Automatic Detection of Paint Defects using Machine Learning

DESIGN DOCUMENT

Team: sddec21-17
Client: Danfoss Power Solutions
Adviser: Mohamed Selim, Ph.D.

Team Members:
Brett White
Corrine Dornbach
Josh Moeller
Khairi Norizan
Lucas Keller

Contact Us: sddec21-17@iastate.edu
Website: https://sddec21-17.sd.ece.iastate.edu

Revised: April 25, 2021

Executive Summary

Development Standards & Practices Used

Our development standards will follow the IEEE Software Engineering guidelines and standards. These standards will ensure that we deliver a high-quality product that satisfies the needs and wants of Danfoss. When developing the design and the model itself, we will use Agile, an iterative design process. Major development stages are broken into chunks that can be adapted throughout development, allowing us to maintain a small buffer of research-orientated features that can be used in future implementation while simultaneously involving our Danfoss contact in the design process.

Summary of Requirements

Our development and design requirements have a focus on a tested deep learning solution that is shipped with analysis in the areas of the defective pump classes. The primary requirements are as follows:

- Use and shipment of deep learning software to classify defective pumps from sample images
- Both hardware and software integration to automatically capture, analyze, and save images
- The solution must be capable of inference in < 10 seconds for a pump within the camera field of view
- Precision requirement of 0.7 for binary defect classification
- Recall requirement of 0.8 for binary defect classification, false positives are generally acceptable
- General descriptive statistics for each defect class should be reported by end of the semester.
- The solution must be capable of running 24 hours a day, while still performing inference in < 10 seconds for a given pump

Applicable Courses from Iowa State University Curriculum

Com S 474 - Introduction to Machine Learning:
Provides background knowledge on ML techniques & some experience with implementation.
Stat 330 - Probability and Statistics for Computer Science:
Useful for evaluating which paint defects are best to focus on in terms of return on investment. Helpful for understanding the bias-variance tradeoff present in our model.
SE 309 - Software Development Practices:
Provides knowledge on best practices when working in team environments. Especially useful for source control / CICD and project planning methods (like AGILE).
HCI 575 - Computational Perception:
Provides knowledge on image processing techniques especially in areas of morphology, image analysis, image transformations.

Tangentially related:
Math 207 - Matrices and Linear Algebra:

Useful for image transformations & manipulations that help train our model to understand it. Example: rotations or principal component analysis. Additionally helpful for understanding vectorization present in our model that allows us a real-time solution.

New Skills/Knowledge Acquired that was not Taught in Courses

Deep Learning: Object Detection
Important to implement this solution. In order to provide a solution to the problem, it is necessary to understand the Deep Learning algorithms.

Software and Hardware Components Integration
Since the solution requires hardware such as cameras, thus, the knowledge in software and hardware integration is important. The integration will be involving the model and the cameras.

Table of Contents

1 Introduction

1.1 Acknowledgment

We would like to express our gratitude to our Senior Design Instructor, *Dr. Lotfi Ben Othmane*, for giving us the opportunity to participate and work on this project. We would also like to thank our Faculty Advisor, *Dr. Mohamed Selim*, for his insightful comments and valuable guidance throughout the research process and project development. Importantly, we would like to acknowledge the team at Danfoss, especially *Mr. Mohamed Eldakroury*, for providing the necessary resources to implement our system. Danfoss has supplied us with factory simulations, data, and images on defective pumps, spreadsheets of defect information, and knowledge of the problem.

1.2 Problem and Project Statement

Problem Statement

> Danfoss Central Paint and Packaging manufactures painted pumps. Over an 18-month period, there have been a large number of defective pumps produced where the paint job has either been incomplete or poor. Approximately one-eighth of these defects were classified as high-impact defects which indicates that they were most likely scrapped. The combined cost of scrap and rework for both high-impact and low-impact defects for this period is very high, and Danfoss would like to reduce, or even completely eliminate, this cost.

Solution Approach

> Our solution is to design and implement a defect detection system that classifies the different occurrences of pump types: non-defective, no paint, low paint, splattered paint. When a defective pump is detected, the system sounds an alarm to notify a staff member who can repair the malfunctioning painting robot. This helps achieve our goal to reduce the number of defective units produced by 40% and, therefore, reduce the overall financial loss.

This solution is important because it will help Danfoss Central Paint and Packaging reduce the number of defective hydrostatic pumps produced by the painting process. This will reduce the number and length of delays in providing pumps to clients and save Danfoss a substantial amount of money by reducing the need for rework or scrap of pumps. In order to approach this, we are designing a defect detection station to capture images of a hydrostatic pump to determine if the pump is defective. When a defect is detected, the system notifies the workers.

This solution is meant to reduce the time it takes to remake pumps and subsequently save Danfoss money. By having this system in place, we can detect any defects faster and more accurately and consequently fix them faster.

1.3 Operational Environment

This solution is intended for use within the Danfoss Central Paint and Packaging lanes. The hardware and software will be installed in such a way that the lighting is constant. Furthermore, this implementation will have no exposure to water or extreme temperatures.

