Transactions on Machine
Learning and Data Mining
Vol. 14, No. 2 (2021) 55 - 69
© 2021, ibai-publishing
P-ISSN: 1865-6781, E-ISSN 2509-9337
ISBN 978-3-942952-89-7

# Bearing Lubricant Defect Segmentation Using Synthetic Data

Richard Bellizzi[1], Jason Galary[2], Alfa Heryudono[3]

[1,2] Applications Development & Validation Testing Lab, Nye Lubricants, Fairhaven MA 02719, USA
rbellizzi@nyelubricants.com, jgalary@nyelubricants.com
[3] Department of Mathematics, University of Massachusetts Dartmouth, Dartmouth MA 02747, USA
aheryudono@umassd.com

**Abstract.** Lubricant testing often requires a post-test examination of specimens to obtain the desired critical measurement. Advancing these analysis methods aids in developing higher-performing products by allowing for improved insight into the lubricants' performance. Building off previous work in Computer Vision and Machine Learning, this work aims to extend the use of these methods into the lubricant testing realm. Minimizing defects is a desirable outcome since part of the lubricants' role is to protect the bearing's surface. While large-scale defects are easy to interpret, it becomes difficult to differentiate between test results when comparing bearing examples with less apparent defects. Providing a more consistent, granular analysis of these tests can help lubricant development withstand stringent requirements.
R-Mask CNN methods provide an option to apply instance segmentation techniques to classify areas of interest, allowing for an image with multiple instances of these defects. Since big data is the fuel for a system like this, there are certain limitations regarding the number of examples for lubricant bearing surface defect data. Leveraging data amplification techniques allows for a synthetic 'big' data set to accommodate the model's needs. This paper lays out how these tools work synergistically to provide a model that can operationalize for a company sooner than waiting to generate a complete set of ideal data.

**Keywords:** Machine Learning, Computer Vision, Defect Segmentation, Synthetic Data

## 1 Introduction

In the first stage of this work, bearing defect detection provided general classification, which proved that machine learning could extract features allowing a model to recognize the presence of corrosion accurately [2]. While this established that machine learning could identify corrosion on bearings, improving this methodology to include additional defects provides a lubricant company, such as Nye, with a higher performing model. Computer Vision and Machine Learning methods are used in various industries to automate manual, repetitive visual inspection methods. The current bearing analysis process meets these criteria since the analysis is repetitive, manual, and the analyzed data, the bearings, are uniform in nature. This uniformity, or the idea that a clean sample always looks like another clean sample, provides the feasibility needed to consider these methods. This system focuses on bearings with various defects since obtaining a model that generalizes to new data requires a robust training space to extract the necessary features. Previous work [2] showed that standard Convolutional Neural Network (CNN) models could accurately distinguish corrosion defects on these bearing samples. A Mask R-CNN [1,6,14] model builds off this recognition feature and adds instance segmentation and categorization of the desired characteristics within the images. Corrosion defects and various 'other' defects provide the source labels in the data set used to train the Mask R-CNN [6]. The masks, therefore, combine these two types of defects into a 'defect' class for the model to recognize not only corrosion but these additional surface defects that occur. Additionally, the 'clean' images provide an excellent background to synthesize simulated defects when considering these bearings' uniform nature [1,9].

Several core approaches provide the backbone for the bearing defect detection and segmentation model. Considering the data-hungry nature of machine learning, which requires large amounts of data to generalize well, an intermediate data engineering step introduced to meet this demand bridged the gap. A synthetically generated extensive data set further supplements the training of this model, starting with a limited image data set provided by Nye Lubricants [1,9,14]. Testing new bearings and then scanning them to store as training data requires a decent number of resources. This resource bottleneck limits the advantage that big data provides. By amplifying this data set and synthetically increasing it to a considerable size, the Mask R-CNN [6] has enough training data to extract the necessary Bearing Surface defect detection and segmentation features. [1,9,14]

Advancing these industry analysis methods aids in developing higher performing products by allowing for improved insight into the lubricants' performance. Building off previous work in Computer Vision and Machine Learning [1,2,3,6,9,10,14], this work aims to extend the use of these methods into the lubricant testing realm. Minimizing defects is always the goal for a lubricant since part of its role is to protect the bearing's surface. Interpreting the fine-grained results of high-performing products is possible given these new methods. As a tool alongside current methods, confusion on different interpretations of results have a comparison method now to differentiate the results discretely.

## 2 Synthetic Data Generation

The dataset supplied by Nye Lubricants consists of various bearing images like the examples shown in Figures 1 and 2. These images provide a starting point for amplification once scanned on the Bearing Corrosion Analyzer (BCA) system used to digitize bearing corrosion test results. These bearing images show the bearings' internal race after subjection to cycles of motion and stationary resting periods. After disassembling these bearings and isolating the bearing race for visual inspection, a technician assigns a rating of 0 to 5 in increments of 1. The bearing undergoes scanning on the BCA system, and a TIFF file is stored to establish a working dataset. There are several examples of defects that Nye Lubricants considers in this work. The current standard classifies specifically only corrosion that occurs on these test bearings. With a digital representation of the data, further analysis can extrapolate additional disturbances to the bearing's surface. When visually analyzing the bearings, specific parameters must be met for a defect to qualify as corrosion. While visually showing a disturbance, these specifications mean that certain disturbances may not be corrosion and therefore not considered in the rating. Nye's goal is to build on this method and supply both the standard rating and a secondary rating that encompasses these additional defects. In this way, product development has enhanced feedback from these tests.

