Machine Learning for Semi-Autonomous Maintenance in a Hazardous Environment

Frantsishak Akulich, Christiane Alford, Mazdak Khoshnood, and Muhammad Ishaq Memon
University of Illinois at Chicago, Chicago, IL, 60607

Implementing maintenance automation with machine learning (ML) in a hazardous environment saves time and money, but also decreases human exposure to life threatening conditions. The goal is to develop a robotic system that semi-autonomously performs maintenance on radioactive parts of an accelerator beamline enclosure at Fermi National Accelerator Laboratory, specifically for the Neutrinos at the Main Injector (NuMI) Project. The first step in this project is the creation of an ML assisted device capable of detecting nuts and bolts on the beamline's front window flange. Using an NVIDIA microcontroller and a 3D stereo camera, a deep learning (DL) model based on convolutional neural networks (CNN) is trained on a custom dataset of 3D images representing flanges captured by means of set camera paths from a rendered SolidWorks model. The final product will comprise a model based on SSD Inception V2, the Intel Realsense D435 depth camera, and a Jetson Nano microcontroller housing all necessary software and python scripts for object detection. From preliminary results for TensorBoard, the model trained using the dataset of SolidWorks images detects nuts and bolts with an average precision and recall of 74.5% and 61% respectively. This translated into the model detecting nuts and bolts on a 3D printed flange in real-time within a classification score of 80% to 90%. These results demonstrate that a model can be accurately trained using rendered CAD image training data and can be generalized to real world applications. By including depth analysis to the object detection script, the distance between the target and the camera is obtained and used to create a 3D coordinate system. The current work forms the basis for future development of ML and DL techniques for robotics applications.

Table of Contents

MACHINE LEARNING FOR SEMI-AUTONOMOUS MAINTENANCE IN A HAZARDOUS ENVIRONMENT I

TABLE OF CONTENTS II

LIST OF FIGURES 1

LIST OF TABLES 2

NOMENCLATURE 2

I. INTRODUCTION 3

A. Problem Statement 3
B. Sponsor Background 3
C. Literature Survey 3
D. Design Criteria 4

II. TECHNICAL CONTENT 5

E. Assumptions 5
F. Metrics 5
G. Proposed Solutions and Selected Design 6

III. METHODOLOGY 8

H. Experimental Method 8
1. Machine Learning Model 8
2. Creation of Dataset 10
3. Intel Realsense D435 12
4. Custom Object and Coordinate Detection Script 13
I. Technical Drawings 14
J. House of Quality 15

IV. RESULTS 16

V. CONCLUSIONS AND FUTURE WORK 19

APPENDIX 19

ACKNOWLEDGMENTS 21

REFERENCES 21

List of Figures

Figure 1. Example of image of a hex nut labeled for the first dataset. 10
Figure 2. Example of image of the back of the 3D printed flange. 10
Figure 3. Example of image of the flange from third dataset. 11
Figure 4. Example of images of the flange from SolidWorks dataset with no background (left) and applied background for increased accuracy (right). 11
Figure 5. Ground truth (right) is compared to detections (left) where the model confuses the grass for a nut with a classification score of 72%. 12
Figure 6. Example image from final dataset using a background with varying features. 12
Figure 7. Flow Chart of Object and Coordinate Detection Algorithm 14
Figure 8. Rendered CAD model of 3D printed, prototype flange with nuts on the back of the flange (right) and bolts on the front of the flange (left) inserted into holes. 15
Figure 9. Ground truth (right) is compared to detections (left) from a SolidWorks image of a flange with nuts where classification probabilities ranges from 79% to 99%. 18
Figure 10. Ground truth (right) is compared to detections (left) where the model assesses that there are no nuts or bolts in the image. 18
Figure 11. Ground truth (right) is compared to detections (left) from SolidWorks image of a flange with bolts where classification probabilities are all 99%. 18
Figure 12. Output of image of flange with detection boxes labeling the detected nuts and 3D coordinates of each nut with the camera lens as the origin. 19
Figure 13. Excerpt from NuMI target assembly drawing. 20
Figure 14. Detailed drawing of window flange from NuMI target. 20
Figure 15. Detailed drawing of prototype flange for 3D printing. 21

List of Tables

Table 1. Decision Matrix for Microcontroller 7
Table 2. Decision Matrix for Camera 7
Table 3. Decision Matrix for Model Architecture 8
Table 4. House of Quality 16
Table 5. Results from Training TensorFlow Model with 24-hour time limit and 20,000 steps 17

Nomenclature

CNN	=	Convolutional Neural Network
DNN	=	Deep Learning Neural Network
CAD	=	Computer Aided Design
SSD	=	Single-Shot Detector
ML	=	Machine Learning
DL	=	Deep Learning
3D	=	three dimensional
AI	=	Artificial Intelligence
GPU	=	Graphics Processing Unit
CPU	=	Central Processing Unit
API	=	Application Programming Interface
CSI	=	Camera Serial Interface
USB	=	Universal Serial Bus
tf	=	TensorFlow
RAM	=	Random-access memory
HDMI	=	High-Definition Multimedia Interface
SFTP	=	SSH File Transfer Protocol

I. Introduction

The goal of the project is to use both computer vision and machine learning to identify and locate bolts on flanges which may be arbitrarily oriented in space in a hazardous environment. First, the principles of machine learning and the various applications and techniques for which it can be used, as well as, a review of various patents of similar scope are examined so as to avoid any infringements of copy right for the duration of this project. Following this information, a full methodology of the structure and implementation of the design and experimental procedure is performed and preliminary results are shown in this report.

A. Problem Statement

The sponsor of this project, Fermilab, wishes to implement both computer vision and machine learning to identify and locate nuts and bolts which may be arbitrarily located oriented in space. Due to the highly radioactive conditions that are found in the environment where the various projects at Fermilab are conducted, it is too dangerous to have humans perform any maintenance on the various instruments that are key to the continuity of said projects. Therefore, by having a robot take over to perform this maintenance, the project life cycle could potentially be extended and the likelihood of human exposure to radioactive materials would be decreased. The implementation of machine learning allows for the system to be adaptable to the environment in which is it placed, as allows for less immediate human supervision over the tasks to which it is being applied to.

B. Sponsor Background

Established in 1967, Fermilab is the United States' leading particle physics laboratory, working with advanced particle accelerators to enhance the understanding of the world and universe. Fermilab aims to lead the world and nation in neutrino science, and development of particle colliders for further development of scientific knowledge. One of the many projects currently underway at Fermilab is the Neutrinos at the Main Injector Project, otherwise known as the NuMI Project. NuMI is a set of facilities that are being used to search for neutrino mass by looking for neutrino oscillation [1]. Currently, the lab is building a new particle beamline to direct a nearly pure beam of neutrinos from the Main Injector towards two detectors which will be able to count the neutrinos and all three subtypes of neutrinos and further determine the existence of the particles' mass [1].

The project described in this report is a step towards the objective of having a robotic system semi-autonomously perform maintenance or end-of-life procedures on various highly radioactive components from the newly constructed accelerator beamline enclosure for the NuMI Project. By creating a computer vision system with machine learning to locate fasteners, a robotic arm could then be programmed to move in proximity of and further remove the fasteners from the accelerator beam which would eliminate the possibility of human exposure to radiation.

C. Literature Survey

This section of the report is comprised of the analysis of the various concepts of machine learning that will be implemented for the duration of the project. Literature reviews of deep learning, image classification, object detection, image segmentation, transfer learning, and convolutional neural networks follow below. Following this, an analysis of various patents that may be along the scope of this project is conducted to determine how our project contributes to the greater body of creation and engineering.