1.4 Requirements

Main Requirement

The main requirement for defect detection involves Danfoss's production line for various pumps. Primarily, we plan to implement an early detection system that initiates a stop signal for the assembly line and therefore reduces the number of defective units by 40%.

The defect detection will be focused on identifying 3 types of defects:

- No paint
- Light paint
- Splatter/orange peel paint

Functional Requirements

A primary functional requirement of our solution includes the creation of a defect-detection station that is mostly independent of the Danfoss assembly line. Independent in this context means not connected to their signaling system and network infrastructure. This independent station will utilize up to two cameras to analyze the pumps and will prevent the continued painting of defective pumps before the paint-drying chamber on the assembly line. Once the paint defect is detected, a physical alarm would then be triggered which requires human intervention.

Economic Requirements:

The total budget supplied by Danfoss is $5,000. Beyond monetary assistance, they will also be aiding us with the development of a dataset consisting of several thousand images. At the moment, paint defects have afflicted Danfoss Power Solutions with large losses over an 18-month period. Hence, the overall goal for this solution is to reduce the number of paint defects by 40%.

Software/Hardware Requirements:

The *software requirements* for this solution include the development and integration of a deep learning algorithm that will classify pumps as defective or non-defective.

The *hardware requirements* primarily include two cameras and a computing station to detect a pump, capture several images, and perform classification. Additionally, the solution requires an alarm to audibly inform workers of the presence of a defective pump.

1.5 Intended Users and Uses

The final system is intended for use by Danfoss Power Solutions in their central paint and packaging lanes to help detect paint defects on hydrostatic pumps. Our solution will reduce the time to detect defective units and therefore cut down on the number of total defects. The primary audience of our implementation is assembly line workers present on the floor who are not actively watching for defects. Besides intervention upon alarm signals, interaction will be reduced to maintenance like basic cleaning.

1.6 Assumptions and Limitations

Assumptions	Limitations
• Pumps near the exit of the paint station and packing lanes will be orientated in the same direction • Lighting will be consistent for all images captured • Pumps with a low sample size will be handled effectively by the ML model	• Defect detection station must be confined to one area so that it doesn't disrupt the rest of the product line • The camera must be placed in such a way that painted surfaces are visible to the vision system • Total budget of $5,000 for all computing equipment and cameras needed

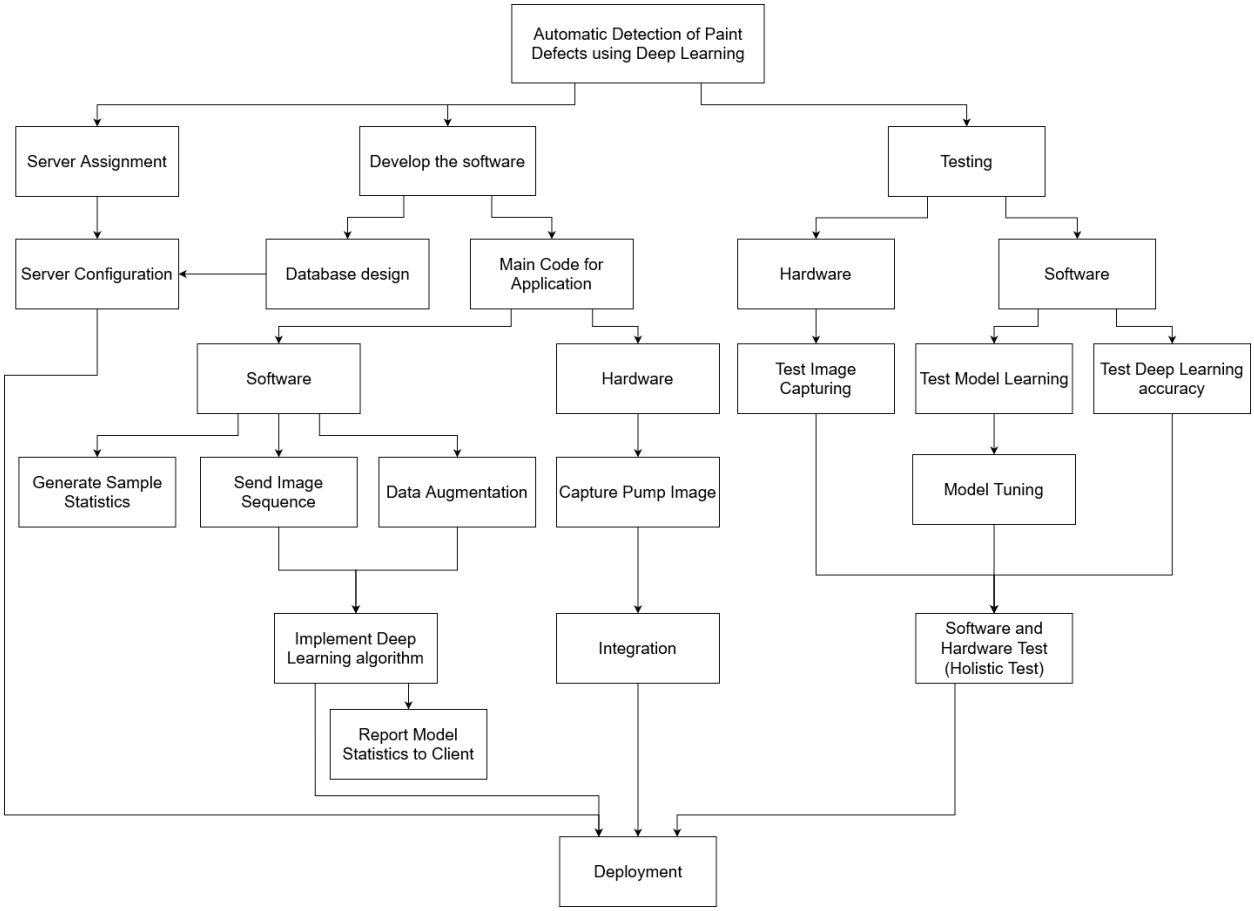
1.7 Expected End Product and Deliverables