**Fig. 1.** Source Bearing with One Type of Defect

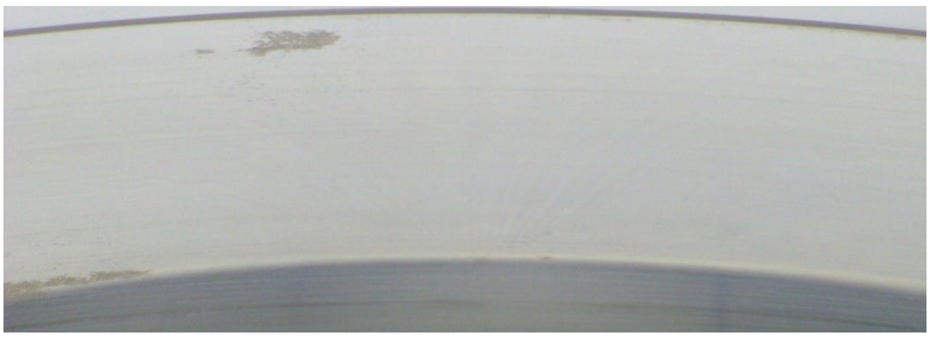


**Fig. 2**. Source Bearing with Multiple Types of Defects

The supplied bearing data consists of around 900 RGB TIFF images with dimensions of 2590 x 1101, and they provide the initial starting point for amplification. Using the method presented by Immersive Limit [9], these 900 images separate into two different components, a background image, and a foreground image. The foreground represents the defects that might occur during operation in a corrosive environment. Immersive Limits' work [9] builds off the MS Common Objects in Context (COCO) [10] work available on GitHub, which has been utilized in numerous projects to segment and classify images. A MATLAB tool, called the Image Labeler, supplied the platform for which the Nye bearing images were processed and labeled at a pixel level. The resulting pixel label masks provide a way to isolate and crop different defect examples for use as foregrounds. After isolating and extracting these examples, the new foreground images were saved as PNG files with transparent backgrounds, allowing them to combine on top of 'clean' bearing background images. Around 40 clean bearing images provided the background source for synthetic image generation, while a combined total of 450 defect samples contributed to the foreground image set. After all the image components were cleaned and prepared for amplification, adapted python source code, modeled after the Immersive Limit [9] image generation example, was used to synthesize new bearing images. This synthetic generation entailed taking a background image and randomly selecting different defect foregrounds for recombination. After applying random transformations in rotation, scale, and brightness, the foreground defects overlay onto the bearing background producing a new 'synthetic' example of a bearing with defects. In this way, the original sample data set evolves to a set of 100,000 synthetic images, with some examples shown in Figure 3. This amount of data approaches Big Data levels, which provides feasible training data to the final model. [1,6,9,14]

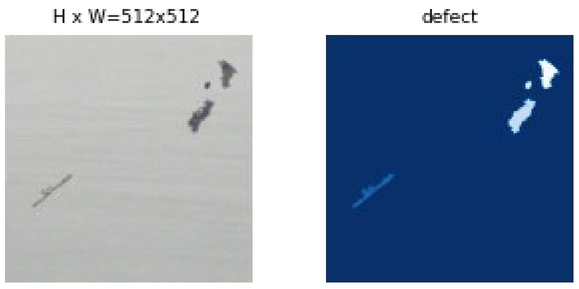


**Fig. 2.** Samples of Synthetically Generated Bearing Data with Masks

## 2.1 R-Mask CNN Method

The recent model for developing this system is the Mask R-CNN, which builds on Faster R-CNN and other work to improve segmentation in images [3-6,13]. Developed by the Facebook AI research team, this model has multiple components that allow it to locate regions of interest and then isolate anything contained in that region of interest. The first step in this model is the Region Proposal Network (RPN), responsible for establishing a bounding box containing a candidate object. Additionally, while the model identifies these areas of interest, it also classifies the object contained within the bounding box in parallel. These components so far are standard in the R-CNN [3] and Faster R-CNN models [13], with the latter visualized in Figure 4. These models provide a dual output response where the user obtains a box isolating the object and a label for that object. Introducing a binary mask in tandem with these responses, like in Mask R-CNN, yields a robust segmentation model [6]. A mask consists of a binary representation of the image with the objects represented by their class label. This added detail provides the model with a pixel-to-pixel level of analysis and is desirable in most imaging methods, especially for bearings where defects can be minor. A cutting-edge method like this is slow to spread into industries where the technical skills required are still developing while data sources are limited. This work shows that these techniques can provide valuable insight into product development and digitizing data. It also pushes to invoke more research and practice into these methods to promote industrial growth within lubricant manufacturing and research. [6]

