RYERSON UNIVERSITY
FACULTY OF ENGINEERING AND ARCHITECTURAL SCIENCE
DEPARTMENT OF AEROSPACE ENGINEERING

**Implementation of Object Recognition Algorithm to enhance Manufacturing and
Maintenance Tasks on an Aircraft**

Abhinav Sundar

AER 870 Aerospace Engineering Thesis – Final Report

Submission Date: April 13, 2020

Faculty Advisor: Dr. Joon Chung

Acknowledgements

I would like to thank my research advisor, Dr Joon Chung, for his guidance and support through the progression of this thesis. I would like to thank Cassandra Czobit, and the MIMS Lab Research group, Pratik Pradhan, Isabel Jojart and Vishaal Venkatesh for continuous support and guidance. Finally, I would like to thank my family, especially my mom, for their strength, motivation and support throughout my research.

Abstract

The objective of this thesis was to evaluate the viability of implementation of an object recognition algorithm driven by deep learning for aerospace manufacturing, maintenance and assembly tasks. Comparison research has found that current computer vision methods such as, spatial mapping was limited to macro-object recognition because of its nodal wireframe analysis. An optical object recognition algorithm was trained to learn complex geometric and chromatic characteristics, therefore allowing for micro-object recognition, such as cables and other critical components. This thesis investigated the use of a convolutional neural network with object recognition algorithms. The viability of two categories of object recognition algorithms were analyzed: image prediction and object detection. Due to a viral epidemic, this thesis was limited in analytical consistency as resources were not readily available. The prediction-class algorithm was analyzed using a custom dataset comprised of 15 552 images of the MaxFlight V2002 Full Motion Simulator's inverter system, and a model was created by transfer-learning that dataset onto the InceptionV3 convolutional neural network (CNN). The detection-class algorithm was analyzed using a custom dataset comprised of 100 images of two SUVs of different brand and style, and a model was created by transfer-learning that dataset onto the YOLOv3 deep learning architecture. The tests showed that the object recognition algorithms successfully identified the components with good accuracy, 99.97% mAP for prediction-class and 89.54% mAP. For detection-class. The accuracies and data collected with literature review found that object detection algorithms are accuracy, created for live-feed analysis and were suitable for the significant applications of AVI and aircraft assembly. In the future, a larger dataset needs to be compiled to increase reliability and a custom convolutional neural network and deep learning algorithm needs to be developed specifically for aerospace assembly, maintenance and manufacturing applications.

Table of Contents

1.0	Introduction.....	vii
1.1	Computer Vision and Deep Learning	1
1.2	Scope and Structure of the Thesis.....	2
2.0	Background.....	3
2.1	Convolutional Neural Networks	3
2.2	Overview of Dataset Format and Compared CNNs	6
2.3	Current Commercial Application.....	9
3.0	Algorithm Design.....	11
3.1	Design Objective and Requirements.....	11
3.1.1	Design Objectives	11
3.1.2	Design Requirements and Verifications:	12
3.2	Methodology	13
3.2.1	Ground Station Setup.....	14
3.2.2	Dataset Collection and Preparation.....	15
3.2.3	Transfer-Learning	17
3.2.4	Accuracy and Application testing	20
4.0	Results and Discussion	21
4.1	Prediction-class	21
4.2	Detection-class.....	23
4.3	Degree of Requirements Matrix.....	29
5.0	Conclusion	30
6.0	Future Considerations	31
	References.....	32
	Appendix A – Prediction class.....	34
	Training Algorithm	34
	Training Algorithm Raw Results	35
	Application Algorithm.....	36
	Application Algorithm Raw Results.....	37
	Appendix B – Detection-class	39
	Training Algorithm	39
	Modal Testing Algorithm	40

Application Algorithm	41
Application Algorithm Raw Results	42

List of Figures

Figure 1: Local Receptive Field extraction method [7].....	4
Figure 2: Feature map creation [8].....	5
Figure 3: InceptionV3 Inception Module A.....	7
Figure 4: InceptionV3 Inception Module B.....	8
Figure 5: InceptionV3 Inception Module C.....	8
Figure 6: Dataset Preparation - Prediction-class.....	16
Figure 7: Dataset Preparation - Detection-class.....	17
Figure 8: Transfer-learning Process - Prediction-class.....	19
Figure 9: Transfer-learning Process - Detection-class.....	20
Figure 10: Model Testing - Prediction-class.....	21
Figure 11: Application testing - Prediction-class.....	22
Figure 12: Object Instances versus AP	23
Figure 13: Model Testing - Detection-class.....	24
Figure 14: Detection Results - Lexus RX 350.....	24
Figure 15: Detection Results - Lamborghini Urus.....	25
Figure 16: Detection Results - Tesla Model X	25
Figure 17: Detection Results - Hyundai Elantra	26
Figure 18: Detection Results - Mercedes G-Wagon	26
Figure 19: Training Algorithm Raw Results - Prediction-class.....	35
Figure 20: Aviation Inverter Prediction Results	37
Figure 21: MaxFlight Inverter Prediction Results	37
Figure 22: PE (Green) Wire Prediction Results.....	37
Figure 23: PE (Green+Yellow - 240V) Wire Prediction Results	37
Figure 24: U (Red) Wire Prediction Results.....	38
Figure 25: Application Algorithm Code - Detection-class.....	41
Figure 26: Lexus RX 350 Detection Results Raw	42
Figure 27: G-Wagon Detection Results Raw.....	43
Figure 28: Hyundai Elantra Detection Results Raw	44
Figure 29: Tesla Model X Detection Results Raw	45
Figure 30: Lambourghini Urus Detection Results Raw.....	46

List of Tables

Table 1: Design Requirements - Prediction-class.....	12
Table 2: Design Requirements - Detection-class.....	12
Table 3: Design Requirement Verifications – Prediction-class.....	13
Table 4: Design Requirement Verifications – Detection-class.....	13
Table 5: Dataset Format Comparison - Detection-class	16
Table 6: CNN Comparison - Prediction-class [10][12]	18
Table 7: CNN Comparison - Detection-class [16].....	18
Table 8: Degree of Requirements Matrix - Prediction-class	29
Table 9: Degree of Requirements Matrix - Detection-class	29

Nomenclature

AP	Accuracy Precision
ATTOL	Autonomous Taxi Take-Off and Landing
AVI	Automatic Visual Inspection
CNN	Convolutional Neural Network
CV	Computer Vision
IoU	Intersection over Union
mAP	mean Accuracy Precision
Microsoft COCO	Microsoft Common Objects in Context
MIMS	Mixed-Reality Immersive Motion Simulation
ML	Machine Learning
Pascal VOC	Pascal Visual Object Class
R-CNN	Region-based Convolutional Neural Network
ResNets	Residual Networks
R-FCN	Region-based Fully Connected Network
RPN	Region Proposal Network

1.0 Introduction

The main objective of this thesis was to test the viability of object recognition algorithms for the implementation into manufacturing, assembly and maintenance tasks. The aerospace industry can be broken into three hardware sectors: manufacturing, maintenance and assembly. A few examples of tasks in these sectors are sheet metal cutting, visual inspections and landing gear assembly. Artificial Intelligence has various applications across various industries; however, it is young to aerospace industry. Large companies such as Boeing and Airbus have recently started to utilize object recognition in system inspections. With the implementation of an object recognition algorithm the aerospace hardware sectors will see an increase of worker efficiency and effectiveness because of the reduction of human error. Two object recognition algorithms were tested: prediction-class and detection-class. The primary objective and secondary objectives, such as cloud-based dataset storage and model creation, were also taken into consideration to design and implement algorithms that recognize a mean accuracy persicion (mAP) of 90% and 85% for prediction-class and detection-class, respectively. The dataset preparation format was selected based off accuracy between Microsoft COCO and Pascal VOC. For the creation of the object recognition models, CNNs such as ResNet-50, InceptionV3 and YOLOv3 were researched and compared based on mAP and speed. Deep learning CNNs require large amounts of images and annotations for an accurate model to be created. This is achieved by large corporations due to the quantity of researchers and amount of time. The time constraint of three months, inaccessibility to a variety of aerospace components, and the difficult quarantine situation caused by the global pandemic, COVID-19, did not give enough time and resources to complete this research to its totality. For the proof of concept, this thesis used undersized and under-diversified datasets comprised of aerospace and manufacturing industry-related components.