We are expected to deliver a working prototype system for detecting defective paint jobs on pumps by December 2021. The system will be able to capture images of pumps, accurately classify pumps as defective or non-defective, and notify staff when a defect is detected.

2 Project Plan

This section describes our planned tasks and timeline of completion. Other than planned tasks and timeline of completion, the risk assessment is also described according to our planned tasks. We then enumerate the resources required to complete this solution.

2.1 Task Decomposition



2.2 Risks And Risk Management/Mitigation

Sample Set

Sample sets carry a moderate risk because the number of samples received or taken partially corresponds to the accuracy of the model. To ensure this risk stays low, Danfoss will supply us with 1000+ images of each pump set in high-resolution captured at two different angles.

Sample Set Risk Assessment: 2/10

Inaccurate Model

An inaccurate model can pose a high risk during the development process. By creating a descriptive list of statistics for each sample class, we can mitigate the risk while also increasing the overall accuracy of the design and model.

Inaccurate Model Risk Assessment: 7/10

Hard To Detect Defects

Hard-to-defect detects are difficult to spot with an untrained eye and can house a small amount of risk in the overall detection process. By using high-resolution capture, these defects can be easier to spot and detect in the model's vision system.

Defect Detection Risk Assessment: 3/10

Server Assignment

Server assignment will carry a relatively low risk as it is composed solely of server configuration.

Server Configuration Risk Assessment: 2/10

Software Development

Software development will house most of the product risk. Issues can arise early on in the development process that can later affect the data augmentation or imaging sequence. These issues then can have an overall effect on the implementation of the deep learning algorithm and eventual deployment.

Software Development Risk Assessment: 7/10

Testing

The testing phase will carry a relatively low risk. The major points of risk in testing arise from the model's learning. If images are labeled incorrectly, it can have an effect on the overall accuracy of the model. We will want to be careful not to wipe out any data or worse, destroying our model during the testing process.

Testing Risk Assessment: 4/10

Deployment

The overall deployment risk was mitigated due to the decision by Danfoss that the final solution will not be present on the assembly line or connected to the overall Danfoss network.

Server Deployment: Risk Mitigated

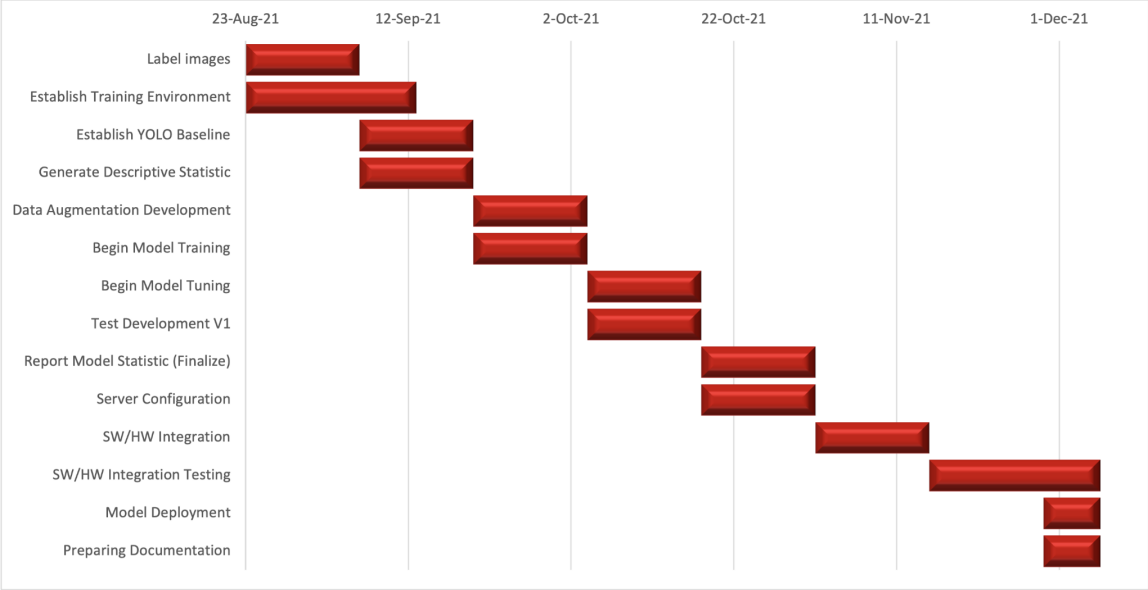
2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Communication is still ongoing with the client on what metrics or evaluation criteria are defined for the product. It is clear that a target of 40% of errors should be mitigated, but it is unclear what an acceptable false-positive rate would be for our alarm signaling or classification. We have chosen to aim for a recall of 0.8 and a precision of 0.7. This focus on recall should help reach the target of 40% defect reduction.

