

University of Southern Queensland  
Faculty of Health, Engineering and Sciences

# Machine Vision for Locomotive Control

A dissertation submitted by

**Anthony Wallin**

in fulfilment of the requirements of

**ENG4111 and 4112 Research Project**

towards the degree of

**Bachelor of Engineering (Honours) (Mechatronic)**

Submitted October 2021

# Abstract

Trains have existed since the 19<sup>th</sup> century. The locomotives and rolling stock have advanced greatly since the first iterations. The infrastructure that support these mighty machines has not kept up with the rapid rate of advancement. The purpose of this paper was to investigate a new method of automatic locomotive control using the latest in machine vision technology.

Automatic trains are very niche, used almost exclusively in light rail. The gap in automation between rail and other modes of transport steadily grows. Many things contribute to this, but a key factor is that the vast majority of infrastructure was build before the turn of the 21<sup>st</sup> century with little consideration for new technologies. The latest automation uses short range wireless communication and complex servers to operate the trains. This makes such methods troublesome to implement into cross country freight networks.

The latest advancements in artificial intelligence and machine vision have brought algorithms with complex neural networks at the core. Many industries have already begun implementing such algorithms. Most prominently in the transportation industry, the technology has been implemented into cars to create an 'autopilot' feature. Tesla's range of cars and soon trucks is the most successful implementation to date.

The algorithm that was tested for viability was YOLOv2. Six variations were created, tested then compared against each other. Data collected from a train driving simulator was used as training data and testing data. MATLAB was used to build and test these algorithms.

Results showed that the smaller algorithm that was created was superior in all cases. The results also proved that YOLOv2 based algorithms can identify rail signals. This has opened many paths for future work, from neural network and algorithm optimisation to taking the work done here and applying it to real locomotives.

## ENG4111/ENG4112 Research Project

### **Limitations of Use**

The Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

University of Southern Queensland  
Faculty of Health, Engineering and Sciences

## ENG4111/ENG4112 Research Project

### **Certification of Dissertation**

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Anthony Wallin



## Acknowledgements

I would like to acknowledge my supervisors, Tobias Low and Derek Long. Their experience and input provided the necessary guidance to complete this dissertation. Thanks go to my family for providing inspiration and helping to touch up the final form of this dissertation. My final thanks go to me, thank you for not quitting on yourself and making it to the end of this long journey.

# Table of Contents

Abstract.....	2
Acknowledgements .....	5
List of Figures.....	8
List of Figures.....	10
Introduction .....	11
1.1 Statement of Task.....	11
1.2 Objectives of work.....	12
1.3 Overview of Dissertation .....	12
Background .....	13
Literature Review .....	16
3.1 Failure to update infrastructure.....	16
3.2 Current Train Automation.....	17
3.3 Machine Vision and Neural Networks.....	19
3.3.1 Machine vision algorithms .....	21
3.3.2 Machine vision present in the rail industry.....	22
3.3.3 Machine vision in the transport industry .....	23
3.4 Review summary .....	24
Methodology .....	25
4.1 Research questions.....	25
4.2 Data collection .....	25
4.3 Machine Vision creation and testing overview .....	27
4.4 Neural Network Modifications and Algorithm training.....	28
4.5 Training data explanation .....	28
Results.....	30
5.1 Training results.....	30
5.2 Test results.....	33
5.3 F-scores and precision-recall curves.....	36
Analysis .....	47
6.1 Training Data Analysis.....	47
6.2 Testing results Analysis.....	47
6.3 The story told by the F-scores and precision-recall curves.....	48
Discussion.....	50
7.1 Results Discussion .....	50
7.2 Consequences of Implementation .....	50
Conclusions .....	52
8.1 Future work.....	52

8.2 Conclusion.....	53
References.....	54
Appendix A – Project Specification .....	55
Appendix B – Code.....	57
Neural Network creation.....	57
Algorithm training.....	57
Algorithm Testing.....	60
Creation of F-score.....	61
Appendix C – Raw data .....	63

# List of Figures

<i>Figure 1: The Salamanca, the first commercially successful locomotive (The Mechanic's Magazine, 1829) .....</i>	<i>11</i>
<i>Figure 2: Example of a block system with signals (The Railway Technical Website, 2019) .....</i>	<i>13</i>
<i>Figure 3: British lower-quadrant semaphore stop signal (absolute) with subsidiary arm (permissive) below (David Friel, 2005) .....</i>	<i>14</i>
<i>Figure 4: Searchlight Signal, Atlanta, Georgia, USA (Todd Sestero,2006) .....</i>	<i>15</i>
<i>Figure 5: Train 15X2 route from Central Station towards eastern entry of Roma Street Station, Platform 8 (Australian Transport Safety Bureau, 2017) .....</i>	<i>17</i>
<i>Figure 6: Grade of Automation Diagram (International Association of Public Transport,2012) .....</i>	<i>18</i>
<i>Figure 7: typical high-level architecture of a modern CBTC system (RailSystem,2021).....</i>	<i>19</i>
<i>Figure 8: abstract neural network (Shaw NP, 2020).....</i>	<i>20</i>
<i>Figure 9: R-CNN process diagram (Ross Girshick, 2014).....</i>	<i>21</i>
<i>Figure 10:YOLO process diagram (Joseph Redmon, 2016) .....</i>	<i>22</i>
<i>Figure 11: Searchlight signal showing an 'all clear' signal () .....</i>	<i>26</i>
<i>Figure 12: Screenshot of train simulator in action with Searchlight signals ahead.....</i>	<i>26</i>
<i>Figure 13: YOLO18 Day Complete Identification.....</i>	<i>33</i>
<i>Figure 14: YOLO18 Day Partial Identification .....</i>	<i>33</i>
<i>Figure 15: YOLO18 Day False Positive.....</i>	<i>34</i>
<i>Figure 16: YOLO18 Day No Pickup .....</i>	<i>34</i>
<i>Figure 17: Signal Detection - Daylight results.....</i>	<i>35</i>
<i>Figure 18: Signal Detection - Night results .....</i>	<i>35</i>
<i>Figure 19: Signal Detection - All Conditions results .....</i>	<i>36</i>
<i>Figure 20: Daylight trained YOLO18 tested against the Daylight data.....</i>	<i>36</i>
<i>Figure 21: Daylight trained YOLO50 tested against the Daylight data.....</i>	<i>37</i>
<i>Figure 22: Night trained YOLO18 tested against the Daylight data .....</i>	<i>37</i>
<i>Figure 23:Night trained YOLO50 tested against the Daylight data .....</i>	<i>38</i>
<i>Figure 24: All conditions trained YOLO18 tested against the Daylight data .....</i>	<i>38</i>
<i>Figure 25: All conditions trained YOLO50 tested against the Daylight data .....</i>	<i>39</i>
<i>Figure 26:Daylight trained YOLO18 tested against the Night data .....</i>	<i>40</i>



<i>Figure 27: Daylight trained YOLO50 tested against the Night data .....</i>	<i>40</i>
<i>Figure 28: Night trained YOLO18 tested against the Night data .....</i>	<i>41</i>
<i>Figure 29: Night trained YOLO50 tested against the Night data .....</i>	<i>41</i>
<i>Figure 30: All conditions trained YOLO18 tested against the Night data .....</i>	<i>42</i>
<i>Figure 31: All conditions trained YOLO50 tested against the Night data .....</i>	<i>42</i>
<i>Figure 32: Daylight trained YOLO18 tested against the All conditions data .....</i>	<i>43</i>
<i>Figure 33: Daylight trained YOLO50 tested against the All conditions data .....</i>	<i>44</i>
<i>Figure 34: Night trained YOLO18 tested against the All conditions data .....</i>	<i>44</i>
<i>Figure 35: Night trained YOLO50 tested against the All conditions data .....</i>	<i>45</i>
<i>Figure 36: All conditions trained YOLO18 tested against the All conditions data .....</i>	<i>45</i>
<i>Figure 37: All conditions trained YOLO50 tested against the All conditions data .....</i>	<i>46</i>

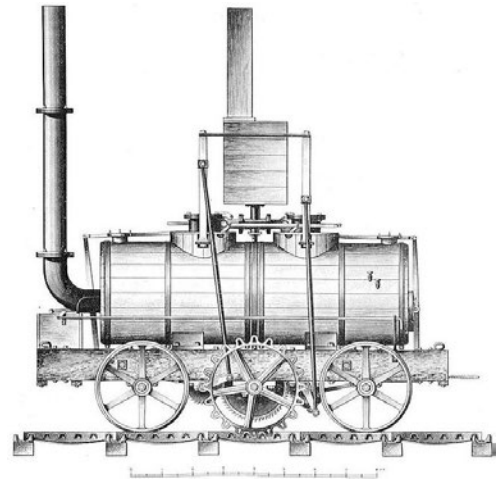
## List of Figures

<i>Table 1: Training data sets explanation.....</i>	<i>27</i>
<i>Table 2:YOLO18 Day Training Results.....</i>	<i>30</i>
<i>Table 3: YOLO18 Night Training Results .....</i>	<i>30</i>
<i>Table 4: YOLO18 All conditions Training Results .....</i>	<i>31</i>
<i>Table 5: YOLO50 Day Training results.....</i>	<i>31</i>
<i>Table 6: YOLO50 Night Training results.....</i>	<i>32</i>
<i>Table 7: YOLO50 All conditions Training results .....</i>	<i>32</i>
<i>Table 8: Daylight data F-score table.....</i>	<i>39</i>
<i>Table 9: Night data F-score table.....</i>	<i>43</i>
<i>Table 10: All conditions F-score table .....</i>	<i>46</i>
<i>Table 11: All Daylight trained YOLO18 F-scores.....</i>	<i>48</i>
<i>Table 12: Condensed F-score table .....</i>	<i>49</i>

# Chapter 1

## Introduction

Trains have existed since the 19<sup>th</sup> century. The first commercially successful locomotive, the Salamanca, was built by Mathew Murray in 1812, used to transport coal from Middleton to Leeds in England. Since then, trains have evolved with the advancements in technology. Electric power was implemented into locomotives during the 1880s, mainly using overhead wires to deliver the required power. Diesel locomotives would start to be prototyped at the turn of the 20<sup>th</sup> century but would not become what is now commonplace today till 1914 when Hermann Lemp developed the prototype diesel-electric locomotive control system.



*Figure 1: The Salamanca, the first commercially successful locomotive (The Mechanic's Magazine, 1829)*

The infrastructure for railroad industry has also evolved during this time. Different methods of signage and signalling, different rail gauges, new track layouts and interlocks. Compared to the advancement of the locomotives and rolling stock however, the implementation of advanced concepts into the existing infrastructure severely lagged. The automation of locomotive control was first implemented in the 1960s with the opening of the Victoria line in the London underground but has remained a comparatively niche technology. This is in stark contrast to other transportation sectors, planes and boats have had auto-pilot functionality for decades and cars are now being outfitted with auto-drive functionality.

### 1.1 Statement of Task

This project is to examine existing methods of railroad automation, find, and develop an early solution that bypasses the requirement of a large infrastructure overhaul using the latest in machine vision techniques.

## 1.2 Objectives of work

There are 5 core objectives this project is to complete:

- Explore the current state of train control and railway infrastructure.
- Highlight the unique challenges of railway automation.
- Review the advancements in machine vision.
- Investigate possible system/algorithms for implementation
- Develop a prototype algorithm that detects rail signals as a proof of concept.

## 1.3 Overview of Dissertation

This dissertation is organised as follows:

**Chapter 2** reviews the evolution of railroad infrastructure.

**Chapter 3** investigates key literature surrounding railroad automation and machine vision.

**Chapter 4** discusses the methodology undertaken in this dissertation.

**Chapter 5** is dedicated to the experimentation results.

**Chapter 6** analyses the results contained in **chapter 5**.

**Chapter 7** discusses the relevance of the results and the far-reaching consequences.

**Chapter 8** suggest further work and concludes the dissertation.

## Chapter 2

### Background

Early railway infrastructure was very simple, consisting of rails and mechanical switching points. This posed some challenges when running multiple trains on the same stretch of track. The solution of the time was to break the route down into sections of track between switching points. When a train conductor received the timetable for the train, not only did it have the expected times of arrival at subsequent stations but also the times of when the train 'controlled' each section of track on its route.