1.1 Computer Vision and Deep Learning

Computer vision (CV) backed by a deep learning enabled object recognition algorithm successfully recognized target objects and provided all the required information for various manufacturing and maintenance tasks. CV is the ability for computers to analyze and understand the visual world. In the manufacturing world, a major use for computer vision is fault detection. This ability for fault detection can be integrated to an automatic assembly line to increase efficiency and effectiveness [1]. With the increase in the amount of data available for CV tasks, the scope of the algorithms has grown. Machine learning initially used statistical learning algorithms such as linear regression, logistic regression, decision trees and support vector machines (SVM) to do pattern analysis and object detection. The method of deep learning using neural networks has been developed recently for the purpose of feature extraction and pattern analysis [2]. Object recognition is a CV task for identifying objects in images or video. This task can be broken down into three main categories: image classification, object localization and object

detection. This thesis utilized a prediction-class algorithm based on the image classification task; and a detection-class algorithm based on the object detection task, which consists of an algorithm that produces a list of object categories present, bounding box location, and scale of every instance of each object category [3].

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstractions [4]. A major benefit to deep learning algorithms over older learning algorithms was the ability to continuously increase performance as the neural networks and datasets increase in size and complexity [5] [6]. This was key to manufacturing applications as deep learning enabled for object detection could continuously increase accuracy with the addition of more data. These algorithms can improve the accuracy performance of object recognition codes.

1.2 Scope and Structure of the Thesis

The hardware sectors of the aerospace industry are pushed by accuracy and efficiency. Tasks such as assembly requires a large amount of time for part identification. Object recognition algorithms would, for example, be trained to conduct automatic visual inspections of the aircraft and aid aircraft technicians to complete landing gear assemblies by quickly locating the required part. Object recognition algorithms in the aerospace industry also provides a significant support to technician training. With the detection features of an object detection algorithm, an effective augmented reality-based training application can be created to decrease technician practical training time and aid newly trained technicians to identify target components. This thesis intended to improve the task of part identification by removing human error from the process, allowing worker efficiency and performance to increase.

To give a fundamental understanding into deep learning-enabled object recognition algorithms, this thesis reviews the ground CNN structure and the underlying functions that enable the breakdown and processing of image features; beneficial modifications to the original CNN; and commercial applications of CV in the commercial sector. The methodology is described to show how the design requirements were achieved. Finally, the results are discussed and assessed based on the research and design objectives.

2.0 Background

The prediction-class and detection-class algorithms used modified CNNs. These CNNs were developed to achieve better results for the object recognition task. The CNNs used were InceptionV3 and YOLOv3. the original CNN was updated for the explicit purpose of increased accuracy and speed, each with layer additions and modifications. The theory behind these CNNs are covered this section, with trade-off studies covered in the methodology section.

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are the main type of neural networks used in object recognition. Previous networks have each neuron from the previous layer connected to the neurons in the next layer. Due to this design, limitations on the size and processing ability of those networks are apparent. However, with the use of the three basic ideas of a CNN, a faster, deeper and larger network can be created. These concepts are as follows: local receptive fields, shared weights and pooling [7]. Rather than connected an entire image to the next layer, CNN layers take an input layer and the network connects a local region of neurons to one neuron of the next layer. The application of CNN layers creates a feature map. As a result, that neuron of the following layer learns the weight and bias of the local region, commonly known as the local receptive field. The local receptive field was then slid over the image and then respectively assigned to a neuron, with the field moving from left to right and top to bottom. Figure 1 demonstrates the relation between the feature map and the local receptive field:

Figure 1: Local Receptive Field extraction method [7]

These weights and biases are shared over the next resultant layer of neurons, and each of that resultant feature map can extract one type of feature. Convolutional layer includes multiple feature map that extract various types of features from the target image., Figure 1 displays how the neuron weight (5x5) of the preceding layer is projected and recorded on the following neuron. This translates the 28X28 image into a 24X24 feature map [7]. The shared weights and biases are referred to as the kernel or filter. The convolutional layers method of feature extraction reduces the number of parameters calculated drastically, when compared to the older fully connected layers. The convolutional layers' sliding local receptive field's extraction method can be mathematically represented by equation 1 [8]:

$$G[m, n] = (f * h)[m, n] = \sum_j \sum_k h[j, k] f[m - j, n - k] \quad (1)$$

The figure 2 below accurately explains how the feature map is created using an input image and set kernels/filters:

Figure 2: Feature map creation [8]

The value of each feature map neuron is a two-part iterative equation. The first part is the weighted average (z) of the input local receptive field values (x) for each neuron is the function of the activation function (a , or x for the first layer) of the previous layer, the set weight of the neuron (w), and the set bias of the neuron (b). The second part is the value of the neuron which is calculated by multiplying the non-linear activation function (g) with z [9]. Therefore, for the first feature map for the first neuron, the weighted average, the value (y), and the next layer's activation function (a) would be:

$$x = a_1^0 \quad (2)$$

$$w^T * x = w_1 x_1 + w_2 x_2 \dots w_n x_n \quad (3)$$

Where n equals the number of input local receptive field values.

$$z_1^1 = w_1^T * x + b_1 \quad (4)$$

$$a_1^1 = y = g^1(z_1^1) \quad (5)$$

Where subscript denotes neuron number and superscript denotes layer number [9]. Weight and bias are shared values for the entire layer. However, these equations can be simplified into a matrix for the whole layer. The weight average of the values (Z) equation of each layer involves two unknowns with two equations. The layer values are calculated with the activation function of the previous layer (A), the defined weight of the current layer (W), the defined bias of the current

layer (b), and the non-linear activation function (g). Equations 6, 7 and 8, pertain to the first layer [9]:

$$\mathbf{x} = \mathbf{a}^0 \quad (6)$$

$$\mathbf{Z}^1 = \mathbf{W}^1 * \mathbf{x} + \mathbf{b}^1 \quad (7)$$

$$\mathbf{A}_1^1 = \mathbf{Y} = \mathbf{g}^1(\mathbf{Z}^1) \quad (8)$$

The non-linear activation function, g , allows for greater flexibility, increased speed of learning and the creation of complex functions during the learning process. As the ReLu activation function is piece-wise linear function, the parameter definitions become easier, therefore it is often used for feature maps. Finally, the last element of a CNN is the pooling layers. A pooling layer is a layer which simplifies the output of a convolutional layer, takes each feature map output from the convolutions and prepares a condensed feature map [7].