In the paper "Deep Learning for Computer Vision: A Brief Review," Voulodimos describes deep learning and the different schemes used for machine learning [2]. The concepts of deep learning and Convolutional Neural Networks are examined specifically in this report for the project. Deep learning is defined as a technique used in computer vision that aims to mimic the human brain's neural network system. By creating computational models with multiple processing layers, this allows the machine to learn and represent data with levels of abstraction that imitate how the brain perceives information [2]. Deep learning can be applied for computer vision problems such as image classification, object detection and image segmentation which will be discussed later in this report.

The paper goes on to discuss the deep learning model, Convolutional Neural Networks (CNNs) which is shown to be very successful when applied to computer vision applications – it should be noted that all nomenclature, including CNN, is described on page 3 of this report. CNN is broken into three main neural layers. These layers are (1) convolutional layers, (2) pooling layers, (3) fully connected layers and each layer has a different role in the entire network and each layer is connected to the next [2]. In using a CNN model with an established architecture, it is subjected to pretraining which initializes the network with pretrained parameters so as to accelerate the learning process and enhance the overall capability of the network [2]. It is concluded that CNNs outperform traditional

machine learning approaches in computer vision and recognition tasks while also being relatively easier to train. Because of this, the CNN model to deep learning will be used for the project at hand.

As stated by Voulodimos, "object detection is the process of detecting instances of semantic objects of a certain class in digital images and video," [2]. To give a more colloquial depiction, object detection essentially takes an image, draws a box that bounds around the target object (or multiple instances of said object) [3]. Object detection can also work as an image classification system which takes the objects in the image and classifies then in the class to which they belong. For the purpose of the project, nuts and bolts are the class of objects that the model is being programmed to search for and the model is to classify the nuts and bolts separately on the flange. However, the limitation of object detection is that it will not send any information to the program about the shape of the object which is where image segmentation can be useful.

Image segmentation takes a given image and "draws a pixel-level mask on a specific object" and by doing so allows the model to receive more information about the size and shape of the object that is being detected via machine learning [3]. This can be useful for the project moving forward with potentially programming a robot arm to move to remove the nuts and bolts from the flange on which they lie. It is also found that by training a model that to simultaneously use image classification, object detection and image segmentation the model is able to do each task more successfully [3]. Therefore, for the machine learning project at hand, the model will be trained for all the tasks above.

Building a model from scratch can be incredibly time consuming and therefore unrealistic for the timeline of this project. Therefore, transfer learning will be implemented based on pre-trained models. These pre-trained models are models that have been trained to perform a similar task to the model that is desired and therefore can be modified to be used for the task at hand. According to Yosinski, Clune, Bengio and Lipson, transfer learning can actually build more accurate models while also saving time [4]. This is because it is computationally time consuming to train a model, therefore repurposing a pre-trained model is done. There are three methods by which one can repurpose the pre-trained model (1) train the whole model, (2) train some layers in the CNN and freeze the others or (3) freeze the convolutional base. Based on the type and size of the dataset being used, one of the three methods is chosen [4]. As the dataset for the project is built, one of the methods above will be chosen and implemented.

The first patent being reviewed is a robot implemented item manipulation [5]. It involves an inventory managing robot that analyzes a set of items for the target to be manipulated. It uses computer vision to identify the target, determines how to pick it up, and where it needs to be placed. The robot can determine if it cannot perform the task and requests human assistance. The patent does not specify what type of robot will perform the manipulation, or what type of camera will be used to determine the location of the target item [5]. The hybrid human/robot solution used here is relevant to the project at hand's design as computer vision is to be used (specifically convolutional neural networks) to identify a target that needs to be manipulated by a robotic arm, then a human will take over to perform the manipulation. In the project's case only one class of item is to be identified instead of multiple.

A process of the installation of nuts and collars onto fasteners has previously been created [6]. The process mainly highlights the use of these robots in the aerospace industry. The device uses a robot supporting a mounting base and carry a gimbal and rotary drive to make up a lightweight sensing robot with a fastening/torque tool [6]. The process is relatable to our project as a device is to be mounted on a robotic arm which will eventually be program to remove the nuts and bolts from a flange once the target objects have been identified.

Deep machine learning methods and an apparatus has been patented that will be related to object manipulation by the end effector of a robot. By applying the machine learning method with the training of a deep neural network, the system will be able to predict a measurement to be relayed to the robot arm which ends in the successful grasp of the targeted object [7]. This relates to our project as we will be training a neural network and having the information sent to a robotic arm to move towards the located nuts and bolts.

Another patent was published describing a semantic-labeled representation of 3D space taken from data from a video. The objects to be located in space are defined within a 3D coordinate system which relates to our project as our camera system will be set up with a common 3D coordinate system for space mapping of the environment that our device is to be placed in [8].

A machinery arm has been invented with a visual spatial attention method that is based on convolutional neural networks. This includes the collection of target object visual data, the training of the CNN model, and finally the verification of the model [9]. It relates to the project by means of using a CNN to train a model and have that information sent to a robotic arm for industrial use.

D. Design Criteria

This section outlines the necessary design criteria. These criteria are based on the sponsor's needs and the overall goals for the project and are separated by the overall design requirements as well as the desired end product. The

design requirements are the functional requirements and specifications of the device being created. These requirements are that the device must be able to accurately differentiate between bolts, nuts and various other features that may be present on the bolt circle. The device has to be able to determine the location and orientation of the previously identified fasteners with reference to a common coordinate origin. Finally, the device is required to be able to be scalable or otherwise applicable to different sizes of robot arms and varying sizes of nuts and bolts.

The desired end product must be mountable on a robotic arm and must be capable of enclosing all the necessary hardware for the computer vision and machine learning tasks. The end product is also comprised of the software that will be able to identify and locate arbitrarily oriented nuts and bolts in 3D space. Finally, the end product will include the programming of a robotic arm to move in proximity of the location of nuts and bolts on a bolt circle.

II. Technical Content

In these subsections, the method followed to complete the project is presented. The subsections are broken down into the project assumptions, metrics and proposed solutions which all lead to the selected design of the project.

E. Assumptions

Before moving to the design process and design of experiments of the projects, assumptions must be established so as to narrow the scope of the project. By setting limits to the extent to which the project is conducted, a higher success rate for the project is established for the given timeline. The following section describes the assumptions that are made in regard to machine learning for semi-autonomous maintenance in a hazardous environment.

The first assumption made is that the device will be first be comprised of a microcontroller in order to fulfill the design requirements of being mountable on a robotic arm and capable of enclosing all necessary software. It must have enough storage capacity to house all necessary libraries and packages that will be used to create a machine learning and computer vision system. The two microcontrollers that will be considered for the project are the Jetson Nano and Jetson TX2. Comparisons of these devices are found in the Proposed Design Solutions in this report.

Another assumption for the project is that the camera to be used is a 3D stereo camera. This assumption is made due to the design requirement that the device must be able to determine the location and orientation of the previously identified fasteners to a common coordinate origin – which cannot be determined using a 2D camera. Furthermore, the underlying principle of deep learning and using CNNs is based on the human brain and human experience and the human experience is based on a three-dimensional reality [10]. Therefore, having a 3D stereo camera collecting data on the environment would reflect a more accurate image of the camera's surroundings and allow for the creation of a virtual environment to base a coordinate system about which a robotic arm can be adjusted to move towards the nuts and bolts detected by the camera in 3D space [10]. It can be concluded that by using both CNNs and 3D stereo camera, the vision system would be able to create a more accurate model of reality which would then allow the entire system to be more accurate. Another assumption is to be taken that while programming the robotic arm, the camera must be set to lie within this range of distance from the targeted nuts and bolts on the bolt circle.

Since the goal of the machine learning device is to perform maintenance and end-of-life procedures on highly radioactive components from an accelerator beamline enclosure, it can be assumed that the device attached to the robot arm will only be used once before being becoming obsolete and having to be thrown away with all other contaminated parts of the beamline and maintenance devices. Therefore, the device, software and hardware must be easily reproducible for future maintenance projects of similar scope. This can be done so by backing up the system to a separate hard drive that can be uploaded onto other Nvidia Jetson devices as the project uses various scripts and programs that are not unique to the device that will be presented in this project.