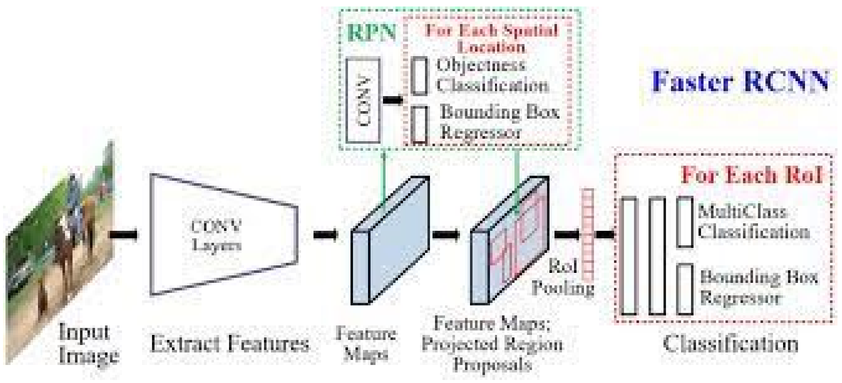


**Fig. 3.** Faster R-CNN Framework

Faster R-CNN is a system that motivated and provided the backbone for the Mask R-CNN [6,13]. The model provides the RPN system, which defines potential bounding boxes through two CNN components. Feeding the image into a convolutional network to generate features that map into an additional CNN that predicts the proposed regions of interest ends up being computationally faster than the Selective Search Algorithm used in Fast R-CNN and R-CNN [3]. Faster R-CNN uses an RoI Pooling method to combine overlapping region propositions to finalize the isolation and segmentation of the objects contained in the input image [13]. Mask R-CNN introduces a RoIAlign function that accomplishes this same feat through bi-linear interpolation to localize the necessary regions within objects, shown in Figure 5 [6]. This layer takes the output from the initial convolutional network and then generates a classification and location for each instance segmentation based on the information from the proposed region network and the feature extraction network. This RoIAlign method allows for improved mask generation on a pixel level, making it promising for the scale of detail required for bearing corrosion detection. This efficiency is desirable due to the real-time capability of computing these detections, making it possible to outfit test rigs with timelapse video captures of corrosion development with the necessary segmentation extractions sampled at intervals.

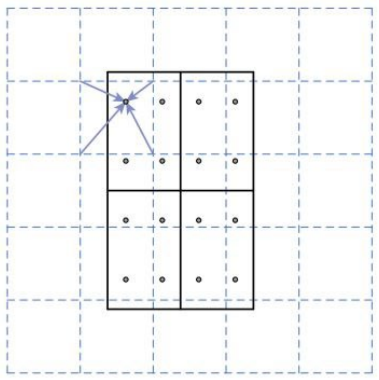


**Fig. 4.** RoI Align Bi-linear Interpolation Example

The R-Mask CNN [6] method has been used in various examples to provide accurate segmentation models, such as in the Weeds example [9], which inspired this approach. Since these different use cases were possible, transferring this technology to bearing defects is a viable option for Nye. Various-sized data sets allowed a way to understand the R-Mask CNN models' effectiveness when utilizing sizeable synthetic data sets [1,6,9,14]. The several data sets used during training increase at a factor of 10 five times to yield $10^1$, $10^2$, $10^3$, $10^4$, and $10^5$ images per set, each used in analyzing the effectiveness of synthetic bearing data. Using an approach like this allowed for an understanding of the limitations that synthetic data brings. Given that synthetic data relies on its core source for feature extraction, the possibility of saturation is likely to occur where increasing the number of synthetic data yields no improvements. Utilizing the COCO utilities on Github [1,10] and the examples provided by both the original COCO creators [10] and the Immersive Limit work [9], the synthetic data sets were integrated and processed utilizing Python source code adaptations [1,5,9,14].

Overall, R-Mask CNN [6] models are a well-established computer vision implementation, and it is well known for its use in automated driving technologies. This methodology is robust, and when used in the correct aspect, its capability extends into similar use-cases for defect detection [8,12]. Bearing defects on a uniform surface present a suitable scenario for this type of model. The only changes taking place are the various foreground occurrences, while a clean bearing always is the same. The core components utilized for this system come from the MRCNN Python library [1,5,9]. This library is a popular starting point for various other projects due to its flexibility for adaptations to different data sets. This adaptation feature utilizes transfer learning concepts, such that the model network is composed of the well-known Resnet101 CNN [3] layer structure. This pre-trained network initializes using various options for the starting weights, such as the MS COCO [10] weights or the ImageNet weights [5]. The MS COCO [10] weights were selected due to the network's improved segmentation training using these weights, which translated well when segmenting the contoured bearing surface defects. These weights were used for all data sets to maintain consistent results between data sets. The advantage of a model using the MS COCO weights is the initial training on 350K images with 80 different categories, which provides an extremely promising avenue for transfer learning on a small data set like Nye's bearing data [10].