The remaining milestones to be met for Fall 2021 are:

1. Label images from client & configure training environment
2. Perform analysis on defect classes and data augmentation on the dataset
3. Train the model, tune the model
4. Test the pipeline, integrate software and hardware
5. Deploy the model, deliver analysis & model statistics

2.4 Project Timeline/Schedule



The above timeline assumes 2 week sprint periods ending and starting every other Monday.

2.5 Project Tracking Procedures

By utilizing GitLab, we can create multiple tickets for each project milestone that help us break down major sections into subsections that can be assigned to each member with a due date to keep development on track. Slack is being used for communication and sharing information among the group members. Additionally, we use Microsoft Teams and Zoom to meet with our client and faculty advisor, respectively, while email is used to communicate with our client, faculty advisor, and anyone else with whom we need to communicate outside of meeting times. Together, GitLab, Slack, Microsoft Teams, Zoom, and email provide us with our primary project management tools.

2.6 Personnel Effort Requirements

In general, each team member contributes approximately 8 to 10 hours per week towards the implementation of our defect detection system. These hours can be broken down into 3 categories: technical implementation, communication, and planning.

First, it is essential to the implementation of our solution that each team member contributes 3+ hours a week in technical development. Without technical development, the solution is solely theory and paperwork. Beyond technical development, 2 to 3 hours a week are spent on communications such as meetings, emails, and Slack messages. Without communication, our ability to prioritize tasks is lost. Lastly, 2 to 3 hours per week are put towards planning such as writing documentation or creating system diagrams.

2.7 Other Resource Requirements

There are few resources that we need in order to accomplish this deep learning solution. Predominantly, we need Danfoss to provide us with a dataset of more than 1000 images from each defect class with descriptions that will be used for the image labeling process. These two requirements will be vital for high accuracy model training.

To provide the aforementioned resources, we also need Danfoss to purchase image capturing equipment for which Danfoss plans to utilize in our prototype and their long term solution. Because the equipment is essential to our solutions viability, we have communicated requirements to the image capturing equipment and its orientation. The requirements for the image capturing equipment are that it needs to be FHD or higher resolution, have (reasonably) controlled lighting, and capture pump images from two controlled angles. Meaning, we require camera placement such that two cameras capture four total pump sides for each pump being analyzed.

As a stretch goal, Danfoss will provide us a temporary computing station that allows us to deploy our model and feed the captured images for real time analysis. It has been communicated that Danfoss IT would not allow an inference station long term, but it would be an excellent demonstration of our model's capabilities and we plan to deploy to their servers if given the opportunity.

Lastly, besides the resources from Danfoss, we require computing resources from Iowa State University ETG. Specifically, we need GPU compute resources to accelerate model training. Since the training time for high epochs is quite significant, utilizing Iowa State GPU will allow model tuning and testing at a much accelerated rate. Without ISU compute power, we anticipate that our solution accuracy will suffer; the training process from randomized weights will likely take 2-3 days with our anticipated sample set.

2.8 Financial Requirements

Danfoss will finance up to $5000 for the deep learning solution development. Therefore, all image capture and Danfoss-specific computing equipment related to this solution will be purchased and provided by Danfoss. Given the component requirements communicated to Danfoss, our component price estimates are below.

Component	Approximate Price
Camera Equipment: Two FHD Cameras	$1,000 x 2 = $2,000
Computing Station: PC, Display, interfacing	$2400
Stands: Cameras, PC, etc…	$600
Total Financial Estimate	$5,000

3 Design

3.1 Previous Work And Literature

For our solution, we have looked into various topics on deep learning algorithms, Convolutional Neural Networks, and the YOLO model for object detection. We plan to adopt these aspects and algorithms in a way to accurately detect the paint defects of pumps.

Deep Learning Algorithms:
A class of machine learning that uses multiple layers to extract higher features from raw input. In image processing, this can be used to identify image edges and various objects like faces, digits, or letters.

Convolutional Neural Network:
A deep learning algorithm that is able to take an input image and assign weights to various parts of the image. It has preprocessing built into the neural network. The convolutional network works to mimic the connectivity pattern of human neurons in the visual cortex. By using a convolutional neural network we can capture the spatial and temporal dependencies in the image.