Finally, after conducting various literature and patent reviews from the Introduction section above in this report, transfer learning will be applied to the machine learning process for the first test at creating the machine learning system. Since many models have already been created and published for tasks of relatively similar scopes to this project, a model will be chosen based on accuracy and an optimized time for training the model, and transfer learning will be applied by means of creating a unique dataset of nuts and bolts and training the premade model to detect the objects presented in our project's dataset.

F. Metrics

Since the scope of the project pertains to more involvement with creating programs and applying already established models and scripts to the unique task of locating nuts and bolts, the key metrics of the project are the percent accuracy of the object detection program, the time taken to detect the fastener to which the camera is pointing to, and the distance between the location of the camera to the located fasteners which is determined by the

range by which the stereo camera can accurately detect an object in its field of view. The costs of the project solely pertain to the cost of the Jetson Nano, the Intel Realsense D435 depth camera, and the various accessories needed to use the microcontroller, and the cost of the robot arm which will be determined by our project advisor, Jon Komperda, as the 3D printed, prototype bolt circle was printed by the UIC on-campus facility, the Engineering Makerspace, and the 3D stereo camera and Jetson TX2 were provided to the team by Fermilab. Therefore, the price of the project is fixed to the prices of these devices.

One of the most important metrics for the project is how accurately the system will be able to detect a nut or bolt on a bolt circle. The accuracy of the object detection prediction is directly linked to the success of the overall project. Therefore, optimizing the accuracy is a key metric for the project – the higher the accuracy, the higher likelihood of successful object detection. However, the more accurate the model, the more RAM is required to perform the training which can eventually lead to the Jetson Nano crashing. Therefore, this must be taken into consideration when choosing which model to use for training the machine learning process. This can be seen as the SSD Inception V2 Model has a range of accuracy falling between 85 and 95%, however, this model requires such a large amount of RAM that it may crash the GPU or CPU used for training. By using the SDD Lite Mobilenet V2 model, the accuracy falls between 60 and 85% and the system is more likely to properly complete the training process. Both the SSD Inception V2 Model and the SSD Lite Mobilenet V2 model will therefore be tested for training on the Jetson Nano, a regular CPU laptop, and on UIC's supercomputer, Dragon. This will test to see how long it may take to train each model using the various resources presented. Once the models have been trained, the accuracies are visualized in TensorFlow's TensorBoard. This is done by comparing the bounding boxes around the targets set at "ground truth" compared to the boxes around the object detected by the newly trained model. Finally, the loss function, precision and recall will be compared between the two models to determine which would be better used for the project.

The time taken to detect the fastener to which the camera is pointing to is very much dependent on the model being used for transfer learning with the dataset pertaining to the project. However, using TensorFlow allows for access to optimizer options, or tf.OptimizerOptions, which can be applied to the results presented in the Tensor Graph once training has already been attempted and optimized certain TensorFlow operations to save time. By testing both custom models with the camera, it can be determined which model is superior based on the time it takes to accurately detect the target objects.

Finally, the distance between the location of the camera to the located fasteners, which is determined by the range by which the stereo camera can accurately detect an object in its field of view, is measured and is to be taken into account when determining the accuracy of the coordinate detection of the computer vision and machine learning system. This can then be used to program a robotic arm to move in proximity to the nuts and bolts on the bolt circle. The camera acquired from Fermilab, the TaraXL stereo camera, has a detection range from 50 cm to 300 cm from the lens of the camera, compared to the Intel Realsense D435 which has a detection range of 10.5 cm to 1000 cm. Therefore, when establishing a coordinate system using the point cloud software for space mapping from either 3D stereo camera, these ranges must be taken into account. This will also depend on the distance between the end of the arm that will point to the located fastener and the location of the camera lens on the device attached to the robotic arm.

G. Proposed Solutions and Selected Design

After reviewing the literature, existing patents and taking the above into consideration, it can be determined that though many different machine learning applications exist in the realm of image classification, object detection and image segmentation, nothing has yet to be applied to the specific task and environment for this project. From this, a decision matrix, Table 1, was created for choosing between the Jetson Nano and the Jetson TX2 as the microcontroller for the project. The memory capacity as well as compatibility with the two provided 3D stereo cameras were given the highest scores of 5 as these are key for moving forward with the project. The storage capacity was given a score of 3 as memory is more vital to the project compared to storage. The cost was given a low score of 2 as both devices were provided either by UIC or by and Fermilab. The size was given a score of 4 as Fermilab wants the device to be able to be connected to a robotic arm.

Table 1. Decision Matrix for Microcontroller

Weight	4	2	3	5	5	19
Percentage	21%	11%	16%	26%	26%	100%
Option	**Size**	**Cost**	**Storage**	**Memory**	**Compatibility**	**Score**
Jetson Nano	Handheld size but would be difficult to attach to a robot arm	Cheap ($99)	128GB micro-SD	4GB LPDDR4, 25.6 GB/s	Not compatible with Tara3D stereo camera. Compatible with Intel Realsense D435 camera	Provided by UIC
	50	80	70	35	85	62
Jetson TX2	Larger and would not be able to attach to a robot arm	About four times as expensive as the Nano, but provided by Fermilab ($399)	32 GB eMMC 5.1	8GB 128-bit LPDDR4, 58.3 GB/s	Compatible with Tara3D stereo camera	Provided by Fermilab
	30	20	50	70	90	58
Notes:	High = smaller size	High = low cost	High = more storage	High = more memory	High = better compatibility	

Next a decision matrix, Table 2, was made to compare the different types of cameras that may be used for the object detection. The resolution was given the highest score of 5 as this is most important when having the system detect the target objects. The cost was given a score of 2 as the cameras were all provided by either Fermilab or fell within the budget for UIC for the project. Compatibility was given a score of 3 because the camera chosen must be compatible with the Jetson interface, however, most cameras are compatible, so this was not a problem to overcome. The size was also given a score of 3 as the camera will be attached to a robot arm and must be able to move with the robot arm without inhibiting the motion.

Table 2. Decision Matrix for Camera

Weight	5	2	3	3	13
Percentage	38%	16%	23%	23%	100%
Option	**Resolution**	**Cost**	**Compatibility**	**Size**	**Score**
TaraXL stereo camera	1512 x 480	$349	Compatible with TX2	Fits on top of Nano (100 mm x 30 mm x 35 mm)	Provided by Fermilab
	75	20	80	75	67
Raspberry Pi	Terrible resolution	Cheap	Easy to set up and compatible with all Jetson platforms	Very small and fits in slot on Nano box	Disqualified for bad resolution
	5	70	80	80	50
Intel RealSense D435	1280 x 720	$179	Compatible with TX2 and Nano	Fits on Nano (90 mm x 25 mm x 25 mm)	Provided by UIC
	80	60	80	76	76
Notes:	High = better resolution	High = low cost	High = more storage	High = more memory	

Next a decision matrix, Table 3, was made to compare the different types of model architectures that may be used for training the custom machine learning model using transfer learning. The average recall and precision were given the highest scores of 10 each as these are equally important when having the system correctly detect and differentiate between the target objects. The speed of detected the objects was given a score of 5 as this was subject to if a human eye may notice a notable difference.

