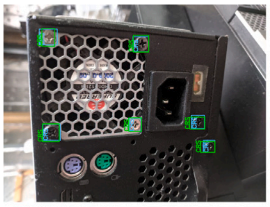
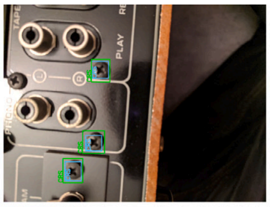
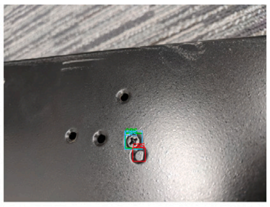
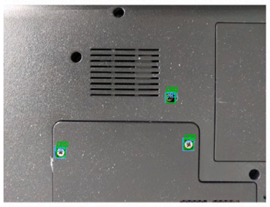
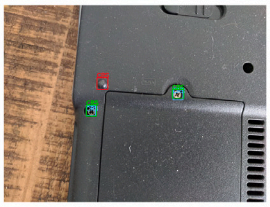
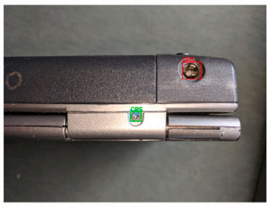
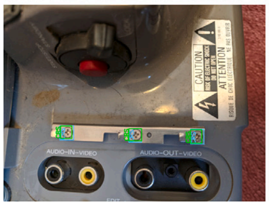
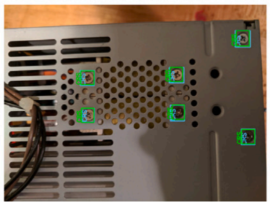
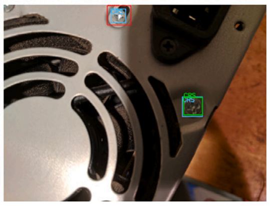
presented images with a blend of screws and screw-like objects. Grates, connectors, and holes are often screw-like in appearance and can be a source of difficulty for classical CV techniques. As shown, the approach used in this paper is robust in differentiating screws from screw-like objects. The top right picture in Table 4 shows the only FP Model C predicted in Test Set A, which is a circular indent in an electronics case. Model C scored 92.60% AP on Test Set A with an average speed of 2.535 FPS. This result reaffirms the value of using DL techniques for fastener detection as they can exhibit high performance and speed when optimized.

Test Set B contains a total of 131 screws. Model C correctly predicted 130 screws while making only 2 FP predictions. Both FP cases are shown in the center and mid-right pictures in Table 4, where the model mistook a power connector and another circular feature as a screw. Still, Model C is robust when presented with images containing holes that do not contain screws. Model C scored 99.20% AP on Test Set B with an average speed of 2.532 FPS.

Test Set C contains a total of 200 screws. Model C correctly predicted 197 screws while making only 1 FP prediction. The FP case is shown in the bottom right picture of Table 4, where the model did surround the CRS in a bounding box; however, the IOU was less than 50%, which resulted in a FP. Model C scored 98.39% AP on Test Set C with an average speed of 2.628 FPS. These results show that model C is highly robust when given the specialized task of CRS detection in boxed electronics.

Fig. 6 shows the FPS for all seven input resolutions on all three test sets. Average FPS decreases as input resolution increases for all test sets. This is expected since more computation is needed for evaluating higher resolution images. The FPS curves for Test Sets A, B, and C are nearly identical and follow the same trajectory. The minor variations between both curves can be attributed to the slight inconsistency of the GPU's processing speed. It should be noted the CV time spent using classical techniques is dependent on the number of screws and screw-like objects in an image [4,56]. Fig. 6 shows even when three different

Table 4
Model C detected output images from Test Sets A, B, and C.

Test Set	Detected Output Images		
A			
B			
C			

*Some images have been rotated 90° to better fit the table as the test sets contain both portrait & landscape images.

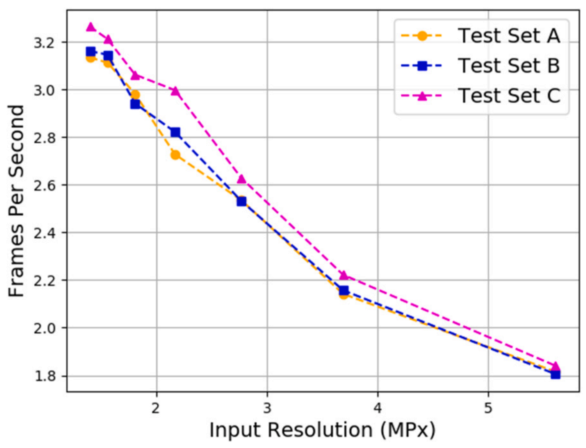


Fig. 6. Image processing speed reported as a function of input resolution for all test sets.

datasets with varying numbers of screws and screw-like objects are evaluated, the CV time is dependent almost exclusively on input resolution. This suggests an image with few screws would likely be evaluated in the same amount of time as an image with many screws.

