THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF ENGINEERING

Utilizing Deep Learning and Computer Vision to Detect Defects in a 3D Printed Product in a Manufacturing Environment

ALYSON CAROLINE FARKAS
SPRING 2021

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Engineering
with honors in Letters, Arts, and Sciences

Reviewed and approved* by the following:

Robert Avanzato
Associate Professor of Engineering
Thesis Supervisor

Sally Sue Richmond
Assistant Teaching Professor of Information Science, Engineering
Second Reader

David Ruth
Associate Professor of History
Honors Advisor

* Electronic approvals are on file.

ABSTRACT

The objective of this project is to implement deep learning and computer vison practices to detect defects within 3D printed plastic parts in a manufacturing environment using a conveyer belt and a robot arm. Deep learning is an important technological development within artificial intelligence, and is successful in different application areas including: manufacturing, medical analysis, and self-driving cars. The study focuses on using transfer learning to identify and classify defects in 3D printed food trays that are designed to hold food in a car. MATLAB is used to develop and train a deep learning neural network based on a pretrained network. Images of trays classified as good and bad quality were used to train and test the network. Several studies were completed by changing training parameters to determine optimal settings for repeatable and accurate classifications of manufacturing defects. The trained network was able to classify good and bad quality trays with a success rate ranging from 90%-100%. A camera positioned above the conveyer belt was used to classify trays based on the trained deep learning network. A robot arm was integrated to sort good and bad quality trays from the conveyer belt to designated containers. The robot integration and accurate classifications demonstrated the potential for increased efficiency in quality assurance and manufacturing procedures. The study suggests that deep learning with transfer learning is a viable option to test 3D printed parts for defects in a manufacturing environment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

My sincere thanks and gratitude to

Professor Robert Avanzato
for mentoring me, providing insightful expertise,
and always dedicating time to answer questions

Professor Sally Richmond
for providing insight, proofreading, and editing

Dennis Wozniak
for helping me use the 3D printer within the
lab space at Great Valley

Family and Friends
for always supporting me and
encouraging me to do my best

Chapter 1

Defect Detection in Manufacturing

In manufacturing procedures, defect detection is a vital step within quality assurance to ensure a product meets customer needs and requirements, adheres to strict quality and safety standards, and decreases economic and environmental impact by reducing waste. When defect detection is integrated within the manufacturing process, defects can be identified earlier causing less product waste, and can lead to product refinement stages in which products can be manually inspected and finished if defects can be repaired. [1]

In manufacturing settings, different manufacturing processes lead to common sources of defects. Within casting and foundry work of metals, common defects include hot tears, cold shut, sand wash, sand blow, scab, shrinkage, and hard spots, which can have significant implications on structural integrity. Other defects are common with fusion welding due to thermal stresses that develop during heating and cooling of the piece. These defects include distortion, porosity, cracks, slag inclusions, lack of fusion, lack of penetration, undercutting and underfilling, which can cause insufficient welds leading to an increase in residual stress or causing a joint to fracture or fail below intended loads or stress. [2]

From mass production where there is a high production volume to job shop production where orders are smaller and more refined to the customer base, the “global market requires near perfect quality” as the “detection of rare quality events has become an issue of paramount importance.” [3] With newer technologies within the AI field emerging such as deep learning, defect detection can be categorized as a “binary classification problem” where algorithms can

learn and classify based on compiled data. [3] A proposal for using machine learning through "intelligent supervisory control systems" along with a "learning process and pattern recognition strategy" can make defect detection a part of a "predictive model" using a logistic regression algorithm. [3] These algorithms and datasets have been validated through two test cases described within a journal paper as one involved ultrasonic metal welding and the other a laser spot welding process. Poor quality welds were detected and classified using machine learning. This indicates that using deep learning for defect detection is being explored by manufacturers and will be of growing interest in the upcoming years. [3]

Common Defects in Plastic Extrusion

Additive manufacturing such as the 3D printing of plastics may cause multiple facets of defects including: blemishes, warped spots, insufficient laying or infill, incomplete layers, stringing, and surface nodules. These defects are attributed to physical deformities which may not directly impact the product's useability, but can contribute to product failure on the market due to poor appearance and possibly failing specific safety standards. Since blemishes are often difficult to spot and cannot be quantifiability determined, human inspection of such defects leads to discrepancies as it is up to the individual to determine if the product meets non-quantifiable standards. To ensure this process is streamlined and repeatable, techniques beyond human detection are being implemented to make blemish and defect detection more reliable and lead to a more efficient quality assurance process.

The use of computer vision and artificial intelligence has been tested and is being used in 3D manufacturing settings. For example, a common defect in 3D printed objects is stringing

which is caused by “printing parameters” or the geometry of the object. [4] The stringing effect results in areas of the print, usually between gaps, to have strings of excess plastic which leak from the nozzle during coordinate changes. By adding images of the object with areas of high stringing visibility into the trained neural network, classification of defective parts can be made and leveraged within the assembly line processes through the use of streaming cameras and deep learning. [4]

Product of Interest

Within a Design and Manufacturing class, the problem of blemish and defect detection in 3D printed parts was personally encountered in a group project. The objective of the class was to focus on the design and development of a product to prototype. The group’s idea was to develop a portable food tray that can be attached to a car air vent for easy accessibility to food while parked, and to reduce distractions due to eating and dropping food while driving. Figure 1 displays the SolidWorks drawing and a finished prototype of the food tray that users attach to an air vent, while placing sauce in the smaller tray, and main meal components in the two other compartments.

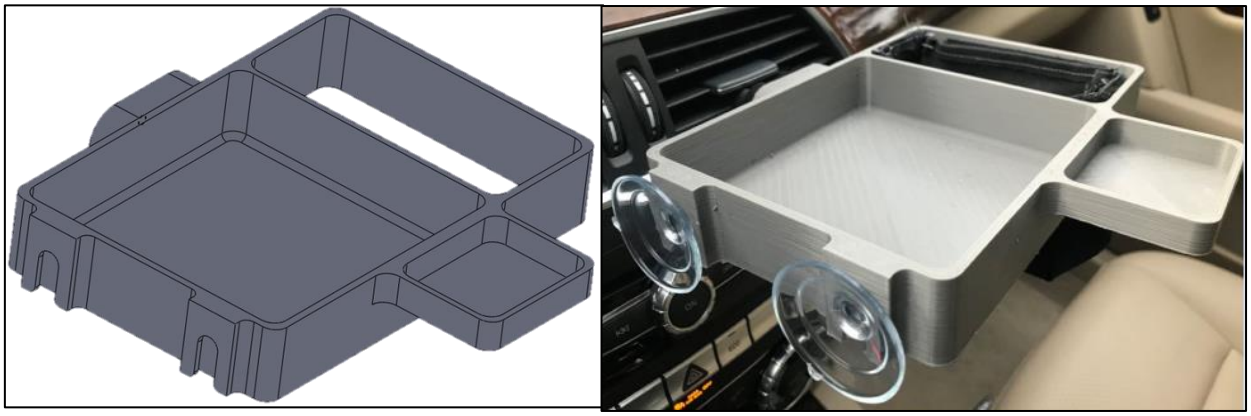


Figure 1: SolidWorks Model of Food Tray (L) And Prototype Attached to Air Vent in Use (R).

The food tray was made using a Robo 3D printer and printed using PLA 1.75mm plastic filament. The printer uses a heated bed to build material on and an Elmer's stick glue was also applied to the bed for build stability and adhesion with initial PLA layers.

When trays were printed, a common defect was blemishes on the bottom surface as the tray had a large contact surface on the heat bed. When layers are continually built upon a spot where filament may have been thinner or skipped due to inaccurate extrusion, areas of high and low divots occur which produce defects in the final 3D printed part. These defects cannot be analyzed until after the part is completed and often lead to discrepancies within the team if the tray would be suitable for customers despite blemishes. Figure 2 provides an example of blemishing on the bottom surface of the tray to show the uneven divots within the plastic layer. These divots are the area of interest for quality assurance as they affect the overall appearance of the product, decrease cleanability, and depending on the depth of the blemish, affect structural integrity of the tray.

Figure 2: Image of Blemish on Bottom of Food Tray.

To ensure this product could be easily washed, a vital requirement was that the surface shall not be porous nor contain crevices which may lead to food getting stuck, causing bacterial growth and unsanitary surfaces. This problem of blemish detection provided the real-life motivation to determine a method to detect defects using technology and lead to advances in quality assurance processes which can increase efficiency, decrease waste, and increase profitability by increasing consumer ratings. From here, researching and testing a method to streamline an otherwise non-quantifiable process of deciding which parts have too many blemishes provided the purpose for this research.

Chapter 2

What is Deep Learning and How Can it be Applied

Deep learning is subset of machine learning which applies artificial intelligence as computers are taught to "learn by example."[5] Deep neural networks are trained to "perform classification tasks" using upwards of 150 layers and sets of labeled data with neural network architecture. [5] Neural networks break up the inputs into layers that extract data as it can be trained for supervised and unsupervised learning, classification, regression, pattern recognition and clustering. Supervised learning will be used for the following endeavor as it is trained based on sample inputs to produce specific outputs and works well within modeled and dynamic situations in order to classify data. For classification, algorithms can learn to classify new sets of data from previously inputted labeled data. This, paired with pattern recognition, allows for classifying data based on key features. [5] A benefit of using deep learning is that it automatically extracts key features from image datasets as opposed to traditional machine learning where features need to be manually identified and extracted with specific algorithms.

The type of neutral network used for deep learning applications is a convolutional neutral network (CNN) in which it learns features based on inputted data, uses 2D layering, and is ideal for processing images. A CNN is ideal for deep learning as it automatically extracts features from datasets and the network can be retrained. [6] The inputted images have unique features which are extracted by the network and trained using these features. For example, each input image is comprised of different shapes and colors. The neural network will use the input and apply different layers to learn and detect edges and finally complex shapes. [7] With each layer is added complexity to the features of the image. [5] The unique features are learned by the network through a process of convolution, rectified linear units, and pooling in order to create

connected layers to support classification. Convolution is where the input data passes through filters in which certain features are activated from the images. A rectified linear unit speeds up the process as only activated features can be passed through to the next layer. The output is simplified using pooling since parameters the network learns are condensed. [6]

Transfer Learning: A Subset of Deep Learning

The neural network used for this study is AlexNet which is pretrained using millions of images in 1000 classification categories and an image input size of 227x227 pixels with three layers (RGB). AlexNet can be used to train other collections of images using transfer learning. [8] Using transfer learning as opposed to creating a new neural network has multiple benefits as a smaller dataset of labeled images can be used for the inputs, computation can be done using a CPU, training time only takes a few minutes, and model accuracy is good depending on the pretrained network. When using transfer learning, the early layers are already pretrained which detect simple features such as edges, blobs, and colors. Only the last classification layers that are specific to the training dataset are modified, depending on the number of classifications the network needs to predict. [9]

Current Applications Across Industries

Deep learning can be applied in different facets of industries from automated driving, industrial automation, and in medical research. Within automated driving, important features such as pedestrians, traffic signals, and stop signs are inputted into the network to increase safety and decrease accidents. Other applications within industry can allow for safer interactions with

heavy machinery and moveable robots as humans can be detected and warned if within unsafe distances. [10] Research reported in [11] suggests that emerging artificial intelligence software will allow robots to grasp and move objects in a manner similar to human motion which can increase human and robot interactions within warehouse environments, leading to more automation and safer integration of humans and robots working side by side. [11] In the medical field, deep learning has been used to accurately differentiate cancerous from noncancerous cells by using high quality images extracted from microscopes. Currently, due to COVID, using AI within the medical field has increased, for example by using image classification procedures to analyze lung tissue and determine COVID infections within CT scans, helping to assist in proper diagnosis. [12] These different applications indicate the diversity of deep learning applications and how they can be integrated in daily problems across industries and help to augment human perceptions and data analysis to improve accuracy.

Benefits of Using AI in Manufacturing

There are key advantages to integrating machine learning and AI within manufacturing procedures, leading to overall improvements in quality assurance, efficiency, and reliability of products. Since AI can be trained to detect common defects or whether products are within specifications for size, this can reduce waste as products can be verified earlier in the manufacturing process, improve quality assurance procedures, and increase overall product quality. Increased product quality can lead to greater customer satisfaction, which can result in expansion of the product or company. Since high-speed cameras and machines work at an increased efficiency when compared to humans, automating tasking such as sorting and visually

looking for defects will decrease production time, leading to less bottlenecks, optimizing production, and increasing inventory. [13]

These examples of deep learning applications in various industries indicate that deep learning is being used positively for society; however, there are growing ethical implications as to how AI advancements within industry may affect society, and how these implications can be addressed before it is commonly adopted in manufacturing. Some of these implications are discussed in the next section.

Ethical Implications of Using AI

Although using artificial intelligence and deep learning can provide a repeatable and efficient process for blemish detection, there are common ethical implications of using this technology within companies such as bias due to training data and when it is applied to systems that impact people and their livelihood.

Since deep learning depends on large datasets to be inputted into the network, areas of bias can occur due to the types of data used, how they are inputted, and the diversity of the data. The predictions otherwise made by humans can also be replicated by the network causing potential bias as to who can get loans, mortgages, apply for credit cards, or even insurance quotes. Depending on the dataset used, the algorithms lead to racial and financial bias and could lead to liabilities within companies. [14]

A social implication of AI technology within industry is replacing humans with automated high-speed cameras, deep learning algorithms, and robots, as it could decrease human jobs and lead to an increase in unemployment for warehouse workers. Decreasing human

workers would decrease wages and salaries and decrease overhead costs for the company allowing for more products to be fulfilled in a timely manner and therefore increase profit. A more profitable company may allow for opportunities for people in other sectors.

To help curb these ethical implications within industry applications, AI algorithms should not replace human analysis procedures, but aid in decision making to decrease bias. Datasets can also be trained using more diverse images, while mining data from larger outlets to decrease bias.

Chapter 3

How to Proceed with MATLAB and Transfer Learning

In order to proceed using transfer learning for blemish detection in food trays, several steps must be completed to create a valid and useable dataset to train the neural network. Reference Appendix A: Software and Hardware for the necessary software downloads used within this chapter. This chapter outlines the procedure from creating test specimens, capturing raw images, cropping images, using transfer learning with AlexNet, and researching and determining hyperparameters for testing.

Creating Test Specimens

All prints of the food tray were produced using the Robo 3D printer and printed using PLA 1.75mm plastic filament in MakerBot Cool Gray. The printer uses a heated bed to build material on and stick glue was also applied to the bed for build stability and adhesion with initial PLA layers. In order to create test specimens that can accurately be compared, machine settings for the 3D printer were all held constant using the Matter Control software at the following parameters: the prints were set to medium quality, a standard layer height of 0.2mm, a first layer height of 0.3mm, a fill density of 0.2 with a grid infill type, and support material used with line support type, at 50% overlap and 3 interface layers. The heat bed is set to a constant temperature of 50°C and the extrusion nozzle is set to a temperature of 210°C. [15] Figure 3 is a picture of the 3D printer used. Technical specifications for the Robo 3D are within Appendix A: Hardware and Software.

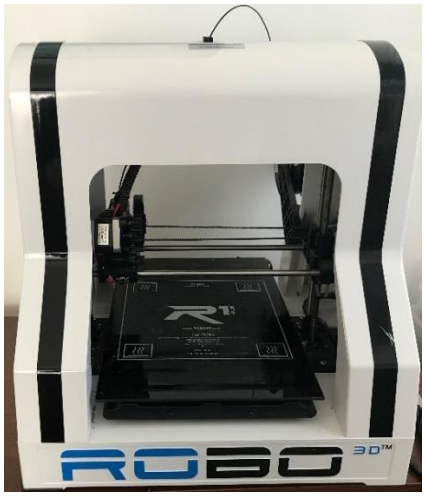


Figure 3: Robo 3D (3D Printer)

Test specimens were created in SolidWorks as shown in Figure 4. The drawing shows full scale dimensions; however, the trays were scaled down to 25% of the original size, shown in Figure 5, to decrease material usage when printing, allow for more trays to be produced, and decrease production time. Scaling the tray down does not affect blemish defects as they are apparent in both small and large trays, allowing for an accurate representation of the problem, but at a smaller manufacturing level. Once test specimens were created, images needed to be captured which is discussed in the following section.

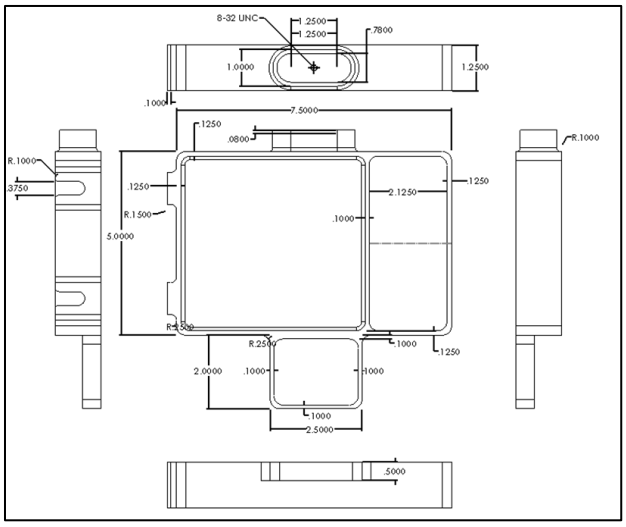


Figure 4: SolidWorks Dimensioned Drawing of Food Tray in Inches.

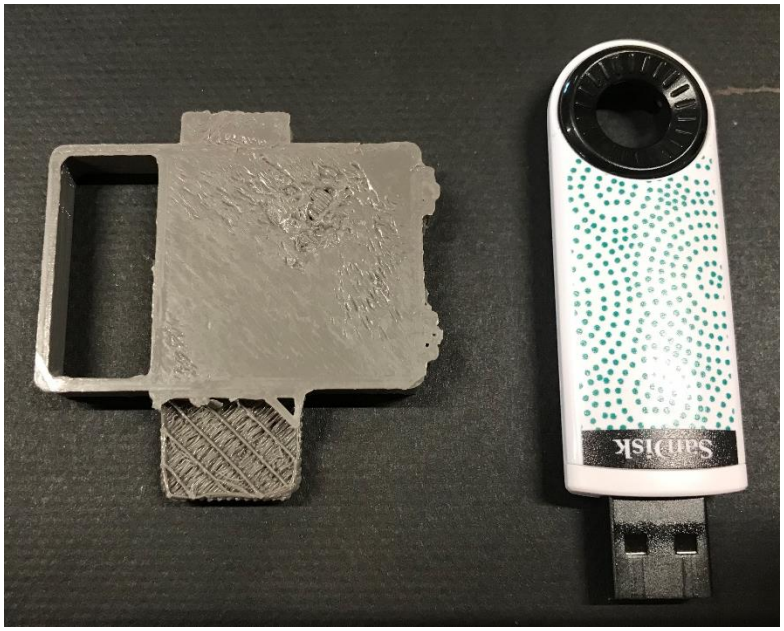


Figure 5: Printed Food Tray at 0.25 Scale Next to USB for Reference Size.

Capturing Raw Images

In order to capture raw images, factors such as focus, lighting, background, shadowing, range, and overall image clarity were analyzed to determine where and how the images are captured. A Logitech C920 HD USB camera was used and mounted on a stand with specifications available in Appendix A: Hardware and Software. The stand is tilted 15° and the camera positioning is tilted 7°. For the camera, the minimum focus is 2 centimeters. The camera and conveyer belt are parallel to the table (180°). In order to stay in range of the camera focus and ensure that each picture would be of comparable focus quality, the height between the camera and conveyer belt was set as a constant. Since all trays are no more than 0.5 inches in height, this confirms that all pictures will be in range and at the same scale, making for pictures to encompass the full tray with limited background and no blur. Figure 6 is a representation of the camera setup which indicates where the camera is in relation to the conveyer belt.