Table 3. Decision Matrix for Model Architecture

Weight	5	10	10	25
Percentage	20%	40%	40%	100%
Option	**Speed (ms)**	**Average Recall**	**Average Precision**	**Score**
Faster RCNN	89	0.573	0.7002	
	30	57	70	57
SSD Lite Mobilenet	31	0.673	0.8288	
	75	67	83	75
SSD Inception V2	42	0.687	0.847	
	70	69	85	76
Notes:	High = faster speed	High = better recall	High = better precision	

Therefore, taking the above assumptions and decision matrices scores into consideration, this leads to the following proposed design of the system for our project. The SSD Inception V2 model architecture will be used to train the custom model, while the Nvidia Jetson Nano will be used as the microcontroller for running the pretrained object detection and classification script via the Intel Realsense D435 depth camera. However, as the Jetson Nano does not have the RAM compacity to perform the training of the model, a local directory on the supercomputer, Dragon, will be set up to be used to train our model using transfer learning. A custom object and coordinate detection script will be created for detecting nuts and bolts on a flange and displaying the 3D coordinate location of each target object.

III. Methodology

This section describes the design process, design of experiments, and methods chosen in order to optimize the results. It is broken down into three sections: Experimental Method, Technical Drawings and a House of Quality.

H. Experimental Method

This section describes the experimental methods used for each subsection of the project. This includes the experimental method for the machine learning process, the depth camera set up and finally applying the resulting machine learning model to the camera depth tracking and object detection system.

1. Machine Learning Model

In order to begin working with the Jetson Nano, the Jetson Nano Developer Kit SD Card Image has to be downloaded from the Nvidia website and flashed onto a micro-SD card. For the project, an SD card with 128 GB of storage capacity was chosen. A keyboard and mouse are connected to the Nano via USB slots and a monitor is connected by an HDMI cable. For testing purposes, a Raspberry Pi camera is connected to the Jetson Nano which will later be replaced by the Intel Realsense D435 depth camera.

Once the Jetson Nano is booted, it can be configured for machine learning applications. This is done by installing system package prerequisites, installing Keras and TensorFlow and installing the Jetson Inference engine. Once Python has been installed, the Python package manager, the Python environment can be configured by setting up virtual environments which help having to maintain separate micro-SD cards for each development environment that may be used on the Nano. Once on the create virtual environment, TensorFlow and Keras can be installing within the environment, and Jetson Inference can be compiled for installation in the same environment. Jetson

Inference contains sample vision systems that can be used for object detection, image classification and image segmentation. A library of programming functions for computer vision, OpenCV, is also installed in the virtual environment.

At this point in the experimental process, the Nvidia Jetson Nano, imagenet-camera sample was tested for image classification using the Raspberry Pi camera as the input camera. The imagenet-camera binary is executed with the Residual Neural Network model, or ResNet, as the image classification model. By point the camera at various objects, the system was able to identify and displaying the accuracies of each classification of the objects that it was targeting. However, the accuracies ranged from 10 to around 60% and would even vary with any shift in the camera focus or movement of the object of camera. For the project, the accuracies should fall within a smaller range and the object detection system will be programmed for the sole purpose of identifying fasteners rather than a variety of objects as is found using the given demos from the Jetson Nano.

The first attempt at solving the problem posed by Fermilab will begin with a custom object detection system created using TensorFlow and OpenCV and applying transfer learning to an existing highly complex model. The first model chosen for this project was the SSD Lite Mobilenet V2 since it was created to run on mobile devices and requires less RAM than other complex models.

To begin building the custom object detection, a custom dataset of images of various nuts and bolts was created. The nuts and bolts in the dataset requested by Fermilab were socket head cap screws, hex head cap screws, taper screw head cap, and hex nuts. However, both hex nuts and bolts will be in the dataset so as to not confused the model for initial testing. An initial dataset of 400 images of these fasteners was created by first taking pictures of the fasteners, and then expanded by using google images of these nuts and bolts. A detailed description of the creation of this dataset is found in the Dataset Creation section of this report. Using LabelImg, an image labeling software for machine learning, each fastener in the images was labeled by drawing a box around the target object and given the general labels ‘nut’ and ‘bolt.’ From the labeling, a .xml file was created for each image containing all information that is to be read by TensorFlow. The image files with their corresponding .xml files are then split into two folders, ‘test’ and ‘train’ containing 20% and 80% of the files respectively. In order for TensorFlow to be able to process these files, the .xml files had to be converted into TFRecord files which contain the required information to train the neural network and act as TensorFlow’s binary storage format. To convert the files, a Python script is run and points to the paths of the dataset with the images and their corresponding .xml files and outputs .record files.

Next, the SSD Lite Mobilenet V2 model configuration file is downloaded onto the Nano. A label map text file is created containing the label names created while labeling the images and a specified number ID for each label created. The model configuration file is then modified to meet the correct number of number classes for the labeling IDs, correct paths for TFRecord files, model training batch size, number of training steps and loss functions defined. Next, the protobuf files responsible for TF Object Detection model configuration and training parameters were compiled. From this point, the neural network can be trained for the custom dataset of the nuts and bolts and will take approximately 20 hours for completion. TensorBoard, a visualization tool for machine learning experimentation, is used to tracking loss and accuracy during training. However, due to the minimal amount of RAM available on the Jetson Nano, training could not be completed, and the Nano would crash with each attempt at training. Similarly, attempting to train the model on a laptop would have taken around 200 days for completion which is not viable for the project.

To solve the problems of lack of RAM and computing power, Dragon, UIC’s supercomputer cluster, which is a CPU and uses Intel Xeon processors, was used to not only complete the training process of the model, but also to speed it up. After creating a local directory for the project on Dragon, all of the software and libraries needed for training that had been installed on the Jetson Nano were installed into this directory. Then in order to execute the training, a submission script is run so as to execute the job on a compute node. At this point, the model was set to train for an initial setting of 72 hours, the maximum allotted time for any given job, so as to set a benchmark for how long training may take. Cyberduck, a SFTP software, was then used to transfer the model training data from the supercomputer to a personal laptop. TensorBoard was then launched to observe the results from training the model. TensorBoard displays the results for the loss function as well as the overall accuracy of the model that was trained. It also displays examples of the images from the testing dataset where is compares the ground truth images to the same image with the detection results. The ground truth image is the image with the set bounding box from LabelImg, while the detection image is the result of the machine learning system attempting to apply a bounding box to the target object. By measuring the differences between ground truth and the detection, the accuracy of the system can be determined.

After observing the initial results, the training dataset was changed from images of singular nuts and bolts, to a dataset of images of various hex nuts and bolts in flanges found both in the city environment as well as found on google images. Following this, another dataset was created from images taken in SolidWorks of model of flanges

found on McMaster Carr filled with hex nuts and bolts. By training the model on these various datasets, the accuracy and loss functions can be compared so as to determine which data produces the most accurate results from the model training.

Finally, after observing the results of the model's training on TensorBoard and in real time, the model was trained using the SSD Inception V2 configuration file. The same steps for training the model with the SSD Lite Mobilenet were performed for training with the SSD Inception V2 model architecture. The results for both model architectures were then compared in terms of speed and accuracy in order to determine which model architecture should be used for training our model for the duration of the project.

2. Creation of Dataset

After observing the results of many different training sessions on the supercomputer, it has become clear that the accuracy and precision of the model are incredibly dependent on the images that comprise the inputted dataset. The first dataset was made up of 231 images taken from an iPhone of a variety of single nuts and single bolts in each image. An example of an image from the dataset is shown below in Figure 1.

Figure 1. Example of image of a hex nut labeled for the first dataset.

Due to the resizing of the image for use in TensorFlow, it can be seen that the hex nut in Figure 1 is flattened and not ideal for machine learning. Since the machine learning and computer vision system will ultimately be applied to the window flange of the NuMI target and since the dataset was deemed unsatisfactory, it was concluded that a new dataset was to be created using the prototype flange.

The second dataset was made of around 800 images also taken from an iPhone of the 3D printed flange shown in Figure 1 in the Technical Drawings section of the report. This flange, like the window flange of the NuMI target, is made up of 32 nuts and 42 bolts. The pictures taken showed varied light and background settings and generally displayed the entire flange at varying distances from the camera, an example is shown in Figure 2 below.

Figure 2. Example of image of the back of the 3D printed flange.