While this worked, accidents occurred that required adjustments to the method, such as extending the allotted time and having trains meet at the switch points. As rail networks grew, tracks soon had dedicated directions, one-way tracks. To allow for more utilisations of the track, it was broken down into smaller segments known as a 'block.' Initially, a man would stand at the entrance of each block with a stopwatch. This man was to measure the interval between trains and inform the following if it was too close to the previous train and had to decrease speed. While this did allow for more efficient use of the tracks, due to the lack of information beyond the watchmen, collisions did occur.

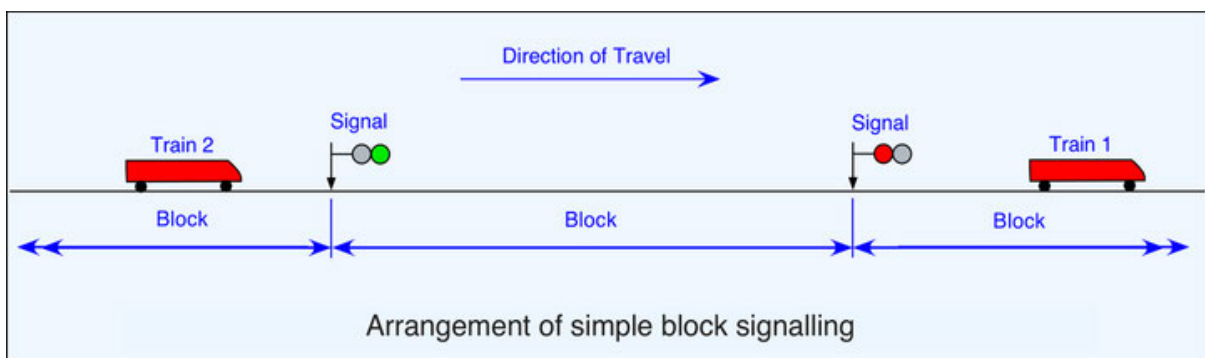


Figure 2: Example of a block system with signals (The Railway Technical Website, 2019)

With the invention of the telegraph in 1841, information could now, for the first time in history, move faster than a train. This alleviated some of the inflexibility of the original system by allowing changes to the schedule to occur while the trains were in operation. With the telegraph also came the next universal infrastructure upgrade, mechanical signals, and signal boxes.

With the ability to communicate nearly instantaneously, major junctions and blocks that occurred on main lines had signal boxes installed. These were buildings that housed the mechanical levers for the switch points and new mechanical signals. Signalmen would change the signal to display 'Danger' to protect a block after a train had entered the block. Signal boxes would communicate with others up and down the line, tracking a train's movements and informing them of when to change the mechanical signal as the block cleared. This flow of information allowed for more efficient and safer train movement and relegated the need to exclusively rely on a timetable to smaller branch lines that did not have many trains operating. Today, exclusive reliance on timetabling is unheard of and would only happen in the rarest of situations.



*Figure 3: British lower-quadrant semaphore stop signal (absolute) with subsidiary arm (permissive) below (David Friel, 2005)*

With blocks being standardised by the introduction of mechanical signals, specialised rules were developed to improve their efficiency. These rules have crystallised into two universal block types, absolute and permissive. An absolute block is the standard block type. It requires that on a 'danger' signal, the proceeding block is not to be entered. This rule can be broken only in certain circumstances such as a train breakdown with express permission from the signalman. A permissive block allows for a train to pass a 'danger' signal but at a significantly reduced speed. Most countries have restricted permissive blocks to freight trains.

Advancements in electrical understanding and lens making in the late 19<sup>th</sup> century gave way to the introduction of electric signals and automatic block systems. This eliminated the need for oil lamps and, more significantly, manned signal boxes. A basic automatic block detects a train using a track circuit.

Either a low DC voltage or very specific frequency AC voltage is applied to the rails at the end of a block. At the block entrance is a relay or other circuitry that connects the rails together. When the block is clear, the relay is energised. When a train is in the block, the circuit is shorted across the steel axles, deenergising the relay and causing a 'danger' signal to display at the entrance of the block. Automatic blocks are connected to allow for warning signals to be displayed multiple blocks before the obstructed block.

Signals have evolved greatly over time from flags and hand signals to oil lamps and mechanical signals to electric lights. Within the modern electric signals there is a myriad of different designs, from the most basic stop/go light to an array of lights that can give the drivers vastly more information. The Searchlight signal is one of the oldest designs still in operation today as its first implementation dates to the 1920s with the patent existing in the 1918. Originally it used a single light source and when the track condition changed, the lens would move, providing the new track signal. Modernised versions have replaced this setup with a single lens but using different light emitting diodes to provide the colour required. As standard the light is surrounded by a large black

dish, which gives the impression that the searchlight signal is but a modernisation on an even older design, the Hall enclosed disk signal. Searchlight signals have a tricolour code that informs the train. Red for Stop, there is danger ahead, Orange/yellow for proceed with caution and Green for proceed as normal. This design has faded in popularity and was replaced as the most popular signal type by the vertical colour light signal.



*Figure 4: Searchlight Signal, Atlanta, Georgia, USA (Todd Sestero,2006)*

# Chapter 3

## Literature Review

### 3.1 Failure to update infrastructure

Early 20<sup>th</sup> century steam trains began to push the boundaries of speed with several locomotive models exceeding 120km/h while fully laden, few even surpassed 180km/h. This rapid speed increase caused concern for regulators. Many deaths in the industry caused a wave of legislation to be enacted in the United States of America. These required the implementation of new technologies and equipment onto existing trains. The 1922 ruling by the Interstate Commerce Commission required the installation of 'automatic train control' systems. By today's standards, the implemented systems were less for control and more for train protection. Designed to display the previous signal in the cab via various methods, it provided as a reminder for the engineer or on faster trains the primary method of information as the wayside signage would pass by too quickly. Even this old system has not made it to every active railway line.

A more advanced system that automatically stops the train if a Signal Passed At Danger (SPAD) event occurs does not exist on most mass transit trains in Australia. A close call occurred in Brisbane on September 5<sup>th</sup> 2017 due to lacking an automatic stop function. The offending passenger train 1W33 was approaching Roma Street Station when it passed the stop indicator of Signal RS57. 1W33 passed through points 226 and entered platform 8. This put it on a collision course with oncoming passenger train 15X2 that had just left Central Station moments earlier. The SPAD alarm was raised at the Queensland Rail Management Centre. Emergency calls were made to both drivers and the trains stopped with only 550 metres separation. No damage to the rolling stock or injuries were reported. Infrastructure damage was caused as points 226 was not set for train 1W33 to pass through (Australian Transport Safety Bureau, 2017).





Figure 5: Train 15X2 route from Central Station towards eastern entry of Roma Street Station, Platform 8 (Australian Transport Safety Bureau, 2017)





If train 1W33 had passed Signal RS57 one minute earlier, there would have been a collision as another train, train 1K56, was at platform 8 of Roma Street Station exchanging passengers. This event caused serious delays across the entire network for hours as the points had to be repaired. Unfortunately, SPAD events are not uncommon and have been on the rise in 2021. Reported by the Brisbane times, SPAD events had doubled between August 2020 and March 2021. From 1.7 to 2.4 SPADs per million train kilometres (Felicity Caldwell, 2021). This needs to change.

### 3.2 Current Train Automation

There are two different types of train automation. First is Automatic Train Protection (ATP), the other is Automatic Train Operation (ATO). The previously mentioned SPAD automatic stop is one such example of a train-based ATP. An infrastructure example of an ATP is an interlock system that displays a 'danger' signal if the proceeding switch points are incorrectly positioned.

An ATO is the generally perceived idea when discussing vehicle automation. In the railroad industry there are multiple levels that define the degree of automation (or Grades of Automation, GoA) present. This rating system has 5 levels, GoA 0 to GoA 4. GoA 0 is a train with zero automatic assistance, the safety and reliability of the train rests on the operation team. Below is a graphic produced by the International Association of Public Transport to better illustrate the higher levels of automation in a metro situation.

# Grade of Automation

Grade of Automation	Type of train operation	Setting train in motion	Stopping train	Door closure	Operation in event of Disruption
GoA 1 	ATP with driver	Driver	Driver	Driver	Driver
GoA 2 	ATP and ATO with driver	Automatic	Automatic	Driver	Driver
GoA 3 	Driverless	Automatic	Automatic	Train attendant	Train attendant
GoA 4 	UTO	Automatic	Automatic	Automatic	Automatic

ATP - Automatic Train Protection      ATO - Automatic Train Operation

Figure 6: Grade of Automation Diagram (International Association of Public Transport, 2012)

The first instance of train control automation was the Victoria Line of the London Underground. Opened in 1969 by Queen Elizabeth II, to relieve the congestion on other lines of the London Underground network. This system was developed from an existing ATP called pulse code cab signalling that had been created in the 1920s by Union Switch and Signal for Pennsylvania Railroad (PRR).

It works by applying electrical pulses to the running rails that would be detected by a train via induction. Different pulse rates are used to convey information in the train cab. PRR used this technology for in cab signalling and for automatic train stop that had been required by the Interstate Commerce Commission for passenger transit.

The British engineers took this design and took it a step further by mostly eliminating the need of a driver. Instead of displaying the information for the driver to act on, the system would control the train directly. The only thing the driver needed to do was restart the train after it had reached the station. In 2012 the pulse code cab signal system was replaced by the more advanced Communication Based Train Control (CBTC) system that it operates today.

A CBTC system as defined by the Institute of Electrical and Electronic Engineers 1474 standard is a “continuous, automatic train control system utilising high-resolution train location determination, independent from track circuits; continuous, high capacity, bidirectional train-to-wayside data communications; and trainborne and wayside processors capable of implementing automatic train protection (ATP) functions, as well as automatic train operation (ATO) and automatic train supervision (ATS) functions”(Institute of Electrical and Electronic Engineers, 2004).

The Thales SelTrac™ G7 is one of the latest CBTC designs. Such a system has three major subsections: Operations, Wayside, and Trainborne. Operations is the human aspect as well as the central intelligence of the system. Wayside is the all the infrastructure required for a CBTC system to operate. Communication relays, GPS correction equipment, data lines, switch point controllers, and many other critical systems. The trainborne system receives information through the wayside and controls the train directly.

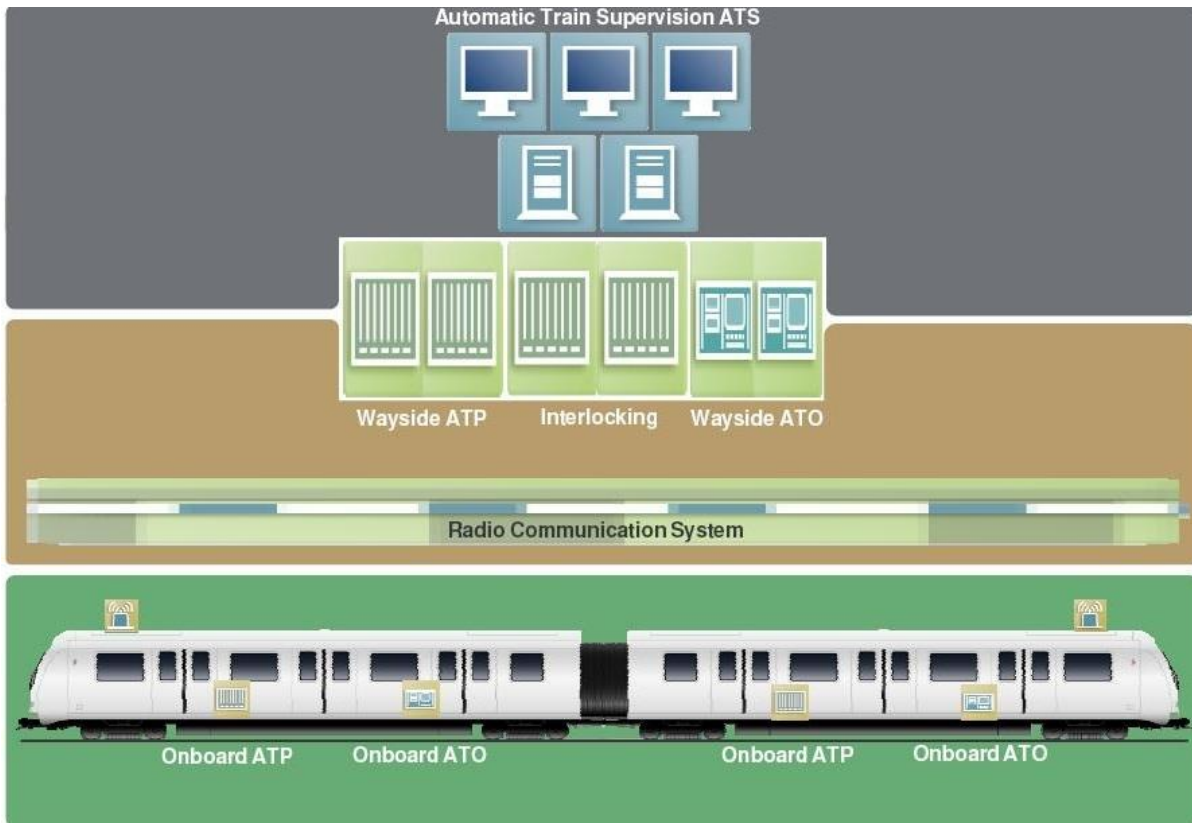


Figure 7: typical high-level architecture of a modern CBTC system (RailSystem,unknown)

The overwhelming majority of these systems exist only in metros and light rail. One of the few examples of heavy freight utilising such a system is the Autohaul system in Western Australia. This system was implemented by Hatachi Rail and Rio Tinto in the Pilbara region of Western Australia to facilitate the transportation of 360 million tonnes of iron ore annually.