Fig. 7a shows the AP vs. mini-batch size, Fig. 7b) shows the AP vs input resolution and Fig. 7c) shows a 3D plot of the AP scored on Test Sets A, B, and C as a function of input resolution and mini-batch size. Models scored highest on Test Set B likely because it has the smallest variety of objects. Models scored lowest on Test Set A likely because it contains the largest variety of objects. The general AP curve for all three test sets is similar, with all curves reaching optimal AP when a mini-batch size of 16 is paired with an input resolution of 2.77 MPx (1664 x 1664 pixels). This indicates the method used in this paper yields a configuration that shows consistently optimal performance across several variations of CRS detection tasks.

The results obtained in this paper show improvement over results that have been reported from previous work, either in the time to detect a screw or in the screw detection accuracy. A summary of this work compared with previous results can be found in Table 5 where the work described by this paper is referred to as Tiny-YOLO v2 (Model C). In terms of detection accuracy, Wegener et al. [6] used a Haar Cascade classifier but was only able to detect 50% of screws. Vongbunyong et al. [54,55] also used a Haar Classifier and reported being able to detect over 80% of screws [54], however they also indicated a high number of false-positive screw detections (82.32%) and false-negative screw detections (35.78%) [55]. Yildiz and Wörgötter [5] reported their custom deep learning model was able to achieve an AP of 80.23% on a testing dataset compared to the 66.47% YOLO v3 was able to do. DiFilippo and Jouaneh [56] tested multiple laptops using contour and blob detection, and the accuracy of detection was based on the color of the laptop and vision system settings. The system performed the best on lighter laptops and detected 86.7% of screws, and the fastest computer vision time was 6.5s per screw. For laptops that had darker cases, the percentage of screws that were correctly identified decreased. The work presented in this paper has a higher percentage, as the best-trained model (Model C) has an AP of 92.60% on Test Set A, 99.20% on Test Set B, and 98.39% on Test Set C. It was also faster than previous systems that reported the time it took to detect a screw [4,55], as one frame took approximately 0.4s to process.

5. Conclusions

In conclusion, fastener detection is a required step for CV based robotic disassembly and servicing applications. The use of DL for this task offers several advantages to classical CV techniques, including higher detection speed and performance. Fasteners typically occupy a small portion of an image, so it is important to use a high-resolution NN to capture maximum detail when detecting images. It is desirable to find the optimal compromise between input resolution and mini-batch size for a given NN as both parameters are dependent on available GPU resources.

This paper presents a method for determining the optimal compromise between input resolution and mini-batch size for CRS detection while utilizing maximum GPU resources. An optimal compromise for an NVidia Tesla V100 GPU with 32 GB of RAM was found with a mini-batch size of 16 and an input resolution of 1664 x 1664 pixels. At this

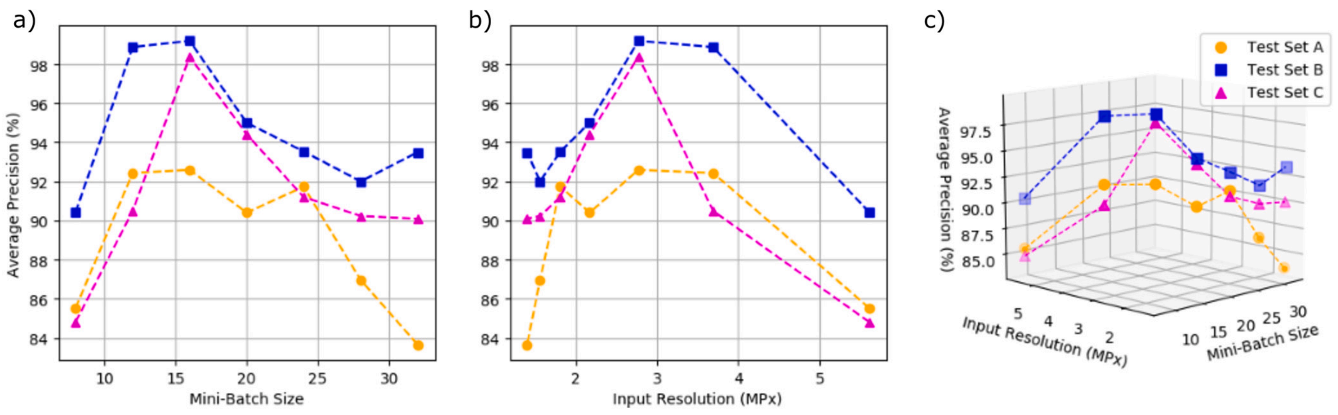


Fig. 7. (a) AP vs. mini-batch size for all test sets (b) AP vs. input resolution for all test sets (c) AP reported as a function of input resolution and mini-batch size for all test sets.