2.2 Overview of Dataset Format and Compared CNNs

The performance of an object detection model is affected by many factors. These include factors such as the type of deep learning algorithm that extracted features, the input image resolutions, the boundary box encoding, data augmentation, the training dataset, the localization loss function and training configurations including batch size, input image size, learning rate and learning rate decay. Dataset format was an influencer for training accuracy. The method of how and what annotation information was saved resulted in a positive or negative variation in mAP. Two main dataset formats used in the object recognition algorithms are Microsoft COCO and Pascal Visual Object Classes (Pascal VOC). The key difference between the two formats was the type of file saved, which was a .json file for COCO and .xml for Pascal VOC. JSON files included info, licenses, categories, images and annotations. However, one JSON file contained all the images in the dataset. One XML file was created per image and included information such as folder, filename, size, object details per instance, truncated and difficult tags, and the bounding box details per object instance [10].

As object recognition task have evolved to model more complex, real life applications, the need for faster and more accurate CNNs has developed. VGGNet originally began by modifying the ground CNN architecture for simplicity, by deploying only 3X3 local receptive field convolutional layers. A max pooling layer was added to the reduce the volume size [11]. VGGNet with its simplicity goal, still followed the base CNN architecture, therefore the weight of the

overall deep learning model was large. A major development in the deep learning field arrived with exotic architectures such as Microsoft’s ResNets and Google’s GoogleNet/InceptionV1 to InceptionV3. ResNet-50, consists of 50 weight layers, is a deep convolutional neural network that has a unique feature called Residual Modules. These modules allowed the data learned in the previous modules to be applied to the current module, therefore the training time and error were reduced. The modules were added after each convolutional layer, which allows it to compile the information and reference that function for the following layers. The goal for ResNet was depth and was achieved by addressing the “vanishing gradient” error. As layers were added, the deep learning network started to converge upon training, this resulted in increased training error and a rapidly degrading accuracy; this was the vanishing gradient issue [12]. ResNet achieved up to 152-layer depth which is 8 times deeper than VGGNet, with a greater accuracy and 5 times lower weight [12]. InceptionV3 worked as a multi-level feature extractor. Its main feature was the method that each convolutional layer was organized to optimize speed and weight. The architecture used various sizes of convolution local receptive fields, such as 1X1, 3X3 and 5X5, to capture features in different scales, thus the accuracy of each convolutional layer is increased. InceptionV3 included three different substructures for feature extraction referred to as Inception Modules A, B and C. Figures 3, 4 and 5, visualize the breakdown of these multi-level feature extracting modules from the paper, *Rethinking the Inception Architecture for Computer Vision* [13].

Figure 3: InceptionV3 Inception Module A

Figure 4: InceptionV3 Inception Module B

Figure 5: InceptionV3 Inception Module C

Detection deep learning networks have two main goals: accuracy and speed. The base CNN structure has been modified according to those goals. Architectures such as Faster R-CNN and R-FCN used region-based feature extractors to accomplish this goal, whereas CNNs such as YOLOv3 combine the ResNet and Inception features and create hybrid networks. R-CNN was a two-stage object detection network. The first stage was a deep FCN, which included the convolutional layers, and the second stage was a Region Proposal Network (RPN). The RPN was the marketed selling point of the Faster R-CNN and consisted of two sibling fully connected layers:

a box regression and a box classification layer. The RPN was put ahead of the convolutional feature map and utilized the feature output to predict object bounds and scores simultaneously. This module enabled the Faster R-CNN to increase overall accuracy from the older Fast R-CNN [14]. Like the Faster R-CNN, the R-FCN was also a two-stage architecture. It consisted of an FCN and an RPN. However, unlike the individual region evaluation method of the Faster R-CNN, the R-FCN was fully connected network with all the layers being convolutional. This means that the second stage, RPN, evaluated the entire region instead of multiple individual evaluations. This enabled the R-FCN to limit errors such as the “a trous trick” and increase the overall accuracy of the architecture [15]. YOLOv3 was the third version of the YOLO9000 object detection system, which aimed at achieving real-time object detection with good accuracy. The system achieved this by mixing the Darknet-19 network used in YOLOv2, and residual networks (ResNets). The convolutional layers were also varied in size like the InceptionV3/GoogleNet architecture with 1X1 and 3X3 size layers, which increased the accuracy and speed. DarkNet-53 has similar performance to a 152-layer deep residual network (ResNet-152 on Faster R-CNN) with twice the speed [16].

2.3 Current Commercial Application

Boeing and Airbus lead the front in the aerospace computer vision field. Boeing’s work dates back to 1999, when they created a video-based foreign object detector. This system was designed as an overhead camera that can automatically detect small pieces of debris that can go unnoticed by human vision [17]. This system was put into the Boeing wing production line of the F/A-18 C/D and E/F. This was a fault-detection inspection system during an assembly line. The system was cost-effective, reduced the need for inspection, decreased part rejections and delivery delays. This technology used objection detection and was built on foreign-object samples, work pieces and tooling to build positive and negative assumptions. Airbus is currently focusing on implementing CV algorithms in the air. Airbus has demonstrated its first fully automatic vision-based take-off system. On January 16th, 2020, Airbus utilized image recognition technology which it incorporated directly onto the aircraft. This achievement is part of the larger Autonomous Taxi, Take-Off and Landing (ATTOL) project [18]. Vision-based taxi and landing tasks are targeted to be completed by mid-2020. Image recognition has allowed for the effective take-off of an aircraft, which entails a lower workload for the pilot and a safer flight due to the elimination of human error. The aerospace industry has applied object recognition to assembly line fault detection and autonomous take-offs to eliminate the effect of human error.

A key manufacturing industry example is the company, NanoNets. This company is providing the model creating resources for manufacturing tasks such as AVI. NanoNets has created their own proprietary CNN to detect faults in the assembly line and defects during a visual inspection. With the promises of fast, accurate, and scalable object recognition algorithms, this company allows for individuals and businesses to create custom models for tasks such as product quality control, defect detection, and crop quality assessment. The use of the algorithm services is

available through APIs, docker, or mobile apps achieves the human factors considerations of accessibility. Nanonets explained that deep learning and CV algorithms application into visual inspection, allows for automatic inspections, lowers the cost of labour and removes human error of imprecision of eyesight with unlimited performance [19]. As deep learning algorithms are scalable, the performance continually increases with an increase in the amount of data available.

3.0 Algorithm Design

The design objectives, requirements and methodology are stated in this section. The focus for this research was to assess the viability of an object recognition algorithm for the implementation into aerospace manufacturing and maintenance tasks. The algorithms shall meet the stated requirements for successful integration. The designed algorithm should be able to efficiently identify the target object and display its results in an effective format. These objectives and requirements were integrated into the design methodology to create two types of object recognition models and algorithms. The design methods for both prediction-class and detection class are stated in this section.

3.1 Design Objective and Requirements

3.1.1 Design Objectives

The object recognition algorithms successfully recognize and display the target object and its name, when given an input image. For the designs to meet the goal the following objectives were tested and met:

OBJ-MAIN-PREDICTION-001: The prediction-class object recognition algorithm should be able to optically recognize the target object.