### 3.3 Machine Vision and Neural Networks

Computer vision in a classical sense is about manipulating collected data through various techniques to achieve a very specialised outcome. A basic example is determining physical defects in a metal can. The can to be inspected is captured from side on. The image is then put through a series of filters that strip the colours away and inspects the perimeter pixels to confirm if the can walls aren't dented. This would happen multiple times a second as the can is rotated. This era of computer vision, while still useful in many circumstances, has stated to close.

The new era is far closer to artificial intelligence due to the many advancements that have been made. Machine learning algorithms is one such advancement. This technology can be seen in use across many IT application. The most prominent applications that everyone has experienced is targeted advertisement and the YouTube suggested videos. While specifics of these algorithms are hidden away to either protect intellectual property or to avoid possible scrutiny, these algorithms are very good at what they do. One method is to implement re-enforcement learning. This adjusts the internal weights and values of an algorithm as the algorithm is used. Raising the value of a factor when a video is watched for example.

This machine learning didn't influence many changes in computer vision but led to the new field of deep learning and this has had incredible effects in the short time that it has taken to apply it to computer vision.

Deep learning refers to the use of neural networks to accomplish a task. A neural network is a stack of interconnected nodes that interpret data by passing data from one layer of nodes to the next with weighting and thresholds determining what is to be passed along. This method of data interpretation was heavily inspired by the human brain, as it mimics the biological communication between neurons in the brain.

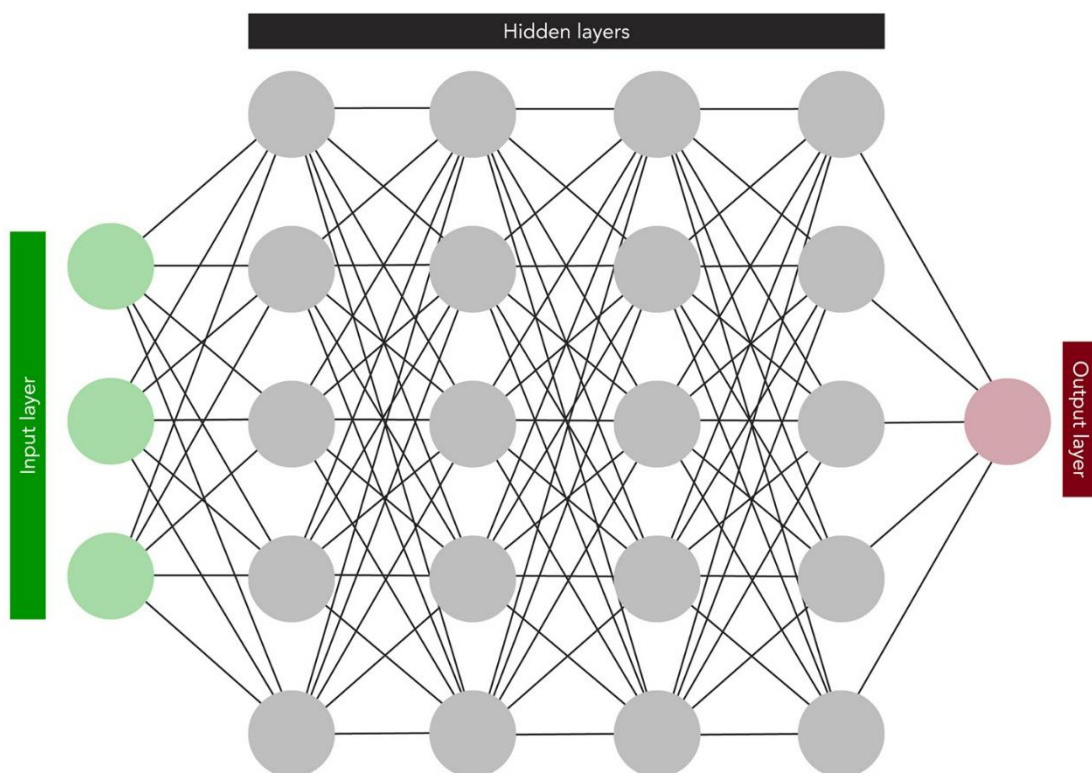


Figure 8: abstract neural network (University of Waterloo, 2020)

The neural network structure can be broken down into three distinct layers, input layer, hidden layers, output layer. The input and output layers are self-explanatory, they interface with the algorithm that uses the neural network. The hidden layers is where the work occurs as these define the complexity of the neural network. Both the number and the width of the hidden layers count toward the overall complexity.

Before a neural network can be implemented into any algorithm, not just machine vision, it must be trained. This is where the weights are applied to the paths that link the nodes and the thresholds that determine if the data is to be passed along. Training is commonly in the form of data that has been pre-processed either by a human or other simpler algorithms to contain the expected output of the neural network. Once trained, the neural network is ready to be meshed with final algorithm.

### 3.3.1 Machine vision algorithms

Many algorithms have been developed that use neural networks as the foundation. All Convolution Neural Networks (CNN) classify objects with some providing localisation in the form of bounding boxes for that object. Classification refers to determining ‘what’ the object is in the image. Localisation determines where the object is and is generally displayed with a bounding box. Two common veins of CNNs are the algorithms based on classifications and algorithms based on regression.

Region-based Convolutional Neural Network or RCNN is an example of the former. Developed by Ross Girshick et al. and presented in their 2014 paper titled: “Rich feature hierarchies for accurate object detection and semantic segmentation”. The algorithm is described to operate in four stages,

Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs (Girshick et al.,2014).

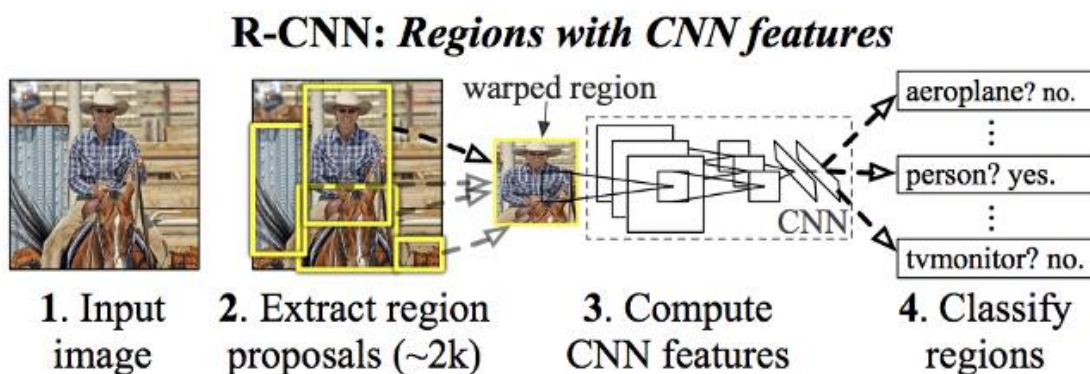


Figure 9: R-CNN process diagram (Ross Girshick, 2014)

RCNN provided not only a speed advantage compared to earlier object detection methods but also double the accuracy. Even with the speed increase, real time object detection was still infeasible for RCNN as images could still take multiple seconds to compute with the worst cases requiring 20 seconds. It also required some substantial computer hardware to run at these speeds. Variants of RCNN have been created to reduce the computing power required and generally speed up the process such as Fast-RCNN and Faster-RCNN.

Algorithms based on regression on the other hand, they skip finding ROIs and run the class prediction and bounding boxes for the entire image in a single run. This reduces processing time significantly and gives an incredible speed advantage but trades some accuracy for that speed. This makes it viable to use in real-time detection. You Only Look Once (YOLO) is one example of a algorithms based on regression. First introduced in 2016 by Joseph Redmon et al in their paper titled: "You Only Look Once: unified, Real-time Object Detection"(Redmon et al., 2016).

YOLO takes a fundamentally different approach to object detection. Instead of trying to speed up existing frameworks like Faster-RCNN, YOLO is designed from the ground up for speed. The presented algorithm worked as follows,

- (1) resizes the input image to  $448 \times 448$ ,
- (2) runs a single convolutional network on the image, and
- (3) thresholds the resulting detections by the model's confidence (Redmon et al., 2016).

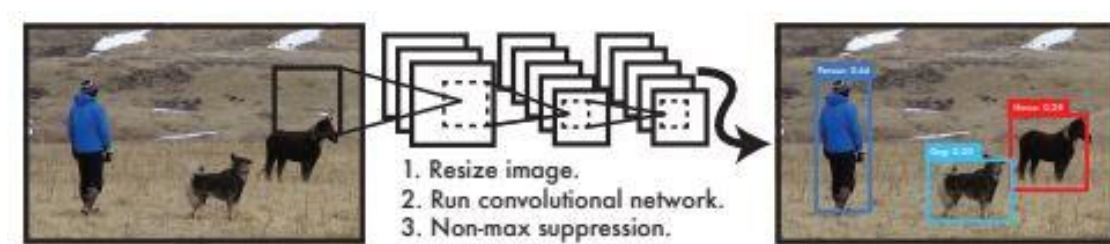


Figure 10:YOLO process diagram (Joseph Redmon, 2016)

YOLO has also received iterative changes by varies groups with YOLOv5 being the latest.

### 3.3.2 Machine vision present in the rail industry

Machine vision has already entered the rail industry but not into the direct control of the locomotives. The majority of implementations have been into the inspection and maintenance of the rolling stock and hardware. Transportation Technology Center Inc released a briefing on the research that had been conducted under the Association of American Railroad Strategic Research Initiatives Program into the application of machine vision. The brief published in 2014, explains and showcases the capability of the machine vision systems that are being developed. These systems allow for detection of minute details that an inspector might overlook or cannot see without magnification, increasing efficiency and safety by better informing train operators of when

maintenance is due. Many other companies have been developing similar systems as efficiency and safety are high priorities for train operating companies.

Pree et al (2012) investigated autonomous trains in open loop situations then built a prototype. The primary focus of the paper is the system architecture and the obstacle recognition that was implemented on the prototype. Unfortunately, this work has not been examined in its entirety as the full report is not available to the public. What has been seen of this document is the use of machine vision in the obstacle recognition system. While not the focus of this project, the inclusion of obstacle recognition is a logical next step.

### 3.3.3 Machine vision in the transport industry

In the broader transport industry, machine vision for autonomous vehicles is a hot topic. Around the turn of the century there were many papers produced exploring the growth of machine vision research. Dickmanns (2002) highlights the impressive developments that was made before the 21<sup>st</sup> century but it also shows a major problem that all engineers must contend with, insufficient technology. Many projects of this era were halted outright, or the scope was reduced due to insufficient computing power. Majority of the advancements in this era were made with the use of black and white CCD-cameras with relatively simple edge feature extraction. To handle more complex scenarios, it was estimated that computational capability would need to improve by at least 3 magnitudes as colour and texture processing was envisioned to be desirable. With the accurate prediction that 'Moore's Law' would continue, companies have returned to the topic of autonomous vehicles.

The most recent implementations of machine vision in autonomous vehicles have been coupled with other sensors such as LIDAR. A project undertaken in Japan by researchers at the Nagoya and Nagasaki universities explored the possibilities of an open-source solution to autonomous vehicles (Kato et al., 2015). In this investigation, machine vision was used to detect and track objects that were moving with the vehicle in the test environment. Not only could this system recognise the moving elements in the environment but also the stationary elements such as traffic lights and signals.