YOLO Model:
You Only Look Once, the YOLO model uses real-time object detection that can achieve high accuracy while being able to run in real-time. It is able to train on full images and optimize its detection performance. This model is known to be extremely fast and intake the entire image in training to encode contextual information about classes and appearance, Redmon et al.

3.2 Design Thinking

There are several defining aspects of our defect detection model: low sample size, preprocessing techniques, and camera setup will play a significant part in making our solution unique.

Relatively low sample sizes combined with high feature samples will likely impact our design decisions more than any other factor. We are not yet on a finalized hardware setup, so anything is subject to change, but it is likely that we have to utilize several techniques to prevent our model from overfitting. Primarily, we believe that regularization to simplify our model and data augmentation to scale our sample size will allow a scalable solution when it would otherwise not be possible in this time frame. This is because the defect rate is relatively low and a dataset has not yet been built.

Assuming the augmentation & regularization work, some aspects to shape our design relating to preprocessing include: histogram equalization to help control lighting conditions, smoothing techniques to filter noise, and standardization to get data in a simplified range while maintaining outliers. We will use a YOLO model which is built for real time object detection.

Lastly, other setups and hardware-related aspects include several driving factors. First, multiple cameras to capture images of every surface will be needed; the cameras will be implemented on a standalone station that isn't connected to the Danfoss assembly network. Additionally, that station will require a physical alarm to sound when a defect has been detected.

3.3 Proposed Design

For Object-Detection, we are using YOLO for both our toy model and the design itself. From our research studies, the YOLO model processes images in real-time at 45 frames per second which means this algorithm outperforms the other detection algorithms, including DPM and R-CNN, when generalizing from natural images to other domains like artwork. Therefore, we think this algorithm would be beneficial for this implementation since Danfoss would need a quick algorithm in order to process multiple batches of pumps.

Like every Object Detection algorithm, YOLO requires a dataset in order to train the algorithm. So, first and foremost, we would need to gather images from our client. After gathering the images we received from the client, we would need to train our algorithm with the dataset we have obtained. The performance of the algorithm relies greatly on the number of images, the number of epochs being used, and of course, the accuracy of the object detection algorithm.

YOLO reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time, so it implicitly encodes contextual information about classes as well as their appearance. Fast R-CNN, a top detection method, mistakes background patches in an image for objects because it cannot see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN, therefore, hit our objective to help Danfoss with reducing the number of defective units and together, reduce the overall monetary loss.

3.4 Technology Considerations

There are many Object-Detection Algorithms such as Fast R-CNN, Faster R-CNN, Histogram of Oriented Gradients (HOG), Region-based Convolutional Neural Networks (R-CNN), Region-based Fully Convolutional Network (R-FCN), Single Shot Detector (SSD), Spatial Pyramid Pooling (SPP-net), and You Only Look Once (YOLO).

However, for learning solutions, we have decided to use YOLO as our Object-Detection Algorithm. Our reasoning is touched upon in the following paragraphs and concluded in the technology considerations conclusion. Note that we only made a comparison between R-CNN and YOLO in this section.

R-CNN:

R-CNN generates approximately 2000 region proposals using selective search from a given image that are all manipulated & reshaped to be the same dimensions and then each are given to its own convolutional neural network. The CNNs generate feature vectors from each region proposal, with features such as texture, color, etc. to be considered. The feature vectors are then used by a support vector machine to determine what category the region proposal is. Bounding boxes are then created by the confidence rating of each of these region proposals.

Benefits:

1. Higher accuracy for smaller objects
    a. Since R-CNN uses various regions to combine predictions, therefore, it is better at analyzing several smaller regions.
2. Good localization due to use of various regions

    a. The varying resolution on the region proposals leads to increased capabilities of detecting objects of different sizes and locations

Disadvantages:

1. Large number of CNNs results in slow processing time
2. Large dataset needed
    a. The need for complex data analysis suggests that R-CNN needs a large sample size to train an accurate model

YOLO:

YOLO Network divides images into grids with G x G cells, and the grid then generates N predictions for bounding boxes (G x G x N boxes in total). Each bounding box is limited to having only one class during the time of prediction, which restricts the network from finding smaller objects.

YOLO unifies the task of object detection and the framing of the detected objects as the spatial location of the bounding boxes is treated as a regression problem. As of this, the entire process of calculating class probabilities and predicting bounding boxes is executed in one single Artificial Neural Network (ANN), which enables optimized end-to-end training of the network, and enables the YOLO network to perform inference in up to 45 FPS.

Benefits of using YOLO:

1. High-speed
    a. YOLO is quite fast and can process 45 frames per second.
2. Image Generalization
    a. YOLO network is able to generalize the image better.
3. Scaling Capabilities
    a. YOLOv5 architecture offers small, medium, large, and XL networks for varying resolution and dataset requirements
    b. YOLOv5 is demonstrably capable of various types of input sequences

Disadvantages of using YOLO:

1. Comparatively low recall and high localization error.
2. Struggles to detect close objects because each grid can propose only 2 bounding boxes

For our given use cases of a single pump analyzed from a static position, the disadvantages of the YOLO model are inconsequential. YOLO also allows for continuous, real-time analysis which meets the requirement to stop the manufacturing process to prevent the continued defective painting of pumps. R-CNN is too slow for this application.