Table 5
Comparison of screw detection with previous results.

Authors	Accuracy (%)	Detection time (s)
Wegener et al. [6]	50	Not reported
Vongbunyong et al. [54,55]	>80 [54]	2.60
	82.32 (False Positive) [55]	
	35.78 (False Negative) [55]	
Yildiz and Wörgötter [5]	80.23 (custom DCNN)	Not reported
	66.47(YOLO v3)	
DiFilippo and Jouaneh [4]	86.7 (Light Laptop — Best Parameters)	6.7
Tiny-YOLO v2 (Model C)	92.6 (Test Set A)	0.4
	99.20 (Test Set B)	
	98.39 (Test Set C)	

configuration, Model C scored 92.60%, 99.20%, and 98.39% AP on Test Sets A, B, and C respectively. A limitation of the models in this paper is FPS must be sacrificed for such high input resolutions. While much faster than classical CV techniques, the fastest model in this paper ran at only around 3 FPS. These results from the best-performing model shows improvement over accuracy and detection time from previous models and approaches that have been presented in literature. Another limitation is the time spent upfront in manually creating a training dataset for the purpose of screw detection. Unless publicly available, the practitioner must generate their own training dataset for their specific application. For this reason, the authors have created a publicly available repository containing the manually generated datasets used in this paper. The repository may be accessed through the following link: https://github.com/Dan-Brogan/Cross-Recessed-Screw_Deep-Learning-Datasets.

Future work should include training a single Tiny-YOLO v2 network to detect multiple types of screws and even other useful features commonly found on electronics. Some considerations for multi-class detection include the requirement for additional training data on multiple object classes and the mAP metric should be used to evaluate performance in place of AP. It is hypothesized that the method used in this paper applies to any GPU; however, future work is needed to investigate this hypothesis. Future work should also investigate the integration of this CV system into a robotic test bed.

CRediT authorship contribution statement

Daniel P. Brogan: Methodology, Funding acquisition, Data curation, Formal analysis, Software, Visualization, Writing – original draft. **Nicholas M. DiFilippo:** Project administration, Supervision, Conceptualization, Writing – review & editing. **Musa K. Jouaneh:** Project administration, Supervision, Conceptualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The first author would like to acknowledge the NASA Rhode Island Space Grant (RISG) Consortium, USA for supporting his work on this project. The NASA RISG Consortium provided financial support independent of influencing the study design; collection, analysis, and interpretation of data; writing of the report; or in the decision to submit the article for publication. We also like to thank Harrison Decker and Indrani Mandal for providing access to the University of Rhode Island AI Laboratory GPU computing resources.

References

[1] Forti V, Baldé CP, Kuehr R, Bel G. The global e-waste monitor 2020. Tech. rep., Bonn/Geneva/Rotterdam: United Nations Univ., Int. Telecommun. Union & Int. Solid Waste Assoc. (ISWA); 2020.
[2] Sthiannopkao S, Wong MH. Handling e-waste in developed and developing countries: Initiatives, practices, and consequences. Sci Total Environ 2013;463:1147–53.
[3] Cui J, Forssberg E. Mechanical recycling of waste electric and electronic equipment: a review. J Hazard Mater 2003;99(3):243–63.
[4] DiFilippo NM, Jouaneh MK. A system combining force and vision sensing for automated screw removal on laptops. IEEE Trans Autom Sci Eng 2017;15(2):887–95.
[5] Yildiz E, Wörgötter F. Dcnn-based screw detection for automated disassembly processes. In: 15th Int. conf. signal-image technol. & internet-based syst. 2019, p. 187–92.
[6] Wegener K, Chen WH, Dietrich F, Dröder K, Kara S. Robot assisted disassembly for the recycling of electric vehicle batteries. Procedia CIRP 2015;29:716–21.
[7] Shan M, Guo J, Gill E. Review and comparison of active space debris capturing and removal methods. Prog Aerosp Sci 2016;80:18–32.
[8] Liou J-C. An active debris removal parametric study for LEO environment remediation. Adv Space Res 2011;47(11):1865–76.