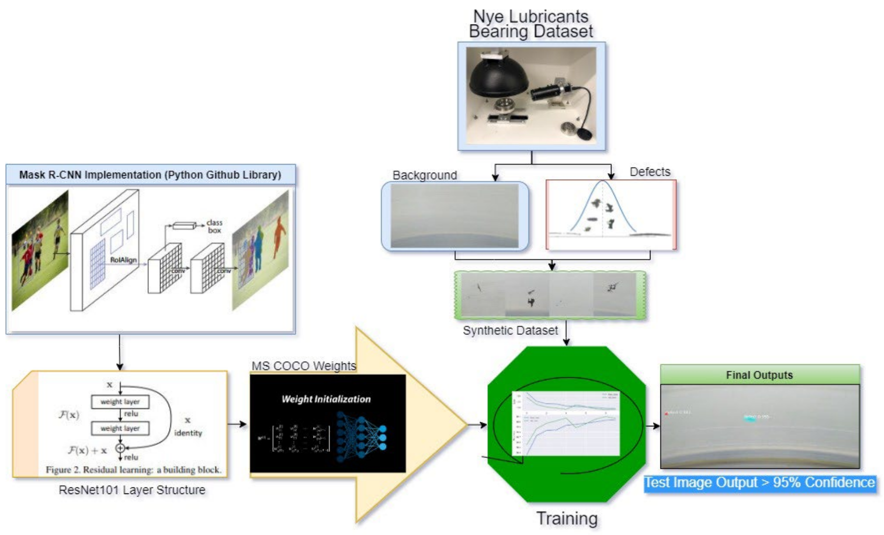

Nye Lubricants Bearing Dataset
Background
Defects
Synthetic Dataset
Mask R-CNN Implementation (Python Github Library)
class box
RoIAlign
ResNet101 Layer Structure
Figure 2. Residual learning: a building block.
MS COCO Weights
Weight Initialization
Training
Final Outputs
Test Image Output > 95% Confidence

**Fig. 5.** Project Flow used to Implement a Bearing Defect Detection System

The R-Mask CNN [1,5,6,9,14] implementation is trained in a Jupyter notebook [1] environment using an Nvidia GeForce GTX 1070 GPU as the hardware for computation. Even with the GPUs accelerated capabilities, the model required a decent amount of time for training. The training parameters were standardized across the different sized data sets to allow for a more direct comparison of each model's capability to learn from synthetic data. With 3000 steps per epoch and a validation frequency of 50, the model trains for two epochs on the head layers and then trains one additional epoch across all layers contained in the network. The model input is a standard normalized bearing image with dimensions (512, 512, 3). The output is an image with bounding boxes identifying the proposed defect regions where highlighted contours overlay onto the different defects segmented by the model. Considering the Weeds model [9] performed with accuracy upwards of 99 percent, the synthetic data method, scoring around 95-98% on bearings, seems to work well given the right amount of training. In Figure 6, the entire process flow maps out these tools to better represent how they come together to provide a functional model sooner than waiting for available data.

$$Precision = \frac{TP}{TP+FP} \tag{1}$$

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

$$mAP = \frac{1}{101}\sum_{i}^{101} Precision(Recall_i) \tag{3}$$

Evaluating the performance of these segmentations is essential as well. The tool provided for most computer vision methods is the Average Precision score [6]. Averaging this value across all validation or test images provides a Mean Average Precision (mAP). These measurements come from the COCO [10] data set standards which provide an effective way to evaluate the final models' capability. Average Precision (AP), defined by the Area Under the Curve calculation performed on the Precision-Recall (PR) curve for each image, allows for measuring performance. The COCO mAP, shown in Equation 3, uses 101 interpolated points along the PR curve to calculate an AP [6]. Batching these calculations provides a mean Average Precision score (mAP). As shown in Equation 1, Precision uses the IoU score to calibrate what is True Positive (TP) and False Positive (FP) amongst the samples. Recall, shown in Equation 2, provides the second component for the AP, and its components are also determined using the same level as Precision. Overall it is calculated by taking the Intersection over Union (IoU), shown in Figure 7, and then calculating both the Precision and Recall. Using the IoU score as a threshold for determining positive matches, these scores compute across all images providing the inputs for the Average Precision calculation. In this work, an IoU of 50 percent, as an intermediate expectation, provided a way to examine the models' capability in this application.

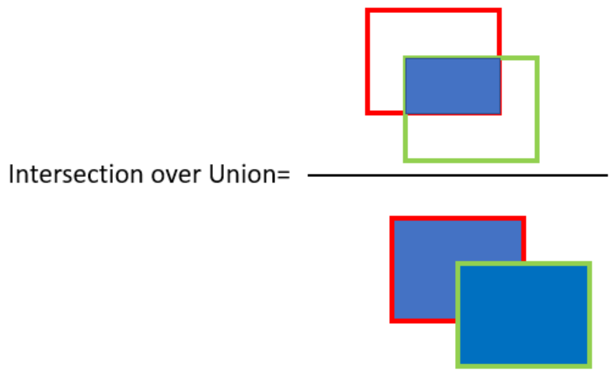


**Fig. 6.** IoU Calculation Visualization

While many more Machine Learning algorithms and methods [7,11] exist, those mentioned make up the core for achieving a system that automates bearing lubricant defect detection. Each method is best displayed by how each unfolded according to these results. For example, the core CNN models provide the region proposal and classification functionality, while the mask segmentation provides the final piece to improve segmentation on a pixel level. Data engineering was required to obtain this pixel-level detail with a focus on image manipulation methodologies. Transfer Learning is used to supplement training time and benefit from Big data training on networks within this application. Further supplementation still exists through modification of the synthetic image generation methods. Utilizing all of these techniques in unison, shown in the overall project diagram Figure 6, presents how to provide a functional system that performs bearing lubricant defect detection with acceptable levels of accuracy.