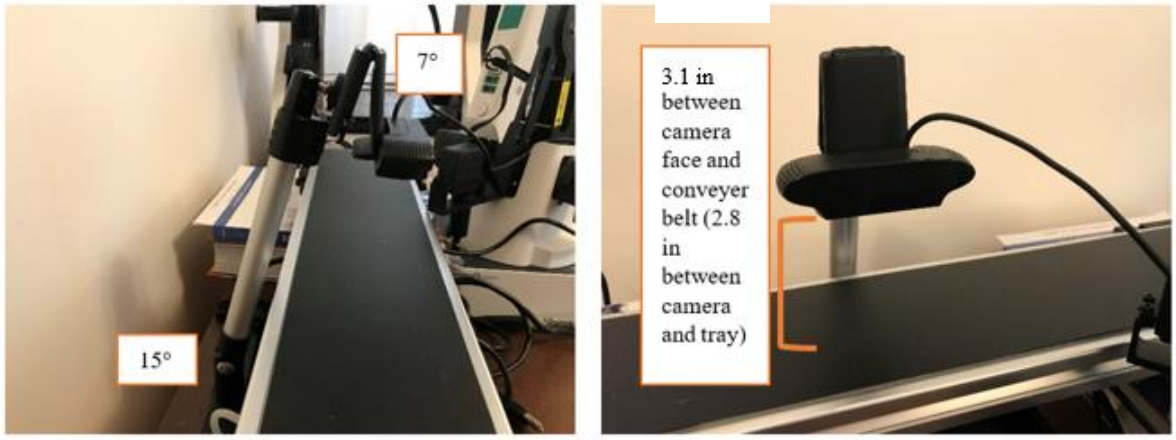


Figure 6: Setup of Camera and Conveyer Belt Used for Capturing Images and Testing.

Lighting settings remained constant as "lighting bias are among the most frequently occurring challenges in image recognition problems." [16] Factors such as what time of day the images are taken, intensity of lighting, and background imagery due to windows and unintentional light sources such as monitors were also considered to create an environment in which the process could be repeatable and mimic a manufacturing environment within a factory setting. Within the room where raw images were captured, all windows had curtains drawn, ceiling lights were on, desk lamp was on, and pictures were taken on an overcast day. By keeping the lighting constant, shadows and variations in lighter and darker hues due to brightness, was decreased. Since transfer learning uses the inputted images for classification, discrepancies in lighting could cause classification errors as they may be used as opposed to the intended blemishes. [16]

In addition, to mimic a manufacturing setting, all images were captured on a conveyer belt at a constant distance of 3.1 inches from the surface of the conveyer to the face of the camera lens. The tray is placed on the conveyer belt, which is black and contrasts with the grey color of the tray, making the tray distinct from the background. When applied in a live setting

with a moving conveyer belt, this also ensures that the background is constant and helps to eliminate unintended shadowing which may cause classification errors.

As with automated manufacturing, robots can place parts equidistant and in the same location repeatedly on a conveyer belt. The assumption that all trays will be placed in the same location on the conveyer was made, and an outline indicating the position of the tray was marked on the conveyer belt. Each tray was placed in the same location when images were captured to ensure the entire tray would be in range. By having a constant location, it also increases efficiency when cropping the images.

Each image was captured in MATLAB using code and commented directions shown in Table 1. This code is meant to capture raw images of good and bad quality trays which will be further processed before being used as a complete dataset for transfer learning.

Table 1: How to Capture Raw Images in MATLAB Using a USB Camera.

```
% How to Capture Raw Images Using MATLAB and a USB Camera
% Download Image Processing Toolbox
% Download Image Acquisition Toolbox
% Download MATLAB Support Package for USB Webcams
% Connect USB camera to USB port in computer
% Open MATLAB, and open a new script to run the following code

% connect to USB camera
cam = webcam(2)

% take a picture using the USB camera
rgbImage=snapshot(cam);

% display the image that was captured using the USB camera
imshow(rgbImage)

% once the picture is captured it is manually saved to a specific file folder as a .jpg
% file folders (Raw_Tray_Good or Raw_Tray_Bad)
% corresponding to the classification of the tray (good or bad)
```

An example of a raw captured image is shown in Figure 7. This image shows the entire bottom of the food tray, whereas only the area outlined in red is the area of concern for quality assurance. Other areas of the tray that may look rough have excess structural material which can be taken off. The main focus of this research is looking at smaller, less identifiable issues such as blemishes which are shown within the boxed area.

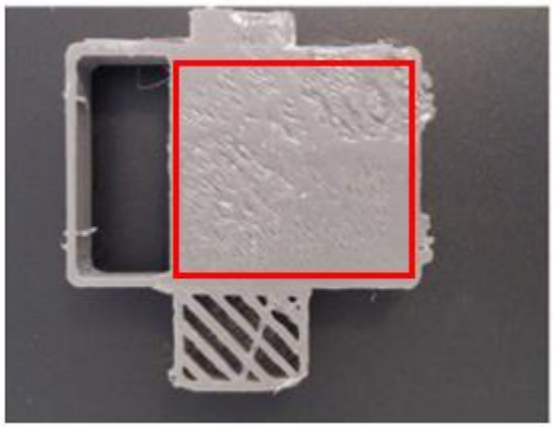


Figure 7: Raw Image of Tray with Red Box Indicating Area of Interest.

Cropping Images

Once images were captured and saved, each raw image needed to be cropped as the area of interest is only a portion of the tray. Cropping was done by importing the saved raw images into MATLAB and using the crop function to make the 480x620 pixel image of the range of pixels comprising of the red outlined area so only the area of interest will be utilized for training a neutral network. This will increase accuracy and avoid misclassifications resulting from other aspects of the tray which should not be analyzed in this quality assurance procedure. The following code in Table 2 was utilized to crop each image.

Table 2: How to Crop Raw Images Using MATLAB and a Preexisting .jpg Image.

```
% How to Crop Raw Images Using MATLAB and a Preexisting .jpg Image
% load the captured raw image into the MATLAB script
im=imread('Tray_x_Bad/Good.jpg');

%show the original image
imshow(im)

% crop the original image to specific parameters only focusing on the
% large surface area on the bottom of the tray
crop=im(120:360,320:590);

% show the cropped image
imshow(crop)

% once the image is cropped save it as a .jpg with a file name denoting the
% classification of the tray and the tray number
```

The cropped images will then be used as the datasets for transfer learning. An example of a cropped image is shown in Figure 8. This tray exemplifies an area of high blemish density indicating that this tray would be classified as a bad quality tray within the transfer learning dataset.

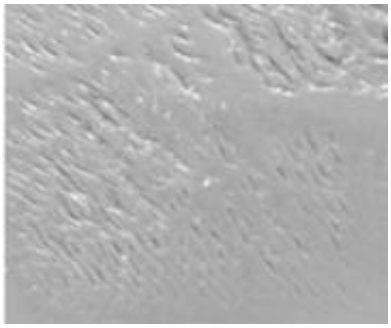


Figure 8: Example of Cropped Image.

A folder titled TrainingData was made and saved to the Desktop. Two folders inside the TrainingData folder were made and titled “Food_Tray_Cropped_Bad_29_Jan_2021” and “Food_Tray_Cropped_Good_29_Jan_2021” depending on the visual inspection. These datasets are known as the true classification labels in which the network will train.

Transfer Learning Using AlexNet

When using a preexisting neural network such as AlexNet to train new images, the result is called transfer learning which is more efficient than starting a new network since it uses smaller datasets, making this an ideal option for the intended application. The code in Table 3 is modified from sample code provided by MATLAB and contains comments that explain what each line of code does. In addition, the readFunction code is downloaded from MATLAB and saved onto the desktop, as shown in Table 4. This code resizes the images and uses a function called from the main code in Table 3.

Table 3: Transfer Learning Code Using AlexNet. [17]

```
%% Access dataset images stored within the TrainingData folder, along with the subfolders
% This commands that the labels created for the 2 sets of images (good and bad trays)
% will be labeled based on their folder names
allImages = imageDatastore('TrainingData', 'IncludeSubfolders', true,...
    'LabelSource', 'foldernames');

%% Split data into training and test sets
[trainingImages, testImages] = splitEachLabel(allImages, 0.8, 'randomize');
 % Need test images, for example if a dataset of 100 images is used, at a
 % 0.8 train to test ratio, 80 images will be used for training while 20
 % images will be used for testing

%% Load Pre-trained Network (AlexNet)
% AlexNet is a pre-trained network trained on 1000 object categories.
% AlexNet is avaliable as a support package on FileExchange.
alex = alexnet;

%% Review Network Architecture
layers = alex.Layers

%% Modify Pre-trained Network
% AlexNet was trained to recognize 1000 classes, we need to modify it to
% recognize just 2 classes since there are only 2 types of classification (good and bad trays)
layers(23) = fullyConnectedLayer(2); % change this based on # of classes
layers(25) = classificationLayer

%% Perform Transfer Learning
% For transfer learning we want to change the weights of the network ever so slightly.
% How much a network is changed during training is controlled by the learning rates.
opts = trainingOptions('sgdm', 'InitialLearnRate', 0.001,...
    'MaxEpochs',10, 'MiniBatchSize',16, 'Plots','training-progress');

% continued…
```

```
%% Set custom read function
% One of the great things about imageDataStore it lets you specify a
% "custom" read function, in this case it is simply resizing the input
% images to 227x227 pixels which is what AlexNet expects. You can do this by
% specifying a function handle of a function with code to read and
% pre-process the image.

trainingImages.ReadFcn = @readFunctionTrain;

%% Train the Network
% This process usually takes about 5-20 minutes on a desktop GPU.
myNet = trainNetwork(trainingImages, layers, opts);

%% Test Network Performance
% Classify each tested image with a predicted label, determine accuracy of the neural network, and create a
% confusion matrix to illustrate the true labels verses the predicted labels.
testImages.ReadFcn = @readFunctionTrain;
predictedLabels = classify(myNet, testImages);
accuracy = mean(predictedLabels == testImages.Labels)
confusionchart(predictedLabels, testImages.Labels)
```

Table 4: readFunction Code. [17]

```
% Copyright 2017 The MathWorks, Inc.
function I = readFunctionTrain(filename)
% Resize the images to the size required by the network.
I = imread(filename);
I = imresize(I, [227 227])
```

Setting Hyperparameters for Training

When training the neural network different types of hyperparameters, settings which affect the training of the network, were changed and monitored such as: epoch size, mini batch size, initial learning rate, and ratio of data being used to train and test.

An epoch is when an entire dataset is passed through the neural network one time. [18] Increasing the number of epochs increases the total iterations used while training the network. When a smaller epoch number is used, it can lead to underfitting, which decreases the accuracy of the training. Using a larger number of epochs can lead to overfitting which causes non-

consistent results. When increasing the epoch size, accuracy of the training also increases, but the optimal number of epochs needs to be realized to prevent overtraining the network which can lead to skewed accuracy. In order to find the optimum number of epochs, the general rule is to stay within the range of 5 to 20 epochs when using transfer learning depending on the diversity of data and the dataset size. [18, 19]

The mini batch size are the smaller groups of data that are fed into the neural network and are approximately 20% to 25% of the training set size. For example, if there are 100 images total, with 80 being used for training and 20 being used for testing, the mini batch size generally can be chosen to be 16-20. Two consequences of using small or large batch sizes are generalization and convergence speed. If large batch sizes are used, the generalization of the network may be poor resulting in inaccurate classification when data samples outside of the trained set are used. When smaller batch sizes are used, the convergence speed will decrease and the rate in which the network learns will slow down, causing greater fluctuations and noise when training leading to a decrease in testing accuracy. Since the data is flowing through an internal CPU instead of an external GPU, batch size will not have a substantial impact on accuracy, but will affect the training time. [18, 20]

Training rate describes the speed in which a network is trained and is typically between zero to one for transfer learning applications; however, individual testing for datasets should be verified as this parameter is often determined through trial and error. A general approach to determine optimal training rates is to start with a larger training rate such as 0.01 and then try training rates one to two magnitudes lower. The key to determining an optimal training rate is to look at loss, as a too small training rate will lead to an increase in loss within the first few iterations when observing a Loss vs Iteration graph. When using a smaller training rate, signs of

loss within the primary iterations in a Loss vs Iteration graph with a continually downward slope would indicate convergence and good training rate. [18, 21, 22]

Increasing the percent of data being used to train, results in decreased data for testing. Typically, an 80% training to testing ratio is used, while for some smaller datasets using a 70% or 60% training to testing ratio may be necessary. For example, if a total of 20 trays were used to train the network, and the training percentage was 60%, 12 trays would be used for training while the remaining 40% (eight trays) would be used for testing. To achieve an optimal testing and training ratio, the dataset size should be increased to improve accuracy. From these parameters epoch size, mini batch size, and training rate will be further explored within the preliminary parametric studies. [18]

Chapter 4

Preliminary Transfer Learning Parametric Studies

Preliminary transfer learning studies were conducted using AlexNet to understand and test the software, along with gauging the accuracy of results. The dataset included 40 trays to be used for training and testing. There were 20 trays classified as good and 20 trays classified as bad that were loaded into the network. The studies aided in determining optimal parameters for accurate learning and training results and also important considerations about the dataset. Study parameters were chosen based on parameter designations determined through Advanced Robot Application lecture notes [18]. Four main studies were conducted and analyzed: comparing constant hyperparameters (two separate studies), changing epoch size, and changing the learning rate. Other supplemental studies were also completed and are located in Appendix B: Parametric Studies Raw Data.

Study 1 and Study 2: Using Constant Parameters

The first parametric study used general parameters and was repeated three times to determine if the results were comparable. Epoch size and mini batch size were set at 10, with a typical learning rate of 0.001 and typical train to learn ratio of 0.8. The three trials and corresponding accuracies are shown in Table 5. The accuracy of the tests ranged from 0.652 to 1. This indicates that due to the randomization of data and small test set, a larger dataset must be used to make the training more accurate and consistent.

Table 5: Preliminary Parametric Studies Constant Hyperparameters Study 1.

Trial	Dataset Size (total # of trays)	Epoch Size	Mini Batch Size	Initial Learning Rate	Train to Learn Ratio	Accuracy
1	40	10	10	0.001	0.8	0.875
2	40	10	10	0.001	0.8	1
3	40	10	10	0.001	0.8	0.652

The training progress of Trial 2 is specified in Figure 9. The Accuracy vs Iteration graph shows that the accuracy remains constant from epoch 4 onward, indicating that 10 epochs are not needed for a smaller test set. The Loss vs Iteration graph also converges at epoch 5 demonstrating that the extra iterations are not necessary for sufficient training.

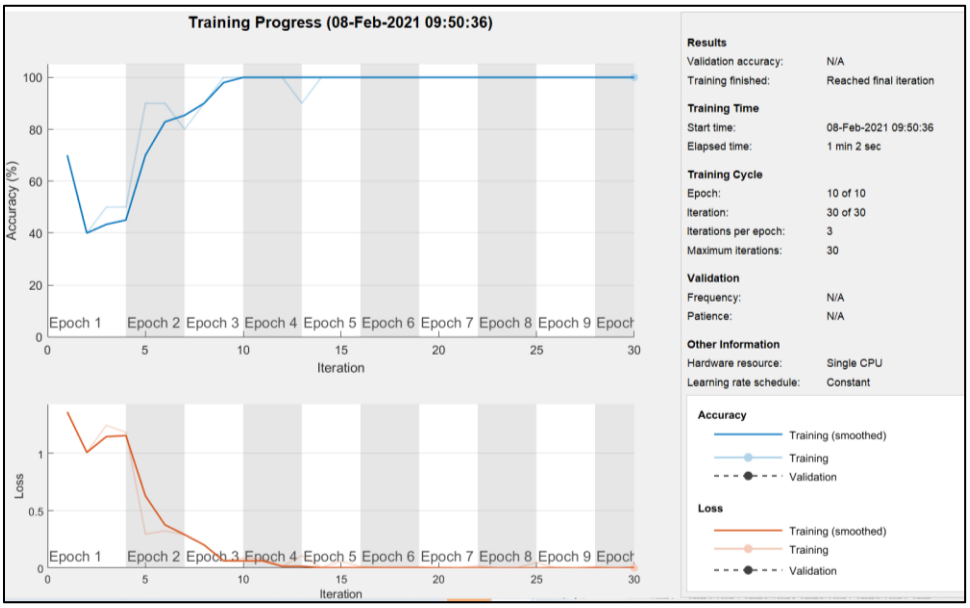


Figure 9: Preliminary Parametric Studies Training Progress for Constant Hyperparameters Study 1 Trial 2.

Another parametric study was conducted using a smaller epoch size of 5 given that data converged around epoch 5 within Figure 9. Three trials were conducted using a mini batch of 10, a general learning rate of 0.001 and a general train to learn ratio of 0.8. The results identified in Table 6 conclude that the overall accuracy of this study was more repeatable as the accuracies were smaller in range, fluctuating from 0.875 to 1.

Table 6: Preliminary Parametric Study Constant Hyperparameters Study 2

Trial	Dataset Size (total # of trays)	Epoch Size	Mini Batch Size	Initial Learning Rate	Train to Learn Ratio	Accuracy
1	40	5	10	0.001	0.8	0.875
2	40	5	10	0.001	0.8	1
3	40	5	10	0.001	0.8	1

Normally, decreasing epoch size decreases accuracy; however, in this study the opposite occurred. The discrepancy indicates that the smaller dataset causes inconsistent accuracies as randomized images are used for training and testing. For example, in Trial 1 the picture shown in Figure 10 failed. Misclassifications, such as this one, are due to inconsistencies in photos as the sliver of black on the left in the image may have been in a tray image classified as good. Since images are randomized per each training, the increase in accuracy could be due to this image being used for training as opposed to testing.

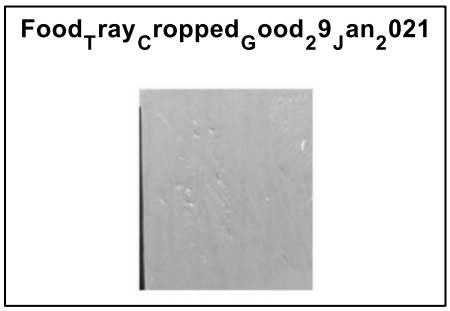


Figure 10: Preliminary Parametric Study, Constant Hyperparameters, Study 2.

Study 3: Changing Epoch Size

Another study was completed where the epoch sizes varied with all other parameters staying constant. Generally, increasing the epoch size increases accuracy and the results for the three trials identified in Table 7 are consistent with this generalization. The change in accuracy is minimal due to the small dataset and randomized images used in training and testing.

Table 7: Preliminary Parametric Study Changing Epoch Size Study 3.

Trial	Dataset Size (total # of trays)	Epoch Size	Mini Batch Size	Initial Learning Rate	Train to Learn Rate	Accuracy
1	40	5	10	0.001	0.8	0.875
2	40	10	10	0.001	0.8	1
3	40	15	10	0.001	0.8	1

Study 4: Changing Initial Learning Rate

Frequently, the initial learning rate is between z0 and 1 for transfer learning purposes. Three trials were conducted within this study to determine how the change in learning rate affects accuracy with the results shown in Table 8. The trials justify that 0.001 is a good number to use for the training rate as the results are comparable to those obtained with a smaller learning rate and is more efficient because it reduces training time.