The iPhone camera was set to take square images so that when resizing the images occurred, they would not be distorted. However, after observing the results of the model training on TensorBoard, it was concluded that the images contained too many nuts and bolts for the system to accurately detect any of the target objects. Also, as seen in Figure 2, the lighting conditions were not permissible to properly identifying the nuts and bolts, and the images were captured at too great of a distance from the flange.

Taking into account the results from the first two datasets, a third dataset was created by taking 1200 images of nuts and bolts on various flanges found around the city of Chicago with the hopes that the varying backgrounds may improve training accuracy. These flanges had a more limited number of nuts and bolts attached to them – ranging between two and four of one of the types of fasteners. Due to the vast majority of the flanges being outside, the lighting was significantly better than overhead lighting. However, after reviewing the training results in TensorBoard, it was concluded that the presence background noise and the appearance of partial nuts and bolts, seen in Figure 3, confused the model and resulted in poor accuracy and precision.

Figure 3. Example of image of the flange from third dataset.

After consulting with Dr. Yurkiv, the next dataset was to be created from the rendering of the SolidWorks assembly file of flanges found at McMaster Carr limited to four nuts and bolts per flange. A dataset of 1400 labeled images was created from four different flanges, an example is presented below in Figure 4. This led to the greatest accuracy and precision results of the datasets thus far. However, because the model was only trained to identify nuts and bolts with a blank background, this led to false detections of target objects when an image in the testing folder happened to have different features in the background, as shown in Figure 5.

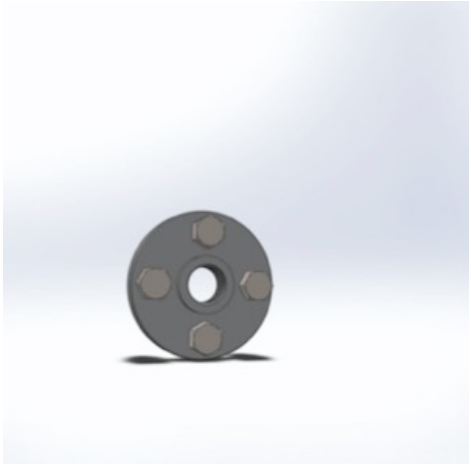


Figure 4. Example of images of the flange from SolidWorks dataset with no background (left) and applied background for increased accuracy (right).

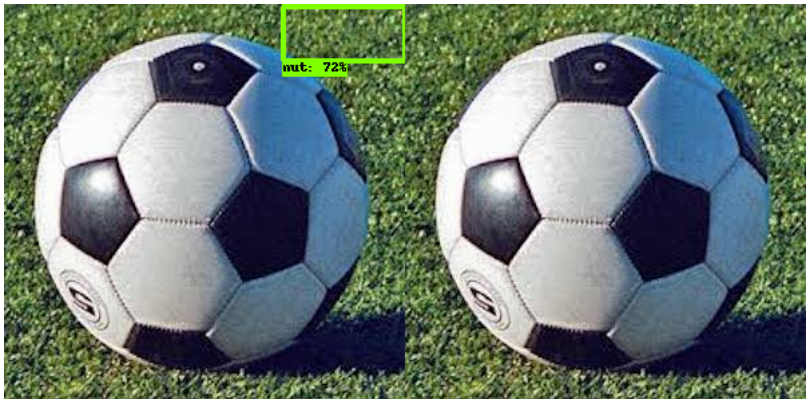


Figure 5. Ground truth (right) is compared to detections (left) where the model confuses the grass for a nut with a classification score of 72%.

Though grass will not be found in Fermilab, more images of the rendered SolidWorks assembly were taken with varying backgrounds found in SolidWorks. The final dataset was therefore made up of 2361 CAD images from SolidWorks with 1405 of the images without backgrounds and 956 images with varied background as shown below in Figure 6. The results from training the model based on the newest dataset led to the best accuracy and precision results of all the datasets.

Figure 6. Example image from final dataset using a background with varying features.

By using SolidWorks to render a model of a flange, a camera path can be set to save images along the pathway. This not only saves time from not taking individual images manually, but also allows for an infinite number of pathways to be set so as to acquire a full range of angles of the flange to be captured for training the model. Ultimately, this will automate the data collection process for the backend of model training, especially as the accuracy and precision of the model increased with the use of this type of dataset.

The training results from all the different datasets and model configurations is found in the Results section of the report.

3. Intel Realsense D435

While the models were being trained, the Intel Realsense D435 depth camera SDK was installed on the Jetson Nano in the same python environment as the previously installed libraries such as TensorFlow and Keras. This was done by installing a Swapfile from the JetsonHacks Github to ease an out of memory issue on the Nano. Next Intel's librealsense was installed on the Jetson Nano. However, in order to allow for the use of the DNN camera function for the Intel Realsense camera, -DBUILD_CV_EXAMPLES=true must be added to the cmake command which can be found in the build shell file. Once this was added to the build .sh file, librealsense was then built on the Jetson Nano. It should be noted that OpenCV is a prerequisite for using the software.

Once the Intel Realsense SDK was successfully installed on the Jetson Nano, various sample codes were made available for use. The specific code that is to be used and potentially modified for the project is the rs-dnn code. This particular sample code uses the Intel Realsense camera with existing DNN algorithms. It was derived from an example that is provided with OpenCV and was modified to function with the Intel Realsense camera as well as take advantage of the depth data. The demo originally loaded an existing Caffe model that will be replaced with the custom models created for detecting nuts and bolts. When running the Intel Realsense rs-dnn demo with the Caffe model on the Nano, bounding boxes appear around target objects in the environment that label both the object being targeted as well as the depth of the object to internal origin in the camera lens.

However, it was found that a custom model could not be imported into this rs-dnn demo due to compatibility issues, therefore, a custom object and coordinate detection script was created to not only detect and differentiate between target object, but also display the 3D coordinates of the targets.

4. Custom Object and Coordinate Detection Script

Once the custom object detection model finished training and saved, a frozen inference graph protobuf file (frozen_inference_graph.pb) is exported using the saved checkpoint files. This must be saved in a folder in TensorFlow's object detection directory on the Jetson Nano which also contains the label map text file created for training the model. The path to the working directory is in user/tensorflow/models/research/object_detection.

The script is created by importing the necessary packages and utilities, such as TensorFlow to utilize the model and the Intel Realsense SDK. Next a path is set to the directory containing the saved frozen inference graph and label map. The input and output data for the object and coordinate detection classifiers are defined as the images for the input, and the detection boxes, classification scores, classes, and XYZ coordinates for the outputs. The Intel Realsense D435 camera is then initialized. A flow chart of the full algorithm is shown below in Figure 7.

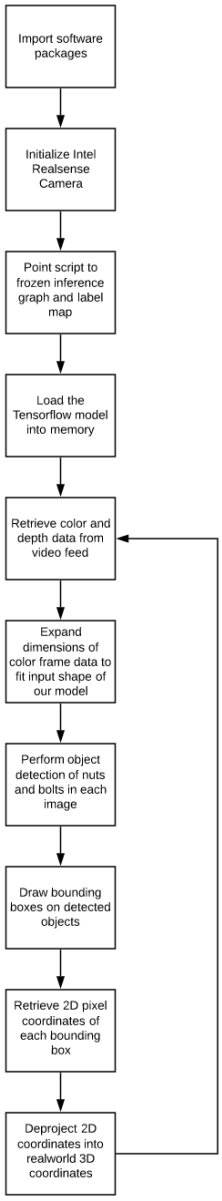


Figure 7. Flow Chart of Object and Coordinate Detection Algorithm

This object and coordinate detection script must be saved in the same working directory as the frozen inference graph. The camera is then placed in front of an image of a flange filled with nuts and bolts and the object and coordinate detection script is run in the terminal of the Jetson Nano. The command must be run while in the directory containing all necessary packages. This will take a minute to load and for the program to begin running.