Consider the results presented as confirmation that these types of analysis systems are feasible in lubricant testing analysis.

## 3 Project Results

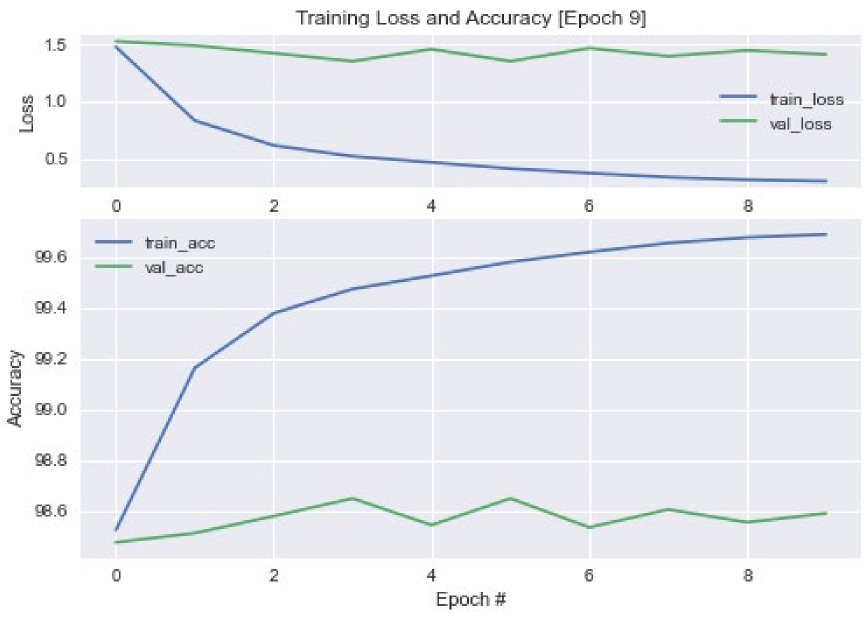


**Fig. 7.** $10^1$ Image Training

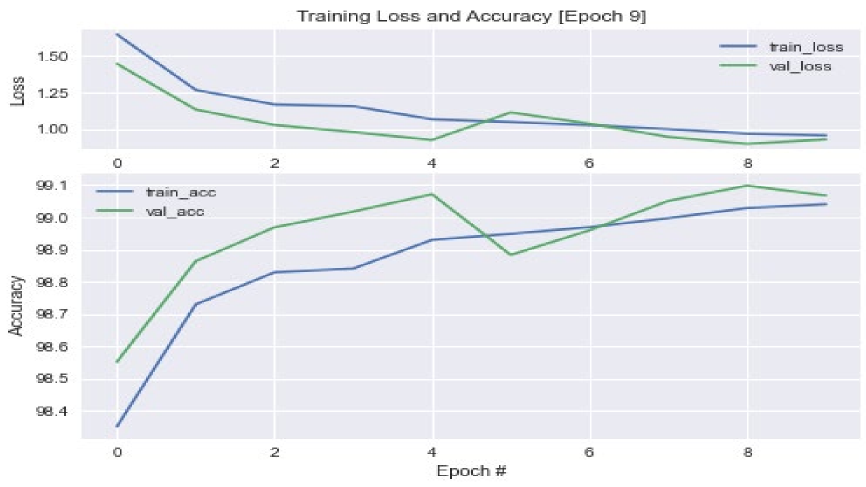


**Fig. 8.** $10^4$ Image Training

**Table 1.** Training Results on 512x512 Synthetic Images

| Dataset for Training | Training Duration (mins) | Training Loss | COCO Batch mAP @ IoU=50 |
|---|---|---|---|
| $10^1$ Image Set | 62.63 | 0.0721 | 0.2861 |
| $10^2$ Image Set | 56.97 | 0.1839 | 0.3043 |
| $10^3$ Image Set | 59.5 | 0.6054 | 0.4026 |
| $10^4$ Image Set | 59.6 | 0.7657 | 0.4183 |
| $10^5$ Image Set | 59.8 | 0.7690 | 0.4410 |

During model development, software flow and execution testing of the source code occurred on smaller data sets, and further debugging searched for any issues that the bearing source data caused [1,5]. After resolving these aspects, setting up separate notebooks allowed for individual runs for the five different-sized data sets. The model's performance shows that accurate segmentation on bearing defect data is possible when training on these larger synthetic data sets. The model is confident on test predictions, with an example shown in Figure 10, primarily since a minimum prediction threshold of 95 percent filters weak predictions out. In this way, the model is an improvement over models developed in previous work [2]. The segmentations on test images, visually inspected to compare the models' results with technician results, compared decently well. The test bearing used is a newly acquired bearing specimen not included in supplying any of the foregrounds. The segmentation and confidence in the predictions were adequate comparatively. As expected, the model segmented slightly more defects than the technician. Typically, only corrosion is covered, so the model can indeed pick up additional disturbances. While this means they were not equivalent, in all of the areas where the technician identified corrosion, the model detected a defect as well and with confidence over 95 percent.