- Justification: The algorithm must be capable of recognizing the target component of the inverter system with an input picture.

OBJ-MAIN-PREDICTION-002: The prediction-class object recognition algorithm should be able to print the names and accuracy precision (AP) of the target object, with the highest percentage at the top.

- Justification: In order to inform the user what object is recognized, the algorithm must display its names and APs.

OBJ-MAIN-DETECTION-001: The detection-class object recognition algorithm should be able to optically recognize the target object.

- Justification: The algorithm must recognize the object to calculate location, bounding boxes and AP's with object detection model.

OBJ-MAIN-DETECTION-002: The detection-class object recognition algorithm should be able to output an image to with all target objects detected with bounding boxes, locations, names and AP's.

- Justification: In order to inform the user where the recognized object is, the algorithm must display bounding boxes, locations, names, and AP's of all objects detected from the input image.

3.1.2 Design Requirements and Verifications:

Requirements were defined to meet the objectives stated in the previous section. The requirements for each type of object recognition algorithm are as follows:

Table 1: Design Requirements - Prediction-class

Subcomponent	Description	Objective
Ground station	Laptop Computer	To process transfer-learning python code and run object prediction algorithm
Custom dataset	15 000 images of 13 classes of objects of the Maxflight V2002 Flight Simulator inverter system	To provide data to be transfer-learned for the creation of prediction model
Neural Network	InceptionV3 CNN	To provide feature extraction and transfer-learning capabilities
Object Prediction Model	Custom .h5 trained model	To identify target set of objects and display their names and AP's

Table 2: Design Requirements - Detection-class

Subcomponent	Description	Objective
Ground station	Dell G7 Laptop with Nvidia Geforce GTX 1050 GPU	To process transfer-learning python code and create the object detection model
Custom dataset	100 images of six class of objects from two different SUVs	To provide data to be transfer-learned for the creation of detection model
Neural Network	YOLOv3 CNN	To provide feature extraction and transfer-learning capabilities
Object Detection Model	Custom .h5 trained model	To identify target set of objects and display their names, bounding boxes, locations and AP's
Mobile Station	Samsung Galaxy Note 9 Smartphone	To run object detection algorithm with the custom model

To ensure the above objectives were met following verification procedures were created to test each subcomponent requirement:

Table 3: Design Requirement Verifications – Prediction-class

Test Category	Qualification	Testing Method
Model Testing	Obtain a mean accuracy precision (mAP) of at least 90%	AP analysis of model after creation. AP's are listed while experiments are being run.
Object Recognition Algorithm Testing	Algorithm must be able to recognize target object in image and list the AP's and names of predicted objects in order from highest to lowest AP	Trials with parts of the inverter system, and different inverters

Table 4: Design Requirement Verifications – Detection-class

Test Category	Qualification	Testing Method
Model Testing	Obtain a mean accuracy precision (mAP) of at least 85%	Separate AP analysis of model after creation, displaying IoU's, AP's and mAP's of each experiment
Object Recognition Algorithm Testing	Algorithm must be able to recognize all target objects in image and display the AP's, names, locations and bounding boxes of detected objects	Trials with images of various SUVs.

3.2 Methodology

The coding environment, libraries, design and testing processes are stated in this section. Both object recognition algorithms aim to show their viability for aerospace manufacturing and maintenance applications. Commercial applications of object recognition algorithms use proprietary neural networks design for the sole purpose of the intended object recognition task. However, this report focuses on the viability of an object recognition algorithm using pre-trained convolutional neural networks such as InceptionV3 and YOLOv3. The language Python was used with artificial intelligence-enabling libraries: ImageAI and Tensorflow. Both prediction-class and detection-class algorithms were trained with custom datasets and pre-trained CNNs, which were selected by a trade-off study. The functionality was verified, and the viability was confirmed with an accuracy and application test.

3.2.1 Ground Station Setup

The ground station used for both prediction-class and detection-class algorithms was a Dell G7 laptop with a Nvidia Geforce GTX 1050 GPU. The training, testing and object recognition codes were made on the language Python. All the codes utilized a deep learning library called “ImageAI”. This library allowed for the creation of the training, testing and application codes with its self-contained deep learning and computer vision capabilities. It can be used for object detection, video detection and object tracking, which are all essential for the successful implementation into the aerospace manufacturing and maintenance field. For the purpose of this thesis, the functions used were from the “ObjectPrediction” and “ObjectDetection” classes. Within those classes, functions were called to set the model path, set the input picture path, set output picture path, load the model, detect objects and print the objects name and AP, for the application codes. The training and testing codes utilized functions that set the model type, called the model, set the dataset path, set the training configuration values, trained the model and evaluated the model [20]. Enabling both the use of the ImageAI library and the training and testing codes was the Tensorflow library. This library, created by Google, enables fast numerical computing and was used to transfer-learning onto deep learning models. A major advantage with this library is that it can run on single CPU systems, GPUs, mobile devices and large scale distributed systems [21]. The code utilized tensorflow GPU version 1.13.1 for both prediction-class and detection-class. This enabled the training, testing and application codes to run at faster speeds.

The prediction-class training and application codes were all run from the Dell G7 laptop. The dataset was setup in folders and those paths were set using ImageAI functions. Once the pretrained InceptionV3 model, TensorFlow GPU version 1.13.1 and ImageAI version 2.1.5 were downloaded, the training code was run on the Nvidia GeForce GTX 1050 GPU. Once the model was trained, the object prediction code was run on an aviation inverter, the flight simulator’s inverter and wires. The full codes for training and application with explanations can be found in Appendix A.

The detection-class training, and testing codes were run from Google colab to display the viability of a cloud-based dataset and training system. A Google drive folder called “detection” was mounted onto the Google Colab notebook and the dataset of 100 images and annotations of two different SUVs were uploaded to it. Once the pretrained YOLOv3 model, TensorFlow GPU version 1.13.1, and ImageAI version 2.1.5 were downloaded, the trained and test codes were run using Google colab’s Nvidia K80 GPU. The completed model was then tested on Google Colab and then transferred to the mobile station for application testing. The detection-class code was tested on a Samsung Note 9 smartphone using a mobile python 3 IDE called “Pydriod 3”. This program allowed for the effective functioning of the object detection code. The full codes for training, testing and application with explanations can be found in Appendix B.

3.2.2 Dataset Collection and Preparation

Dataset collection and preparation was a key factor in the effectiveness and efficiency of the transfer-learning step. The larger and more diverse the data set collected, the better the reliability of the mAP will be. If the model is trained from the repetition of the same image, the algorithm will only be able to recognize that specific object in that specific attitude. For the prediction-class algorithm, due to a scarcity of aerospace components and the quantity of images needed for good accuracy (minimum 500 per class of object) [20], only one system was selected to for training. The inverter system of the MaxFlight V2002 Flight Simulator was selected for its relation to aerospace inverter systems. This acted as a proof of concept for the implementation into aerospace manufacturing and maintenance systems. The dataset collected contained 15 552 images from different angles and lighting scenarios. Dataset preparation followed a train path and test path with the images divided approximately 75% and 25%, respectively. Under each path, the images were broken down into their respective object class folders (i.e. inverter, analog, digital, etc.). The path flow of the dataset preparation is shown in the figure 6.