Tesla, one of the most famous vehicle manufacturing companies, has implemented machine vision to great success into their vehicles. Their website gives a sample of how complex a system can be. The Autopilot neural network that operates the vehicles autonomous mode is comprised of 48 different neural networks that required 70000 GPU hours to train. Many layers of additional algorithms and code exist beyond the neural networks to provide the decision making and vehicle control.

### 3.4 Review summary

Condensing the collected information, automatic train operation has been used since the 1960s and the latest methods of automation make rail operation safer and more efficient than ever. Implement of automation into existing infrastructure requires substantially more work than implementing it into new infrastructure, leaving many existing without the upgrade. This puts the general public at risk due to human error that can occur by the operators.

Machine vision has received a significant advancement with the application of neural networks. Many publicly available algorithms exist but are primarily based on either RCNN or YOLO architecture. Machine vision has not had much implementation to the rail industry but has seen success in the wider transport industry with companies such as Tesla.



# Chapter 4

## Methodology

### 4.1 Research questions

These are the guiding questions that have shaped the experimentation and evaluation:

- Can advanced machine vision techniques be used to detect Searchlight signals?
- If so, what algorithm provides the best results?
- How does using different neural networks effect the performance of the algorithms?

### 4.2 Data collection

Data collection is important but with attaching cameras to real trains unrealistic for this stage, a train driving simulator was used. Options in this genre of simulators is light with only one major developer producing the most content. Train Simulator 2022 by Dovetail games was selected out of experience and ease of access.

The route selected was the Southern Pacific railroad Donner Pass, specifically the track connecting the Roseville railyard and the Sparks railyard. The Donner pass is one of the most important sections of tracks as it has become a major arterial passage for the United States of America. Crossing the Sierra Nevada mountain range to link Sacramento, California to Spark, Nevada, providing a critical link in the First Transcontinental Railroad.

Construction began in late October of 1863 by the Central Pacific Railroad at Sacramento, California. The line would not be completed and opened until December 13<sup>th</sup>, 1867. Central pacific would only operate this line for 18 years before Southern Pacific in 1885 would acquire the right to permanently operate and maintain the line. A second line was built in the 1920s and would become known as "Track #2". This track reduced the grade of the pass to between 1.3% and 2.4%, a significant reduction from Track #1's grade average of 2.5%.

Today, this stretch of railroad is operated and maintained by Union Pacific Railroad who acquired this section in 1995 when the parent company, Union Pacific Corporation, acquired Southern Pacific Transport Company.

Donner Pass currently operates Searchlight signals in conjunction with trackside speed signs, and this is accurately modelled within the simulator. While the simulator is an excellent recreation of the Donner Pass, there are practical limitations that prohibit this method of prototyping new machine vision algorithms from being solely used. First is the game engine that this simulator uses. The original release of this game engine was late 2013. Below is a screenshot aboard a train waiting at a signal, beside it is a signal in the real world.



Figure 12: Screenshot of train simulator in action with Searchlight signals ahead



Figure 11: Searchlight signal showing an 'all clear' signal ()

Comparing figures 11 and 12, it can be seen that there is a sizeable difference between the two. Video game graphics have progressed substantially since the era that this simulator's engine was made. A newer simulator was released in 2020 called Train Sim World 2(TSW2) by the same developers. This simulator has more realistic graphics, but it comes at a significant trade off in useability that made it unsuitable for use in this project.

Second is that only a single camera can be replicated and there only select locations that it can be placed. In a physical system, there would be multiple cameras to cover all possible blind spots. The camera location during data collection was on the left-hand cab exterior of an EMD SD70M locomotive, set with a 4x zoom. Two hundred images were collected in daylight with an additional 250 captured across the transition from day to night. A single video was captured and edited to create a testing video for completed machine vision.

### 4.3 Machine Vision creation and testing overview

Two programs were considered for the development of the neural network-based algorithm. First was MATLAB, the other was OpenCV. MATLAB was quickly chosen over OpenCV due to the experience with the program and documentation available. This choice did cause the training and run time of the algorithm to take longer as MATLAB only supports Nvidia GPU acceleration, and the test rig comprised of an AMD 5600X CPU and RX580 GPU.

All images collected were manually marked in the MATLAB Image labeller to obtain the ground truth table for training and validation of the created machine vision algorithms. The different signal colours were separated into their own categories to simplify the training and validation. The following table shows the breakdown of the datasets that were used to train and validate.

*Table 1: Training data sets explanation*

Data set	Image count	Content
Daylight	200	Daylight
'Night'	250	Transitional lighting
All Conditions	550	All collected content

Two different algorithms were to be built and tested. One was a Faster-RCNN, the other was a YOLOv2. These were built with the neural network Resnet 50 as the core. For the training of all algorithms, the image data sets was randomly divided into 2 subsets, one for training and one for validation. Once trained, it was tested using the collected video with MATLAB code capturing the video, running the algorithm, and displaying the output in both a visual form and in matrix.

With the initial results, it was decided that continuing to work with the Faster-RCNN algorithm would require far more time, expertise and hardware to become operational. So, focus was shifted to the YOLOv2 algorithm. A second YOLOv2 algorithm was created with a simpler Resnet 18 neural network to compare how the different networks affected the algorithm output. Once trained, each algorithm variation was tested with 75 images, 50 daylight images, 25 night. From this point forward, YOLO18 and YOLO50 will be used to reference the different algorithms to avoid confusion.

YOLO18 and YOLO50 both had 3 variations, each was trained on a single dataset and compared against the other variations.

## 4.4 Neural Network Modifications and Algorithm training

Neural networks are very complex and require extensive knowledge to get the most out of them. This is one of the reasons a Faster-RCNN algorithm was abandoned for this project as it required significant modification to any neural network that was to be paired with it. For this reason, very limited modifications were done to both neural networks to make them compatible with the YOLOv2 algorithm. The modification is the meshing point between the neural networks and the algorithm. This involves selecting a node to attach the mesh point to along with setting up many particulars that are critical for the algorithm to work.

The mesh points:

Resnet18 – res4a\_relu

Resnet50 – activation\_40\_relu

The other critical components that were determined at this point of creation was image input size, the number of different objects, and the anchor box sizes. The MATLAB code to accomplish this is in the appendix.

Image resolution(pixels): 640 x 640

Number of objects: 3

Anchor box size(pixels): 32 x 32, 16 x 64, 64 x 16

The training also had a few settings that were standardised across the entire testing:

Training images per iteration: 10

Maximum passes through entire training dataset: 50

Learning rate: 0.001

## 4.5 Training data explanation

MATLAB outputs both a table and a simplified graphic that serve to explain the how the well the neural network has taken to the training. Key terms to understanding the following tables: Root Mean Square Error (RMSE), loss, epoch and mini-batch. Epoch is the number of passes through the entire data set. Mini-batch is a sample of the total data set used to iterate the algorithm that has a

defined size. RMSE describes the errors in the predictions. Loss is the combination of three errors, localisation, confidence and classification.

With these terms explained, it is expected that as the training progressed, the RMSE and loss values would trend toward zero with the difference between the mini-batch and validation being minimised. A training that matches this expectation shows that the algorithm has received worthwhile training.

# Chapter 5

## Results

### 5.1 Training results

Table 2: YOLO18 Day Training Results

Training on single CPU.

Initializing input data normalization.

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch RMSE	Validation RMSE	Mini-batch Loss	Validation Loss	Base Learning Rate
1	1	00:00:10	10.88	9.41	118.4259	88.5188	0.0010
5	50	00:04:01	1.03	0.98	1.0665	0.9619	0.0010
10	100	00:07:55	0.74	0.83	0.5530	0.6887	0.0010
14	150	00:11:48	0.57	0.62	0.3220	0.3784	0.0010
19	200	00:15:38	0.58	0.56	0.3410	0.3095	0.0010
23	250	00:19:28	0.46	0.57	0.2085	0.3247	0.0010
28	300	00:23:18	0.34	0.45	0.1152	0.2022	0.0010
32	350	00:27:07	0.45	0.35	0.2053	0.1230	0.0010
37	400	00:30:57	0.25	0.44	0.0643	0.1940	0.0010
41	450	00:34:46	0.29	0.41	0.0843	0.1722	0.0010
46	500	00:38:36	0.24	0.40	0.0568	0.1609	0.0010
50	550	00:42:25	0.21	0.39	0.0461	0.1499	0.0010

Detector training complete.

\*\*\*\*\*

Table 3: YOLO18 Night Training Results

Training on single CPU.

Initializing input data normalization.

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch RMSE	Validation RMSE	Mini-batch Loss	Validation Loss	Base Learning Rate
1	1	00:00:10	10.87	9.79	118.1912	95.7683	0.0010
4	50	00:04:13	0.77	0.85	0.5877	0.7297	0.0010
7	100	00:08:19	0.63	0.69	0.3912	0.4747	0.0010
10	150	00:12:22	0.56	0.63	0.3153	0.4009	0.0010
14	200	00:16:23	0.52	0.56	0.2670	0.3188	0.0010
17	250	00:20:22	0.40	0.45	0.1565	0.2017	0.0010
20	300	00:24:18	0.35	0.42	0.1243	0.1788	0.0010
24	350	00:28:14	0.50	0.60	0.2537	0.3580	0.0010
27	400	00:32:09	0.32	0.49	0.1003	0.2432	0.0010
30	450	00:36:06	0.29	0.55	0.0848	0.3039	0.0010
34	500	00:40:07	0.31	0.66	0.0989	0.4365	0.0010
37	550	00:44:08	0.16	0.53	0.0244	0.2846	0.0010
40	600	00:48:10	0.17	0.52	0.0293	0.2705	0.0010
44	650	00:52:13	0.20	0.42	0.0389	0.1783	0.0010
47	700	00:56:16	0.28	0.46	0.0792	0.2142	0.0010
50	750	01:00:19	0.14	0.53	0.0202	0.2774	0.0010

Detector training complete.

\*\*\*\*\*

Table 4: YOLO18 All conditions Training Results

```

Training on single CPU.
Initializing input data normalization.
=====
| Epoch | Iteration | Time Elapsed | Mini-batch | Validation | Mini-batch | Validation | Base Learning |
|       |          | (hh:mm:ss)  | RMSE       | RMSE       | Loss       | Loss       | Rate         |
=====
| 1 | 1 | 00:00:12 | 10.22 | 17.08 | 104.4209 | 291.7069 | 0.0010 |
| 2 | 50 | 00:03:59 | 1.15 | 1.17 | 1.3248 | 1.3650 | 0.0010 |
| 4 | 100 | 00:07:52 | 0.82 | 0.85 | 0.6749 | 0.7283 | 0.0010 |
| 6 | 150 | 00:11:44 | 0.80 | 0.75 | 0.6454 | 0.5591 | 0.0010 |
| 8 | 200 | 00:15:37 | 0.66 | 0.65 | 0.4351 | 0.4232 | 0.0010 |
| 10 | 250 | 00:19:31 | 0.49 | 0.56 | 0.2443 | 0.3149 | 0.0010 |
| 12 | 300 | 00:23:24 | 0.41 | 0.52 | 0.1660 | 0.2678 | 0.0010 |
| 14 | 350 | 00:27:17 | 0.42 | 0.46 | 0.1798 | 0.2093 | 0.0010 |
| 16 | 400 | 00:31:10 | 0.36 | 0.55 | 0.1312 | 0.2972 | 0.0010 |
| 18 | 450 | 00:35:03 | 0.32 | 0.45 | 0.1038 | 0.2032 | 0.0010 |
| 20 | 500 | 00:38:55 | 0.35 | 0.44 | 0.1258 | 0.1955 | 0.0010 |
| 22 | 550 | 00:42:47 | 0.27 | 0.43 | 0.0713 | 0.1807 | 0.0010 |
| 24 | 600 | 00:46:39 | 0.26 | 0.43 | 0.0699 | 0.1837 | 0.0010 |
| 25 | 650 | 00:50:30 | 0.32 | 0.42 | 0.0995 | 0.1768 | 0.0010 |
| 27 | 700 | 00:54:24 | 0.21 | 0.43 | 0.0437 | 0.1820 | 0.0010 |
| 29 | 750 | 00:58:18 | 0.21 | 0.38 | 0.0456 | 0.1475 | 0.0010 |
| 31 | 800 | 01:02:10 | 0.17 | 0.45 | 0.0304 | 0.2056 | 0.0010 |
| 33 | 850 | 01:06:03 | 0.21 | 0.47 | 0.0438 | 0.2214 | 0.0010 |
| 35 | 900 | 01:09:55 | 0.14 | 0.47 | 0.0200 | 0.2218 | 0.0010 |
| 37 | 950 | 01:13:49 | 0.15 | 0.45 | 0.0237 | 0.2060 | 0.0010 |
| 39 | 1000 | 01:17:41 | 0.15 | 0.45 | 0.0227 | 0.2033 | 0.0010 |
| 41 | 1050 | 01:21:34 | 0.17 | 0.44 | 0.0273 | 0.1927 | 0.0010 |
| 43 | 1100 | 01:25:27 | 0.14 | 0.42 | 0.0187 | 0.1771 | 0.0010 |
| 45 | 1150 | 01:29:23 | 0.19 | 0.38 | 0.0374 | 0.1444 | 0.0010 |
| 47 | 1200 | 01:33:16 | 0.13 | 0.42 | 0.0176 | 0.1764 | 0.0010 |
| 49 | 1250 | 01:37:09 | 0.58 | 0.59 | 0.3344 | 0.3463 | 0.0010 |
| 50 | 1300 | 01:41:04 | 0.37 | 0.40 | 0.1394 | 0.1627 | 0.0010 |
=====
Detector training complete.
*****