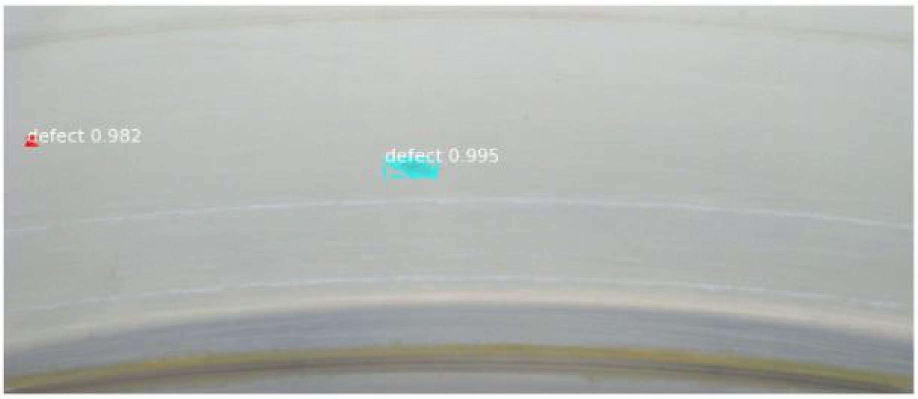


**Fig. 9** Test Bearing Inference on Key Corrosion Locations

The different tools that the GitHub library provides for analyzing the models' performance include a Precision-Recall curve along with a ground truth grid, which is like a confusion matrix of the predictions and the actual mask labels [1,6,10]. Using these tools, comparing the different size data sets is possible with the Average Precision (AP) score as the critical measurement [6,10]. Table 1 shows a comparison of the different-sized data sets that ran successfully on a local GPU device. Training on these data sets was about equivalent as far as training duration goes, but interestingly enough, the training loss increases as the data set sized increases. This effect shows how the smaller data sets force overfitting, where the more extensive models allow for more defect scenarios and more robust training. With this analysis method, the different synthetic data sets compare using the Precision and the segmentation capability as the evaluation metrics.

The final measure used to highlight synthetic data's effectiveness includes a mean score calculated on the different validation data sets. Taking the AP score for each data set across all validation images, a mean score allows for direct comparison between them, calculated by sampling ten different mini batches from the data and the Average Precision computed across each batch [6,10]. These ten batches are then combined and averaged to get an overall AP score for each data set. The method of averaging across different mini batches allows for a view into the models' general accuracy. The resulting Mean APs for each data set size, plotted in Figure 11, show the overall validation AP score performances.