Table 8: Preliminary Parametric Study, Changing Initial Learning Rate, Study 4.

Trial	Dataset Size (total # of trays)	Epoch Size	Mini Batch Size	Initial Learning Rate	Train to Learn Rate	Accuracy
1	40	5	10	0.01	0.8	0.500
2	40	5	10	0.001	0.8	0.875
3	40	5	10	0.0001	0.8	0.875

Chapter 5

Transfer Learning Using a Larger Dataset

As concluded from the preliminary parametric studies, a dataset of 40 trays caused inconsistent training as there were not enough images to be used for both training and testing. To increase accuracy, a larger dataset of 100 trays, 50 classified as good quality and 50 classified as bad quality, were created and tested using transfer learning. By using a larger dataset, more images can be used for training while also increasing the number of images used for testing the network. Four studies were conducted to determine the best parameters to increase the accuracy of the network without over-training. Supplemental raw data and graphs for these studies are located in Appendix C: Larger Dataset Studies Raw Data.

Study 1: Training Using General Parameters

The first trial used general parameters to determine the accuracy of the neural network and the results are identified in Table 9. The epoch size was set at 5 since there is little diversity in classification and the dataset is relatively small in comparison to the thousands of images neural networks are usually trained with. The mini batch size is set at 20%, leading to 16 images used for training. Generally, 0.001 is considered a good initial learning rate and the train to learn ratio is set at 0.8. At these general parameters, the accuracy ranged from 50% to 85% indicating that the accuracy is not repeatable and has a large difference. This suggests that changes in the parameters such as increasing the epoch size and mini batch size, along with decreasing the initial learning rate is necessary to achieve a higher accuracy. Using general parameters only

reaches an average accuracy of 63% which is not acceptable for use in a manufacturing environment as this will increase product lost and waste due to misclassified defects.

Table 9: Transfer Learning Using a Larger Dataset While Testing General Parameters.

Trial	Dataset Size (total # of trays)	Epoch Size	Mini Batch Size	Initial Learning Rate	Train to Learn Rate	Accuracy
1	100	5	16	0.001	0.8	0.800
2	100	5	16	0.001	0.8	0.850
3	100	5	16	0.001	0.8	0.500
4	100	5	16	0.001	0.8	0.500
5	100	5	16	0.001	0.8	0.500

Figure 11 displays the training data for trial 3 which had an accuracy of only 50%. Since the Loss vs Iteration graph did not have a steady negative slope and never converged to 0, this indicates that the number of epochs must be increased to gain a higher accuracy for this dataset. As well, within the Accuracy vs Iteration graph, the network data is underfitted due to the negatively sloping line from epoch 2 to epoch 5, indicating that the small epoch size severely affected the accuracy due to having limited iterations of training.

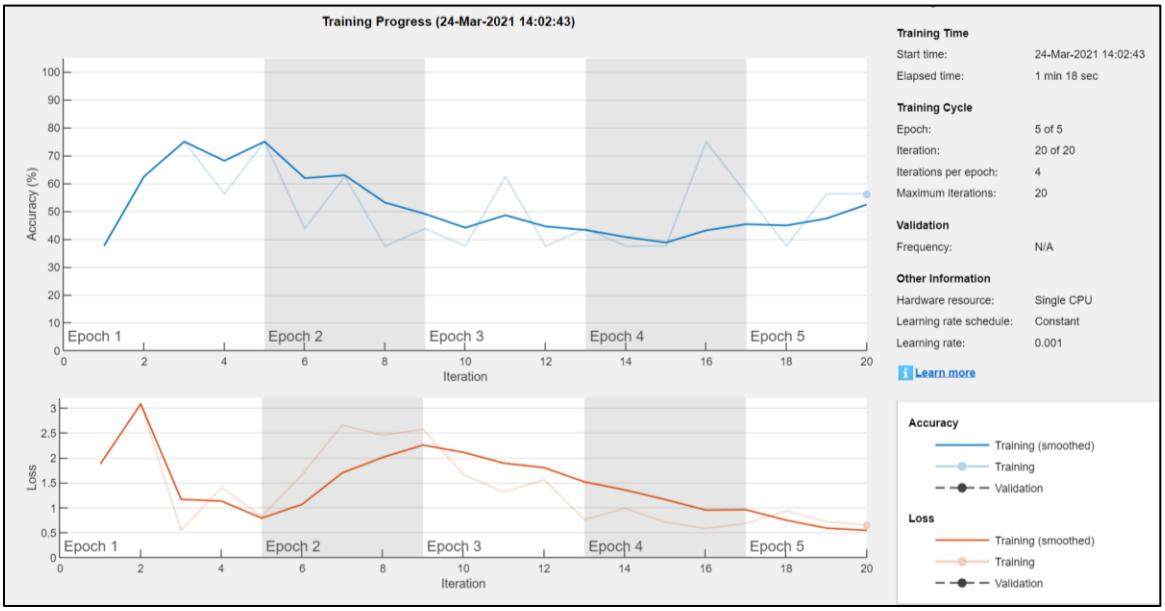


Figure 11: Training Progress for Trial 3 Using General Parameters.

Study 2: Increase Epoch Size Parameter

Since an epoch size of 5 caused underfitting, the epoch size was increased to 10 with all other parameters set as constants, to compare how increasing epoch size increases overall accuracy of training. The results of repeated trials are shown in Table 10. The accuracy for these trials has a range difference of 0.25, but three out of the five trials achieved accuracies greater than or equal to 0.85 indicating a positive trend and overall showing improvement in training.

Table 10: Transfer Learning Using a Larger Dataset While Increasing Epoch Size.

Trial	Dataset Size (total # of trays)	Epoch Size	Mini Batch Size	Initial Learning Rate	Train to Learn Rate	Accuracy
1	100	10	16	0.001	0.8	0.750
2	100	10	16	0.001	0.8	0.650
3	100	10	16	0.001	0.8	0.850
4	100	10	16	0.001	0.8	0.900
5	100	10	16	0.001	0.8	0.900

The increase in epochs resulted in doubling iterations to 40 requires more time for the training to converge. In Figure 12, increasing epoch size caused accuracy to increase gradually over the iterations. It also resulted in the Loss vs Iteration to having a dominant negative slope and to begin converging and remain constant from epoch 8 to epoch 10.

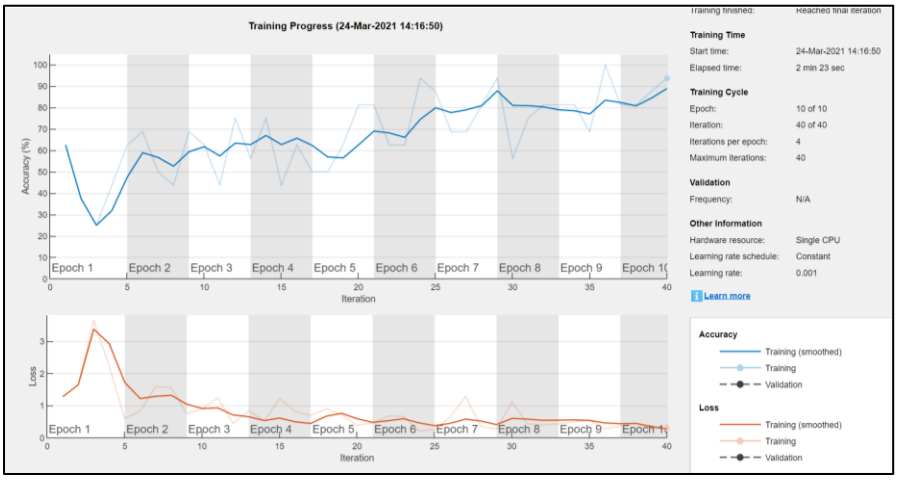


Figure 12: Training Progress for Increasing Epoch Size.

Although increasing epoch size increased accuracy, the goal is to achieve accuracies within the range of 0.90 to 1.00 so that deep learning applications for defect detection significantly improve production quality assurance measures while also reducing waste and possible customer dissatisfaction due to misclassifications. Therefore, more parameters were studied to ensure that the data is not being overfitted or underfitted which can cause generalization errors when applied in similar contexts or severe classification errors when testing using the same dataset.

Study 3: Increase Mini Batch Size from 20% to 25%

Another factor that increases training accuracy is to increase the mini batch size from 20% to 25%. The trials were rerun after increasing the batch size to 25%, from 16 to 20, the results are indicated in Table 11. A smaller batch size can lead to a decrease in convergence among the training data which can cause fluctuations in accuracy. By increasing the batch size to the next general parameter, the accuracy increased, with the greatest accuracy being 0.950 and the average accuracy being 0.85.

Table 11: Changing Mini Batch Size.

Trial	Dataset Size (total # of trays)	Epoch Size	Mini Batch Size	Initial Learning Rate	Train to Learn Rate	Accuracy
1	100	10	20	0.001	0.8	0.900
2	100	10	20	0.001	0.8	0.950
3	100	10	20	0.001	0.8	0.750
4	100	10	20	0.001	0.8	0.850
5	100	10	20	0.001	0.8	0.800

Figure 13 is an example of a confusion matrix which identifies the true classifications of the trays that are being tested compared to the predicted classifications of the trays after they are tested by the network. From the figure, six bad trays were predicted by the network to be bad, while one bad tray was predicted by the network to be good. On the other hand, four good trays were predicted as bad, while nine good trays were predicted as good. This indicates that during trial 3, there is a greater chance that good trays can be misclassified as bad. The scenario is better than bad trays being misclassified as good, but a final parameter will be analyzed to determine if accuracy increases by decreasing misclassification frequency.

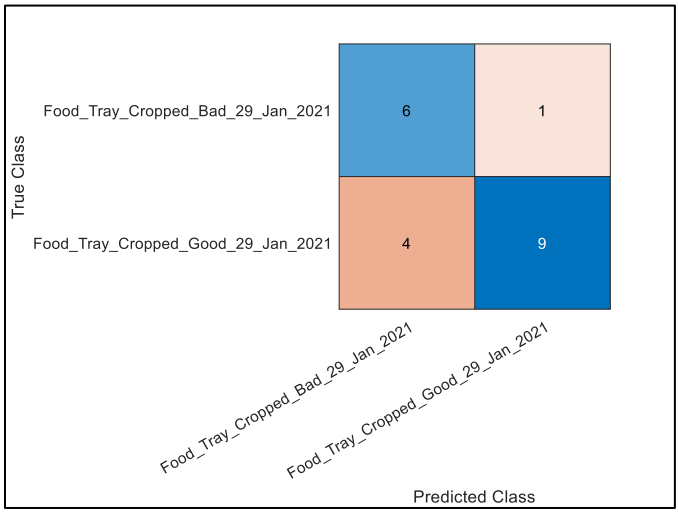


Figure 13: Confusion Matrix.

Study 4: Decrease Learning Rate

Decreasing the learning rate, while keeping all other parameters at the optimum setting as discovered through previously described studies, gave the optimal set of training parameters. Using these settings resulted in the first study to achieve a trial accuracy of 1 along with a minimum accuracy rate of 90% for the other trials, as shown in Table 12. Since decreasing learning rate slows down the process in which the network learns, it will take longer to train the network but will achieve better classification performance. [22] These results indicate that the trained network (myNet) has been optimized.

Table 12: Decrease Learning Rate.

Trial	Dataset Size (total # of trays)	Epoch Size	Mini Batch Size	Initial Learning Rate	Train to Learn Rate	Accuracy
1	100	10	20	0.0001	0.8	0.950
2	100	10	20	0.0001	0.8	0.950
3	100	10	20	0.0001	0.8	1
4	100	10	20	0.0001	0.8	0.900
5	100	10	20	0.0001	0.8	0.900

When analyzing the training progress in Figure 14, the Accuracy vs Iteration graph shows a steady increase in accuracy indicating that parameters have been optimized. Other factors that indicate these are ideal parameters is the lack of positively sloped sections in the Loss vs Iteration graph, as the line keeps approaching zero for convergence with minimal oscillations. [22] By making the training rate smaller and not experiencing an increase in loss within the first few iterations indicates that 0.0001 is an ideal parameter for this specific dataset. This allows the network to learn at a rate that does not learn too quickly which leads to misclassifications nor learn too slowly where the training plateaus at subpar performance. [22]

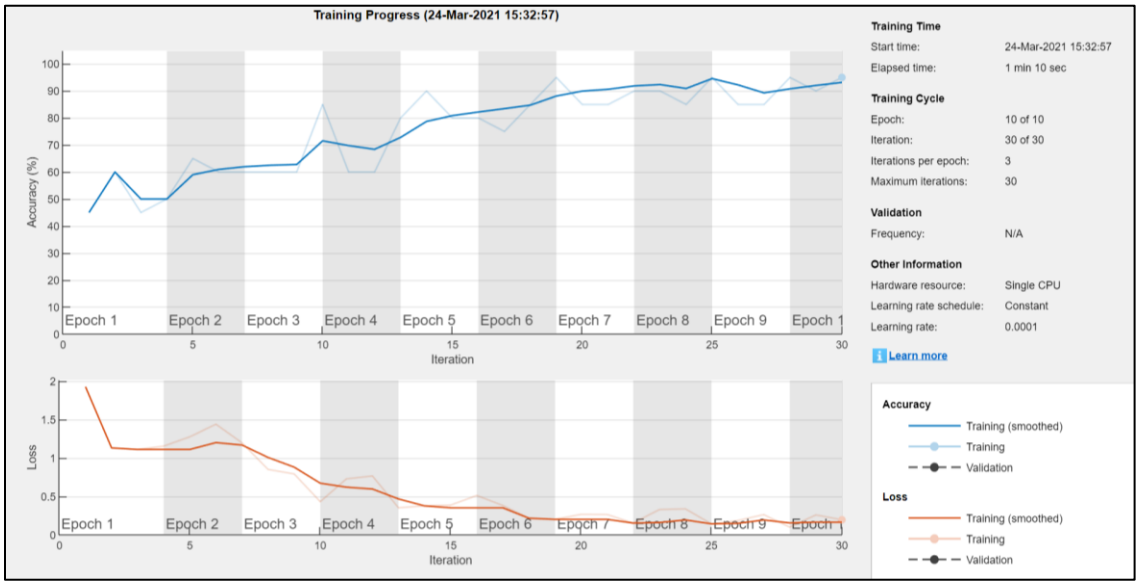


Figure 14: Training Progress for Trial 3.

Conclusion of Larger Dataset Optimal Parameters for myNet

From the four studies outlined within this chapter, decreasing the learning rate had a substantial impact on the accuracy of the trained network when paired with an increase in epoch size and slight increase in mini batch size from the general parameters. The final parameters to be used for further applications when testing, validating, and integrating deep learning into a quality assurance procedure for selecting defects in food trays are outlined in Table 13.

Table 13: Optimal Parameters.

Dataset Size (total # of trays)	Epoch Size	Mini Batch Size	Initial Learning Rate	Train to Learn Rate
100	10	20	0.0001	0.8

A final training was done and used as the myNet variable for the studies described in Chapter 6. The study resulted in a 95% accuracy and optimal results for the training progress graphs as shown in Figure 15.

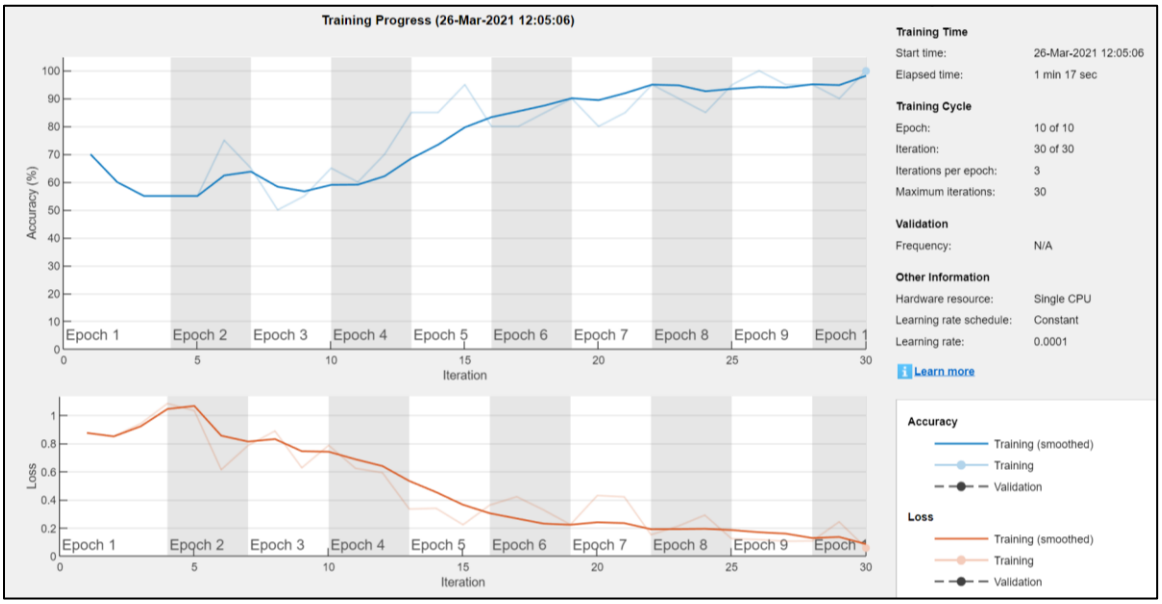


Figure 15: Final myNet Transfer Test.

Chapter 6

Validation Procedures and Comparing Traditional Methods of Roughness Detection

The following two procedures explore how validation of myNet was demonstrated by testing the network live, using trays the network was trained on. Validation on the generalization of myNet was achieved by testing the network using trays printed on other printers. These validation measures ensure that myNet is both accurate enough to classify trays familiar to the network, but also general enough to detect blemishes the network was not specifically trained on. As well, comparing a traditional method of roughness detection against the transfer learning technique applied within this research helps to solidify the decision of using deep learning and justifies assessing the benefits further.

Validate Larger Dataset Using Trays myNet was Previously Trained On

In order to validate the accuracy of myNet and determine if the algorithm was trained and tested with repeatability, a live study was done with ten trays that were loaded into the network, five of which were pre-classified as bad and five which were pre-classified as good. The network has seen these trays previously either through testing or training, so the network should recognize all blemishes and blemish free surfaces. The test was conducted by loading myNet and using the neural network to classify the captured picture using the code described in Table 14. Each tray was placed under the camera so that only the area of concern explained in Chapter 3 was captured. The code captures an image of the tray, resized it to a valid size for myNet, and used the myNet network to classify the image based off of training data.

Table 14: Using myNet to Validate Larger Dataset.

```
% Connect to the camera
   camera = webcam(2)
% Load the neural net
   nnet = myNet;
% Take picture from overhead camera
   picture = snapshot(camera);
% Resize
   picture = imresize(picture,[227,227]);
% Classify the picture
   label = classify(nnet, picture);
% Show the picture
   image(picture);
% Show the label
   title(char(label));
   drawnow;
```

The confusion matrix in Figure 16 summarizes the results of the study and shows that every tray was classified correctly. The results verify that the network was trained with parameters that lead to high accuracy and repeatability as the test correctly identifies 100% of all pretrained trays in classification measures, concluding that the myNet neural network is successfully validated with optimal parameters.