3.5 Design Analysis

After performing several tests with around 106 images of defective and non-defective (see *Section 4.4* for result), the YOLO model has demonstrated capabilities at identifying non-defective and defective pumps. As shown in Section 4.4, we are able to detect both defective and non-defective pumps.

Although we are able to detect defective units, however, we think that we would have problems identifying a low-paint pump (shown below), which would also be considered as a defective pump.

The three dotted circles shown in the above diagram are low-painted areas that consequently classify the pump as a defective unit. One possible method to solve the low-painted area detection is to place bounding boxes only around defective regions. The problem with the aforementioned method is that this method does not work logically with pumps that have no paint.

Another possible method is to train a high-resolution R-CNN to detect defects on the low paint pumps. Unlike YOLO, this method could not be high FPS (Frame per Second). Therefore, training with a high-resolution R-CNN would require a more expensive setup with sensors and a strong computing station. Our proposed solution to detecting challenging defects is to reduce the bounding box area of interest to the specific defect present in the pump. In effect, no paint pumps will have bounding boxes enclosing the pump entirely, but pumps missing paint in only one small region will have bounding boxes enclosing that region only.

3.6 Development Process

For the detection of paint defects, we have decided to follow an agile development process. Agile development allows us to maintain a small buffer of research-orientated features that can be implemented in the future. By following this process we are also able to retrospectively implement continuous improvements to the solution and its overall development. It also allows us to involve the client directly in the development process allowing for the knowledge of individual components to be shared among team members.

For our specific solution, each part will have its own milestone. For example, the YOLO model will have its own milestone separate from the rest of the solution. From this, we will move on to getting the basic model setup, integration of the model, and then testing.

3.7 Design Plan

For the development process, we are following Agile, which is based on iterative development. Requirements and solutions evolve through collaboration between cross-functional teams, therefore, we think this development process would be strong for us and our client. With Agile development, we can focus on developing one milestone at a time. For instance, the developer team

can work on installing the hardware such as the cameras before we can move on to the next development phase.

For the base of our model, we will be using YOLO. This allows us to use real-time analysis at 45 frames per second. YOLO will also be able to generalize the image better and faster than other R-CNN models. While there are disadvantages to using YOLO we can mitigate the most prevalent risks through tuning, training, and testing with our samples Danfoss has provided to us. Following our Gantt chart, we will establish a YOLO baseline then move on into generating a descriptive statistic which will make it easier for us to execute augmentation development and model training.

4 Testing

Testing is an important part of the design process. For our model, we will run hardware and software tests to make sure the accuracy of the YOLO model matches our requirements with the pumps supplied to us. Danfoss samples were labeled then ran through YOLO where it was detected as either defective or non-defective allowing us to tune the toy model's object detection for presentation to Danfoss.

4.1 Unit Testing

Hardware Testing:

- **Image Collection:** We need to verify that the overall quality of pump images captured by the camera(s) is sufficient for the models to process.
- **Buzzer:** The buzzer will need to be tested to ensure that it produces sound and that the sound it produces is loud enough to be heard on the floor. This testing will also include verifying that the wiring for the buzzer and the controller for the buzzer function properly.

Software Testing:

- **Pump Identification:** We need to ensure that our system will recognize the pumps. This will be verified by manually adding metadata to images that specify the appropriate bounding box to capture the pump in the image and then verifying that the model identifies pumps within this bounding box with a border error of no more than 5 pixels on any border in any direction.
- **Pump Classification:** We need to ensure that the model correctly identifies pumps as defective or non-defective. This will be tested by reserving a set of data for testing which the model will not see during training. After the model has been trained, and it passes initial evaluations using the training data, then it will be tested on the testing data. The model will need to demonstrate an accuracy of at least 80%.

4.2 Interface Testing

The camera(s) will be connected to the model so that the image collection to model processing pipeline can be tested to see the impact on model accuracy. The model will be connected to the buzzer to test that when a defect is detected the buzzer is activated.

We have three distinct units: two cameras, a buzzer, and our central computer. Each of these units must be able to interface between itself and the computer.

4.3 Acceptance Testing

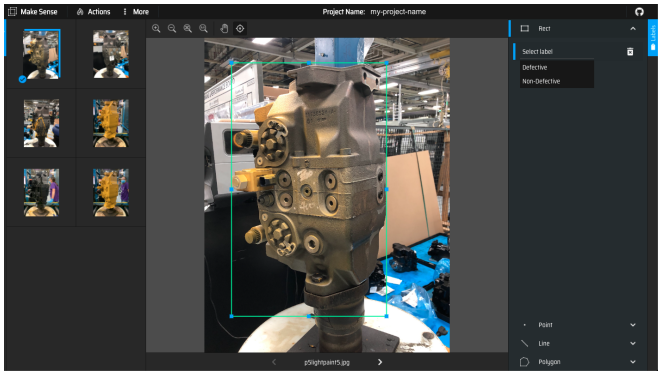
In order to make sure that we meet our requirement specifications, we have demonstrated a YOLO v5 Object Detection Toy Model to our client, Danfoss.