[9] Hu H, Wang D, Gao H, Wei C, He Y. Vision-based position and pose determination of non-cooperative target for on-orbit servicing. Multimedia Tools Appl 2020;79(21):14405–18.

[10] On-orbit satellite servicing study project report. Tech. rep., Greenbelt, MD: National Aeronautics and Space Administration, Goddard Space Flight Center; 2010.

[11] Everingham M, Eslami SA, Van Gool L, Williams CK, Winn J, Zisserman A. The pascal visual object classes challenge: A retrospective. Int J Comput Vis 2015;111(1):98–136.

[12] Redmon J, Farhadi A. YOLO9000: better, faster, stronger. In: Proc. IEEE conf. comput. vis. pattern recognit. 2017, p. 7263–71.

[13] Thtrieu TH. Thtrieu/darkflow. 2019, [Online]. Available: https://github.com/thtrieu/darkflow. [Accessed 7 Nov. 2019].

[14] Chen H, He Z, Shi B, Zhong T. Research on recognition method of electrical components based on YOLO V3. IEEE Access 2019;7:157818–29.

[15] De Gregorio D, Tonioni A, Palli G, Di Stefano L. Semiautomatic labeling for deep learning in robotics. IEEE Trans Autom Sci Eng 2019;17(2):611–20.

[16] Li G, Song Z, Fu Q. A new method of image detection for small datasets under the framework of YOLO network. In: EEE 3rd Adv. Inf. Technol., Electron. Automat. Control Conf. 2018, p. 1031–5.

[17] Cartucho J, Ventura R, Veloso M. Robust object recognition through symbiotic deep learning in mobile robots. In: IEEE/RSJ int. conf. intell. robots and syst.. IEEE; 2018, p. 2336–41.

[18] Redmon J. Darknet: Open source neural networks in C. 2019, [Online]. Available: http://pjreddie.com/darknet. [Accessed 7 Nov. 2019].

[19] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 2012;25:1097–105.

[20] Raj RJS, Shobana SJ, Pustokhina IV, Pustokhin DA, Gupta D, Shankar K. Optimal feature selection-based medical image classification using deep learning model in internet of medical things. IEEE Access 2020;8:58006–17.

[21] Zhao Z-Q, Zheng P, Xu S-t, Wu X. Object detection with deep learning: A review. IEEE Trans Neural Netw Learn Syst 2019;30(11):3212–32.

[22] Liu L, Ouyang W, Wang X, Fieguth P, Chen J, Liu X, et al. Deep learning for generic object detection: A survey. Int J Comput Vis 2020;128(2):261–318.

[23] Gupta A, Anpalagan A, Guan L, Khwaja AS. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. Array 2021;100057.

[24] Lateef F, Ruichek Y. Survey on semantic segmentation using deep learning techniques. Neurocomputing 2019;338:321–48.

[25] Eitel A, Springenberg JT, Spinello L, Riedmiller M, Burgard W. Multimodal deep learning for robust RGB-D object recognition. In: IEEE/RSJ int. conf. intell. robots and syst. 2015, p. 681–7.

[26] Xu J, Zhou W, Chen Z, Ling S, Le Callet P. Binocular rivalry oriented predictive autoencoding network for blind stereoscopic image quality measurement. IEEE Trans Instrum Meas 2020;70:1–13.

[27] Kim Y, Moon T. Human detection and activity classification based on micro-Doppler signatures using deep convolutional neural networks. IEEE Geosci Remote Sens Lett 2015;13(1):8–12.

[28] Zhao Y, Cheng J, Zhou W, Zhang C, Pan X. Infrared pedestrian detection with converted temperature map. In: Asia-Pacific signal and information processing association annu. summit and conf. 2019, p. 2025–31.

[29] Guo Y, Liu Y, Oerlemans A, Lao S, Wu S, Lew MS. Deep learning for visual understanding: A review. Neurocomputing 2016;187:27–48.

[30] Pathak AR, Pandey M, Rautaray S. Application of deep learning for object detection. Procedia Comput Sci 2018;132:1706–17.

[31] Wei X, Yang Z, Liu Y, Wei D, Jia L, Li Y. Railway track fastener defect detection based on image processing and deep learning techniques: A comparative study. Eng Appl Artif Intell 2019;80:66–81.

[32] Zhang K, Wang W, Lv Z, Fan Y, Song Y. Computer vision detection of foreign objects in coal processing using attention CNN. Eng Appl Artif Intell 2021;102:104242.

[33] Zhang Y, Sun X, Loh KJ, Su W, Xue Z, Zhao X. Autonomous bolt loosening detection using deep learning. Struct Health Monit 2020;19(1):105–22.