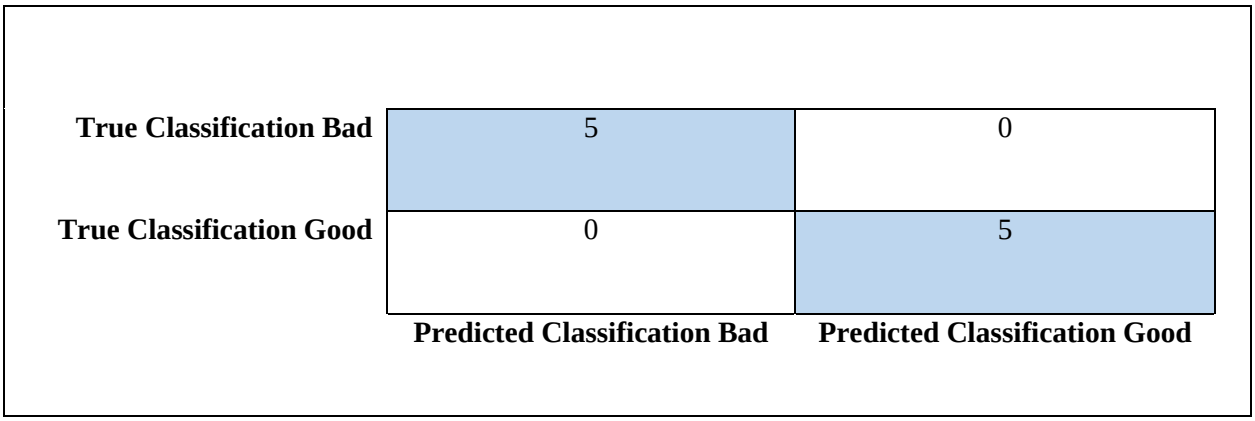
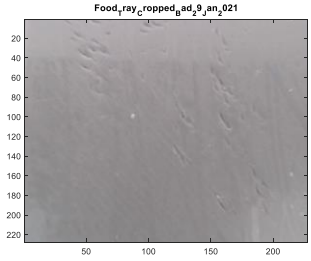
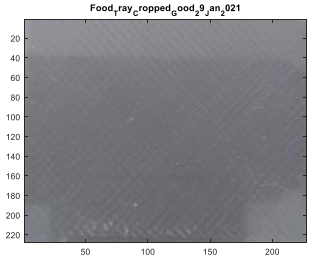
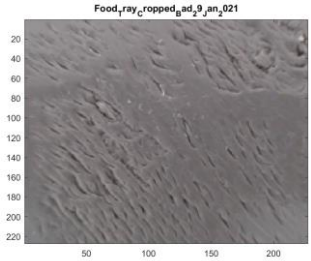
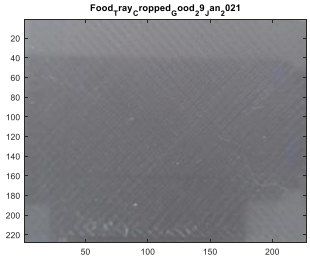
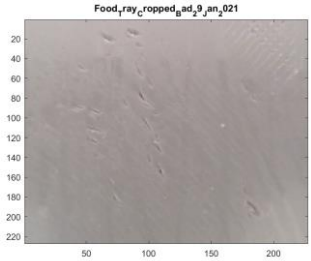
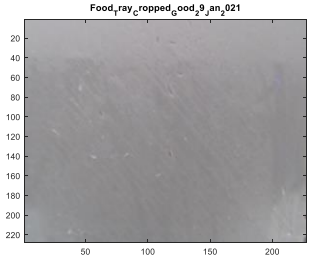
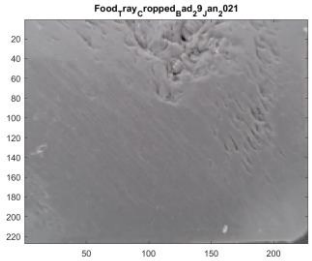
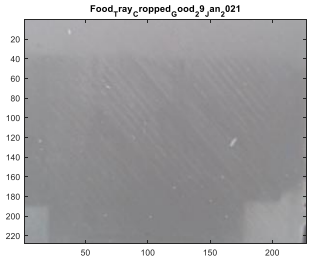
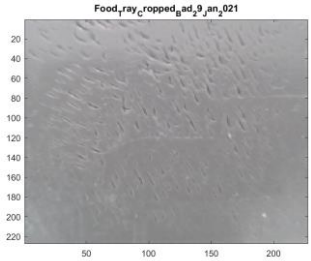
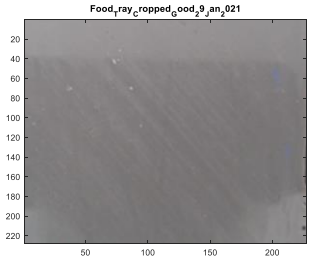


Figure 16: Confusion Matrix for Validation Study.

The images used for validation are shown in Table 15 to illustrate the difference between good and bad trays through the various blemishes.

Table 15: Validation Dataset.

True Classification: Bad Tray	myNet Predicted Classification	True Classification: Good Tray	myNet Predicted Classification
	Bad		Good
	Bad		Good
	Bad		Good
	Bad		Good
	Bad		Good

Using Another 3D Printer to Validate Generalization of myNet

To determine the generalization of the neural network, myNet was tested using food trays printed using a different printer. The specimens were printed using the MakerBot Replicator 3D Printer (5th Generation Model) identified in Figure 17, while using 1.75mm MakerBot Cool Gray filament. The trays were also set to be printed at 0.25 scale. The MakerBot also uses a heated bed, along with structuring material to create the prints which is comparable to how the Robo 3D printer works.

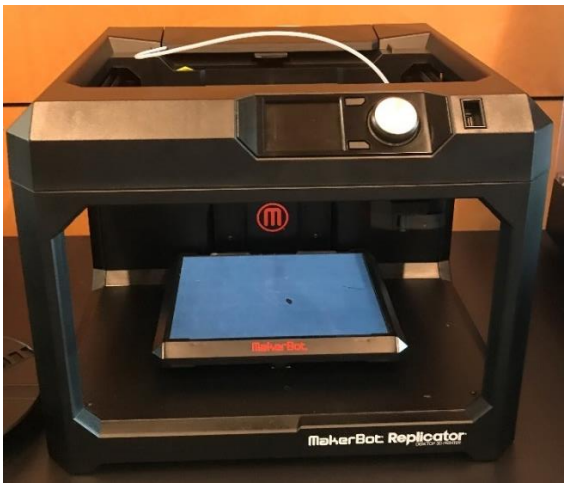
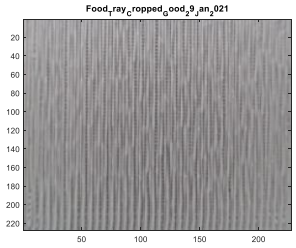
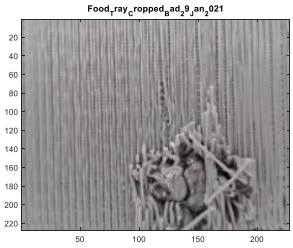
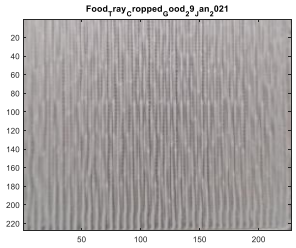
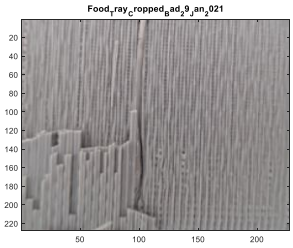
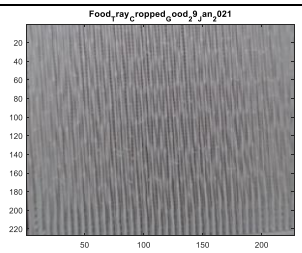
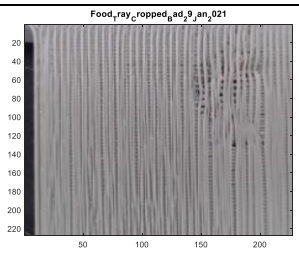
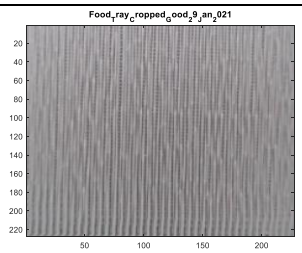
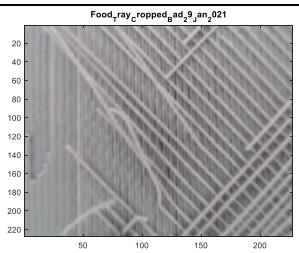
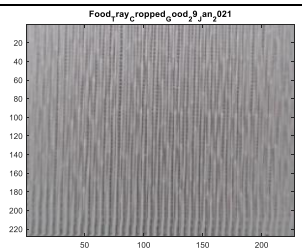
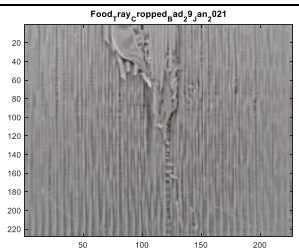


Figure 17: MakerBot 3D Printer.

Table 16 identifies the ten specimens that were used for testing along with the classification determined from the neural network. A total of ten trays were used, five were pre-identified as good quality and five were pre-identified as bad quality. A typical good tray from this printer was of lesser quality than what the network was trained on, with visible lines due to layering and the linear motion of filament extruded. This indicates that the network achieves generalization in that it can be applied to other 3D printers without being retrained. All ten trays were correctly classified by myNet network. Generalization is an important concept concerning transfer learning, as it allows for other sets of data that are similar in characteristics to be

classified accurately. In manufacturing settings, generalization is key, as there are different variations of blemishes that cannot be exactly recreated and trained by the system.

Table 16: MakerBot Tray Generalization Dataset.

True Classifcation: Good Tray	myNet Predicted Classification	True Classifcation: Bad Trays	myNet Predicted Classification
	Good		Bad
	Good		Bad
	Good		Bad
	Good		Bad
	Good		Bad

Alternate Method for Roughness Detection

Surface roughness can be determined using a machine with a small stylus moved across the rough surface at a fixed distance and constant speed. These devices are profilometers and measure the profile of the surface. The machines can only look at individual parts and are most commonly used for metal parts. [23]

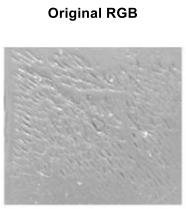
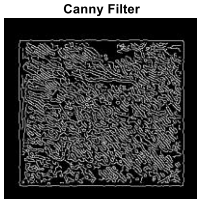
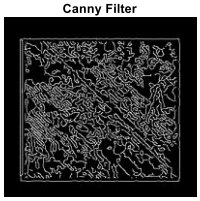
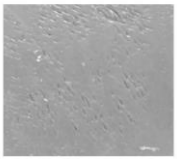
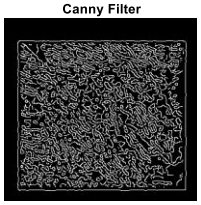
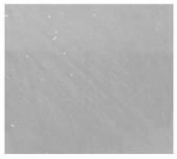
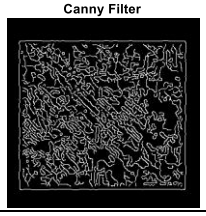
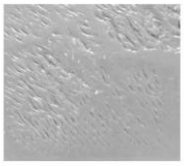
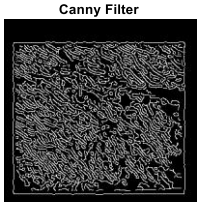
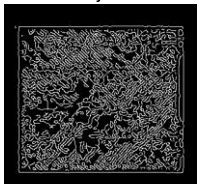
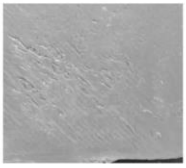
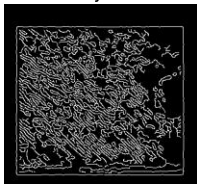
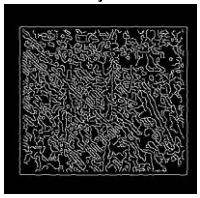
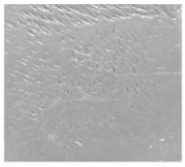
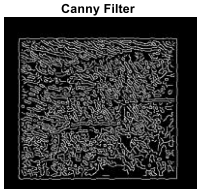
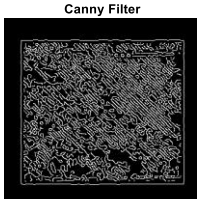
As well, there are traditional computer vision processing methods, such as edge detection that can be used to identify roughness. As an example, edge detection was used on five good and five bad images and then quantitatively compared by summing the total pixels identified as edges. In theory, the trays identified as bad should have a greater amount of edge defined pixels than trays of good quality. Table 17 identifies the code used to process the image and detect edges.

Table 17: MATLAB Code for Edge Detection. [24]

```
RGB = imread('Tray_19_Cropped_Good.jpg'); % get RGB image
imshow(RGB)
title ('Original RGB');

I = rgb2gray(RGB);   %convert to grayscale
figure
imshow(I)      %display image
title ('Grayscale');

BW1 = edge(I,'canny');  %use canny edge detection
imshow(BW1)   % display image
title('Canny Filter');  % title image

totalpixels = sum(sum(BW1(:,1:227)))
```

Once five bad trays and five good trays were processed, the sum of edges or white pixels were compared in each classification category as shown in Table 18. This table displays the original image of the tray in comparison to the binary edge detection image. Black indicates areas of no edges, while white indicates areas where edges are detected.

Table 18: Results of Using Canny Edge Detection to Compare Bad and Good Trays.

Bad Tray #	Original RGB	Canny Filter	Sum of White Pixels	Good Tray #	Original RGB	Canny Filter	Sum of White Pixels
1			7452	5			6251
2			7139	6			5763
3			7405	9			6838
4			6747	12			6541
5			7215	19			6783

After analyzing the data in Table 18, the conclusion is that edge detection using this technique and dataset is ineffective for this 3D printed quality assurance application. The results show that it cannot sufficiently distinguish between good and bad trays to generalize and classify images. The summation of all white pixels, determined from using edge detection, was found for each good and bad tray. Since 3D printers rely on layering techniques and extruding plastic in

linear passes for high surface areas, these faint lines can be visually observed in both good and bad trays. Edge detection treated these lines as edges and wrongfully identified them as blemishes. For example, in the five bad and five good trays that were tested, the summation of pixels ranges from 6747 to 7452 for the bad trays and 5763 to 6838 for the good trays. In order to classify using edge detection, these numbers would need to be split up into common ranges. Since the numbers overlap within the two ranges, accurate identification of good and bad trays cannot be achieved using this method. Other similar computer vision techniques could be investigated, but is not within scope of this research project. The deep learning approach that was selected within the study does not require any low-level image analysis like edge detection. [24]

Chapter 7

Applying Deep Learning into a Manufacturing Environment

After transfer learning was deemed as a reliable and repeatable solution for detecting defects in 3D printed trays, the manufacturing aspect of the system was designed. The system consists of a robot arm, conveyer belt, vacuum gripper, USB camera, IR sensor, and a laptop running the myNet network. The details of the system are described in the following sections.

Robot Arm Integration

A robot arm was selected as the sorting method for the classified trays because they are common in the broader manufacturing context and allows for transfer learning to be applied in a realistic environment that resembles current processing and robotics applications. A robot arm can sort efficiently, speed up repetitive processes to avoid bottlenecks, and increase manufacturing velocity. Integrating classification into a fully automated system also provides a streamlined quality assurance process and demonstrates the repeatability and accuracy of the integrated solution.

The robot arm used is the Dobot Magician which has 4 degrees of freedom and is considered an articulated manipulator. Areas of movement of the Dobot include the base which rotates ±90° from the center, the rear arm which can move from 0° to 85°, the forearm which can move from -10° to +90°, and the rotation servo which ranges from ±90° from the center. A vacuum suction cup with a diameter of 20mm and a pressure of -35 KPa is used as the end effector to pick up and sort good and bad trays. [25] The diagram within the Dobot User Manual

is shown in Figure 18 which illustrates the different ranges applicable to the robot. These were taken into consideration when determining the final manufacturing set up.

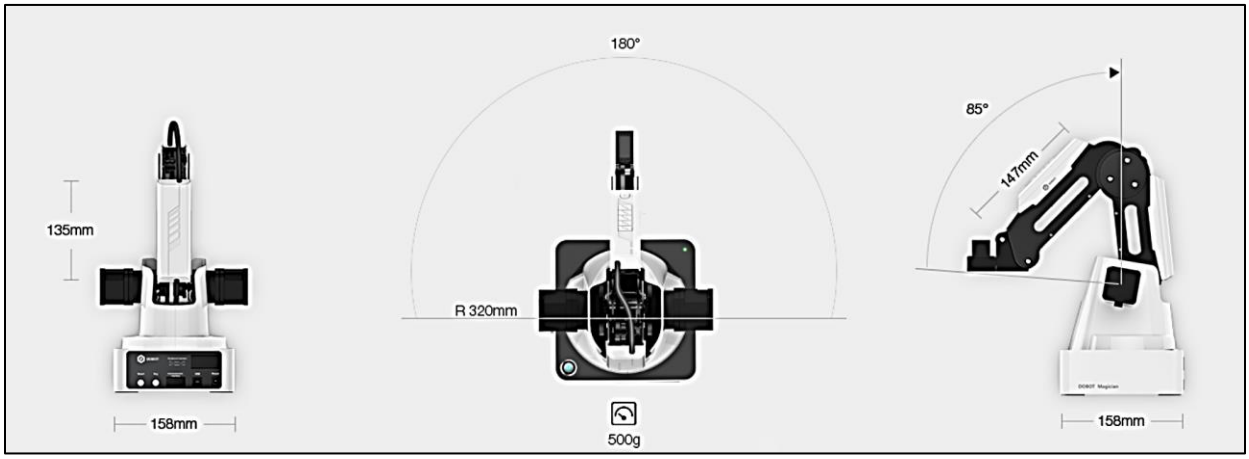


Figure 18: Range of Movement for Robot Arm. [25]

Figure 19 displays a picture of the robot arm with the vacuum suction cup end effector attached.

Figure 19: Dobot Robot Arm.

The system block diagram is outlined in Figure 20 which shows how the hardware and software portions of the system relate. The system consists of the robot arm which is attached via electrical cables to the conveyer belt, IR sensor and vacuum gripper. A USB cable connects the robot arm to the laptop computer which runs both Windows 10 and the virtual machine Linux operating systems. The robot arm is powered using a cable plugged into a standard AC electrical outlet.