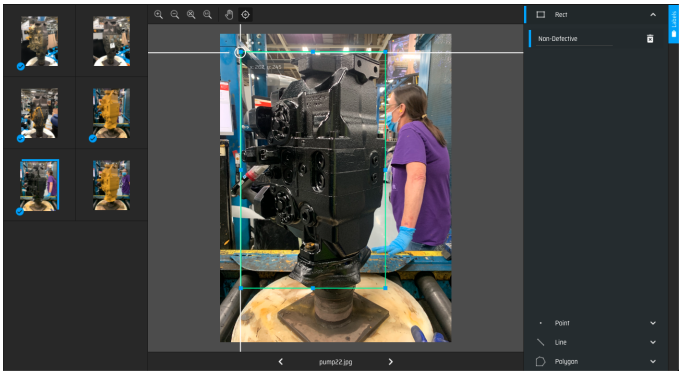
For this to be carried out, Danfoss needed to provide us with a few defective and non-defective image samples. Once Danfoss had provided us with the images, we then performed image labeling for defective and non-defective pumps. Lastly, we trained YOLO v5 with the toy dataset to obtain overfitting. We believe this is a solid representation of model learning.

4.4 Results

We have only received a few pump images from our client for our toy model which are about 54 non-defective pump images and 26 defective pump images. Therefore, we were only able to perform elementary defect detection testing with YOLO v5 due to the limited sample size. Below are sample results from our image labeling and dataset training.

Image Labeling Process





The above images show the labeling process using makesense. In the AI interface, we draw a border tight around the pump in the image then assign a label to the image to designate a defective pump with a 1, and a non-defective pump with a 0.

Image Labeling Result

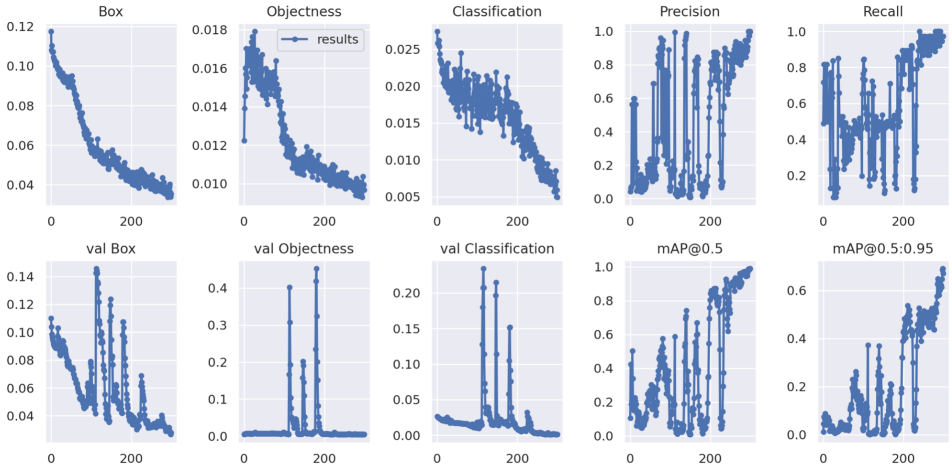
`<x> <y> <width> <height>` - float values relative to width and height of image, it can be equal from (0.0 to 1.0]

```
0 0.401350 0.442368 0.631360 0.769470
```

```
1 0.431624 0.460684 0.645014 0.729915
```

The labels, in the format above, are then converted into YOLO format to be used in the prototype model.

Model Performance Visualization



Object Detection Result

5 Implementation

The implementation plan for Fall 2021 has been continually revised and developed such that it revolves entirely around our Gantt chart and functional decomposition, broken down into 2-week development sprints. Tasks with higher risk potential have a focus on completion earlier in the semester: tuning, labeling; tasks that are depended upon heavily by other tasks are prioritized for the beginning of the semester: configuration, data augmentation, generating statistics.

The development of our solution can be broken down into 5 logical phases:

1. **Label images from client & configure training environment**
    a. We plan to receive a large number of images from our client to use for training and testing the model; the only issue is that the samples are not labeled. Anticipating 5000 images to label, considerable time will be invested into discarding problematic images, labeling retained images, and asking the client to confirm the labels. Our conservative estimates are that labeling 5000 images will take around 30-40 hours considering data handling, etc.

    b. At the beginning of the Fall 2021 semester, we will email ETG requesting computing resources utilizing ISU GPU compute power. This will help reduce training times from several days to as short as a day or two, hopefully. Beyond that, we will link our training environment up with our GitLab repository for efficient deployment, CICD if necessary.