Figure 6: Dataset Preparation - Prediction-class

The dataset collection and preparation for the detection-class algorithm was more complex as the training format required images and corresponding annotated xml files. The dataset was comprised of 100 images and annotations of two types of SUVs: a 2016 Hyundai Santa Fe Sport and a 2015 Lexus RX 350. Two dataset formats were compared for the dataset preparation. The main objective of the dataset format selection was to test the accuracy. The formats compared were the Pascal Visual Object Classes (VOC) and Microsoft Common Objects in Context (COCO) formats.

Table 5: Dataset Format Comparison - Detection-class

	Microsoft COCO	PASCAL VOC
mAP (%) [22]	21.6	73.4

As seen in the table 5, PASCAL VOC is 3.4 times more accurate for object recognition tasks. The PASCAL VOC dataset preparation format and path flow is shown in figure 7.

Figure 7: Dataset Preparation - Detection-class

3.2.3 Transfer-Learning

Transfer-learning is the process of integration of a new related dataset to an older dataset to increase accuracy and speed [23]. The more accurate the feature extraction, the higher the mean accuracy precision of the new model. There are three main parts of transfer-learning: the input, the neural network and the output. The inputs are custom datasets and pretrained models. The neural networks used for prediction-class and detection-class are InceptionV3 and YOLOv3 CNNs, respectively. The outputs are two new models.

Two neural networks were compared for prediction-class and three neural networks were compared for detection-class. As the detection-class model was intended for live feed object detection, the main requirement for the model was speed. The algorithms were listed as successful

if the results were properly displayed and accurate; and the mAPs of the results were above the required mark of 90% and 85%, respectively.

For the prediction-class model, the compared CNNs were ResNet-50 and InceptionV3. Both these CNNs have many feature extracting convolutional layers and have high object recognition accuracies. The two main features compared between the ResNet-50 and InceptionV3 architectures are: CNN weight and accuracy on the ImageNet dataset. This data is from the ImageNet project in which the dataset is comprised of 1.2 million images belonging to 1000 different classes (Reference from cv-tricks.com). The comparison between the CNNs are listed in table 6, with the winning values italicized:

Table 6: CNN Comparison - Prediction-class [10][12]

	ResNet-50	InceptionV3
Weight (MB)	105	<i>96</i>
Accuracy (%)	93.0	<i>94.4</i>

For the same dataset, InceptionV3 outperformed the ResNet-50 in both weight and accuracy. Therefore, the prediction-class model was trained using the InceptionV3 pretrained CNN.

For detection-class, the comparison focus was on mAP and mainly inference time, which is the time it takes for the model to predict the testing data. The neural networks compared for the detection-class model were Faster R-CNN (Region-based Convolutional Neural Network), R-FCN (Region-based Fully Connected Network), and YOLOv3. The compared neural networks have all been trained with the Microsoft COCO object detection dataset format. The COCO format is not optimal for object detection, therefore the researched data displayed low APs for all the neural networks. The trade-off study between neural networks was compiled in table 7, with the winning values italicized:

Table 7: CNN Comparison - Detection-class [16]

	Faster R-CNN	R-FCN	YOLOv3
mAP (%)	<i>34.9</i>	29.9	33
Inference time (ms)	>100	85	<i>51</i>

Prediction-class and detection-class transfer-learning was a two-step process with dataset collection and preparation, and the model creation (via transfer-learning), as the second step. The training algorithm was created using the ImageAI manual, which stated all the functions and their options. Figure 9 visualizes the flow of the transfer learning and the InceptionV3 general architecture:

Figure 8: Transfer-learning Process - Prediction-class

Figure 9 explains the flow of the transfer learning and the YOLOv3's DarkNet-53 based feature extractor:

Figure 9: Transfer-learning Process - Detection-class

3.2.4 Accuracy and Application testing

During the training process, the AP value for each experiment was listed. Once the highest AP was reached, the training process was stopped to ensure overfitting does not occur. The AP was then verified with the satisfactory mark of 90%. The model was then tested with the object prediction code a generic aviation inverter, and parts of the MaxFlight V2002 Flight Simulator inverter system. The AP results were recorded in a table.

For the detection-class code, the model was tested with a testing code that listed the mAP of each model created. The training code saved models for each experiment only if the loss and accuracy was better than the previous experiment. The model with that highest mAP was selected to be used for the application testing. The selected model and JSON dataset file were moved to the mobile station, and the station was setup for object recognition tasks. With Python 3, ImageAI and Tensorflow, input images were given to the object detection code and outputs were obtained. The code was tested with similar vehicles to prove accuracy. The output images and accuracies were recorded.

4.0 Results and Discussion

4.1 Prediction-class

Prediction-class training took 16 653 seconds (4 hours 37 minutes 33 minutes) to reach 100% accuracy (7 experiments). In contrast, The mAP of the model was 99.97%, The generic inverter image was found online [24]. The results, discussion and limitations were explained in this section. The prediction-class dataset was evaluated based on accuracy. The model and application testing revealed exceptional accuracy. Figures 10 and 11 explained the results from the model testing and application testing, respectively.

Figure 10: Model Testing - Prediction-class

Figure 11: Application testing - Prediction-class

The prediction-class algorithm achieved and surpassed its design requirement. Model testing showed an impressive maximum mAP of 99.97% at the seventh experiment. This proves that with a sizeable dataset, the ability for the InceptionV3 CNN and the prediction-class algorithm to complete an image classification task on aerospace-related components and subcomponents. A key result was the identification of the Maxflight's inverter system's various wires. Such as small component, once colour-coded, was identified by the prediction-class algorithm with almost 100% accuracy. The accuracy of the generic aviation inverter was 99.56%, as shown in Figure 11, which was also higher than the satisfactory mark of 90%. From the aspect of accuracy, the prediction-class proves that it can be utilized for assembly, manufacturing and maintenance applications.

The results showed exceptional accuracy, but there were limitations to its feasibility. These values were exceptionally high for object recognition algorithms and are an outlier when compared to literature review. AP's of 99% are unreliable and unrealistic because the model was created with an undersized and under-diversified dataset. The model was trained using only one inverter system. In reality, this algorithm would not be able to recognize the sub-components of any other inverter system. The issue was related to the testing part of the dataset. As the testing images were of the same system, the model achieved exemplary results. Even with numerous pictures from various angles and lighting conditions, the fact was that this model can very accurately recognize only this inverter system. As exhibited in figure 11, the accuracy slightly decreased with the generic aviation inverter. If the application testing was done on a generic aviation wire, the code would not be able to recognize it successfully. To mediate this error, the dataset would need various images of components from a diverse collection of inverter systems, thus making the dataset large and diverse. The inaccessibility of multiple aviation components were

a major limitation for this thesis. The dataset collected was large but not diverse in component brand and type. Another key limitation was the image classification object recognition task. This task was only able to recognize the main object in an image. For the purposes of aerospace manufacturing and maintenance applications, this model would not suffice to remedy this issue the object detection algorithm was utilized. Both accuracy and feasibility were analyzed to show the viability for object recognition in the aerospace applications.

4.2 Detection-class

Detection-class training took 30 832 seconds (8 hours 33 minutes 52 seconds), 2 seconds per step for 160 steps per experiment, for a total of 100 experiments. The results, discussion and limitations were explained in this section. The images that were tested were taken from the internet [25]–[27], and are attached to the application results in appendix B . The detection-class results were recorded and plotted to show the relation of accuracy versus, object instances and number of experiments. These trends can be seen in figures 12 and 13. Furthermore, the application testing results were compiled into five figures (Figures 14-18), which showed the highest accuracy of each object class detection. The code results from the algorithm can be found in Appendix B.