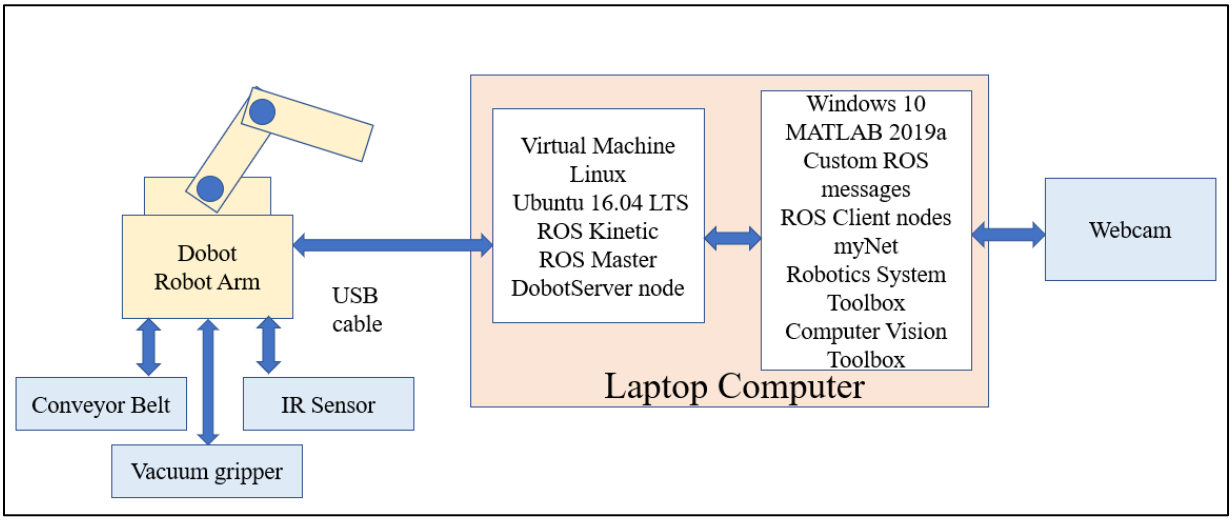


Figure 20: System Block Diagram of Integrated System. [26, with modifications]

As for the software portions of the system, the laptop runs Windows 10 whereas the robot arm can only be controlled using Linux. Using VMWare, a virtual machine runs Ubuntu Linux and the ROS service. This allows for both MATLAB and ROS to be run on the same computer despite using different operating systems.

In order to integrate the robot arm into the process of detecting, classifying, and sorting good and bad trays within a manufacturing environment, several software steps must be done prior to integrating the system. Within MATLAB, ROS Custom messages were installed to provide communication between the robot arm and MATLAB. Information regarding ROS

Custom message procedure and download is located in Appendix D. Commands in Ubuntu are initiated to start and run ROS. Information regarding ROS is explored in Appendix D.

All commands to control the robot arm, conveyer belt, vacuum gripper, IR sensor and webcam are coded within MATLAB as they access ROS custom messages, ROS client nodes, myNet, deep learning toolbox, robotics system toolbox, and support package for USB cameras. These commands flow to the ROS software running on Ubuntu which communicates with the robot arm through the USB cable.

The process to initialize the integrated system is outlined within the pseudocode in Table 19. This pseudocode is the basis for writing the MATLAB code used to maneuver the robot arm and integrate the conveyer belt, vacuum gripper, IR sensor, and USB camera while loading in the myNet neural network to classify good and bad trays. The integration is set to turn on the conveyer belt with trays on it and stop the conveyer belt once a tray has been detected by the IR sensor. Then the overhead USB camera would take a picture of the tray and classify the image based on the trained neural network, myNet. Once the tray is classified, the tray moves along the conveyer belt and stops at a predetermined position. From here, depending whether it is classified as a good or bad tray, the robot arm will initiate the vacuum end effector, pick up the tray and place it in its respective bin labeled “Keep” or “Reject”.

Table 19: Pseudocode.

```
% create IRSetclient, IRSetcmd, and call IRSetclient
% create IRGetclient, IRGetcmd (set port = 2; no call)
% create PTPSetclient, PTPSetcmd (leave empty; no call)
% create VACclient, VACcmd (leave empty; no call)
% create Emotorclient, emotorcmd (set ports, enable and speed; no call)

for  k = 1 : 20        % process 20 objects
    % turn on conveyor belt
    % turn off conveyor belt when block is detected by IR sensor
    while 1              % loop forever until IR sensor detects block
       if block is detected by IR sensor
            turn off conveyor belt
                    take picture of tray
                    classify tray label

                            if label = bad
                                    turn on conveyer belt for specified time
                                    stop conveyer belt
                                    turn on vacuum arm
                                    move vacuum arm to tray position
                                    pick up tray
                                    move to bad box and drop

                            ifelse label = good
                                    turn on conveyer belt for specified time
                                    stop conveyer belt
                                    turn on vacuum arm
                                    move vacuum arm to tray position
                                    pick up tray
                                    move to good box and drop
                            break
                            end % end if
          end % end if
    end     % end while loop
end     %  end of FOR loop
```

Final Integration and Test of Quality Assurance

The purpose of the research was to determine an effective solution to detect defects using deep learning. The final integration test successfully concluded from average optimal parameter accuracies between 90% - 100% with repeatability in validation and generalization procedures. To integrate deep learning classification into a manufacturing setting, the set up described previously using the Dobot robot arm along with a conveyer belt and designated boxes for good and bad classified trays to be placed mimics a small manufacturing environment. The robot arm manufacturing display used for testing is shown in Figure 21. Here the robot arm is located to the left, with the camera and IR sensor located in the middle of the conveyer belt. The boxes used for sorting are located in front of the conveyer belt. This setup was chosen as it allowed for the robot arm to be within range of the sorting boxes and the conveyer belt to sort trays.

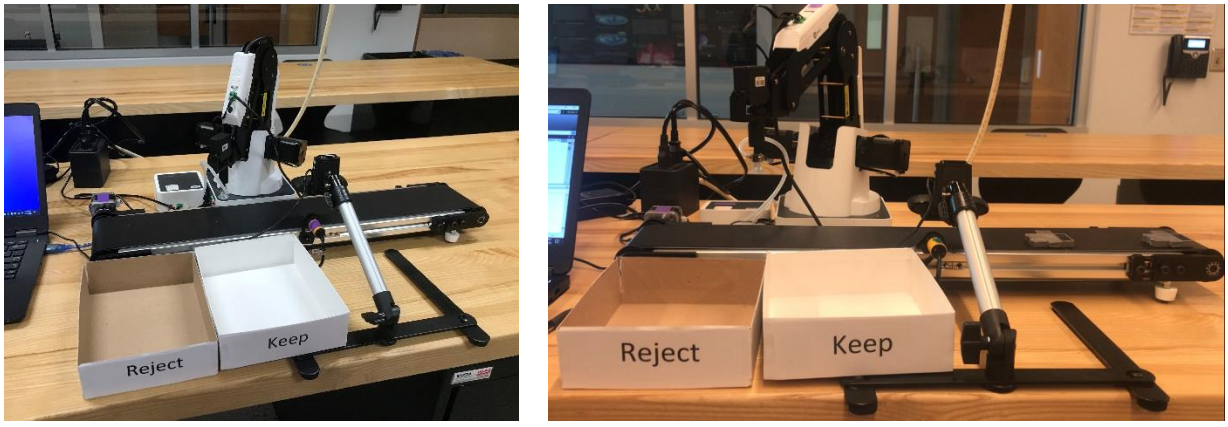


Figure 21: Manufacturing Environment.

In order to integrate the system, pseudocode discussed in the previous section was used to create the final code which is outlined and commented in Table 20.

Table 20: Code to Control Robot Arm System.

```
%% Set up service clients
%set IR client
IRSetClient = rossvcclient('/DobotServer/SetInfraredSensor');
IRSetcmd = rosmessage(IRSetClient);
IRSetcmd.EnableCtrl=1;
IRSetcmd.InfraredPort = 2;
call(IRSetClient, IRSetcmd);

%get IR
IRGetClient=rossvcclient('/DobotServer/GetInfraredSensor');
IRGetcmd=rosmessage(IRGetClient);
IRGetcmd.InfraredPort = 2;

% create point to point client
PTPclient=rossvcclient('/DobotServer/SetPTPCmd');
PTPcmd = rosmessage(PTPclient);

%create vacuum client
VACclient=rossvcclient('DobotServer/SetEndEffectorSuctionCup');
VACcmd=rosmessage(VACclient);

%% Connect to camera and network
camera = webcam(2)   % Connect to the camera
nnet = myNet;      % Load the neural network variable

%% Start loop
for i=1:20    % repeat loop 20 times
   EMotorClient = rossvcclient('/DobotServer/SetEMotor');   % create service client for conveyer belt
   emotorcmd = rosmessage(EMotorClient);   %create empty message to send
   emotorcmd.Index =1;          % set the port number where motor is plugged into robot
   emotorcmd.IsEnabled = 1;    % enable motor
   emotorcmd.Speed = 4000;    % set motor speed
   call(EMotorClient,emotorcmd);  % send to command to move the conveyer belt

   % wait until object is detected on conveyer belt
   while 1
      IRResult = call(IRGetClient, IRGetcmd);
      IRdetect = IRResult.Value
      pause(0.1)
      if IRdetect ==1                % if IR is 1, that means an object was detected
         disp('Object detected')       % display object detected
         emotorcmd.Speed = 0;       % stop the conveyer belt
         call(EMotorClient,emotorcmd);      % stop the conveyer belt
         break    % break out of while loop
      end       % end if statement
   end     % end while loop
   pause(1)   % pause



%continued…
```

```
picture = snapshot(camera);    % take snapshot from overhead camera
picture = imresize(picture,[227,227]);        % resize
label = classify(nnet, picture);    % Classify the picture
imshow(picture);    % Show the picture
title(char(label));  % Show the label
drawnow;   % display image

% if the label indicates the tray is bad quality
if strcmp(char(label),'Food_Tray_Cropped_Bad_29_Jan_2021')    % compare character strings
   disp('Bad Tray Found')          % if the character strings match, the tray is bad quality
   emotorcmd.IsEnabled =1;   % move conveyer belt fixed distance beyond sensor
   emotorcmd.Speed = 5000;  % set motor speed
   call(EMotorClient, emotorcmd);   % call command to move conveyer belt
   pause(3)     % move conveyer belt for designated time
   emotorcmd.Speed = 0;   % stop speed
   call(EMotorClient, emotorcmd);   % call command to stop conveyer belt
   pause(1)   % pause

   % The arm was manually placed where the tray would be located
   % The following code is used to display the position of the arm
   % Pose_client = rossvcclient('/DobotServer/GetPose');
   % Pose_cmd = rosmessage(Pose_client);
   % call (Pose_client, Pose_cmd)

   % Move the robot arm to the tray
   PTPcmd.X = 196.97;
   PTPcmd.Y = 11.84;
   PTPcmd.Z = -13.31;
   call(PTPclient,PTPcmd);
   pause(1)

  % turn on suction
   VACcmd.EnableCtrl = 1;
   VACcmd.Suck=1;
   call(VACclient,VACcmd);
   pause(2)

  % move to “Reject” box
   PTPcmd.X = 282.5;
   PTPcmd.Y = -16.88;
   PTPcmd.Z = 1.42;
   call(PTPclient,PTPcmd);
   pause(3)

   % turn off suction
   VACcmd.EnableCtrl = 0;
   VACcmd.Suck=0;
   call(VACclient,VACcmd);
   pause(1)



% continued…
```

```
  elseif strcmp(char(label),'Food_Tray_Cropped_Good_29_Jan_2021')   % compare character strings
      disp('Good Tray Found')   % if the character strings match, the tray is good quality
      emotorcmd.IsEnabled =1;   % move conveyer belt fixed distance beyond sensor
      emotorcmd.Speed = 5000;   % set motor speed
      call(EMotorClient, emotorcmd);   % call command to move conveyer belt
      pause(3)   % move conveyer belt for designated time
      emotorcmd.Speed = 0;   % stop speed
      call(EMotorClient, emotorcmd);  % call command to stop conveyer belt
      pause(1)  % pause

      % The arm was manually placed where the tray would be located
      % The following code is used to display the position of the arm
      % Pose_client = rossvcclient('/DobotServer/GetPose');
      % Pose_cmd = rosmessage(Pose_client);
      % call (Pose_client, Pose_cmd)

     % Move the robot arm to the tray
      PTPcmd.X = 196.97;
      PTPcmd.Y = 11.84;
      PTPcmd.Z = -13.31;
      call(PTPclient,PTPcmd);
      pause(1)

     % turn on suction
      VACcmd.EnableCtrl = 1;
      VACcmd.Suck=1;
      call(VACclient,VACcmd);
      pause(2)

      % move to “Keep” box
      PTPcmd.X = 273.5;
      PTPcmd.Y = -152.88;
      PTPcmd.Z = 17.42;
      call(PTPclient,PTPcmd);
      pause(3)

     % turn off suction
      VACcmd.EnableCtrl = 0;
      VACcmd.Suck=0;
      call(VACclient,VACcmd);
      pause(1)
   end   % end if statement
end   % end for loop
```

The final test of the system consisted of placing 20 trays on the conveyer belt, ten which had clear blemishes and should be classified as bad, and ten good with smooth surfaces and should be classified as good. The code ran successfully as the conveyer belt moved allowing the tray to reach the IR sensor in line with the USB camera. Once the tray was detected, the conveyer belt stopped and the camera took a picture of the tray which appeared on the monitor and was correctly labeled "Bad". The conveyer belt then turned on, moving the tray for a specified time. From here, the robot arm was initiated to move to the predetermined position of the tray, turn on the vacuum gripper, pick up the tray, move it to the "Reject" bin, and release the vacuum grip. This process was repeated for the other 19 trays with classification and sorting accuracy.

Results for the final integration and test are recorded in proof-of-concept videos demonstrating successful classification and sorting. These videos can be accessed via the links outlined in Table 21. The videos show the robot successfully sorting trays and correctly identifying them acceptable and reject and placing them in the correct bins for further processing. Test Video 1 shows that the robot arm successfully sorted a good quality and bad quality tray into their correct boxes. Deep learning was used for these classifications. Test Video 2 is a partial recording of the robot arm and camera system successfully classifying and sorting 20 trays (ten good quality and ten bad quality trays). The screen recording link is an example of images being classified when testing.

Table 21: Final Integration of Robot Arm Successfully Sorting.

Test Video 1	Example (2 trays)	Video Link
Test Video 2	Final Implementation (Partial recording with 4 trays)	Video Link
Screen Record 1	Screen Recording Example for Classification (20 Trays)	Video Link

Chapter 8

Conclusions and Future Directions

In conclusion, the impact of using a larger dataset of 100 trays for transfer learning proved more successful than the preliminary trainings of 40 trays since more trays can be used for training and testing while also having more variations in blemishes, leading to greater generalization. The optimal parameters of 10 epochs, mini batch size of 20, training rate of 0.0001, and train to learn ratio of 0.80 lead to an overall accuracy ranging from 90% to 100% for predicted classification using myNet. These accuracies indicate that using deep learning for blemish detection purposes is a viable option. A high accuracy paired with a 100% validation rate and 100% generalization rate demonstrates that myNet can be used for a variety of 3D printers with similar quality outputs without the need to be retrained. Given that the classification of images was successfully integrated with a robot arm to demonstrate applications within a manufacturing quality assurance setting helped to solidify the procedure and show the efficiency of the process. Limitations of this research study include the possibility of low inter-rater reliability and a relatively small sample size.

Further research regarding transfer learning using other pre-existing networks for blemish detection training such as ResNet and GoogleNet can be done. [28] Comparing AlexNet to other neural networks can determine which network provides the highest accuracy for the given dataset and optimized parameters. As well, the generalization of myNet can be further tested using trays printed with 3D printers of varying qualities to determine if other surface blemishes unique to different printers can be correctly classified on this network. In addition, more images can also be added to the dataset to increase accuracy.

A future implementation for the optimal dataset concluded from this research is to employ object detection to see where the object is located on the conveyer belt to take into consideration possibilities of the tray being orientated in different positions. Using an object detection algorithm such as YOLO2, which uses transfer learning, provides real time detection, position and size of the object, as well as labeling the classification of the object. [28] When using object detection methods, bounding boxes need be to manually or automatically placed around detected objects and can be time consuming based off of the size of the dataset. Object detection can be processed within MATLAB as there is the ground truth labeling tool which can be used to automate the labeling of training data. By applying a more advanced algorithm, the classification system can be utilized in less strict manufacturing settings where trays do not need to be placed in the same orientation on a conveyer belt. [28]

Appendix A: Software and Hardware

Necessary Software and Hardware

Table A 1: Necessary Software.

Software	Additional Toolboxes to Download
Matter Control	• Software needed to import .stl file and print using Robo 3D
MATLAB R2019a	• MATLAB Support Package for USB Webcams • Image Acquisition Toolbox • Image Processing Toolbox • Computer Vision Toolbox • Deep Learning Toolbox • Deep Learning Toolbox for AlexNet Network • Statistics and Machine Learning Toolbox • ROS Custom Messages
VMWare	• ROS Ubuntu 64-Bit

Table A 2: Necessary Hardware.

Hardware	Additional Materials
ROBO 3D Printer	1.75mm Cool Gray MakerBot Filament
MakerBot	1.75mm Cool Gray MakerBot Filament
Logitech C920 HD Pro Webcam	Camera holder
Conveyer Belt	IR Sensor
Dobot Magician Robot Arm	Vacuum arm end effector

Technical Specifications for Robo 3D

Figure A 1: Specifications for Robo 3D Printer.

Includes

- Fully assembled printer and power cable
- SD printing card
- Upgraded rods and linear bearings
- Upgraded linear motion system
- Prints in PLA, ABS, as well as **LAYWOO-D3** , **t-glase** , and other specialty filaments
- One 300g spool of Robo blue filament to get your project started
- Six-month parts replacement warranty

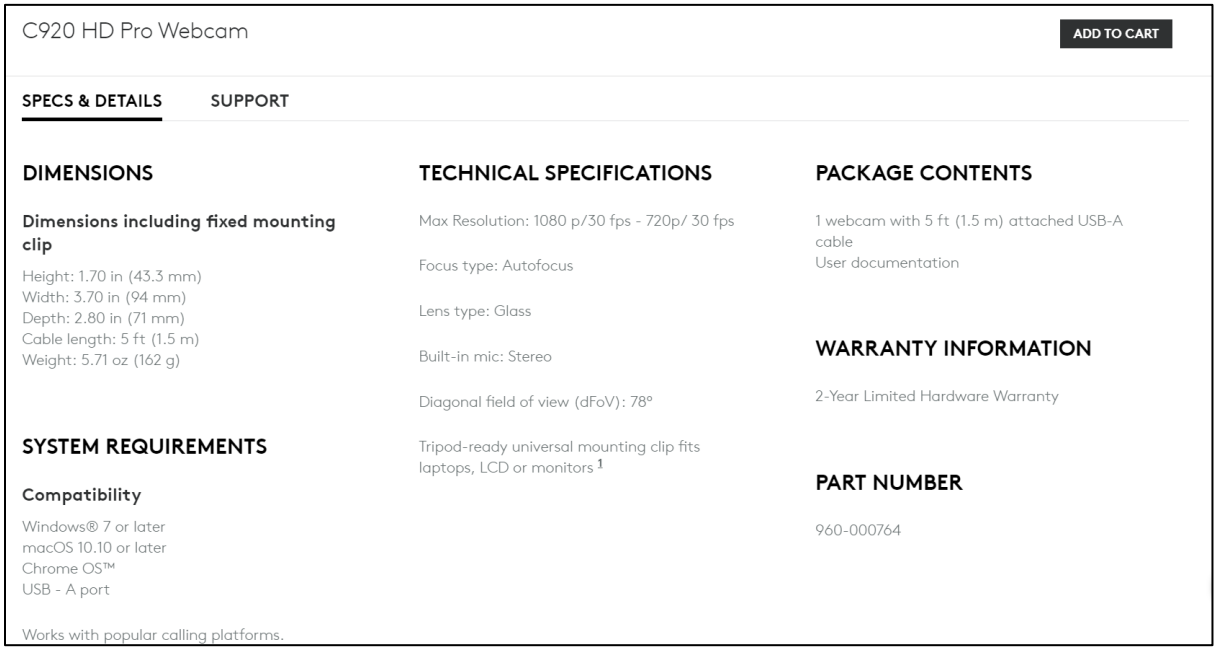
Tech Specs

- With Spool Holder: 15 x 22 x 18.25 in (38.1 x 55.88 x 46.35 cm)
- Without Spool Holder: 15 x 17 x 18.25 in (38.1 x 43.18 x 46.35 cm)
- Single Head Extrusion
- 8.4 x 8.4 heated surface on 10 x 10 bed
- Large 10 x 9 x 8 inch build volume
- Prints in PLA and ABS as well as multiple specialty filaments
- 0.4mm Hexagon All Metal hotend nozzle diameter; prints with any material available.
- Uses 1.75mm filament
- Plastic injection molded parts for longer lasting 3D printer
- 8 Im8uu bearings for precise movement
- 5 Nema 17 motors with 48 oz/in holding torque
- 12V 30amps switching power supply (good for both 115v and 230v outlets)
- 100 Micron Resolution

"Robo 3D R1 + Fully Assembled 3D Printer," *MatterHackers*. [Online]. Available: https://www.matterhackers.com/store/printer-kits/robo-3d-r1. [Accessed: 05-Apr-2021].