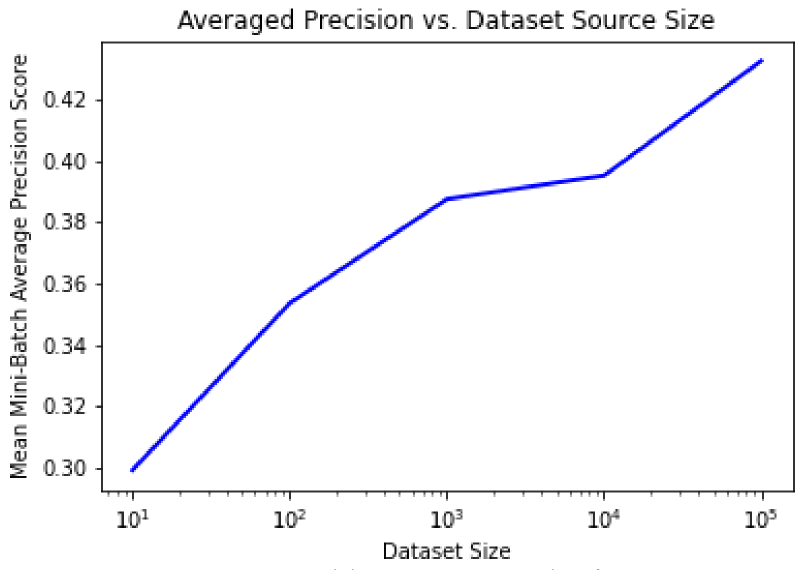


**Fig. 10.** Mean Average Precision Score Progression from Dataset to Dataset

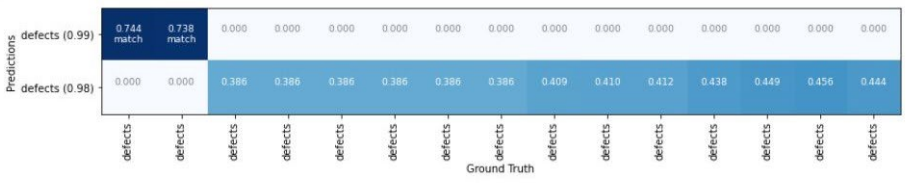
## 4 Conclusions

After training all these different sized data sets, adding additional synthetically generated images improves the model's segmentation accuracy, as shown by monitoring the overall Average Precision score [6,10]. Figure 11 shows how, as data set sizes increase, the model improves its overall localization of the different defects, represented by an increasing mean Average Precision score. For example, both the $10^4$ image-set and the $10^5$ image-set yielded high prediction confidences of 95 percent and higher on new test data. Even with similar performance, however, the Average Precision score [6,10] for each data set still shows the improvements in segmentation obtained by training on more synthetic data. Increasing too much yields expectations that increasing synthetic data too much may end up saturating the models' feature learning. Because of this, the AP score eventually reaches a plateau, but this is only if the source data is not changed. Operationalizing this model allows for additional defects to be segmented and stored for additional source data in future training. Already, this setup provides a pipeline to further improve the model in robustness and accuracy. Since the $10^5$ model offers 95 percent to 99 percent confidence, operationalizing the trained network and deploying it onto the Bearing Analyzer system alongside the current algorithm supports research efforts with added functionality. Since the models' AP score is steadily approaching a satisfactory threshold of greater than 0.5, this deployment provides benefits in continuous model training and accuracy improvements while still obtaining automated defect segmentation in bearing specimen analysis. [6,10]

Continuous improvement is standard in any development, even more so when a model still has errors in its predictions. Considering the pixel level labeling that sourced the foreground images, the model may retain some training data noise. For example, the model may merge pixels near each other, grouping the predictions even though the ground truth information labels them as separate entities. Machine Learning systems consume data to improve; therefore, adjusting the data inputs will only further aid the end application development. Further work plans to explore how to represent this scenario better to provide insight into the models' capability to accurately segment desired defects. Isolating what a 'desired defect' would mean requires that there needs to be some measure of transparency, size, or discoloration for each defect to quantify to the model when to ignore some disturbances while including others. Since this work builds on a standard test method, the current stipulations for determining a defect focus solely on corrosion, so nothing but arbitrary standards exist for these additional disturbances. By establishing an internal standard for this, the improvements to the products resulting from this work may inspire industry standards to a higher-performing level.

Some key components still need to be added to improve the model further. Since the synthetic generation code only overlays a foreground if it does not cross an image edge, some defects on the image boundaries were not recognized or segmented, as seen in Figure 10. Manually injecting additional data that encompasses masks along the image's edges can fix this boundary bias issue alongside additional synthetic generation code adaptations [1,5,9]. The circular nature of the bearing forces the camera's rectangular viewing angle to take snapshots where some defects occur half in one frame and a half in the next. Re-training with these examples should help the model adjust accordingly, allowing it to establish recognition in the middle and edges of the camera view. Another bias exists in the model since the focus of this work was on granularity. The defects that occur as small disturbances are the examples that human technicians struggle to differentiate. Due to this, the defect foregrounds ranged from minor disturbances to medium disturbances to the surface. This exclusion of significant disturbances added a bias in the model to segment the examples under a specific size. With the edge issue mentioned previously, the larger defects also lacked representation, and therefore, the model also struggles to segment defects that cover large areas of the surface.

Since the ground truths show numerous instances, but the model predicts only a few, the combination of defects seems like another core issue. The model predictions are compared to the validation set of ground truths to determine the training accuracy. While these ground truths come from the labeled data, the synthetic combination may create the masks that cause this ground truth to blow up or overlay on top of each other, adding more instances than appear. This ground truth effect, shown in Figure 11, represents how the predictions are high accuracy, but the ground truths do not line up. The ground truth dictates more predictions than the model yields, affecting training as the model combines separate ground truths into one prediction. This bearing models' segmentation and prediction capability on the different defects allows for further break down into an improved multi-class defect data set. The additional information helps identify some of the better features required for classifying these defects as either corrosion or these 'other' surface defects as a next step. In the meantime, Nye gains a model adept at segmenting the different bearing images, simplifying the final test specimen analysis for the technicians. Technicians can use their rationalization in tandem with this additional information to judge whether a defect is an example of corrosion or not.



| Predictions \ Ground Truth | defects | defects | defects | defects | defects | defects | defects | defects | defects | defects | defects | defects | defects | defects | defects |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| defects (0.99) | 0.744 match | 0.738 match | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| defects (0.98) | 0.000 | 0.000 | 0.386 | 0.386 | 0.386 | 0.386 | 0.386 | 0.386 | 0.409 | 0.410 | 0.412 | 0.438 | 0.449 | 0.456 | 0.444 |

**Fig. 111.** Example Ground Truth Matrix on Validation Image

Overall different applications require different needs, and the Weeds models' implementation [9] performs a single class segmentation based on the weeds data. That model aimed to generalize weeds versus the background grass without necessarily separating the different weeds themselves [9]. A model trained to recognize surface defects versus the bearing background provides a single class segmentation solution to provide a working model to Nye. This initial setup improves upon knowing whether a bearing has defects after testing since it can go one step further and segment those defect instances for the users' interpretation. A feature like this does provide an avenue to continue to process test results until a more robust data set is available. A wider variety of the two different classes may help provide enough source data for synthetic images with ground truths containing more robust scenarios than this arrangement. Having more instances of these defects, along with the appropriate bias improvements, would provide the granularity to isolate the bearing surface's different defect occurrences. Overall, the Bearing Defect model behaves as needed for aiding in the technicians' breakdown and evaluation of test specimens. With constant human feedback to reinforce correct predictions over missing or incorrect predictions, the network improves continuously and further adapts to better-performing products. Overall, the Synthetic Data

Generation method with Mask R-CNN [1,5,9] methodology provides a functional trained model to Nye Lubricants with continued development and growth in mind.