[34] Wang Z, Li H, Zhang X. Construction waste recycling robot for nails and screws: Computer vision technology and neural network approach. Autom Constr 2019;97:220–8.

[35] Li Y, Zhao W, Pan J. Deformable patterned fabric defect detection with fisher criterion-based deep learning. IEEE Trans Autom Sci Eng 2016;14(2):1256–64.

[36] Ding C, Lu L, Wang C, Ding C. Design, sensing, and control of a novel UAV platform for aerial drilling and screwing. IEEE Robot Automat Lett 2021;6(2):3176–83.

[37] Zheng Y, Chen Z, Lv D, Li Z, Lan Z, Zhao S. Air-to-air visual detection of micro-UAVs: An experimental evaluation of deep learning. IEEE Robot Autom Lett 2021;6(2):1020–7.

[38] Chen WH, Wegener K, Dietrich F. A robot assistant for unscrewing in hybrid human-robot disassembly. In: IEEE int. conf. robotics and biomimetics. 2014, p. 536–41.

[39] Tan C, Sun F, Kong T, Zhang W, Yang C, Liu C. A survey on deep transfer learning, In: Int. conf. on artificial neural networks. 2018, p. 270–9.

[40] Wang C-C, Samani H. Object detection using transfer learning for underwater robot. In: Int. conf. advanced robotics and intelligent systems. 2020, p. 1–4.

[41] Raza K, Hong S. Fast and accurate fish detection design with improved YOLO-v3 model and transfer learning. Int J Adv Comput Sci Appl 2020;11:7–16.

[42] Montalbo FJP. A computer-aided diagnosis of brain tumors using a fine-tuned YOLO-based model with transfer learning. KSII Trans Internet & Inf Syst 2020;14(12).

[43] George J, Skaria S, Varun V, et al. Using YOLO based deep learning network for real time detection and localization of lung nodules from low dose CT scans. In: Medical imaging 2018: computer-aided diagnosis. vol. 10575, International Society for Optics and Photonics; 2018, p. 105751I.

[44] Al-Masni MA, Al-Antari MA, Park J-M, Gi G, Kim T-Y, Rivera P, et al. Simultaneous detection and classification of breast masses in digital mammograms via a deep learning YOLO-based CAD system. Comput Methods Programs Biomed 2018;157:85–94.

[45] Jia Z, Bhatia A, Aronson RM, Bourne D, Mason MT. A survey of automated threaded fastening. IEEE Trans Autom Sci Eng 2018;16(1):298–310.

[46] Lee NK, An Y, Tsung F. Studying effects of screw-fastening process on assembly accuracy. Int J Adv Manuf Technol 2005;25(5–6):493–9.

[47] Cherubini A, Passama R, Fraisse P, Crosnier A. A unified multimodal control framework for human–robot interaction. Robot Auton Syst 2015;70:106–15.

[48] Pitipong S, Pornjit P, Watcharin P. An automated four-DOF robot screw fastening using visual servo. In: IEEE/SICE int. symp. system integration. 2010, p. 379–83.

[49] Hu Z, Wan W, Koyama K, Harada K. A mechanical screwing tool for parallel grippers—Design, optimization, and manipulation policies. IEEE Trans Robot 2021.

[50] Bolmsjö G. Supporting tools for operator in robot collaborative mode. Procedia Manuf 2015;3:409–16.

[51] Matsuno T, Huang J, Fukuda T. Fault detection algorithm for external thread fastening by robotic manipulator using linear support vector machine classifier. In: IEEE int. conf. robotics and automation. 2013, p. 3443–50.

[52] Li R, Pham DT, Huang J, Tan Y, Qu M, Wang Y, et al. Unfastening of hexagonal headed screws by a collaborative robot. IEEE Trans Autom Sci Eng 2020;17(3):1455–68.

[53] Gil P, Pomares J, Diaz SvPC, Candelas F, Torres F. Flexible multi-sensorial system for automatic disassembly using cooperative robots. Int J Comput Integr Manuf 2007;20(8):757–72.

[54] Vongbunyong S, Kara S, Pagnucco M. Basic behaviour control of the vision-based cognitive robotic disassembly automation. Assem Autom 2013.

[55] Vongbunyong S, Kara S, Pagnucco M. Learning and revision in cognitive robotics disassembly automation. Robot Comput Integr Manuf 2015;34:79–94.

[56] DiFilippo NM, Jouaneh MK. Using the soar cognitive architecture to remove screws from different laptop models. IEEE Trans Autom Sci Eng 2018;16(2):767–80.