```

Table 5: YOLO50 Day Training results

```

Training on single CPU.
Initializing input data normalization.
=====
| Epoch | Iteration | Time Elapsed | Mini-batch | Validation | Mini-batch | Validation | Base Learning |
|       |          | (hh:mm:ss)  | RMSE       | RMSE       | Loss       | Loss       | Rate         |
=====
| 1 | 1 | 00:00:40 | 9.67 | 11.81 | 93.5687 | 139.4192 | 0.0010 |
| 5 | 50 | 00:17:52 | 1.12 | 1.22 | 1.2445 | 1.4779 | 0.0010 |
| 10 | 100 | 00:35:21 | 0.72 | 0.80 | 0.5116 | 0.6479 | 0.0010 |
| 14 | 150 | 00:52:57 | 0.61 | 0.68 | 0.3678 | 0.4563 | 0.0010 |
| 19 | 200 | 01:10:43 | 0.54 | 0.65 | 0.2895 | 0.4196 | 0.0010 |
| 23 | 250 | 01:29:08 | 0.33 | 0.59 | 0.1069 | 0.3437 | 0.0010 |
| 28 | 300 | 01:46:53 | 0.43 | 0.60 | 0.1855 | 0.3625 | 0.0010 |
| 32 | 350 | 02:04:05 | 0.35 | 0.61 | 0.1209 | 0.3733 | 0.0010 |
| 37 | 400 | 02:20:55 | 0.28 | 0.52 | 0.0773 | 0.2687 | 0.0010 |
| 41 | 450 | 02:52:33 | 0.34 | 0.58 | 0.1154 | 0.3355 | 0.0010 |
| 46 | 500 | 03:11:35 | 0.15 | 0.63 | 0.0231 | 0.3974 | 0.0010 |
| 50 | 550 | 03:29:32 | 0.14 | 0.64 | 0.0208 | 0.4143 | 0.0010 |
=====
Detector training complete.
*****

```

Table 6: YOLO50 Night Training results

Training on single CPU.

Initializing input data normalization.

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch RMSE	Validation RMSE	Mini-batch Loss	Validation Loss	Base Learning Rate
1	1	00:00:48	9.11	23.16	82.9501	536.5822	0.0010
4	50	00:18:38	1.10	1.50	1.2125	2.2395	0.0010
7	100	00:36:20	1.20	1.23	1.4381	1.5071	0.0010
10	150	00:54:14	0.95	0.91	0.9022	0.8328	0.0010
14	200	01:12:53	0.75	0.82	0.5578	0.6748	0.0010
17	250	01:31:13	0.65	0.70	0.4272	0.4916	0.0010
20	300	01:49:07	0.55	0.66	0.3069	0.4353	0.0010
24	350	02:07:08	0.56	0.63	0.3160	0.4017	0.0010
27	400	02:25:43	0.44	0.58	0.1919	0.3342	0.0010
30	450	02:43:39	0.44	0.61	0.1939	0.3708	0.0010
34	500	03:57:50	0.37	0.64	0.1343	0.4045	0.0010
37	550	04:17:15	0.36	0.59	0.1296	0.3463	0.0010
40	600	04:35:40	0.39	0.55	0.1492	0.3016	0.0010
44	650	04:54:41	0.28	0.57	0.0757	0.3276	0.0010
47	700	05:13:06	0.33	0.49	0.1071	0.2397	0.0010
50	750	05:31:44	0.29	0.51	0.0859	0.2564	0.0010

Detector training complete.

\*\*\*\*\*

Table 7: YOLO50 All conditions Training results

Training on single CPU.

Initializing input data normalization.

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch RMSE	Validation RMSE	Mini-batch Loss	Validation Loss	Base Learning Rate
1	1	00:01:01	9.60	33.37	92.1268	1113.6803	0.0010
2	50	00:18:09	1.44	1.39	2.0619	1.9344	0.0010
4	100	00:36:03	1.99	1.37	3.9558	1.8678	0.0010
6	150	00:54:35	1.37	1.41	1.8651	1.9840	0.0010
8	200	01:12:30	1.25	1.40	1.5750	1.9669	0.0010
10	250	01:30:24	1.20	1.28	1.4513	1.6363	0.0010
12	300	01:48:21	1.29	1.21	1.6712	1.4740	0.0010
14	350	02:06:11	1.18	1.25	1.3928	1.5675	0.0010
16	400	02:24:09	1.20	1.33	1.4502	1.7585	0.0010
18	450	02:42:02	1.28	1.28	1.6502	1.6305	0.0010
20	500	02:59:46	1.37	1.20	1.8808	1.4312	0.0010
22	550	03:17:14	1.39	1.21	1.9295	1.4644	0.0010
24	600	03:34:43	1.36	1.17	1.8574	1.3755	0.0010
25	650	03:52:09	1.35	1.15	1.8166	1.3136	0.0010
27	700	04:09:38	1.04	1.12	1.0835	1.2628	0.0010
29	750	04:27:18	1.03	1.09	1.0565	1.1941	0.0010
31	800	04:45:02	1.24	1.07	1.5364	1.1485	0.0010
33	850	05:02:32	1.11	1.06	1.2335	1.1333	0.0010
35	900	05:20:08	1.04	1.06	1.0782	1.1180	0.0010
37	950	05:37:33	0.93	1.00	0.8726	1.0048	0.0010
39	1000	05:55:01	1.00	0.99	1.0072	0.9782	0.0010
41	1050	06:12:29	1.19	1.01	1.4211	1.0218	0.0010
43	1100	06:30:03	0.81	0.91	0.6506	0.8338	0.0010
45	1150	06:47:33	1.00	0.92	1.0008	0.8432	0.0010
47	1200	07:05:01	0.83	0.95	0.6893	0.9031	0.0010
49	1250	07:22:29	1.08	1.09	1.1601	1.1906	0.0010
50	1300	07:39:52	1.01	1.01	1.0160	1.0164	0.0010

Detector training complete.

\*\*\*\*\*



## 5.2 Test results

The output of each pass was categorised into 4 different results: Complete identification, partial identification, False positive and no pickup.