Once the program is running a window named “Fermi Detection” will appear with the results of the object and coordinate detection. Detection boxes with the objects’ names and classification scores will be drawn around the nuts and/or bolts present in the video feed, and the XYZ coordinate location of each object will be displayed above each detection box. By using a physical ruler, the accuracy of the locations of each object can be determined. The origin of the coordinate system is taken from the Intel Realsense D435 camera lens.

I. Technical Drawings

This technical drawing pertains to the creation of the 3D printed, prototype flange, or bolt circle, that will be used to test the object detection system and eventually test the robotic arm. This flange was based off of the window

flange of the NuMI target assembly. Fermilab provided detailed drawings of the downstream view, Figure 13, and the window flange, Figure 14, of the NuMI target. These drawings can be found in the Appendix of this report.

Figure 8 below depicts the CAD model of the 3D printed flange used as a prototype for the project. As the focus of the machine learning system is on the placement of the nuts and bolts on the bolt circle, these holes were made in the flange and all surface finishing and divots will be done post 3D printing by means of cutting and wrapping with tin foil to emulate the lighting conditions and glare that would reflect off a flange made of metal. Nuts and bolts will be fastened to the 3D printed bolt circle and the object detection system will be tested for the detection of these fasteners. A detailed drawing of the model, Figure 15, is found in the Appendix.

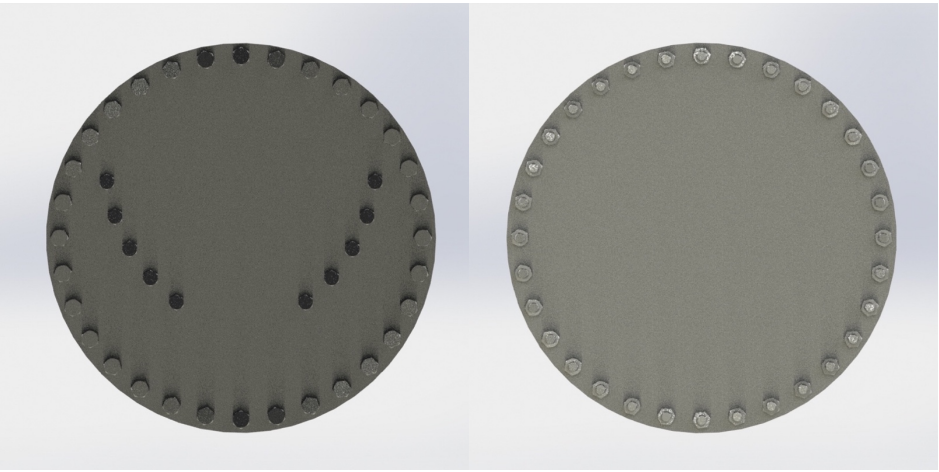


Figure 8. Rendered CAD model of 3D printed, prototype flange with nuts on the back of the flange (right) and bolts on the front of the flange (left) inserted into holes.

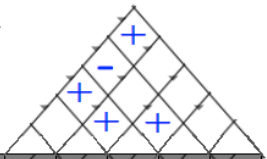
J. House of Quality

The following house of quality in Table 4 was generated in order to determine the correlations between the demand qualities and quality characteristics that Fermilab provided for the project. Each demand quality was given a weight of importance with 5 being the highest and 1 being the lowest, and the relationships between the demand qualities and quality characteristics were ranked by strength on a scale of 1 being the weakest and 9 being the strongest.

Table 4. House of Quality

Title: Machine Learning for Semi-Autonomous Maintenance
Author:
Date:
Notes:

Legend	
⊖	Strong Relationship
O	Moderate Relationship
▲	Weak Relationship
++	Strong Positive Correlation
+	Positive Correlation
−	Negative Correlation
▼	Strong Negative Correlation
▼	Objective Is To Minimize
▲	Objective Is To Maximize
X	Objective Is To Hit Target



Row #	Max Relationship Value in Row	Relative Weight	Weight / Importance	Demanded Quality (a.k.a. "Customer Requirements" or "Whats")	Quality Characteristics (a.k.a. "Functional Requirements" or "Hows") Object Detection	Space Mapping	Move robot arm	Compact hardware	Transferable software platform
				Column #	1	2	3	4	5
				Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)					
1	9	25.0	5.0	Identify and locate nuts and botls in 3D space	⊖				
2	9	25.0	5.0	Differentiate between nuts and bolts	⊖				
3	9	20.0	4.0	Determine location/orientation of identified fasteners with reference to a common coordinate origin		⊖			
4	9	15.0	3.0	Use robotic arm to move in proximity of located fastener		⊖	⊖		
5	9	15.0	3.0	Scalable for various robot arms				⊖	⊖
				Target or Limit Value					
				Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)					
				Max Relationship Value in Column	9	9	9	9	9
				Weight / Importance	450.0	315.0	135.0	135.0	135.0
				Relative Weight	38.5	26.9	11.5	11.5	11.5

IV. Results

These results in Table 5 are results from varying the conditions set for training the TensorFlow model using transfer learning. The training was set to a time of 24 hours with 20,000 steps for the completion of training. The image processing parameters that were taken into account or altered were the batch size, the number of labeled images of nuts and bolts, and how these images were gathered. The training parameters were adjusted by setting the dropout use to “true” with a keep probability of 0.8, setting initial learning late from 0.004 to 0.001, set 12 regularizer from 3.9E-5 to 9.9E-5, and reducing the image size further from 300 by 300 to 256 by 256. These changes were all done by guidance from Steven Penny, the teaching assistant for senior design who’s research us in machine learning. The final step average loss, the average loss and average recall were measured and compared.

Table 5. Results from Training TensorFlow Model with 24-hour time limit and 20,000 steps

	Approach	Model	Batch Size	Number of Labeled Images	**Final step Average Loss**	**Average Precision**	**Average Recall**
Image Processing	First try	SSD Lite Mobilnet	24	231	11	0.06	0.05
	Clean first try images	SSD Lite Mobilnet	24	200	12	0.024	0.023
	Use more images	SSD Lite Mobilnet	24	860	11	0.1	0.1
	Add random object images	SSD Lite Mobilnet	24	1230	11	0.09	0.08
	CAD images + random objects	SSD Lite Mobilnet	24	1048	3.2	0.745	0.61
	CAD images + random objects with varied backgrounds	SSD Inception V2	24	2361	2.416	**0.857**	**0.699**
Adjust training parameters	Set dropout use to true with dropout keep probability 0.8	SSD Lite Mobilnet	24	231	11	0.045	0.022
	Set initial learning rate from 0.004 to 0.001	SSD Lite Mobilnet	24	231	11	0	0
	Set l2 regularizer from 3.9E-5 to 9.9E-5	SSD Lite Mobilnet	24	231	11	0.007	0.005
	Reduce image size form 300x300 to 256x256	SSD Lite Mobilnet	24	231	11	0.024	0.018
	Increase batch size	SSD Lite Mobilnet	32	231	11	0	0

The results from the first attempt at training the model on a dataset are highlighted in yellow with an average precision and recall of 6% and 5% respectively. These results fell far short of the minimum goal threshold of 60%. Therefore, the previously described changes to the training were done along with changes to the datasets, which lead to the training process to be performed for the rest of the project life cycle. By using the SSD Inception V2 model and a dataset made up of CAD rendered images with varied background and random objects labeled to be tested by the model, the results show an average precision and recall of 85.7% and 70% respectively. This is highlighted in the table above in green.

In observing the results from the last training process on TensorBoard, it was seen that when detecting the nuts and bolts in the SolidWorks images the classification probabilities were around 80% to 90%, shown in Figure 9. Along with this, when comparing the ground truth from a soccer ball with a white background to the detections, the model does not draw any detection boxes in the image, shown in Figure 10.