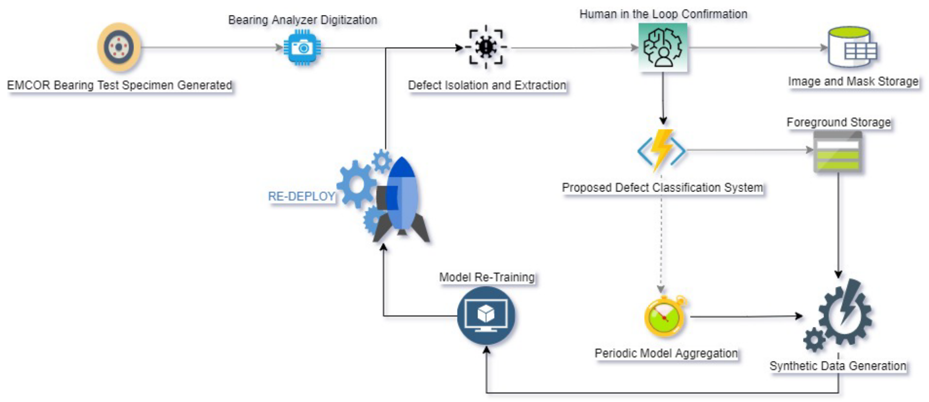

Bearing Analyzer Digitization
Human in the Loop Confirmation
EMCOR Bearing Test Specimen Generated
Defect Isolation and Extraction
Image and Mask Storage
Foreground Storage
Proposed Defect Classification System
RE-DEPLOY
Model Re-Training
Periodic Model Aggregation
Synthetic Data Generation

**Fig. 12.** System Framework for Lubricant Defect Detection

Additional to this expanded quantification of surface defects, it is desirable to standardize and establish a pipeline framework for defect detection models. The inherent extraction capability combined with human feedback allows for new data to propagate back into the model's feature learning. Larger companies have setups established, but generally, there is not a standard but somewhat various guidelines. Assigning some structure around the production style AI system helps other researchers steer their implementations, freeing up focus for the data engineering tasks. A framework consisting of data corrections and long-term storage for data set growth assists development efforts with the continuous improvement culture in mind. Pipelines facilitate this mindset providing connection components for intelligent systems to receive frequent updates. This approach helps account for data drift while also improving the overall capability of the model. An example pipeline flow, displayed in Figure 13, shows how with just one step requiring human intervention, the overall process occurs behind the scenes of the researchers. Strategies like this abstract the complexity of the analysis and system improvements away from the users while still allowing for interpretable and quantifiable data for statistical methods and decision-making tasks. The model and techniques laid out present a way to execute ideas like this, propelling different industrial systems into new data generation sources.

## Acknowledgments

Thanks go to the University of Massachusetts Dartmouth Mathematical Consulting class members for their contributions in brainstorming and feedback on this work. Additionally, thanks to the R&D members, Rob Mulkern and Kevin McBarron from Nye Lubricants, who assisted with generating the data for this work. Finally, thank you to both UMass and Nye for sponsoring this research.

## References

1. P. Perner, U. Zscherpel, C. Jacobsen, A Comparision between Neural Networks and Decision Trees based on Data from Industrial Radiographic Testing, Pattern Recognition Letters 22 (2001), pp. 47-54, PDF-File
2. Bellizzi, R., Galary, J., and Heryudono, A.: Evaluation of convolutional neural networks and transfer learning for bearing corrosion inspection. International Journal of COMADEM (2021).
3. Girshick, R., Donahue, J., Darrell, T., and Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014).
4. Girshick, R., Donahue, J., Darrell, T., and Malik, J.: Region-based convolutional networks for accurate object detection and segmentation. IEEE transactions on pattern analysis and machine intelligence 38(1), 142-158 (2015).
5. Detectron. https://github.com/facebookresearch/detectron, last accessed 2020/12.
6. He, K., Gkioxari, G., Dollár, P., and Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp. 2961-2969, (2017).
7. Jaakkola, T., and Barzilay, R.: Machine Learning MIT Course Book. Draft, MIT, (2016).
8. Johnson, J. W. Adapting mask-rcnn for automatic nucleus segmentation. arXiv preprint arXiv:1805.00500, (2018).
9. Ai weed detector. https://www.immersivelimit.com/blog/ai-weed-detector, last accessed 2020/10.
10. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L.: Microsoft coco: Common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision - ECCV 2014 (Cham, 2014), Springer International Publishing, pp. 740-755.
11. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow, https://github.com/matterport/Mask_RCNN, last accessed 2021/1/24.
12. Ng, A.: Machine Learning Yearning. Online Draft, last accessed 2018.
13. Perner, P., Zscherpel, U., and Jacobsen, C.: A comparison between neural networks and decision trees based on data from industrial radiographic testing. Pattern Recognition Letters 22(1), 47-54 (2001).
14. Ren, S., He, K., Girshick, R., and Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497, (2015).
15. Computer vision tutorial: Implementing mask r-cnn for image segmentation (with python code), https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/, last accessed 2020/12.