Figure 13: YOLO18 Day Complete Identification



Figure 14: YOLO18 Day Partial Identification



Figure 15: YOLO18 Day False Positive



Figure 16: YOLO18 Day No Pickup

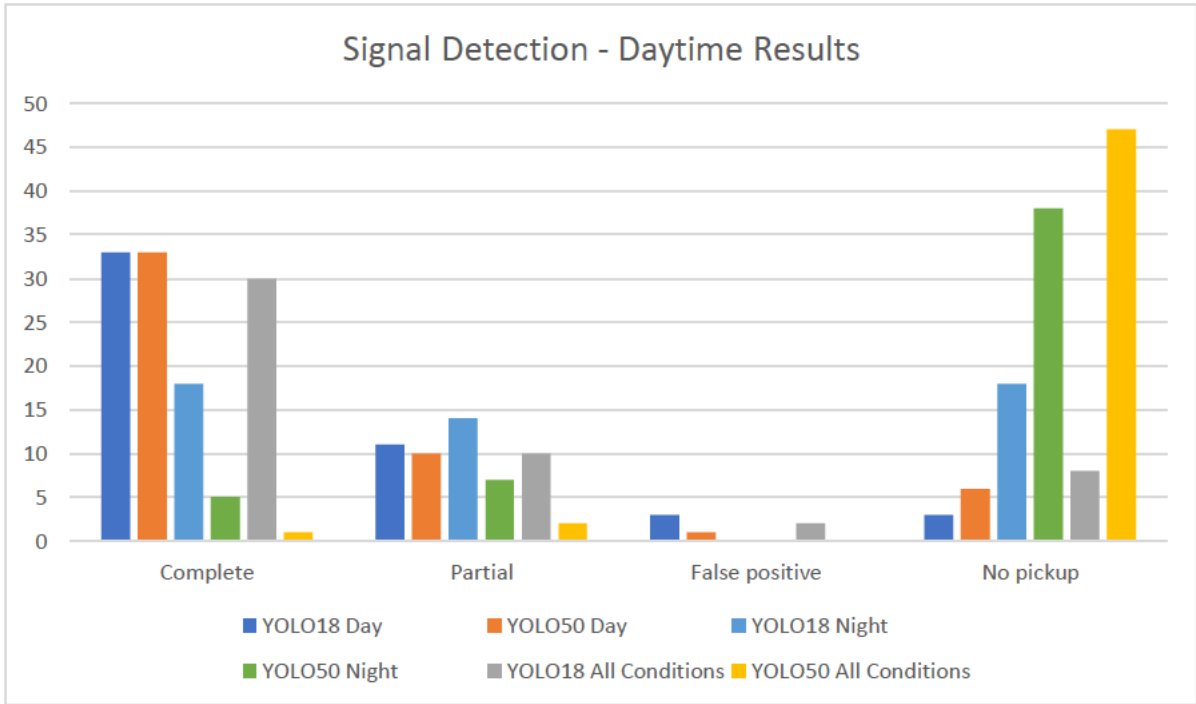


Figure 17: Signal Detection - Daylight results

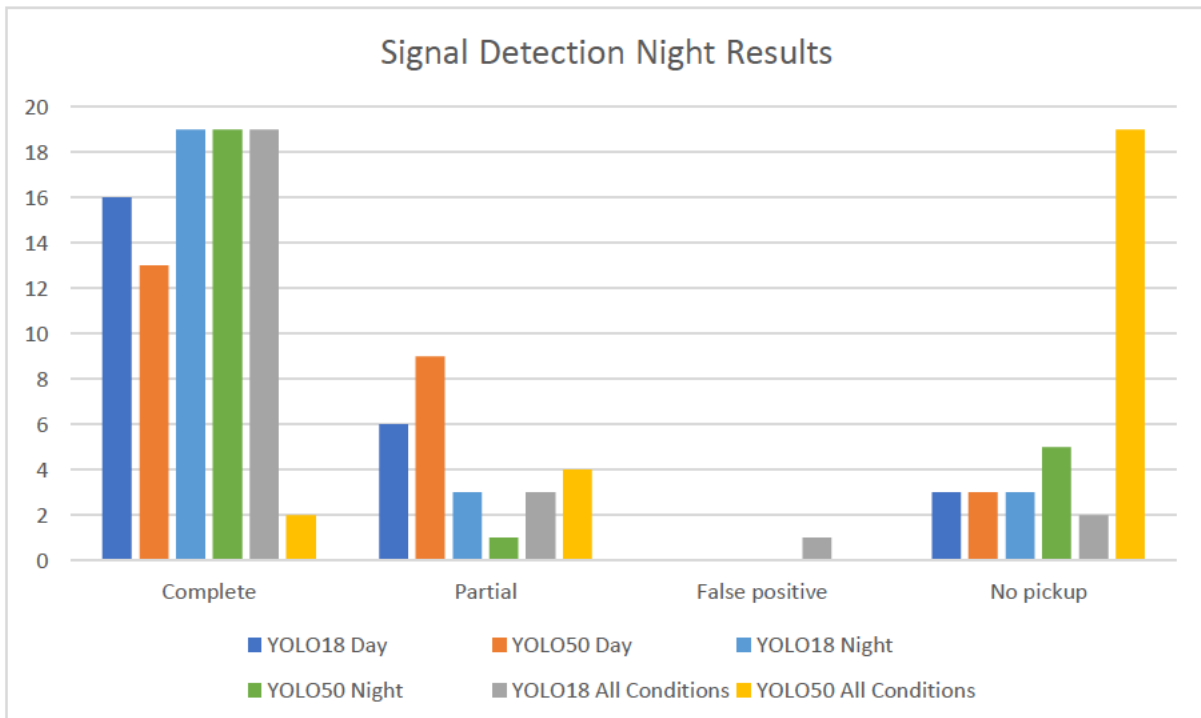


Figure 18: Signal Detection - Night results

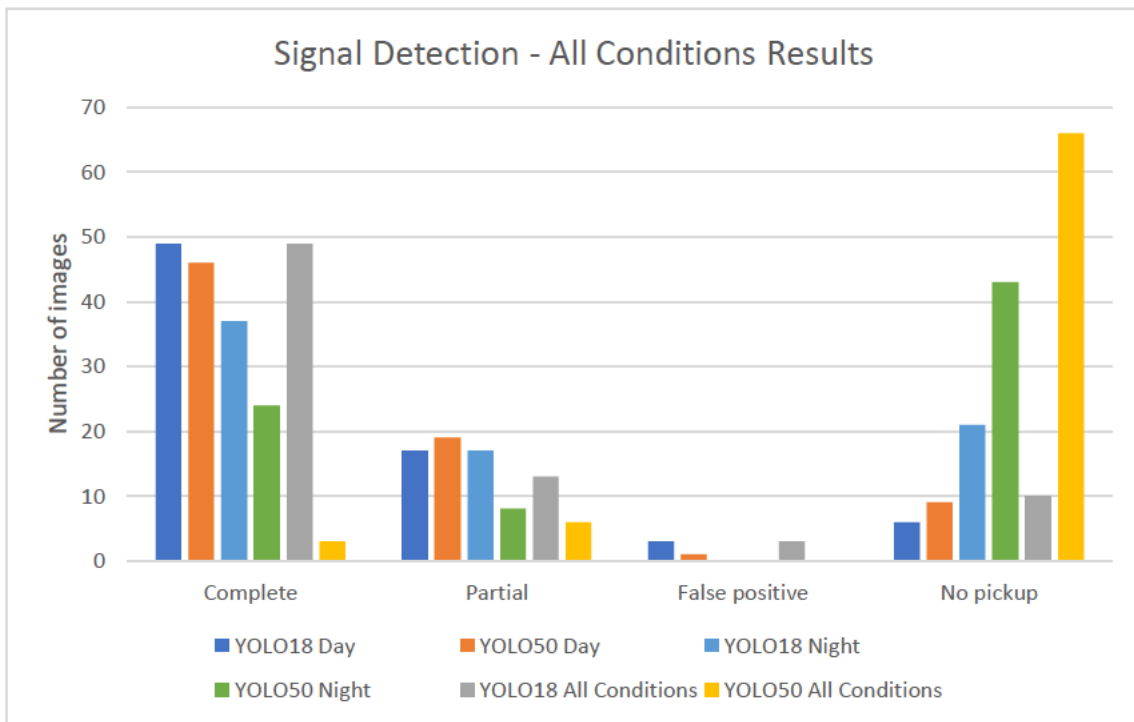


Figure 19: Signal Detection - All Conditions results

### 5.3 F-scores and precision-recall curves

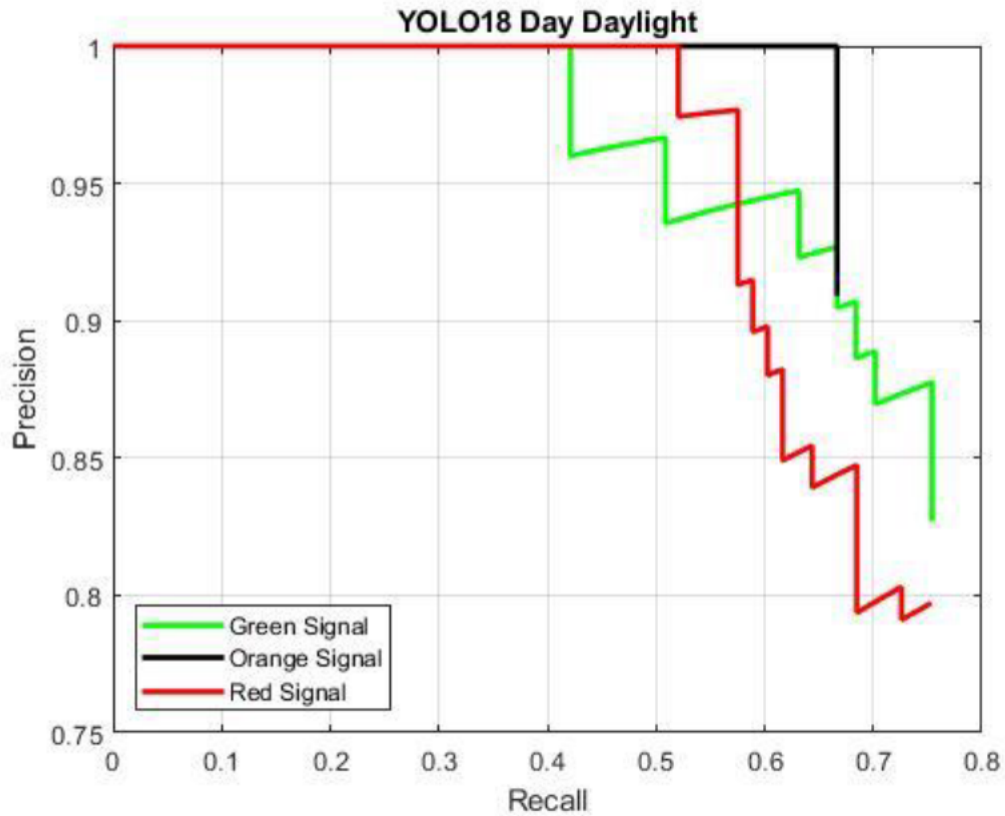


Figure 20: Daylight trained YOLO18 tested against the Daylight data

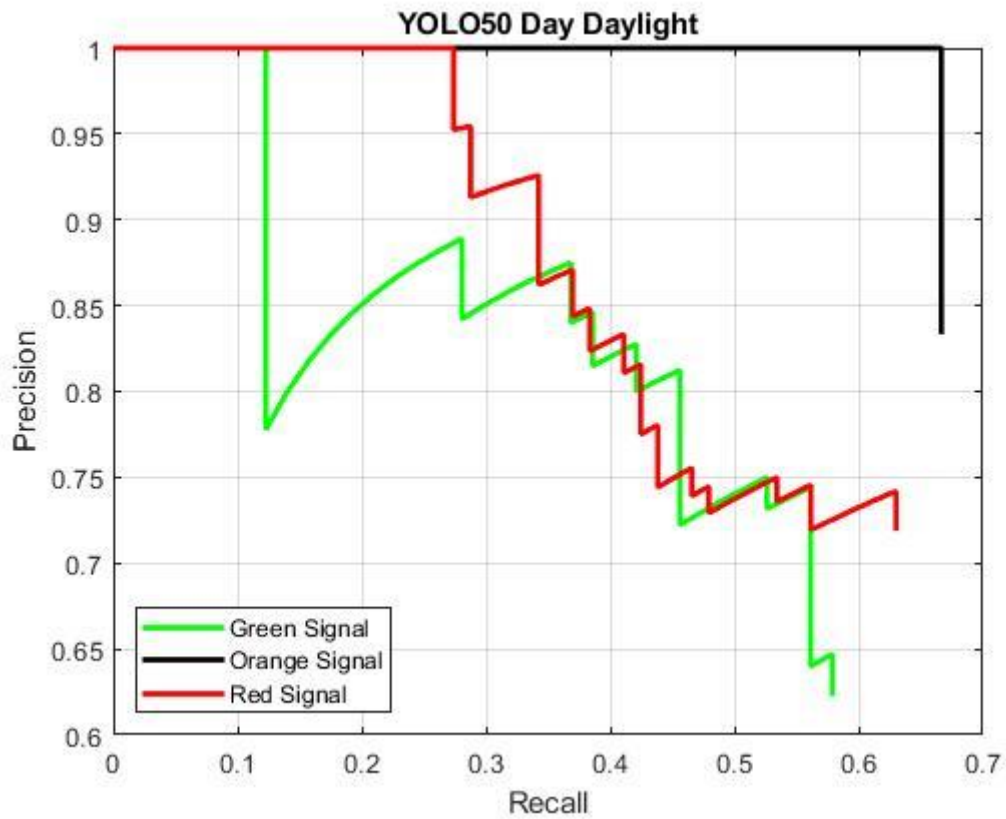


Figure 21: Daylight trained YOLO50 tested against the Daylight data

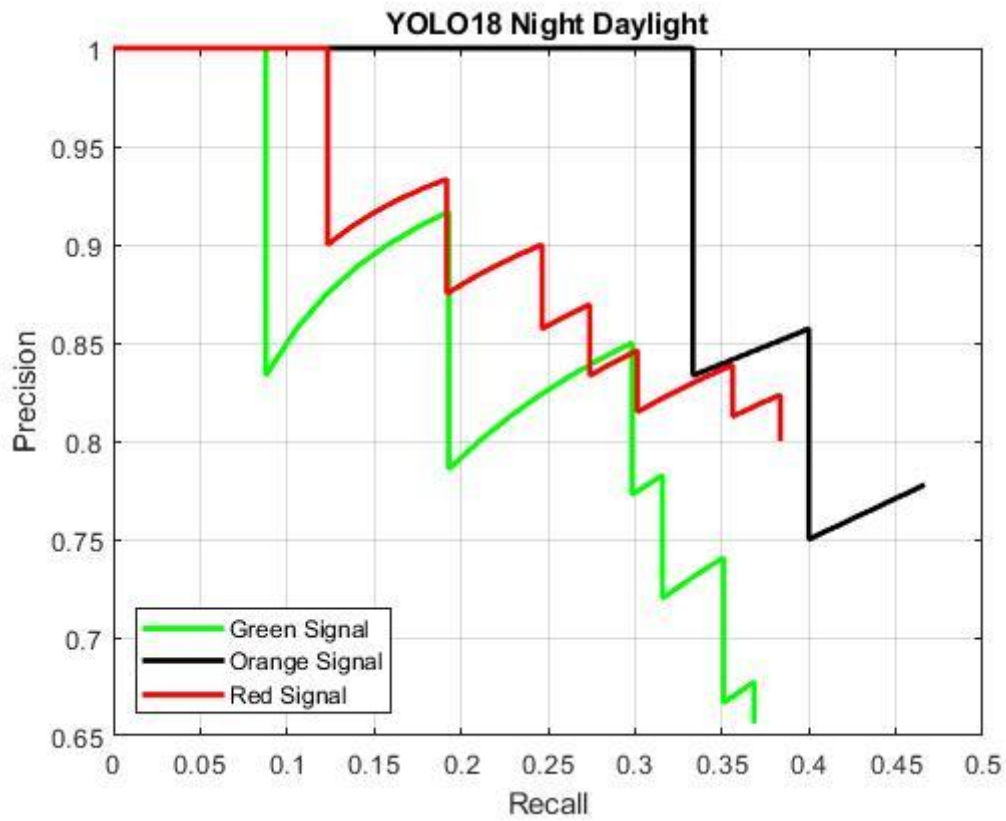