2. **Perform analysis on defect classes and data augmentation on the dataset**
    a. Assuming images are labeled as needed, and our prototype is deployed to an ISU server with a capable GPU, we will begin generating descriptive statistics for each defect class and the data augmentation process. It should be feasible to gather histogram, mean, median, range, and more data for each class and represent them in a digestible format before we begin model training and tuning. Additionally, we seek to perform data augmentation such that 1 sample image can give us 5-10 input images. This should scale our dataset by 5x on a conservative estimate; hopefully, this will allow us to train on high-resolution images with a large YOLOv5 architecture.

3. **Train the model, tune the model**
    a. After labeling and augmenting the images, we will train the model. The training process requires a high-performance computer to train all the labeled images with a high epoch and will take a considerable time to compute. Our conservative estimates are that our dataset will be around 20,000 images after data augmentation. With this input set, we should have no issues tuning the model with a significant verification set. Of course, we will look for the model detecting significance in the areas we previously generated descriptive statistics for. Efforts will be made to ensure the model is not severely underfitting or overfitting.

4. **Test the pipeline, integrate software and hardware**
    a. This process will be performed in parallel with training and tuning. Assurance will need to exist that our data augmentation was performed correctly and that our pipeline is not behaving in any surprising ways. Introductory software-hardware integration will be made to begin efforts to deploy the model to the client-server.

5. **Deploy the model, deliver analysis & model statistics**
    a. Lastly, we will be deploying the model to a Danfoss server environment capable of interfacing with the assembly line if Danfoss IT clears our solution. During this process, we will be delivering our model statistics as well as the descriptive statistics we gathered during stage 2. We want clear communication on our findings during training and development that Danfoss can utilize and replicate in their final solution.

6 Closing Material

6.1 Conclusion

We have researched and decided upon using YOLO as our solution to the problem of real-time analysis of visual data. We have established a planned course of action of Danfoss collecting data over Summer 2021 and our training of the YOLO model Fall 2021. Our goal is to provide Danfoss with a solution to reduce the number of pump paint defects by 40% by providing early detection of defectively painted pumps. We have chosen YOLO as our model because it generalizes the image. This allows YOLO to be faster, operate effectively at a smaller dataset, and is designed to be used in a real-time environment.

6.2 References

B. Dwyer, "Using Your Webcam with Roboflow Models," *Roboflow Blog*, 30-Mar-2021. [Online]. Available: *https://blog.roboflow.com/python-webcam/*. [Accessed: 21-Apr-2021].

Open Data Science, "*Overview of the YOLO Object Detection Algorithm*", Medium, 25-Sep-2018. [Online]. Available: *https://medium.com/@ODSC/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0*. [Accessed: 21-Apr-2021].

Johanness Sjolund & Johanness Ronnqvist, "A Deep Learning Approach to Detection and Classification of Small Defects on Painted Surfaces", Master Thesis, Industrial Engineering and Management, Umea University, Umea, Sweden, 2019. [Online]. Available: *https://www.diva-portal.org/smash/get/diva2:1325010/FULLTEXT01.pdf*. [Accessed: 21-Apr-2021].

J. Solawetz, "How to Train YOLOv5 On a Custom Dataset," *Roboflow Blog*, 02-Mar-2021. [Online]. Available: *https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/*. [Accessed: 21-Apr-2021].

Redmon, Joseph et al. "You Only Look Once: Unified, Real-Time Object Detection." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 779-788.

Saha, Sumit. "A Comprehensive Guide to Convolutional Neural Networks - the ELI5 Way." *Medium*, Towards Data Science, 17 Dec. 2018, towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

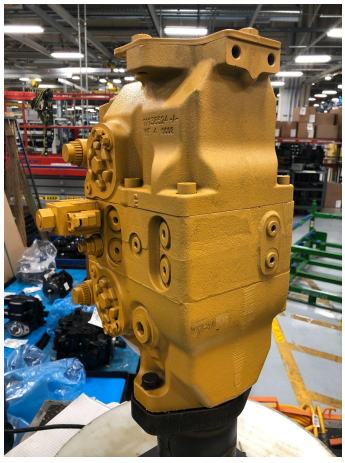
6.3 Appendices

There are a few classifications to which a pump can belong. The ideal classification is non-defective, but there are a few common defective classifications. The main defective classifications are low paint, no paint, splatter paint, and orange peel. Low paint is when a pump is partially painted but not completely painted. No paint is when a pump was not painted at all. Splatter paint is when the paint only appears in streaks. Orange peel is when the paint is not smooth and has a bumpy surface like that of an orange peel.

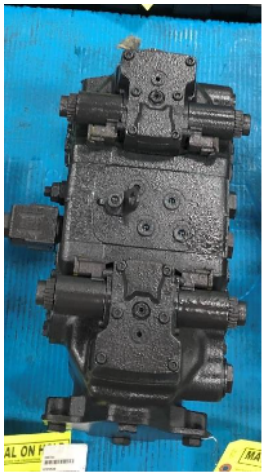
Non-defective

Low paint

Splatter paint



Low paint



Orange peel

No paint