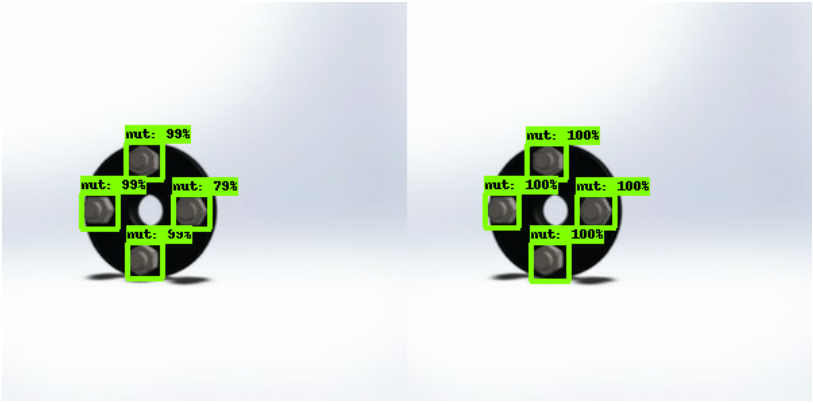


Figure 9. Ground truth (right) is compared to detections (left) from a SolidWorks image of a flange with nuts where classification probabilities ranges from 79% to 99%.

Figure 10. Ground truth (right) is compared to detections (left) where the model assesses that there are no nuts or bolts in the image.

After retraining the model with SolidWorks images with varied backgrounds, the model was able to ignore the background features and accurately detect the target objects as shown in Figure 11.

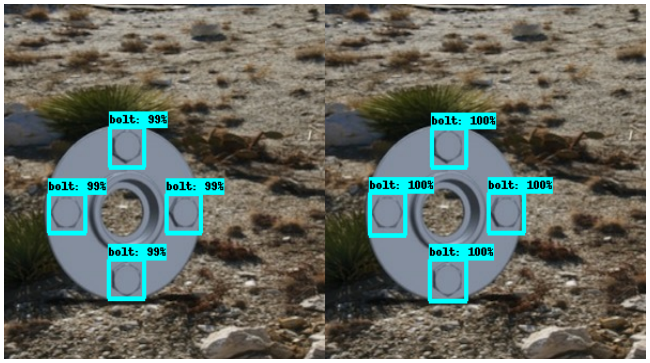


Figure 11. Ground truth (right) is compared to detections (left) from SolidWorks image of a flange with bolts where classification probabilities are all 99%.

Next, the custom model was loaded into the custom object and coordinate detection script on the Jetson Nano which then initialized the camera. The camera was placed in front of an image of a flange filled with nuts and bolts for testing. The results are shown below in Figure 12.

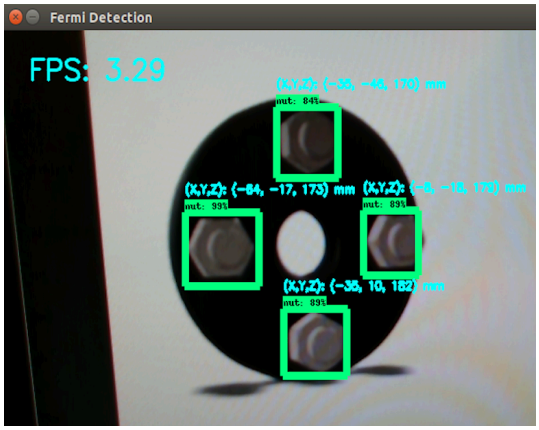


Figure 12. Output of image of flange with detection boxes labeling the detected nuts and 3D coordinates of each nut with the camera lens as the origin.

With a frame rate of 3.29, the custom model was able to detect the target nuts with classification scores ranging between 84% and 99% which falls far above the minimum target score of 60% desired for the project. After physically measuring the distances between the image of the nuts and the camera lens, it was deduced that the machine learning and computer vision system was able to accurately detect and display the 3D coordinates of each of the target objects.

It was also found that the model was able to detect the nuts and bolts on the 3D printed prototype flange with the same classification probabilities, however, due to unforeseen circumstances, the 3D printed flange was no longer accessible, and images of flanges had to be used for testing the machine learning and computer vision system.

V. Conclusions and Future Work

The above decision-making processes narrowed the scope of the project and allowed for final design of the machine learning and camera system. It was concluded that the supercomputer, Dragon, will be used to apply transfer learning to train the model, while the Jetson Nano will be used to run the machine learning and computer vision system conjointly with the Intel Realsense D435 depth camera using a custom object and coordinate detection script. The SSD Inception V2 model will be used for transfer learning with the custom dataset of images of nuts and bolts that was built using images taken within the SolidWorks assembly files of various flanges with nuts and bolts, and varied backgrounds. Finally, it can be concluded that by training a model based on a dataset of CAD images, the machine learning and computer vision system was able to detect nuts and bolts on flanges, as well as, display the 3D coordinates of each object with the camera lens as the origin. This will not only reduce costs but reduce training time and potentially automate other processes that currently risk exposure to radiation.

This is only the first step towards the objective of having a robotic system semi-autonomously perform maintenance or end-of-life procedures on various highly radioactive components for the NuMI Project at Fermilab. By creating a computer vision system with machine learning to locate fasteners, a robotic arm could then be programmed to remove the fasteners from the accelerator beam. Therefore, moving forward, a robotic arm may be acquired and programmed based on the given number degrees of freedom of the robotic arm and the detected XYZ coordinates of the target objects. After researching various six degree of freedom robotic arms, the Niryo One robotic arm was chosen to be the ideal robotic arm for use. This was determined as the University of Salerno successful used this particular robotic arm with the Intel Realsense D435 depth camera and the Jetson Nano [11]. As compatibility between systems is key for the project, this is an ideal scenario moving forward. This robotic arm has been 3D printed, is a six-axis system, and is open source. It can be programmed using a desktop application, an Arduino or Robot Operating System (ROS) which can be installed on the Jetson Nano. This ROS wrapper can be installed and used with the Intel Realsense D435 camera, therefore making this an ideal robotic arm to use.

Appendix

Figures 13 and 14 below show the detailed drawings of the both the downstream view and the window flange of the target. These drawings were used to create the prototype flange that was 3D printed for testing of the machine learning and computer vision system of which its drawing is shown below in Figure 15.

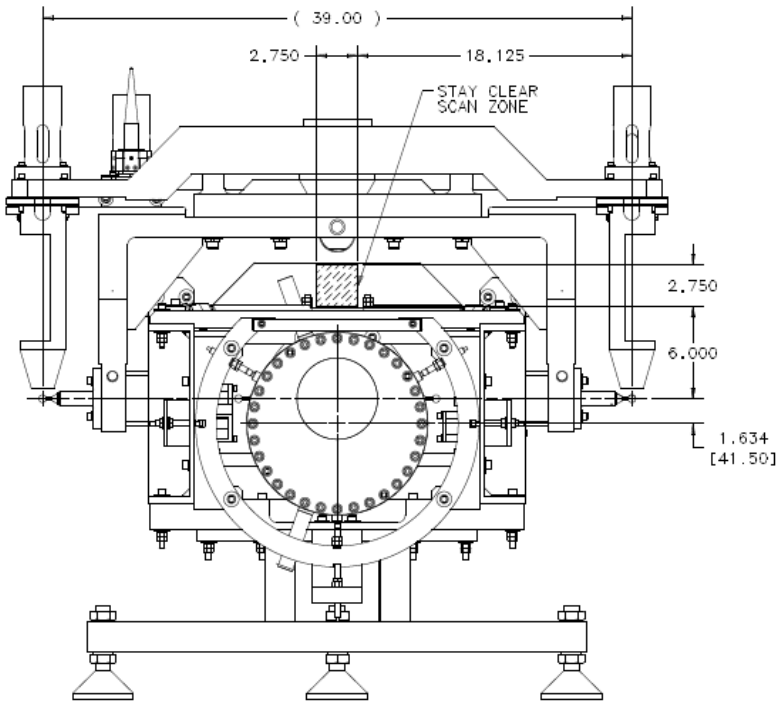


Figure 13. Excerpt from NuMI target assembly drawing.