Characteristics/Specs of Logitech C920 HD Pro Webcam

Figure A 2: Specifications for USB Camera.



Logitech C920 PRO HD Webcam, 1080p Video with Stereo Audio. [Online]. Available: https://www.logitech.com/en-hk/product/hd-pro-webcam-c920. [Accessed: 05-Apr-2021].

Appendix B: Parametric Studies Raw Data

Table B 1: Parameters for Parametric Study Part 1 Study 1.

Part 1: Multiple Trainings with Constant Hyperparameters Study 1					
Trial	**Epoch Size**	**Mini Batch Size**	**Initial Learning Rate**	**Train to Learn Ratio**	**Accuracy**
1	10	10	0.001	0.8	0.875
2	10	10	0.001	0.8	1
3	10	10	0.001	0.8	0.652

Table B 2: Graphs for Parametric Study Part 1 Study 1.

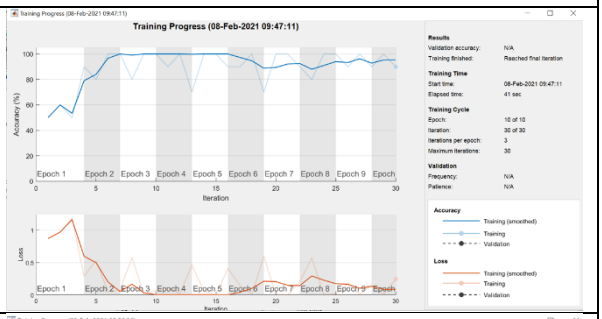
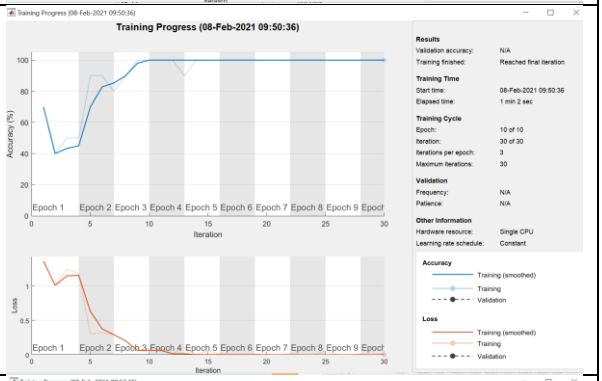
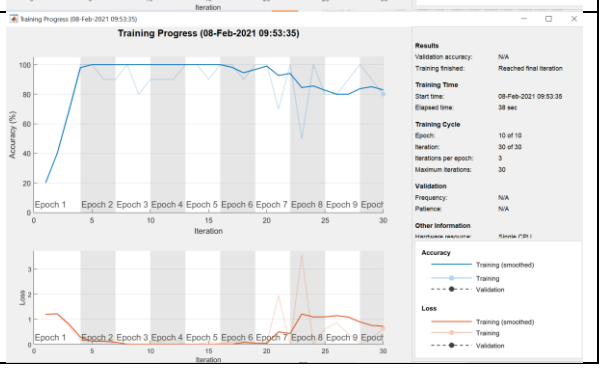
Part 1: Multiple Trainings with Constant Hyperparameters Study 1	
Trial 1	
Trial 2	
Trial 3	

Table B 3: Parameters for Parametric Study Part 1 Study 2.

Part 1: Multiple Trainings with Constant Hyperparameters Study 2					
Trial	**Epoch Size**	**Mini Batch Size**	**Initial Learning Rate**	**Train to Learn Ratio**	**Accuracy**
1	5	10	0.001	0.8	0.875
2	5	10	0.001	0.8	1
3	5	10	0.001	0.8	1

Table B 4: Graphs for Parametric Study Part 1 Study 2.

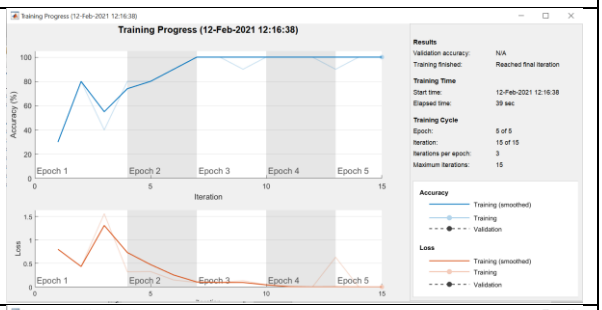
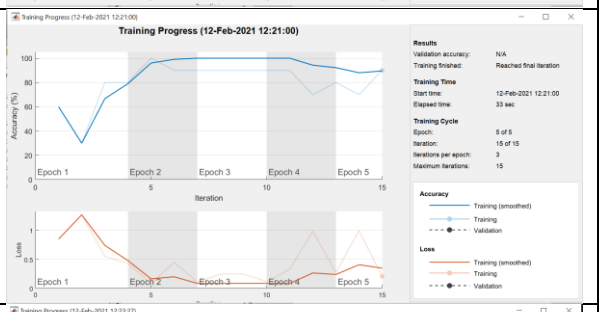
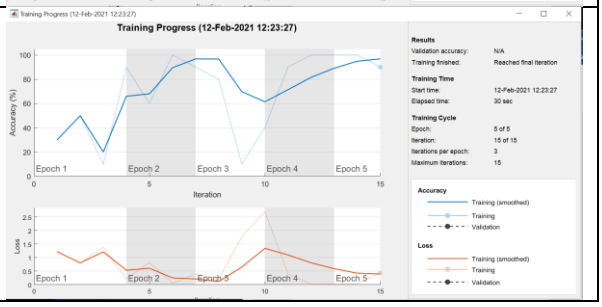
Part 1: Multiple Trainings with Constant Hyperparameters Study 2	
Trial 1	
Trial 2	
Trial 3	

Table B 5: Parameters for Parametric Study Part 1 Study 3.

Part 1: Multiple Trainings with Constant Hyperparameters Study 3					
Trial	**Epoch Size**	**Mini Batch Size**	**Initial Learning Rate**	**Train to Learn Ratio**	**Accuracy**
1	5	20	0.001	0.8	0.875
2	5	20	0.001	0.8	1
3	5	20	0.001	0.8	0.8750

Table B 6: Graphs for Parametric Study Part 1 Study 3.

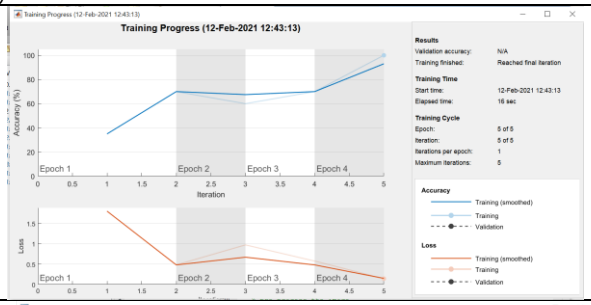
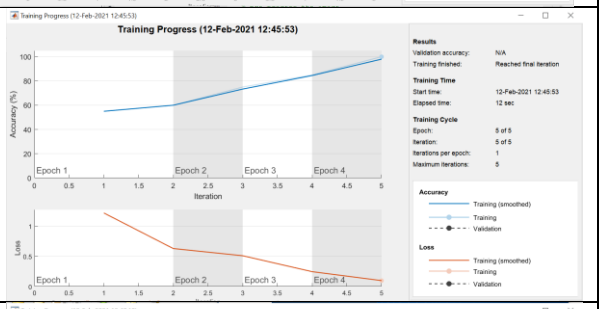
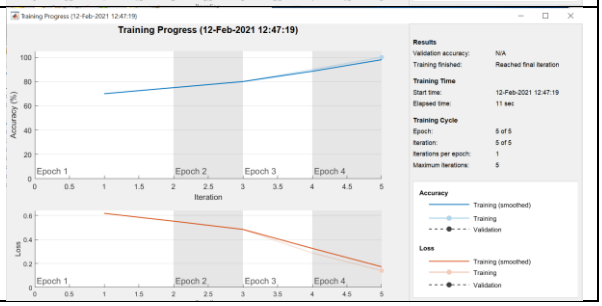
Part 1: Multiple Trainings with Constant Hyperparameters Study 3	
Trial 1	
Trial 2	
Trial 3	

Table B 7: Parameters for Parametric Study Part 1 Study 4.

Part 1: Multiple Trainings with Constant Hyperparameters Study 4					
Trial	**Epoch Size**	**Mini Batch Size**	**Initial Learning Rate**	**Train to Learn Ratio**	**Accuracy**
1	10	10	0.001	0.8	0.8750
2	10	10	0.001	0.8	0.9375
3	10	10	0.001	0.8	1

Table B 8: Graphs for Parametric Study Part 1 Study 4.

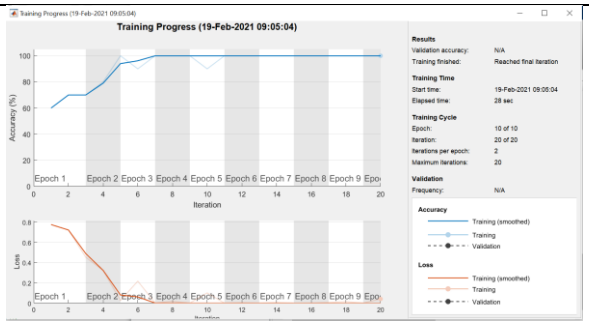
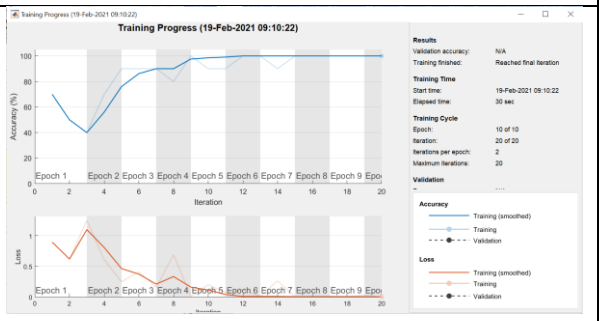
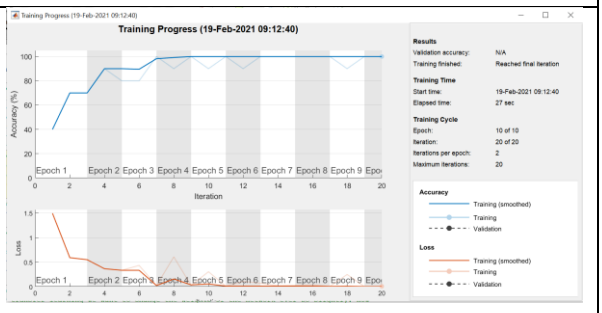
Part 1: Multiple Trainings with Constant Hyperparameters Study 4	
Trial 1	
Trial 2	
Trial 3	

Table B 9: Parameters for Parametric Study Part 1 Study 5.

Part 1: Multiple Trainings with Constant Hyperparameters Study 5					
Trial	**Epoch Size**	**Mini Batch Size**	**Initial Learning Rate**	**Train to Learn Ratio**	**Accuracy**
1	5	10	0.001	0.6	0.8750
2	5	10	0.001	0.6	0.8750
3	5	10	0.001	0.6	0.75

Table B 10: Graphs for Parametric Study Part 1 Study 5.

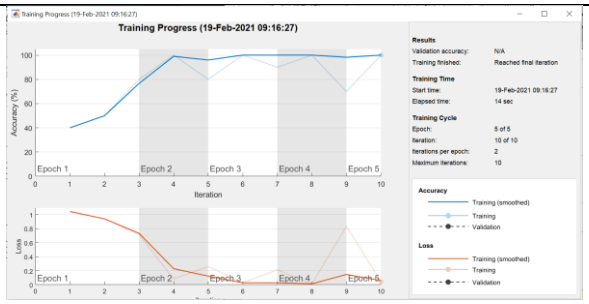
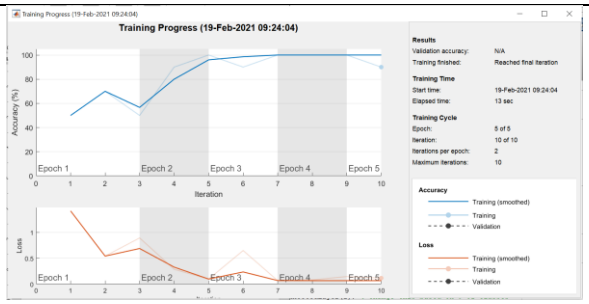
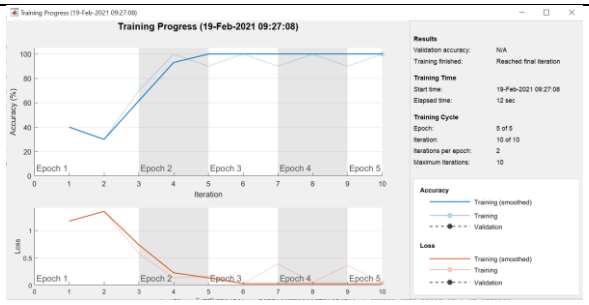
Part 1: Multiple Trainings with Constant Hyperparameters Study 5	
Trial 1	
Trial 2	
Trial 3	

Table B 11: Parameters for Parametric Study Part 1 Study 6.

Part 1: Multiple Trainings with Constant Hyperparameters Study 6					
Trial	**Epoch Size**	**Mini Batch Size**	**Initial Learning Rate**	**Train to Learn Ratio**	**Accuracy**
1	5	20	0.001	0.6	0.8125
2	5	20	0.001	0.6	0.7500
3	5	20	0.001	0.6	0.8125

Table B 12: Graphs for Parametric Study Part 1 Study 6.

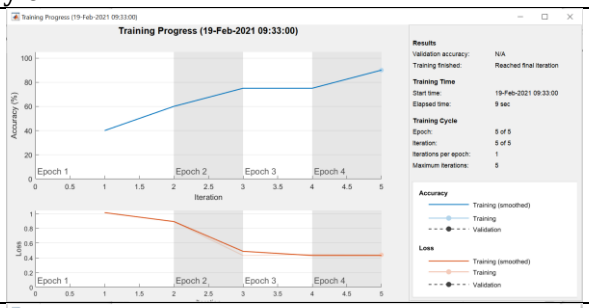
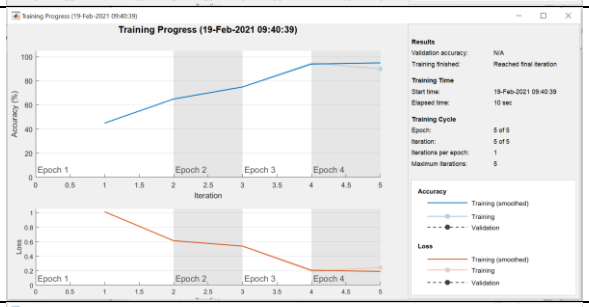
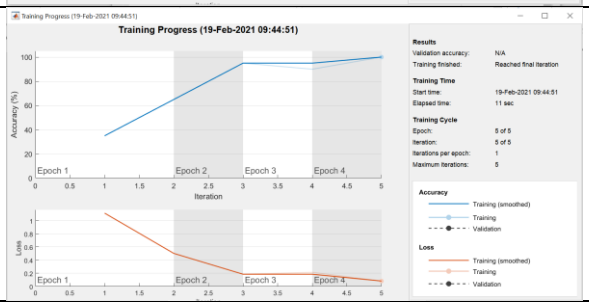
Part 1: Multiple Trainings with Constant Hyperparameters Study 6	
Trial 1	
Trial 2	
Trial 3	

Table B 13: Parameters for Parametric Study Part 2 Study 1.

Part 2: Changing Epoch Parameters Only Study 1					
Trial	**Epoch Size**	**Mini Batch Size**	**Initial Learning Rate**	**Train to Learn Ratio**	**Accuracy**
1	5	10	0.001	0.8	0.875
2	10	10	0.001	0.8	1
3	15	10	0.001	0.8	1

Table B 14: Graphs for Parametric Study Part 2 Study 1.

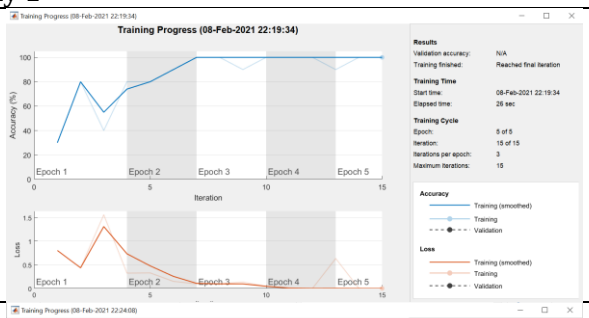
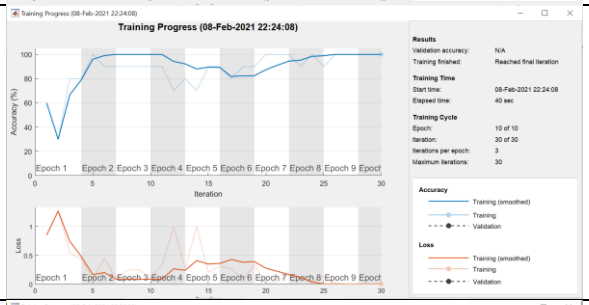
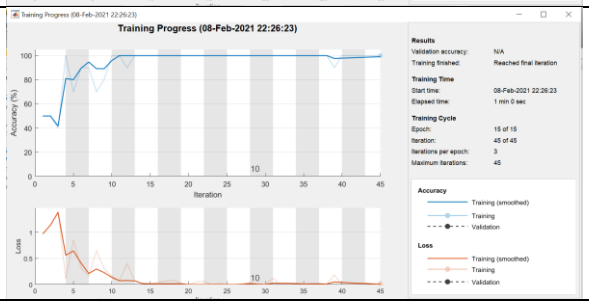
Part 2: Changing Epoch Parameters Only Study 1	
Trial 1	
Trial 2	
Trial 3	

Table B 15: Parameters for Parametric Study Part 3 Study 1.

Part 3: Changing Batch Size Study 1					
Trial	**Epoch Size**	**Mini Batch Size**	**Initial Learning Rate**	**Train to Learn Ratio**	**Accuracy**
1	10	5	0.001	0.8	1
2	10	10	0.001	0.8	1
3	10	15	0.001	0.8	1

Table B 16: Graphs for Parametric Study Part 3 Study 1.