Figure 22: Night trained YOLO18 tested against the Daylight data

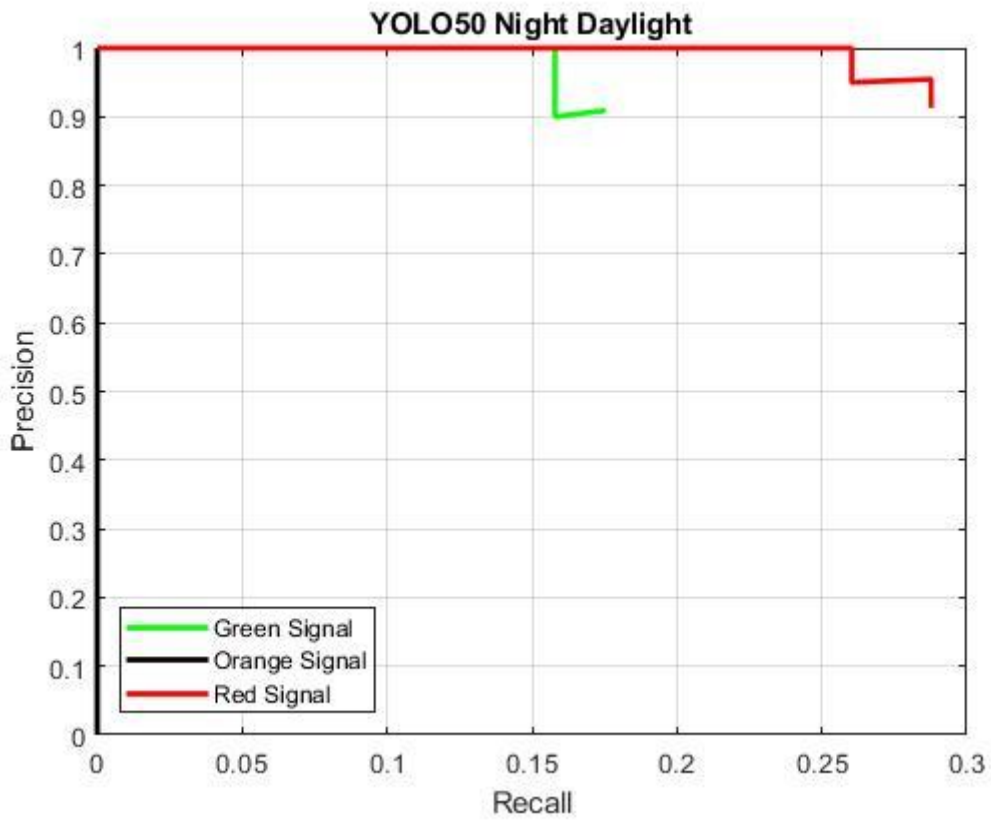


Figure 23: Night trained YOLO50 tested against the Daylight data

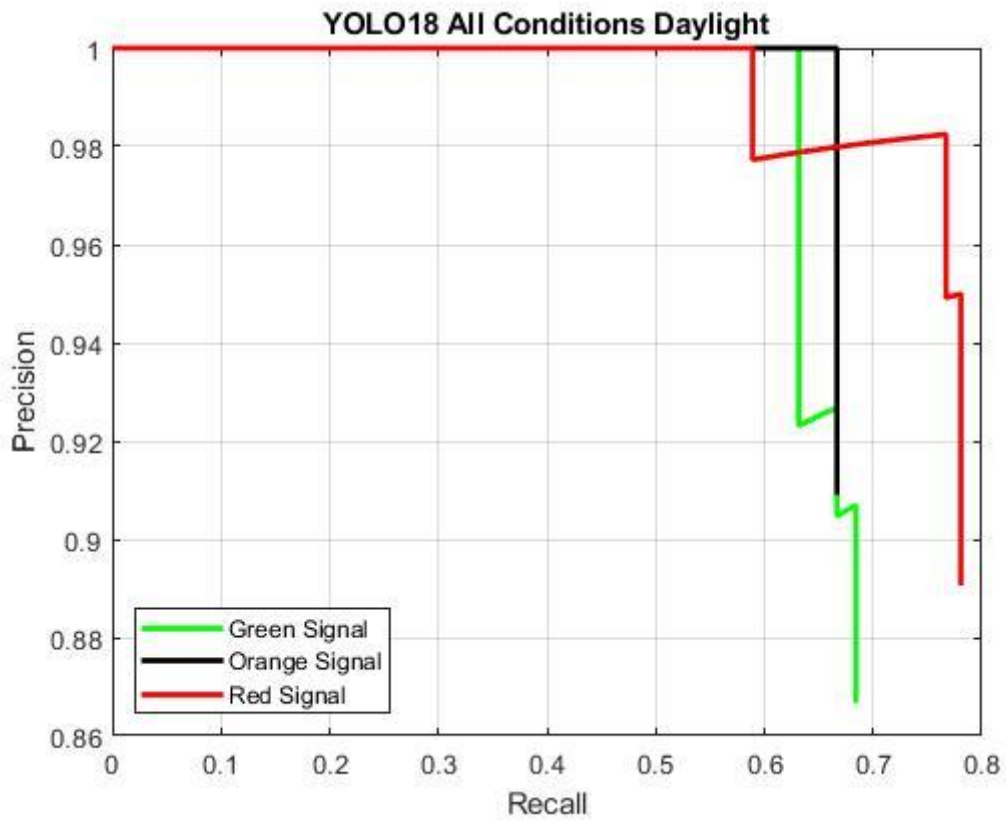


Figure 24: All conditions trained YOLO18 tested against the Daylight data

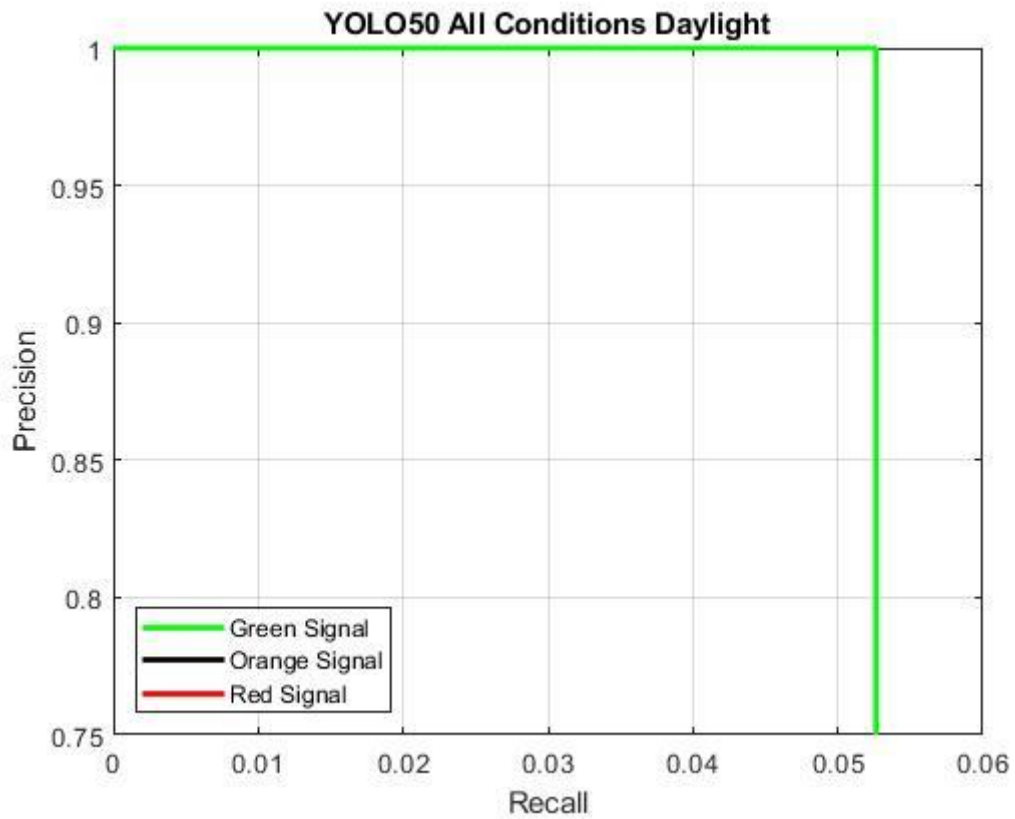


Figure 25: All conditions trained YOLO50 tested against the Daylight data

Table 8: Daylight data F-score table

	Green Signal F-score	Orange Signal F-score	Red Signal F-score	Combined F-score
YOLO18 Day	0.5402	0.4864	0.5398	0.5222
YOLO50 Day	0.4484	0.5060	0.4573	0.4706
YOLO18 Night	0.3216	0.3741	0.3162	0.3373
YOLO50 Night	0.1655	0	0.2561	0.1405
YOLO18 All conditions	0.5103	0.4864	0.5520	0.5163
YOLO50 All conditions	0.0601	0	0	0.0200