Figure 12: Object Instances versus AP

Figure 13: Model Testing - Detection-class

Figure 14: Detection Results - Lexus RX 350

Figure 15: Detection Results - Lamborghini Urus

Figure 16: Detection Results - Tesla Model X

Figure 17: Detection Results - Hyundai Elantra

Figure 18: Detection Results - Mercedes G-Wagon

The detection-class model testing results showed the highest mAP of 89.54% at the thirteenth experiment as seen on figure 13. The mAP function showed a fifth-order polynomial trend with the peak achieved at 89.54%. Model testing showed that the detection-class algorithm using the YOLOv3 CNN achieved and surpassed the required mAP of 85%. The object instance comparison in figure 10, showed the general trend that higher the number of instances of a target object the higher the accuracy. Figure 10 also showed that the higher the complexity of the part the lower the accuracy. This complexity factor was seen when comparing the AP results between the side mirror (97%) and light (89%). As the headlights, rear lights and fog lights all had different shapes and sizes, whereas the side mirror was a distinct protruding rectangular figure. The application testing, shown by figures 14-18, display the strengths and weaknesses of the model. The strengths were seen in figures 14 and 15, with all categories detected on the target SUVs and at accuracies ranging from 94.3% to 99.87%. The raw code output and image detection results are in Appendix B. These validates the set requirements and proves the viability of the implementation of the detection-class algorithm in aerospace applications. Unlike prediction-class, this algorithm detects all the objects within the frame and displays object name and location. This location detection can be used for augmented reality-based assembly training and AVI.

This object recognition outputted the same results as the Nanonets' manufacturing object detection algorithm. The accuracy surpassed industry standard, however, the accuracy obtained is unrealistic due to the undersized and under-diversified dataset. Unlike Boeing's video-based foreign object detection algorithm, which only displayed the results in positive or negative colours, the algorithm designed successfully identified the target and displayed the location, name and AP. This shows that the designed algorithm incorporated better filter than the foreign object detection system. The results were also obtained from a mobile device which proved the accessibility of the algorithm. For the mobile station to be able to process the information quickly and effectively, the model had to be limited in size and detailed in features extracted. The aerospace industry has not applied object detection-enabled maintenance, therefore this implementation of a CNN-based object detection algorithm was unique.

The weakness and limitation of this detection-class algorithm was seen in figures 16, 17 and 18. Due to lack of data, the model detected multiple objects in one location. The lack of data refers to the low number of images used and diversity of objects. The dataset was comprised of images of two SUVs, this limits its ability to detect the components of other types of vehicles. A reliable dataset such as ImageNet includes millions of pictures for thousands of classes of objects. Such a dataset requires a lot of time to collect and format. Figure 16 and 18 showed that irregular SUV figures did not recognize well, as the Tesla Model X has different style doors than most SUVs, and the Mercedes G-Wagon has an unusual boxed figure. These features were not on the SUVs that were trained. A limitation from the overall aspect of the thesis was the inability to access a diverse dataset and manpower. The accessibility of a diverse dataset was affected by the novel coronavirus, COVID-19. The development of this virus forced citizens to stay indoors in quarantine. The lab and curricular resources were restricted causing a personal mental strain and

an inability to complete research tasks. This situation was handled as best as possible, and the datasets that were used for this thesis had to be modified. Large AI-focused companies such as Microsoft and Google have teams of employees working on dataset collection and preparation. This allowed for the creation of large and diverse datasets which then are trained into CNNs for highly reliable models. The detection-class algorithm utilized a hundred images and corresponding annotations. With a more manpower, and a larger, diverse dataset, this detection-class algorithm can be used in a variety of application for aerospace manufacturing, assembly and maintenance.

4.3 Degree of Requirements Matrix

Table 8: Degree of Requirements Matrix - Prediction-class

Requirement	Achieved	Comments
Model Testing	✓	The model achieved an accuracy of 99.97% mAP, which was well above the required 90%.
Object Prediction Algorithm Testing	✓	The algorithm successfully displayed name and AP of target objects.

Table 9: Degree of Requirements Matrix - Detection-class

Requirement	Achieved	Comments
Model Testing	✓	The model achieved an accuracy of 89.54% mAP, which was well above the required 85%
Object Detection Algorithm Testing	✓	The algorithm successfully displayed name, location and AP of all target objects

5.0 Conclusion

Object recognition in the aerospace industry is a relatively new area of study. The viability of object recognition algorithms is based on the ability for the algorithm to detect target components accurately and quickly. This report determined that the accuracy of, both the prediction-class and the detection-class, were viable for the task, with excellent mAPs of 99.97% and 89.54%, respectively. However, the prediction-class could not be used for aerospace tasks as it does not present object location and could only recognize the main target object in the image correctly. This is not the case for object detection, as the detection-class code displays locations, names and accuracies. Limitations for both algorithms were mainly due to lack of dataset size and diversity. Live-feed detection could not be completed for this report due to a sudden change in accessibility situation, however, research has shown that the YOLOv3 architecture can detect objects in live video with acceptable frame rates. This report proved the viability, however, with further research, the functionality of this algorithm can be increased to be commercially usable.

6.0 Future Considerations

Due to restrictions in time and manpower, this thesis was not able to explore the fields of total object recognition viability for aerospace manufacturing, maintenance and assembly task. For a complete viability test of object recognition algorithms, a proprietary CNN must be designed based on the physical properties of the target objects. Future research must consider evaluating the convolution's functions, such as the non-linear activation function, based on the dataset. This research will need to include a proposed network restructuring based on existing architectures. Following the idea of YOLOv3, a hybrid network should be the first building block of the new structure. Once the feature extraction concepts of residual networks and inception modules are brought in, the core CNN can be modified by setting requirements and objectives for parameters such as loss function and kernel/filter features. A proprietary CNN for aerospace manufacturing, maintenance and assembly tasks will be able to show its full viability, strengths and limitations.

References

- [1] “Computer Vision: What it is and why it matters | SAS.”
https://www.sas.com/en_us/insights/analytics/computer-vision.html#manufacturing
(accessed Apr. 12, 2020).
- [2] “Everything You Ever Wanted To Know About Computer Vision.”
<https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e> (accessed Apr. 12, 2020).
- [3] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge.” Accessed: Apr. 12, 2020. [Online]. Available: <http://image-net.org/challenges/LSVRC/>.
- [4] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553. Nature Publishing Group, pp. 436–444, May 27, 2015, doi: 10.1038/nature14539.
- [5] “What is Deep Learning?” <https://machinelearningmastery.com/what-is-deep-learning/>
(accessed Apr. 12, 2020).
- [6] “(444) Andrew Ng, Chief Scientist at Baidu - YouTube.”
<https://www.youtube.com/watch?v=O0VN0pGgBZM> (accessed Apr. 12, 2020).
- [7] M. Nielsen, *neural networks and deep learning*. .
- [8] “Gentle Dive into Math Behind Convolutional Neural Networks.”
<https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9> (accessed Apr. 11, 2020).
- [9] “Deep Dive into Math Behind Deep Networks - Towards Data Science.”
<https://towardsdatascience.com/https-medium-com-piotr-skalski92-deep-dive-into-deep-networks-math-17660bc376ba> (accessed Apr. 12, 2020).
- [10] T.-Y. Lin *et al.*, “Microsoft COCO: Common Objects in Context.”
- [11] “ImageNet: VGGNet, ResNet, Inception, and Xception with Keras - PyImageSearch.”
<https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/> (accessed Apr. 12, 2020).
- [12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition.” Accessed: Apr. 12, 2020. [Online]. Available: <http://image-net.org/challenges/LSVRC/2015/>.
- [13] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens, “Rethinking the Inception Architecture for Computer Vision.”
- [14] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” Accessed: Apr. 12, 2020. [Online]. Available: <http://image-net.org/challenges/LSVRC/2015/results>.
- [15] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object Detection via Region-based Fully Convolutional Networks.” Accessed: Apr. 12, 2020. [Online]. Available:

- <https://github.com/daijifeng001/r-fcn>.
- [16] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement." Accessed: Apr. 12, 2020. [Online]. Available: <https://pjreddie.com/yolo/>.
 - [17] "NASA Tech Briefs - Google Books."
[https://books.google.ca/books?id=6abNhcIGn6YC&pg=RA1-PA13&lpg=RA1-PA13&dq=boeing+object+recognition+algorithms&source=bl&ots=fc5wknUyKl&sig=ACfU3U0pRDg5PpsgPeppGmYTp7Psm8QKmQ&hl=en&sa=X&ved=2ahUKEwi3152C3uToAhUZIXIEHZ-NAIIQ6AEwDXoECA4QKQ#v=onepage&q=boeing object recognition algorithms&f=false](https://books.google.ca/books?id=6abNhcIGn6YC&pg=RA1-PA13&lpg=RA1-PA13&dq=boeing+object+recognition+algorithms&source=bl&ots=fc5wknUyKl&sig=ACfU3U0pRDg5PpsgPeppGmYTp7Psm8QKmQ&hl=en&sa=X&ved=2ahUKEwi3152C3uToAhUZIXIEHZ-NAIIQ6AEwDXoECA4QKQ#v=onepage&q=boeing%20object%20recognition%20algorithms&f=false) (accessed Apr. 13, 2020).
 - [18] "Airbus demonstrates first fully automatic vision-based take-off - Innovation - Airbus."
<https://www.airbus.com/newsroom/press-releases/en/2020/01/airbus-demonstrates-first-fully-automatic-visionbased-takeoff.html> (accessed Apr. 13, 2020).
 - [19] "Everything you need to know about Visual Inspection with AI."
<https://nanonets.com/blog/ai-visual-inspection/> (accessed Apr. 12, 2020).
 - [20] "ImageAI Documentation Release 2.1.5 'Moses Olafenwa' 'John Olafenwa,'" 2020.
 - [21] "Introduction to the Python Deep Learning Library TensorFlow."
<https://machinelearningmastery.com/introduction-python-deep-learning-library-tensorflow/> (accessed Apr. 12, 2020).
 - [22] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger." Accessed: Apr. 12, 2020. [Online]. Available: <http://pjreddie.com/yolo9000/>.
 - [23] Y. P. Lin and T. P. Jung, "Improving EEG-based emotion classification using conditional transfer learning," *Front. Hum. Neurosci.*, vol. 11, Jun. 2017, doi: 10.3389/fnhum.2017.00334.
 - [24] "Astronics Power Conversion Products." <https://www.helis.com/database/news/power-frequency-conversion/> (accessed Apr. 12, 2020).
 - [25] "Mercedes-AMG Unveils Limited-Edition G-Class Models | WardsAuto."
<https://www.wardsauto.com/technology/mercedes-unveils-limited-edition-g-class-models> (accessed Apr. 12, 2020).
 - [26] "Lamborghini Urus News and Reviews | Motor1.com."
<https://www.motor1.com/lamborghini/urus/> (accessed Apr. 12, 2020).
 - [27] "2016 Tesla Model X - New Tesla Model X Prices, Models, Trims, and Photos."
<https://www.motortrend.com/cars/tesla/model-x/2016/> (accessed Apr. 12, 2020).

Appendix A – Prediction class

Training Algorithm

```
from imageai.Prediction.Custom import ModelTraining

model_trainer = ModelTraining()
model_trainer.setModelTypeAsInceptionV3()
model_trainer.setDataDirectory(r"C:\Users\Abhinav\Desktop\simulator")
model_trainer.trainModel(num_objects=13, num_experiments=100, enhance_data=True,
batch_size=32, show_network_summary=True)
```

Training Algorithm Raw Results

```

GeForce GTX 1050, pci bus id: 0000:01:00.0, compute capability: 6.1)
372/372 [=====] - 4377s 12s/step - loss: 0.2501 - acc: 0.9245 - val_loss: 4.0776 - val_acc: 0.2810
Epoch 2/100
371/372 [=====>.] - ETA: 7s - loss: 0.0641 - acc: 0.9311
Epoch 00002: val_acc did not improve from 0.28097
372/372 [=====] - 3407s 9s/step - loss: 0.0641 - acc: 0.9331 - val_loss: 34.2178 - val_acc: 0.0569
Epoch 3/100
371/372 [=====>.] - ETA: 6s - loss: 0.0392 - acc: 0.9099
Epoch 00003: val_acc improved from 0.28097 to 0.79325, saving model to C:\Users\Abhinav\Desktop\simulator\models\model_ex-003_acc-0.793246.h5
372/372 [=====] - 3241s 9s/step - loss: 0.0392 - acc: 0.9389 - val_loss: 1.2377 - val_acc: 0.7932
Epoch 4/100
371/372 [=====>.] - ETA: 6s - loss: 0.0318 - acc: 0.9088
Epoch 00004: val_acc improved from 0.79325 to 0.95452, saving model to C:\Users\Abhinav\Desktop\simulator\models\model_ex-004_acc-0.954520.h5
372/372 [=====] - 3254s 9s/step - loss: 0.0317 - acc: 0.9088 - val_loss: 0.4015 - val_acc: 0.9545
Epoch 5/100
371/372 [=====>.] - ETA: 6s - loss: 0.0242 - acc: 0.9333
Epoch 00005: val_acc did not improve from 0.95452
372/372 [=====] - 2941s 8s/step - loss: 0.0241 - acc: 0.9333 - val_loss: 34.3898 - val_acc: 0.0037
Epoch 6/100
371/372 [=====>.] - ETA: 4s - loss: 0.0445 - acc: 0.9877
Epoch 00006: val_acc did not improve from 0.95452
372/372 [=====] - 2287s 6s/step - loss: 0.0444 - acc: 0.9877 - val_loss: 0.1828 - val_acc: 0.8870
Epoch 7/100
371/372 [=====>.] - ETA: 4s - loss: 0.0017 - acc: 0.9997
Epoch 00007: val_acc improved from 0.95452 to 1.00000, saving model to C:\Users\Abhinav\Desktop\simulator\models\model_ex-007_acc-1.000000.h5
372/372 [=====] - 2242s 6s/step - loss: 0.0017 - acc: 0.9997 - val_loss: 9.5794e-04 - val_acc: 1.0000
Epoch 8/100
371/372 [=====>.] - ETA: 4s - loss: 0.0254 - acc: 0.9560
Epoch 00008: val_acc did not improve from 1.00000
372/372 [=====] - 2259s 6s/step - loss: 0.0253 - acc: 0.9950 - val_loss: 1.9921 - val_acc: 0.3409
Epoch 9/100
371/372 [=====>.] - ETA: 4s - loss: 0.0181 - acc: 0.9950
Epoch 00009: val_acc did not improve from 1.00000
372/372 [=====] - 2257s 6s/step - loss: 0.0181 - acc: 0.9950 - val_loss: 3.0593 - val_acc: 0.7213
Epoch 10/100
218/372 [=====] - ETA: 11:26 - loss: 0.0098 - acc: 0.9977