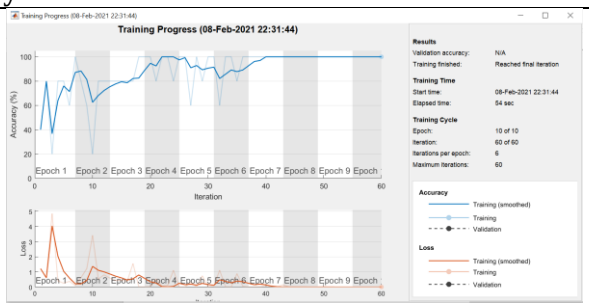
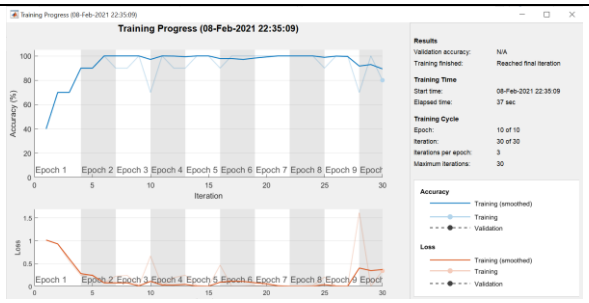
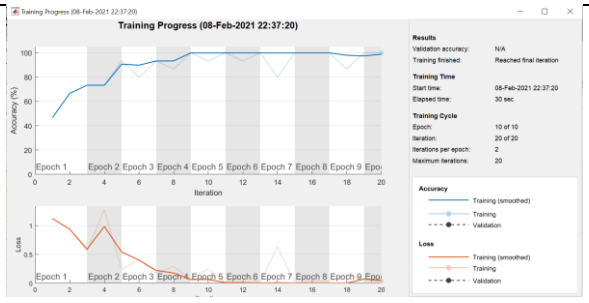
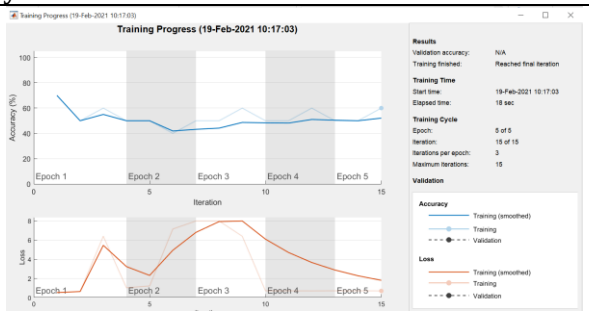
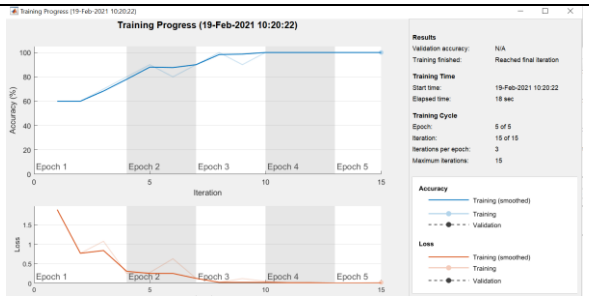
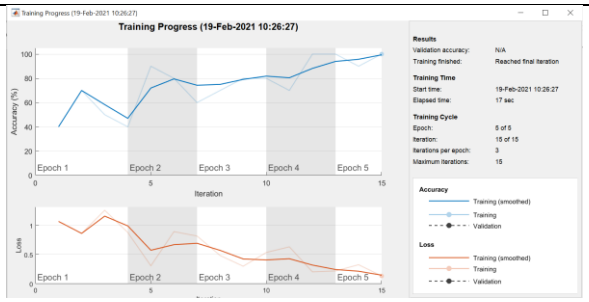
Part 3: Changing Batch Size Study 1	
Trial 1	
Trial 2	
Trial 3	

Table B 17: Parameters for Parametric Study Part 4 Study 1.

Part 4: Changing the Training Rate Study 1					
Trial	**Epoch Size**	**Mini Batch Size**	**Initial Learning Rate**	**Train to Learn Ratio**	**Accuracy**
1	5	10	0.01	0.8	0.500
2	5	10	0.001	0.8	0.8750
3	5	10	0.0001	0.8	0.8750

Table B 18: Graphs for Parametric Study Part 4 Study 1.

Part 4: Changing the Training Rate Study 1	
	
	
	

Appendix C: Larger Dataset Studies Raw Data

Table C 1: Parameters for Larger Dataset, Study 1.

Study 1: General Parameters					
Trial	**Epoch Size**	**Mini Batch Size**	**Initial Learning Rate**	**Train to Learn Rate**	**Accuracy**
1	5	16	0.001	0.8	0.800
2	5	16	0.001	0.8	0.850
3	5	16	0.001	0.8	0.500
4	5	16	0.001	0.8	0.500
5	5	16	0.001	0.8	0.500

Table C 2: Parameters for Larger Dataset, Study 2.

Study 2: Increase Epoch Size					
Trial	**Epoch Size**	**Mini Batch Size**	**Initial Learning Rate**	**Train to Learn Rate**	**Accuracy**
1	10	16	0.001	0.8	0.750
2	10	16	0.001	0.8	0.650
3	10	16	0.001	0.8	0.850
4	10	16	0.001	0.8	0.900
5	10	16	0.001	0.8	0.900

Table C 3: Parameters for Larger Dataset, Study 3.

Study 3: Increase Mini Batch Size					
Trial	**Epoch Size**	**Mini Batch Size**	**Initial Learning Rate**	**Train to Learn Rate**	**Accuracy**
1	10	20	0.001	0.8	0.900
2	10	20	0.001	0.8	0.950
3	10	20	0.001	0.8	0.750
4	10	20	0.001	0.8	0.850
5	10	20	0.001	0.8	0.800

Table C 4: Parameters for Larger Dataset, Study 4.

Study 4: Decrease Initial Learning Rate					
Trial	**Epoch Size**	**Mini Batch Size**	**Initial Learning Rate**	**Train to Learn Rate**	**Accuracy**
1	10	20	0.0001	0.8	0.950
2	10	20	0.0001	0.8	0.950
3	10	20	0.0001	0.8	1
4	10	20	0.0001	0.8	0.900
5	10	20	0.0001	0.8	0.900

Table C 5: Training Progress and Confusion Matrix Graphs for Larger Dataset Study 1.

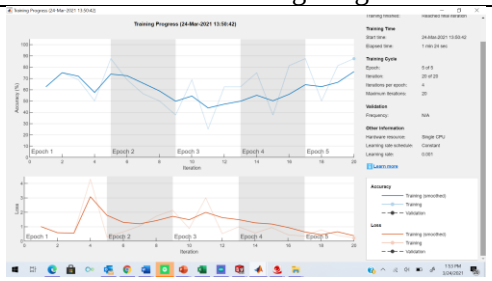
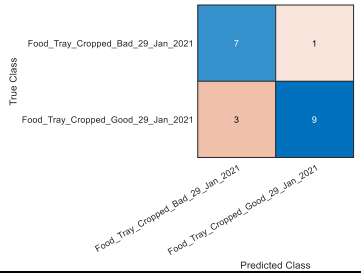
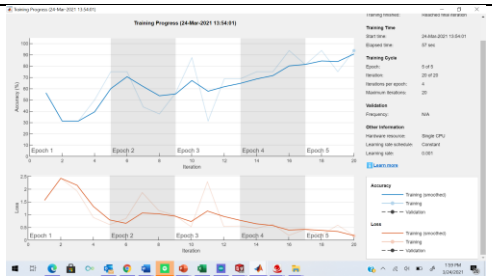
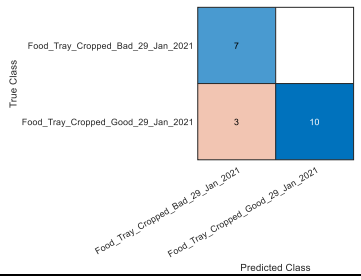
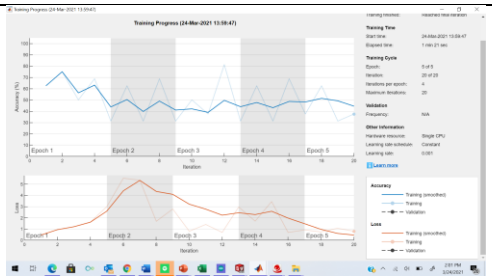
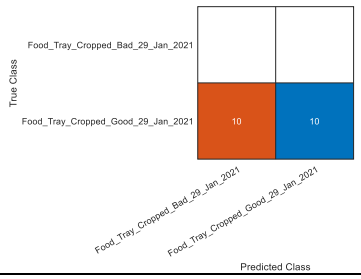
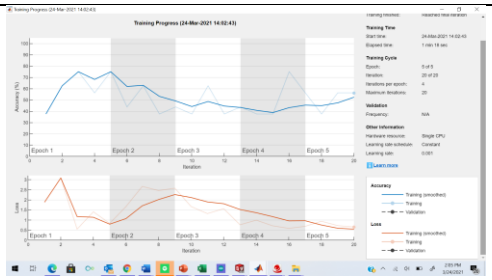
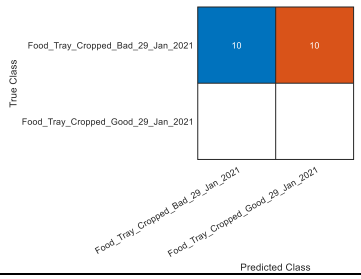
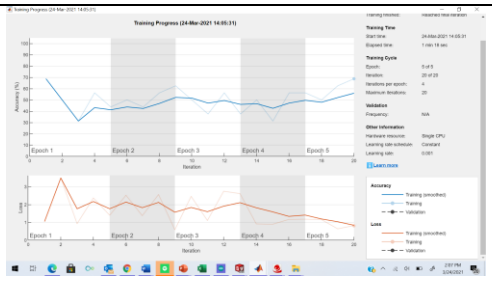
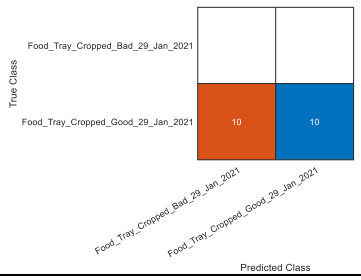
Study 1: General Parameters		
Trial #	Training Progress	Confusion Matix
1		
2		
3		
4		
5		

Table C 6: Training Progress and Confusion Matrix Graphs for Larger Dataset Study 2.

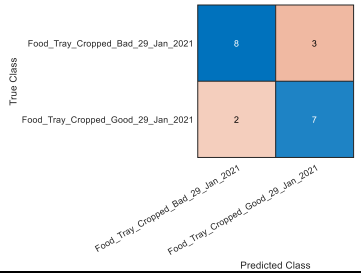
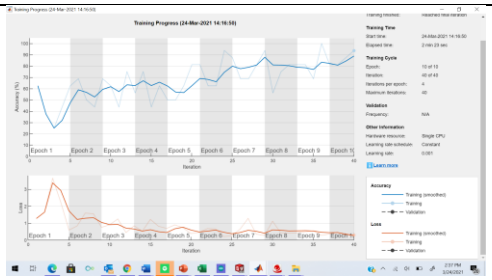
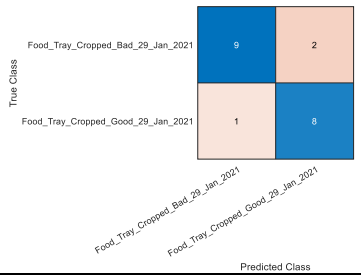
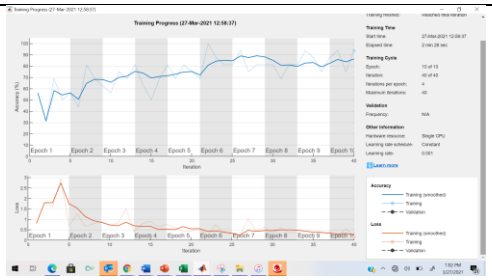
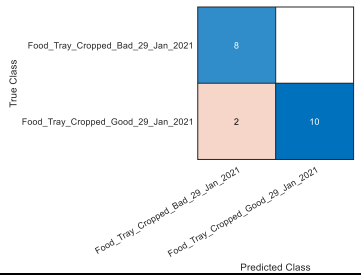
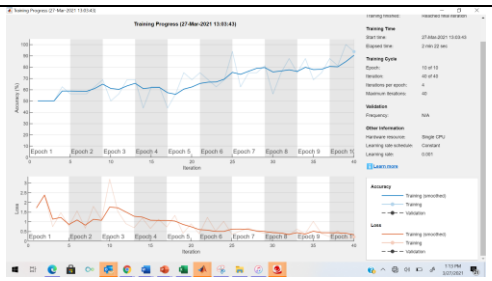
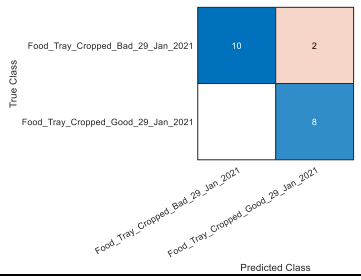
Study 2: Increase Epoch Size		
Trial #	Training Progress	Confusion Matix
1		
2		
3		
4		
5		

Table C 7: Training Progress and Confusion Matrix Graphs for Larger Dataset Study 3.

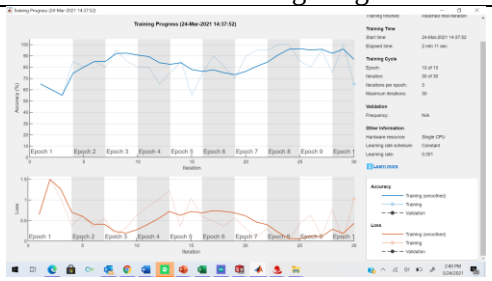
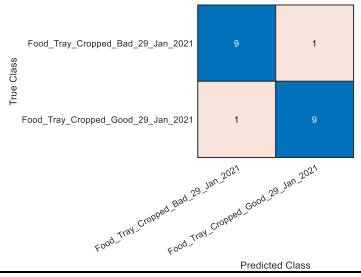
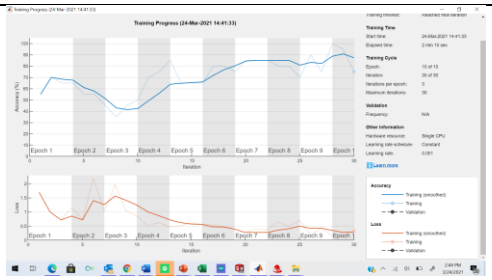
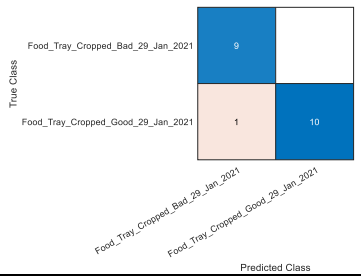
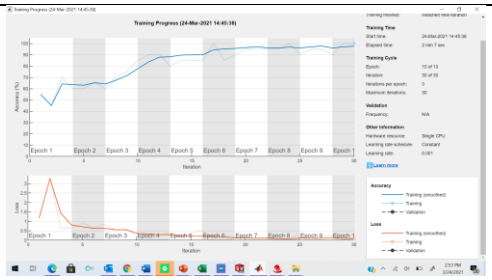
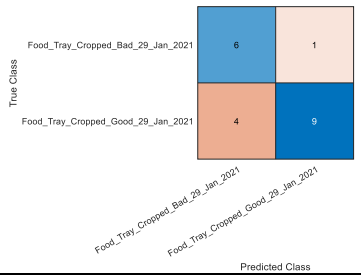
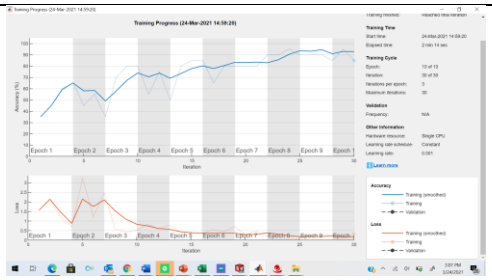
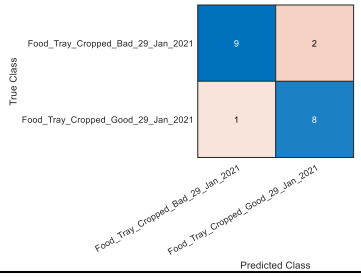
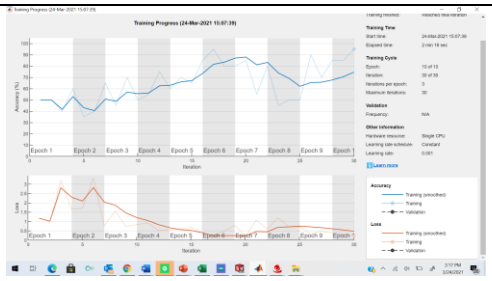
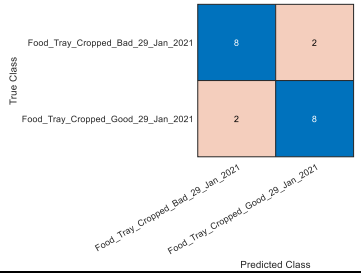
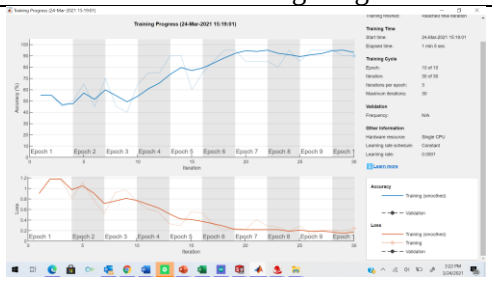
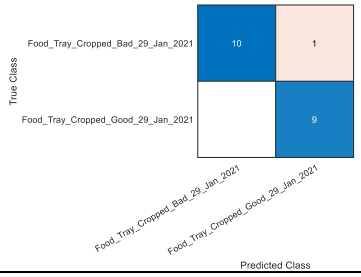
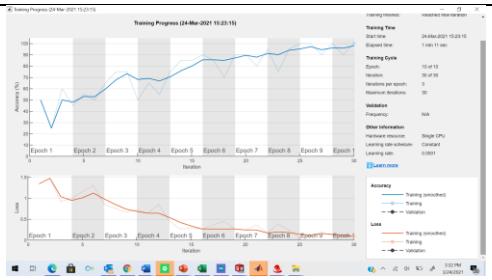
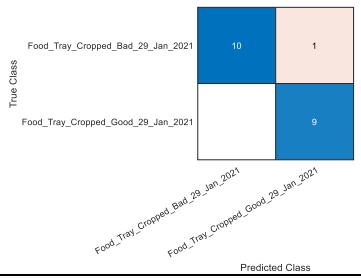
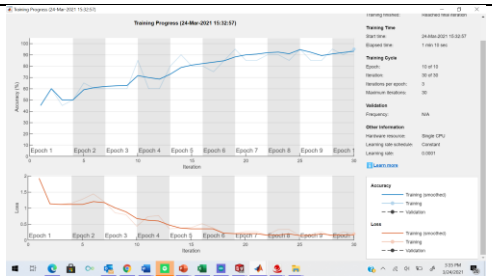
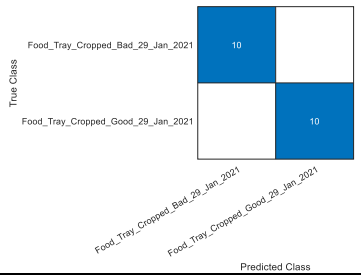
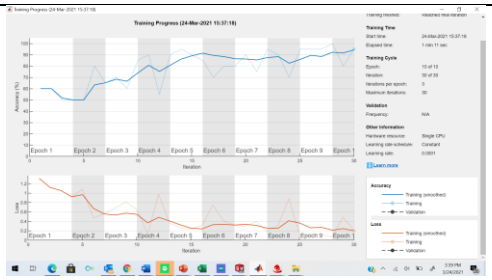
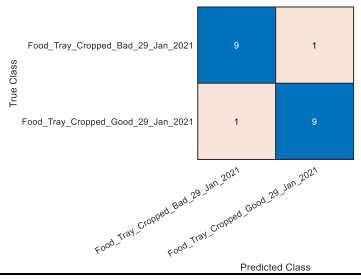
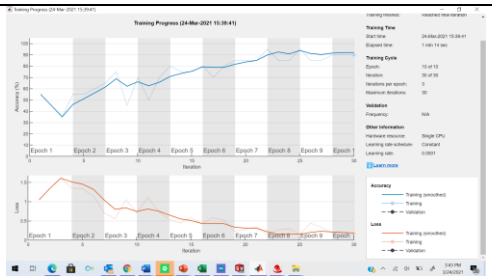
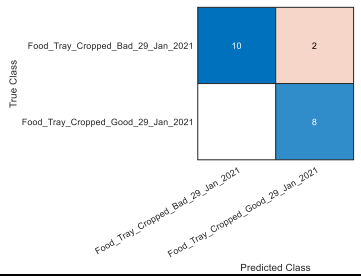
Study 3: Increase Mini Batch Size		
Trial #	Training Progress	Confusion Matix
1		
2		
3		
4		
5		

Table C 8: Training Progress and Confusion Matrix Graphs for Larger Dataset Study 4.