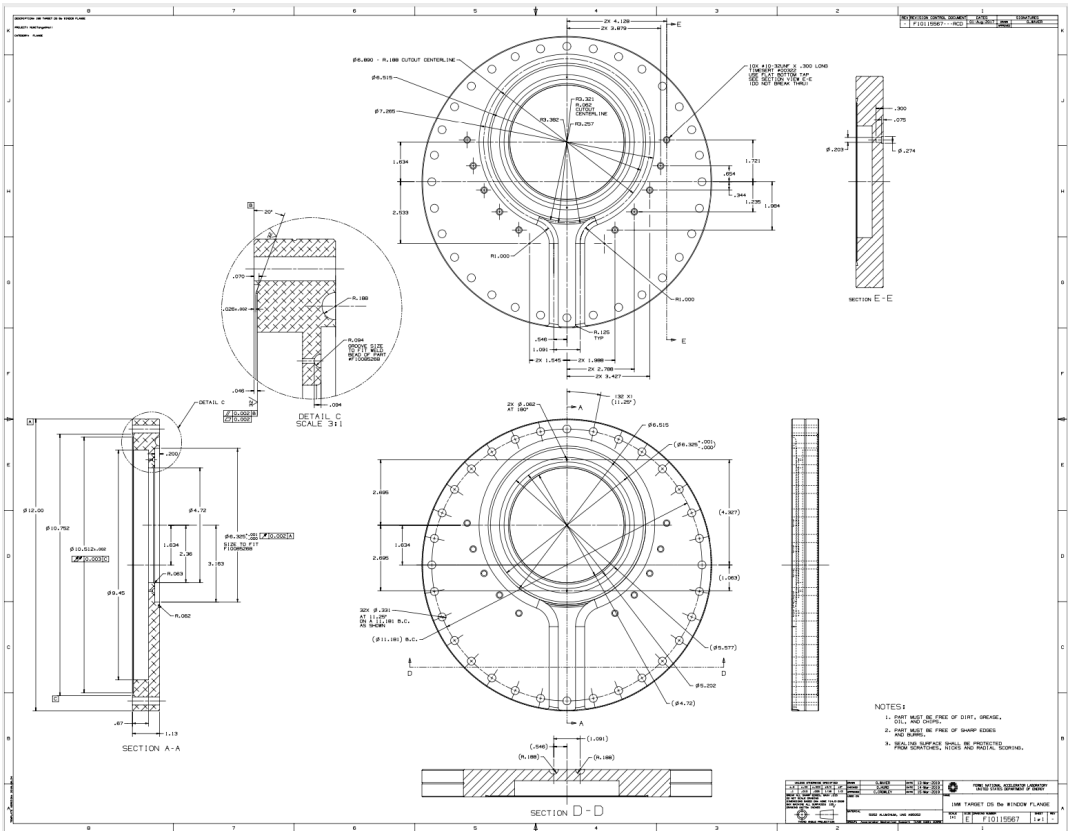


Figure 14. Detailed drawing of window flange from NuMI target.

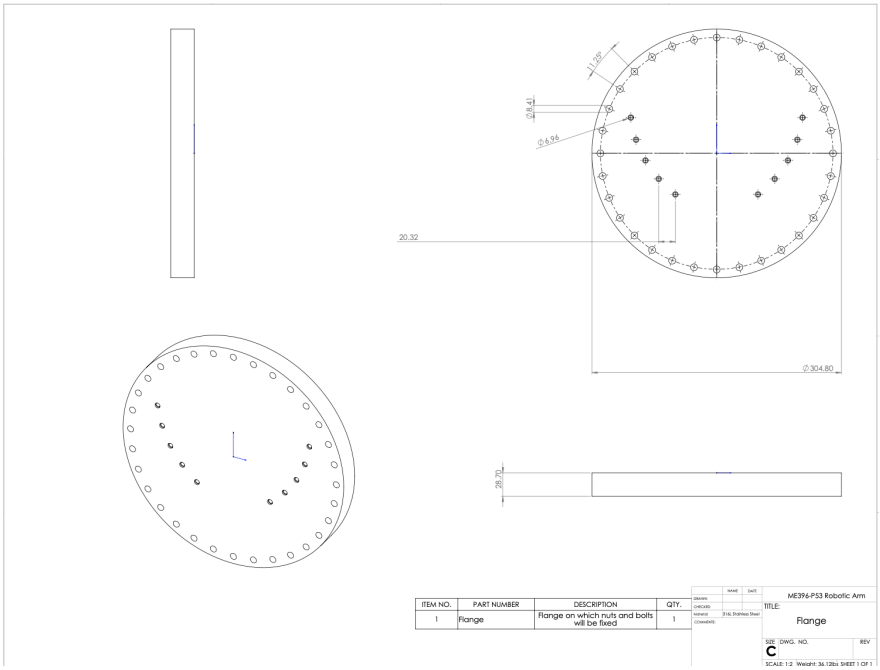


Figure 15. Detailed drawing of prototype flange for 3D printing.

Acknowledgments

The authors of this report thank Patrick Hurh and Katsuya Yonehara at Fermilab for the opportunity to work on such a highly innovative project – their supervision, support and mentorship has been greatly appreciated. The authors also would like to thank the teaching assistants Steven Penny and Marco Ragone for their time and help with programming and machine learning; the Maker Space at the University of Illinois at Chicago for their help in printing the prototype window flange; and Jon Komperda for all the support he provided throughout the project. Finally, a special thanks is given to Dr. Vitaliy Yurkiv for his recommendation of using CAD data for the creation of our final image dataset.

References

[1] Fermi National Accelerator Laboratory, "NuMI-MINOS," [Online]. Available: https://www-numi.fnal.gov/public/numiproject.html. [Accessed 21 11 2019].

[2] A. Voulodimos, N. Doulamis, A. Doulamis and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," *Computational Intelligence and Neuroscience,* pp. 1-9, 2018.

[3] F. Chollet, Deep Learning with Python, vol. 1, Manning Publications Co, 2018.

[4] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, "How transferable are features in deep neural networks?," *Advances in neural information processing systems,* pp. 3320-3328, 2014.

[5] D. Friz, J. Camphous, R. Woolridge, J. Cole, T. Corbett and P. Griffin, "Robotic apparatus and process for the installation of collars and nuts onto fasteners". United States of America Patent 20160082557A1, 2015.

[6] T. Stallman, M. Garber and A. M. Sweeney, "Robot implemented item manipulation". United States of America Patent 10360531, 23 07 2019.

[7] S. Levine, P. Pastor Sampedro and A. Krizhevsky, "Deep machine learning methods and apparatus for robotic grasping". United Statse of America Patent WO2017151206A1, 08 09 2017.

[8] 존 브렌단 맥코맥, 앤커 한다, 앤드류 데이비슨 and 스테판 로이테네거, "Object detection of video data". Worldwide Patent KR20190038808A, 09 04 2019.

[9] 皮思远 and 肖南峰, "A kind of industrial machinery arm visual spatial attention method based on depth

convolutional neural networks". China Patent CN106874914A, 14 05 2019.

[10] A. Kadambi, A. Bhandari and R. Raskar, "3D Depth Cameras in Vision: Benefits and Limitations of the Hardware," in *Computer Vision and Machine Learning with RGB-D Sensors*, Springer International Publishing, 2014, pp. 1-24.

[11] G. Di Prisco, G. Allegretti, M. Schettini and A. Scaldaferri, "Cognitive Robotic Project: Object grasping," Università degli Studi di Salerno, Salerno, 2019.