```

Figure 19: Training Algorithm Raw Results - Prediction-class

Application Algorithm

```
from imageai.Prediction.Custom import CustomImagePrediction
import os

execution_path = os.getcwd()

model_path = "./model/model_ex-007_acc-1.000000.h5"
json_path = "./json/model_class.json"
input_path = "./input/U_Red.jpg"

prediction = CustomImagePrediction()
prediction.setModelTypeAsInceptionV3()
prediction.setModelPath(model_path)
prediction.setJsonPath(json_path)
prediction.loadModel(num_objects=13)

predictions, probabilities = prediction.predictImage((input_path), result_count=5)

for eachPrediction, eachProbability in zip(predictions, probabilities):
    print(eachPrediction , " : " , eachProbability)
```

Application Algorithm Raw Results

```
Inverter : 99.55969452857971
Digital : 0.4192538559436798
W (Black) : 0.016646194853819907
L3 (Blue + Yellow - 240V) : 0.0029364322472247295
V (Blue) : 0.0013560384104494005
```

Figure 20: Aviation Inverter Prediction Results

```
Inverter : 99.99855756759644
Digital : 0.0013745348951488268
L3 (Blue + Yellow - 240V) : 3.4598107845340564e-05
W (Black) : 3.451074803706433e-05
L1 (Black +Yellow - 240V) : 5.084605803062914e-06
```

Figure 21: MaxFlight Inverter Prediction Results

```
PE (Green) : 99.99881982803345
U (Red) : 0.000988748433883302
V (Blue) : 0.00016709589090169175
W (Black) : 1.1668059585190349e-05
PE (Green + Yellow - 240V) : 1.0269037886700971e-05
```

Figure 22: PE (Green) Wire Prediction Results

```
PE (Green + Yellow - 240V) : 99.99884366989136
L2 (Red + Yellow - 240V) : 0.001133296791522298
L1 (Black +Yellow - 240V) : 2.7205305741517805e-05
L3 (Blue + Yellow - 240V) : 1.5455185575774522e-06
BK1 (White + Yellow) : 3.198994680264633e-08
```

Figure 23: PE (Green+Yellow - 240V) Wire Prediction Results

```
U (Red) : 100.0  
L3 (Blue + Yellow - 240V) : 4.752492621662441e-07  
V (Blue) : 2.3156676576263635e-07  
L2 (Red + Yellow - 240V) : 1.6987564133152944e-08  
BK2 (White + Light Green) : 7.85697340521807e-09
```

Figure 24: U (Red) Wire Prediction Results

Appendix B – Detection-class

Training Algorithm

```
from imageai.Detection.Custom import DetectionModelTrainer

trainer = DetectionModelTrainer()
trainer.setModelTypeAsYOLOv3()
trainer.setDataDirectory(data_directory="/content/drive/My Drive/detection")
trainer.setTrainConfig(object_names_array=["SUV", "car", "Truck", "light", "side mirror", "car
seat", "door", "wheel"], batch_size=4, num_experiments=100,
train_from_pretrained_model="pretrained-yolov3.h5")
trainer.trainModel()
```

Modal Testing Algorithm

```
from imageai.Detection.Custom import DetectionModelTrainer

trainer = DetectionModelTrainer()
trainer.setModelTypeAsYOLOv3()
trainer.setDataDirectory(data_directory="/content/drive/My Drive/detection")
trainer.evaluateModel(model_path="/content/drive/My Drive/detection/models",
json_path="/content/drive/My Drive/detection/json/detection_config.json", iou_threshold=0.5,
object_threshold=0.3, nms_threshold=0.5)
```


Application Algorithm

```
1 from imageai.Detection.Custom import
  CustomObjectDetection
2
3 detector = CustomObjectDetection()
4 detector.setModelTypeAsYOLOv3()
5 detector.setModelPath("model/detection_model-
  ex-012--loss-0014.300.h5")
6 detector.setJsonPath("json/detection_config.
  json")
7 detector.loadModel()
8 detections = detector.
  detectObjectsFromImage(input_image="input_im
  age/Lexus.jpg", output_image_path="output/
  Lexus-detected1.jpg",
  minimum_percentage_probability=86)
9 for detection in detections:
10     print(detection["name"], " : ",
  detection["percentage_probability"], " : ",
  detection["box_points"])
11
```

Figure 25: Application Algorithm Code - Detection-class

Application Algorithm Raw Results

Figure 26: Lexus RX 350 Detection Results Raw

```

.
Instructions for updating:
Colocations handled automatically by placer.
SUV : 99.45387244224548 : [90, 43, 788, 469]
wheel : 88.0387544631958 : [305, 269, 474, 418]
car seat : 98.60110282897949 : [400, 130, 477, 174]
car seat : 91.45198464393616 : [480, 150, 572, 199]
side mirror : 92.47829914093018 : [480, 150, 572, 199]
wheel : 98.0442762374878 : [618, 278, 721, 388]

[Program finished]

```

SUV : 99.454

Figure 27: G-Wagon Detection Results Raw

```
.  
Instructions for updating:  
Colocations handled automatically by placer.  
light : 91.849684715271 : [147, 1049, 549, 1505]  
light : 95.07326483726501 : [574, 1103, 1100, 1821]  
light : 92.45089292526245 : [144, 1668, 654, 2262]  
wheel : 87.81088590621948 : [144, 1668, 654, 2262]  
  
[Program finished]
```

Figure 28: Hyundai Elantra Detection Results Raw


```

Instructions for updating:
Colocations handled automatically by placer.
SUV : 99.50832724571228 : [109, 150, 783, 440]
light : 98.60344529151917 : [409, 254, 640, 315]
car seat : 98.94342422485352 : [240, 159, 346, 209]
side mirror : 91.5611743927002 : [254, 184, 360, 233]
light : 88.48536014556885 : [115, 280, 202, 375]
wheel : 98.94772171974182 : [112, 281, 209, 374]
wheel : 97.31044173240662 : [348, 317, 475, 451]

[Program finished]

```

Figure 29: Tesla Model X Detection Results Raw

```

Instructions for updating:
Colocations handled automatically by placer.
light : 98.85995388031006 : [120, 165, 245, 205]
SUV : 99.82632994651794 : [6, 98, 491, 297]
door : 97.62719869613647 : [293, 118, 393, 241]
car seat : 92.0750379562378 : [221, 112, 275, 144]
car seat : 97.27779626846313 : [294, 130, 373, 150]
side mirror : 98.26381206512451 : [294, 130, 373, 150]
light : 93.69239211082458 : [41, 172, 61, 194]
wheel : 97.79163599014282 : [403, 176, 477, 268]

```

[Program finished]

Figure 30: Lambourghini Urus Detection Results Raw