Study 4: Decrease Learning Rate		
Trial #	Training Progress	Confusion Matix
1		
2		
3		
4		
5		

Appendix D: Installing Custom ROS Messages

- This step is done one time to every new laptop that connects with the Dobot using ROS and MATLAB
- **Unzip the custom ROS message folder**
- Install (copy) the folder **inside the Documents folder** (remember the file path; do not rename)
- Note: there is nothing to execute, just **copy the folder** (unzipped) to the Documents folder.
- Open MATLAB
- Enter (in MATLAB) : **prefdir** (Note: this command will display the preferences folder name)
- Copy directory name and set current directory to this directory (copy and paste)
- Enter: **edit javaclasspath.txt** …and add the following line, then save and close (xxxx is user)
- c:\users\xxxx\Documents\custom_msgs2\custom_msgs2\matlab_gen\jar**dobot-1.0.jar**
- Enter: **addpath**('c:\users\xxxx\Documents\custom_msgs2\custom_msgs2\matlab_gen**msggen**')
- Enter: **savepath**
- Close and restart MATLAB (you may also have to reboot machine)
- Enter: **rosmsg show dobot/SetPTPCmdRequest**

- Start **VMware** on Engr laptop and start **Ubuntu 64-bit (ROS)** file (linux OS)
- **Turn on Dobot** and **connect USB cable** to laptop
    - Pop-up window will ask if you want to connect USB device to virtual machine - YES
- Open a Terminal window
- Enter: **roscore**
- Open another Terminal window
- Enter: **rosrun dobot DobotServer ttyUSB0**
    - Note: if above step fails, unplug USB and plug in again, then re-enter command)
- Open another Terminal window
- Enter: **rosservice list** Note: you should see list of dobot services (this is a test)
- Enter: **ifconfig** (to get IP address of ROS virtual machine – record ip address)

Leave all terminal windows running and return to MATLAB

Open MATLAB
Enter: rosinit(ipaddress_of_VirtualMachine)
Enter: rosservice list
Enter: rosservice info /DobotServer/SetPTPCmd

R. Avanzato, Class Lecture, Topic: "A Robot Manipulator Control Labs with ROS and MATLAB and Dobot Articulated Arm v1.4", EDSGN 420, College of Engineering, Penn State, Abington, Pa., Mar., 2021

BIBLIOGRAPHY

[1] Bulnes, F.G., Usamentiaga, R., Garcia, D.F. *et al.* An efficient method for defect detection during the manufacturing of web materials. *J Intell Manuf* **27,** 431–445 (2016). https://doi.org/10.1007/s10845-014-0876-9

[2] S. D. El Wakil, *Processes and Design for Manufacturing*, Third. New York, New York: CRC Press Taylor & Francis Group, 2019.

[3] C. A. Escobar and R. Morales-Menendez, "Machine learning techniques for quality control in high conformance manufacturing environment," *Advances in Mechanical Engineering*, vol. 10, no. 2, p. 168781401875551, 2018.

[4] K. Paraskevoudis, P. Karayannis, and E. P. Koumoulos, "Real-Time 3D Printing Remote Defect Detection (Stringing) with Computer Vision and Artificial Intelligence," *Processes*, vol. 8, no. 11, p. 1464, 2020.

[5] "What Is Deep Learning?: How It Works, Techniques & Applications," *How It Works, Techniques & Applications - MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/discovery/deep-learning.html. [Accessed: 05-Mar-2021].

[6] "Convolutional Neural Network," *MathWorks*. [Online]. Available: https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html. [Accessed: 05-Apr-2021].

[7] "What Is a Neural Network?," *What Is a Neural Network? - MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/discovery/neural-network.html. [Accessed: 05-Mar-2021].

[8] "Transfer Learning Using AlexNet," *MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/transfer-learning-using-alexnet.html. [Accessed: 05-Mar-2021].

[9] "Deep Learning in MATLAB," *MathWorks*. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/deep-learning-in-matlab.html#:~:text=Deep%20Learning%20Toolbox%E2%84%A2%20provides,vision%20algorithms%20or%20neural%20networks. [Accessed: 05-Apr-2021].

[10] "What Is Deep Learning?: How It Works, Techniques & Applications," *How It Works, Techniques & Applications - MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/discovery/deep-learning.html. [Accessed: 31-Mar-2021].

[11] K. Manke, "Deep learning helps robots grasp and move objects with ease," *ScienceDaily*, 18-Nov-2020. [Online]. Available: https://www.sciencedaily.com/releases/2020/11/201118141827.htm. [Accessed: 19-Nov-2020].

[12] A. Saood and I. Hatem, "COVID-19 lung CT image segmentation using deep learning methods: U-Net versus SegNet," *BMC Medical Imaging*, 09-Feb-2021. [Online]. Available: https://bmcmedimaging.biomedcentral.com/articles/10.1186/s12880-020-00529-5. [Accessed: 31-Mar-2021].

[13] "Machine Learning and AI in Manufacturing - The Complete Guide," *Seebo*, 14-Oct-2020. [Online]. Available: https://www.seebo.com/machine-learning-ai-manufacturing/. [Accessed: 05-Mar-2021].

[14] C. Pazzanese, “Ethical concerns mount as AI takes bigger decision-making role,” *Harvard Gazette*, 04-Dec-2020. [Online]. Available: https://news.harvard.edu/gazette/story/2020/10/ethical-concerns-mount-as-ai-takes-bigger-decision-making-role/. [Accessed: 05-Apr-2021].

[15] n.d. *Robo 3D User Manual*. [ebook] Available at: <http://download.robo3d.com/docs/R1/GETTING_STARTED_GUIDE9-10.pdf> [Accessed 05 March 2021].

[16] I. S. A. Krizhevsky, H. G. S. Nitish, M. K. T. W. Karl, L. Shao, J. C. J. Yosinski, Y. B. D. Erhan, P. N. A. Halevy, B. K. A. Esteva, K. T. L. Geert, M. K. T. LL. Joffrey, L. B. Y. LeCun, W. B. K. VC. Nitesh, A. A. J. Justin, Y. B. Y. Wang, J. P. A. B. Camilo, H. G. M. Laurens, J. F. Gabriel J. Brostow, G. L. David, A. G. R. RS. Sutton, S. Hochreiter, A. M. Mateusz. Buda, and T. M. K. DJ. Drown, “A survey on Image Data Augmentation for Deep Learning,” *Journal of Big Data*, 01-Jan-1970. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0. [Accessed: 06-Mar-2021].

[17] “Deep Learning: Transfer Learning in 10 lines of MATLAB Code,” *Deep Learning: Transfer Learning in 10 lines of MATLAB Code - File Exchange - MATLAB Central*. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/61639-deep-learning-transfer-learning-in-10-lines-of-matlab-code. [Accessed: 05-Apr-2021].

[18] R. Avanzato, Class Lecture, Topic: “AI and Deep Learning – Lecture #5: MATLAB Coding v2.5”, EDSGN 420, College of Engineering, Penn State, Abington, Pa., Mar., 2021.

[19] S. Sharma, “Epoch vs Batch Size vs Iterations,” *Medium*, 05-Mar-2019. [Online]. Available: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9. [Accessed: 05-Apr-2021].

[20] R. Haleva, “How to Break GPU Memory Boundaries Even with Large Batch Sizes,” *Medium*, 23-Jan-2020. [Online]. Available: https://towardsdatascience.com/how-to-break-gpu-memory-boundaries-even-with-large-batch-sizes-7a9c27a400ce. [Accessed: 05-Apr-2021].

[21] P. Surmenok, “Estimating an Optimal Learning Rate For a Deep Neural Network,” *Medium*, 14-Jul-2018. [Online]. Available: https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0. [Accessed: 05-Apr-2021].

[22] J. Brownlee, “How to Configure the Learning Rate When Training Deep Learning Neural Networks,” *Machine Learning Mastery*, 06-Aug-2019. [Online]. Available: https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/. [Accessed: 05-Apr-2021].

[23] “What is Surface Roughness: Roughness Measurement Working Principle,” *Ametek Taylor Hobson - Metrology Experts - Precision Metrology Equipment Manufacturer*. [Online]. Available: https://www.taylor-hobson.com/resource-center/blog/2019/october/what-is-surface-roughness. [Accessed: 05-Apr-2021].

[24] R. Avanzato, Class Lecture, Topic: “Computer Vision Lecture #4 (v2.3) Image Processing Part 3”, EDSGN 420, College of Engineering, Penn State, Abington, Pa., Mar., 2021

[25] “Robotics Solution Provider for STEAM Education, Industry & Businesses,” *DOBOT*. [Online]. Available: https://www.dobot.cc/dobot-magician/specification.html. [Accessed: 05-Apr-2021].

[26] Avanzato, R. L. (2020, June), *Development of a MATLAB/ROS Interface to a Low-cost Robot Arm* Paper presented at 2020 ASEE Virtual Annual Conference Content Access, Virtual On line. 10.18260/1-2—34447

[27] R. Avanzato, Class Lecture, Topic: "A Robot Manipulator Control Labs with ROS and MATLAB and Dobot Articulated Arm v1.4", EDSGN 420, College of Engineering, Penn State, Abington, Pa., Mar., 2021

[28] R. Avanzato, Class Lecture, Topic: "AI and Deep Learning – Lecture #4: Object Detection and Data Augmentation v1.0", EDSGN 420, College of Engineering, Penn State, Abington, Pa., Mar., 2021.

ACADEMIC VITA

EDUCATION/COURSEWORK

The Pennsylvania State University, Abington, PA May 2021
Major: *Engineering B.S. (Multidisciplinary Engineering Design Option)*
Relevant Coursework: Engineering Systems Design; Materials and Manufacturing; Advanced Robot Design and Application; Computational Modeling Methods (FEA Analysis); Product Realization

AWARDS/HONORS

(To ensure descriptions for the honors and awards are complete and accurate, the bulleted descriptions are from those provided by Penn State University either from the University website or from award ceremony pamphlets.)

Dean's List Fall 2017-Fall 2020

- In recognition of academic excellence, selected students are named to the Dean's List each semester.

Harold J. and Eileen M. Taub Family Scholarship Spring 2020

- Awarded to high achieving students who share the vision of experiential learning, ethical leadership in action, and values to make an impact within their field of study.

American Military Engineers: Ammann & Whitney Scholarship Spring 2020

- Awarded to engineering students who have outstanding leadership, high ethics, and scholarship achievement.

Dr. Sanford F. Nicol Memorial Award in Engineering Spring 2019, Spring 2020

- A monetary award, presented to an outstanding student in engineering courses, endowed by the family, colleagues, and friends of the late Dr. Sanford F. Nicol, director of academic affairs for more than two decades. Selected by Engineering Department faculty and advisors.

Abington Honors Program Award Spring 2019

- For students who have successfully completed the Penn State Honors Program. These students have maintained a specific GPA in a curriculum including rigorous honors coursework over four semesters. Selected by honors coordinator and honors advisor.

Gold Academic Award Spring 2019

- For students who hold active leadership roles in clubs and have maintained a high-grade point average during their leadership. Selected by Tracy Reed Assistant Director of Student Engagement and Leadership.

WE Design Competition First Round of Competition Finalist Spring 2019

- Women in Engineering Design Competition sponsored by Norfolk Southern and Penn State Altoona.

President's Freshman Award Spring 2018

- Presented annually to undergraduate degree candidates and provisional students who have earned a specific GPA within their first semester.

Abington College Undergraduate Research Poster Fair First Place Winner Spring 2018

- It is the mission of Abington College Undergraduate Research Activities (ACURA) to develop students who are critical thinkers and creative scholars by engagement in scientific experimentation, inquiry-based research, and exploration of the arts through a yearlong research experience within the community of teacher-scholars.

Penn State Abington's Honor Program Fall 2017-Spring 2019

- Entering first-year students are invited to participate in the program based on significant academic achievement and potential. Must participate in honors courses, maintain strong grade point averages, and participate in research, international study or leadership activities.

Abington Fellowship Scholarship Fall 2017

- Each year, a select group of entering first-year honors students are invited to participate in the Abington Fellows program, which provides exceptionally accomplished students unique ways to shape their college educations. Participants receive mentoring from dedicated faculty and additional merit scholarship funding.

RELEVANT PROJECTS

Design and Manufacturing Prototyping Project Fall 2020

Auto Food Tray

- Objective of the class was to focus on the design and development of a product to prototype.
- The chosen idea was to develop a portable food tray that can be attached to a car air vent for easy accessibility to food while parking, and to reduce distractions due to eating and dropping food while driving.
- Prototype designed in SolidWorks and printed using a 3D printer.

Integrated System Design Project Spring 2020

Child Alert Integrated System for Automobiles

- Parents unintentionally forget their children in cars which can lead to accidental death. In order to prevent tragedy, a device needs to be produced to detect and alert if a child is in the car.
- The system must sense a load to determine if a child is present in a car seat. Once the driver side door is opened and a child is detected, an alarm will sound to remind the parent of the presence of their child.

Engineering System Design Project Fall 2019

Device That Prevents Touching Non-Sterile Surfaces in Public Restrooms

- Washing your hands prevents you from getting sick or spreading germs; however, after you wash your hands in a public restroom you must immediately touch a nonsterile surface to exit the bathroom.
- The objective was to form a team and design a device that prevents the spread of germs when opening restroom doors.
- The design processes consisted of: integrating functional and non-functional requirements to understand scope; creating concept designs through research and brainstorming; utilizing performance and requirements matrices to determine the most effective design concept; and modifying and verifying design in reference to specifications, constraints, and customer needs.

Engineering Club Team Project Fall 2019

Making a Floating Arm Trebuchet

- Built a floating arm trebuchet for Penn State Abington's Engineering Club event which used a trebuchet to launch pumpkins.
- This project goes through the design process as the trebuchet was designed, modified, and simulated in SolidWorks, and was built by members of the club.
- The project incorporates teamwork and solid communication skills in order for the trebuchet to be properly built, as well as gaining hands on experience with building something tangible.

Technical Writing Procedure Writing Project Spring 2019

How to Use the Arduino Uno

- Collaborated with a team to produce a manual on "How to Use the Arduino Uno" which included circuit diagrams, commented code, and functions of the components.

- This assignment improved writing skills and integrated knowledge of circuits into one project.

Abington College Undergraduate Research Project: Astrophysics Fall 2017 – Spring 2018
A Survey with Radio Astronomy of Cold H1 Clouds in the Milky Way
- Analyzed absorption features in the plane of the Milky Way and found the velocities of these absorption features, identified which arm of the galaxy the emissions were located, and mapped an area of the local arm of the Milky Way galaxy.
- Presented in Abington College Undergraduate Research Poster and won First Place under the Division of Science and Engineering (Spring 2018)
- Presented at Abington College's Fall Colloquium for Undergraduate Research (Fall 2018)

INTERNSHIPS/WORK EXPERIENCE

Lockheed Martin, King of Prussia, Pennsylvania
Incoming Systems Engineer Associate for RMS June 7, 2021
- Full time position under RMS.

Systems Engineering Intern for RMS (College Student Tech Spec) June 2020 – August 2020
- Implemented updates to project architecture including new system models and architecture diagrams, updated documentation to reflect changes and improvements, and tracked and flagged change requests for organizational and reference purposes.
- Trained in model-based systems engineering to increase efficiency and understanding of effective project architecture paired with project documentation.
- Assisted with Engineering Ahead Program by creating presentations and worksheets, presenting demos, reaching out to presenters, leading and presenting during weekly meetings, and collaborating meeting plans between Penn State and Lockheed.

Systems Engineering Sr Intern June 2019 – August 2019
- Built a system to standardize inventory processes and utilized lean process flow principles to achieve enhanced workflow.
- Documented processes, requirements, and manual instructions for hardware system implementation including both physical hardware placement and software use for hardware tracking.

Penn State Abington, Abington, Pennsylvania
Teaching Assistant for Engineering Mechanics Dynamics Spring 2021
- Holds review sessions, grades discussion forums, and reserves time for office hours to help students with dynamics coursework.

Teaching Assistant for Engineering Mechanics Statics Fall 2020
- Held review sessions, helped grade exams and discussion forums, and reserved time for office hours to help students in statics.

Tutor for First Year Seminar Students Fall 2018
- Assisted first year students with their academic work specializing in Calculus, Chemistry, and Physics.

Teaching Assistant for Penn State Abington's First Year Seminar Class Fall 2018
- Attended seminar class with students and reserved time for office hours during the week to help students in their academic coursework.

Peer Tutor for Penn State's Engineering Ahead Program Summer 2018

- Six-week program in the summer in which I helped students learn the material, provided practice, graded papers, gave academic advice, and gave insight into classes they would take in the fall.

Bucks County Free Library, Yardley, Pennsylvania Summer 2013 – Summer 2017

Volunteer at Yardley Branch of the Bucks County Free Library System

- Helped shelve books and communicated effectively with customers when assisting them.

EXTRACURRICULAR

The Pennsylvania State University, Abington, PA

Engineering Club Treasurer Spring 2019 – Spring 2021

- Responsible for submitting approval requests for monetary aid through the funding committee for special events and club meetings. Collaborates closely with the President and actively plays a role in planning for meetings and attends events and meetings.

Engineering Club President Fall 2018 – Spring 2019

- Oversees the club and provides open communication between the club advisor and office of Student Engagement and Leadership.
- Plans events, meetings, and creates project ideas for meetings. Attends all required President's meetings along with calling meetings between the board members for open communication. Works collaboratively with club members, and molds a space where engineers can come together and accomplish productive activities along with discussion.

Abington Christian Fellowship Active Member Fall 2018 – Spring 2020

- Attends club meetings and weekly Bible studies, and participates in campus outreach events.

THON Member Fall 2017 – Spring 2018

- Participated in canning event in order to raise money for childhood cancer.

QUALIFICATIONS

- Agile project management (Scrum methodology), TeamGantt, Microsoft Word, Excel, PowerPoint, Outlook, OneNote, Skype, technical writing, additive manufacturing, Zoom, AutoCAD, SOLIDWORKS, Multisim, Arduino Uno Microcontroller and Breadboard, C++ Programming, MATLAB, FPGA board programming using Verilog
- Exceptional communication skills, interpersonal relations, fluency in public speaking, ability to work well in collaborative environments, strong problem-solving skills and work ethic, ability to rapidly prototype, knowledge of product development