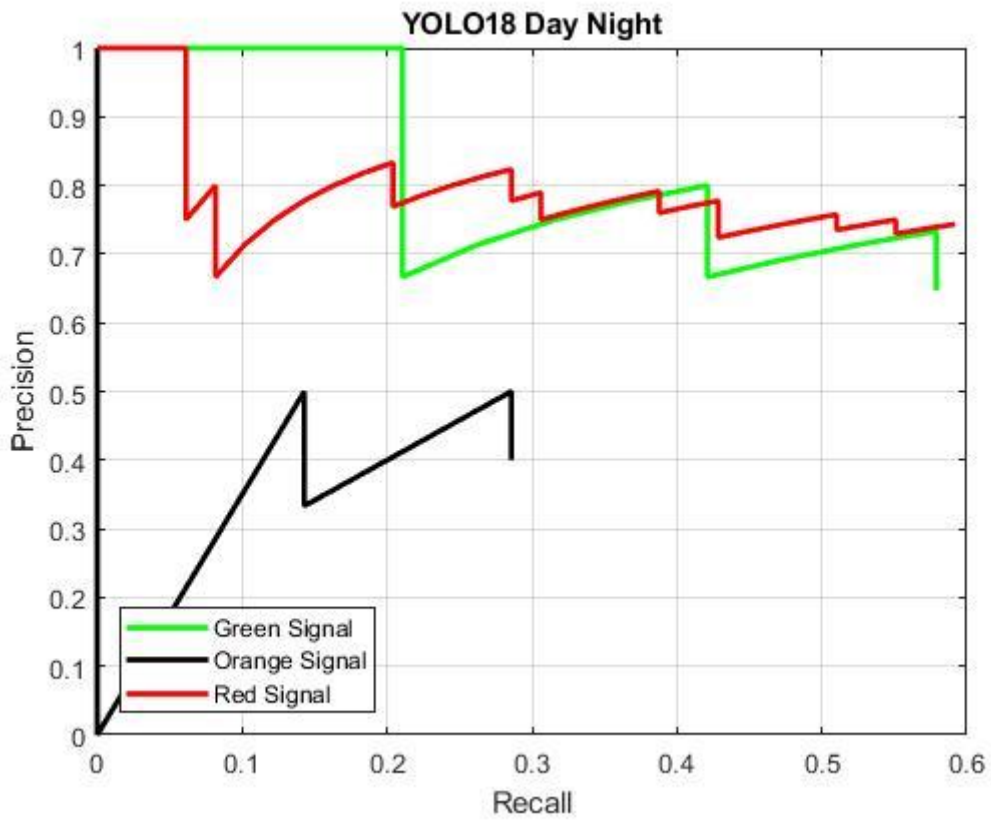


Figure 26: Daylight trained YOLO18 tested against the Night data

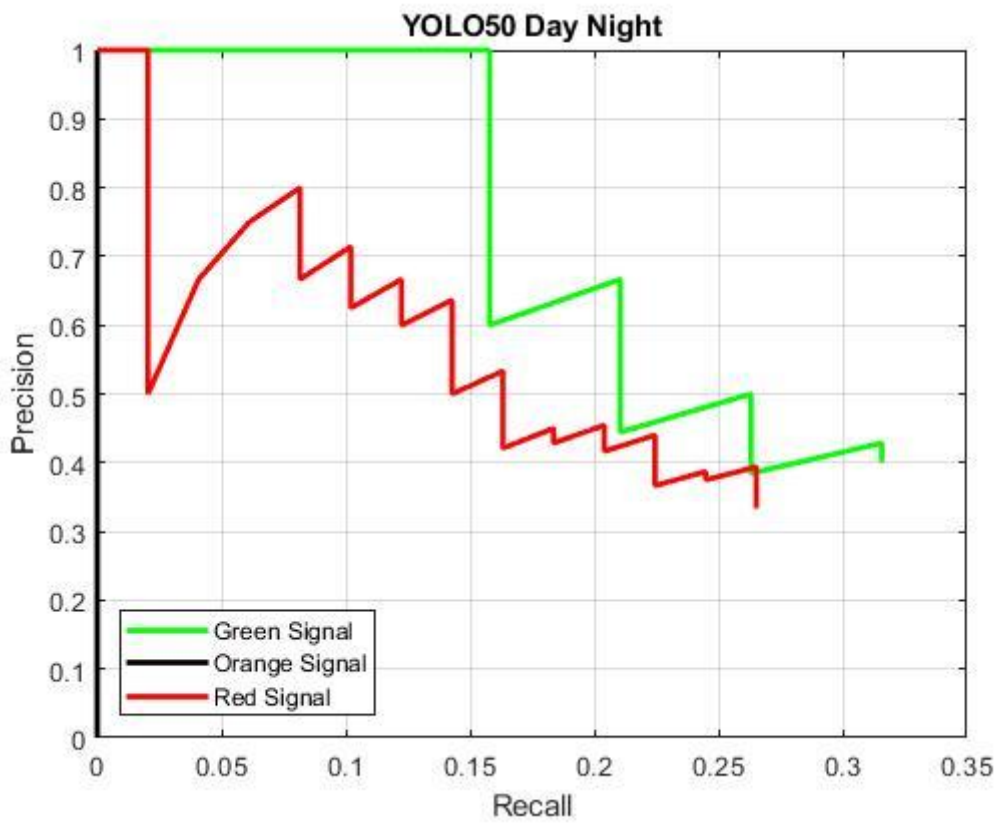


Figure 27: Daylight trained YOLO50 tested against the Night data



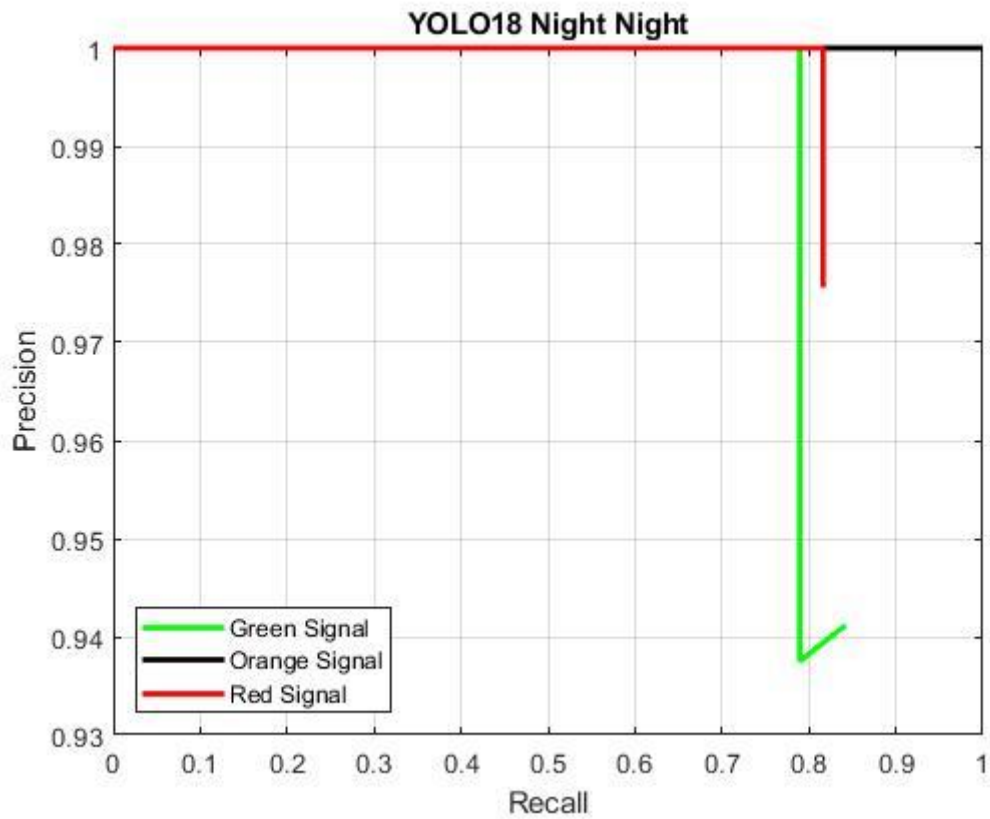


Figure 28: Night trained YOLO18 tested against the Night data

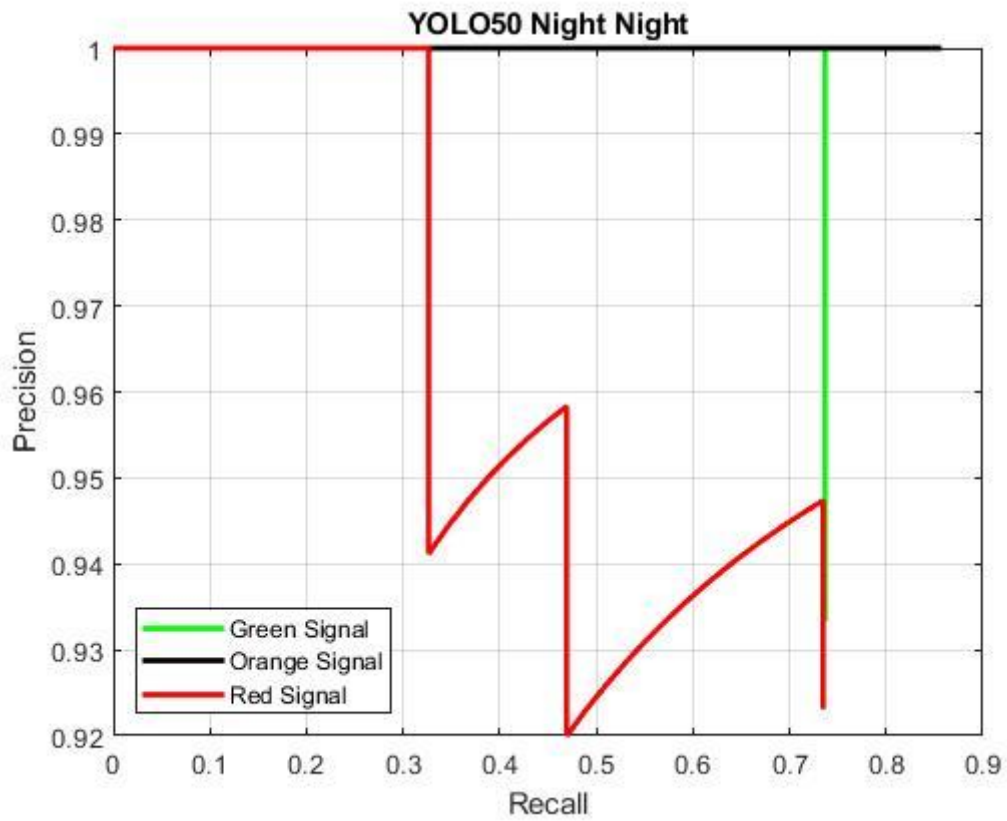


Figure 29: Night trained YOLO50 tested against the Night data

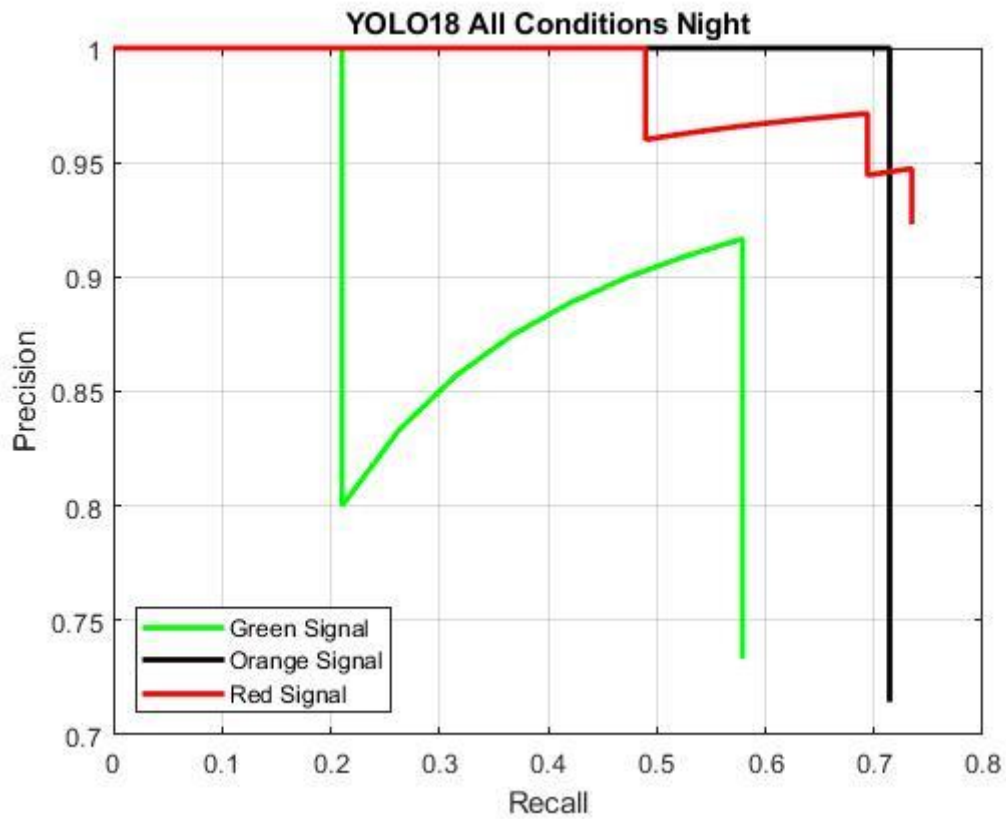


Figure 30: All conditions trained YOLO18 tested against the Night data

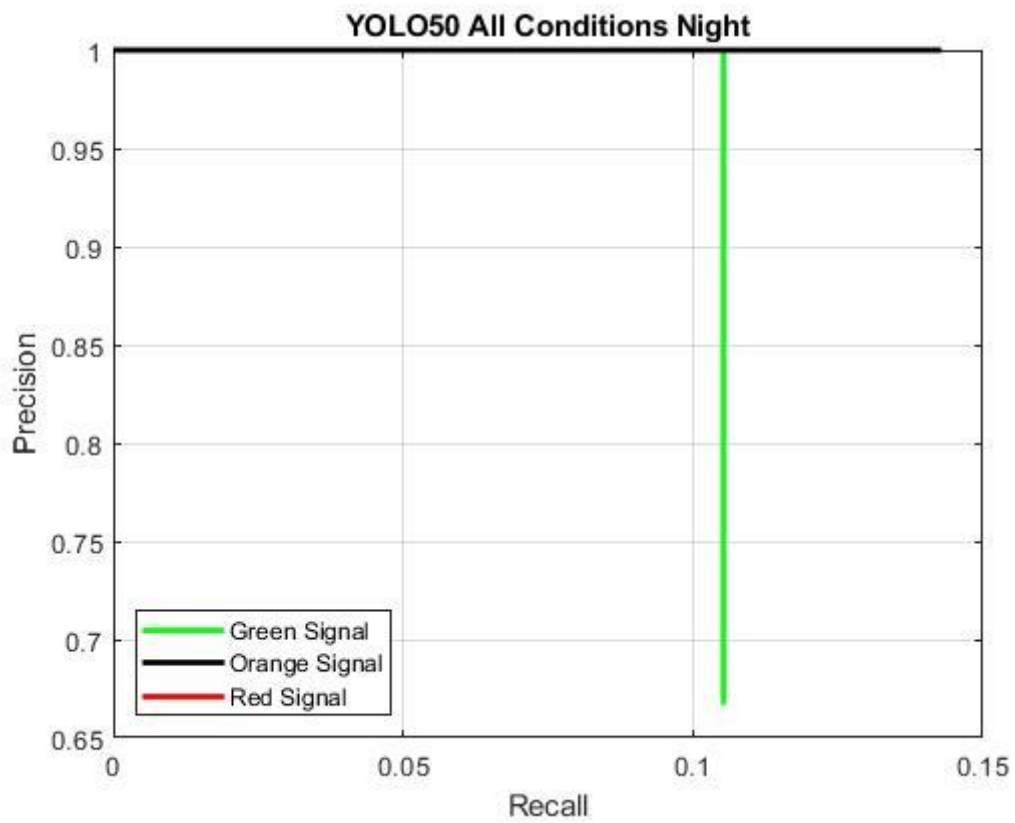


Figure 31: All conditions trained YOLO50 tested against the Night data

Table 9: Night data F-score table

	Green Signal F-score	Orange Signal F-score	Red Signal F-score	Combined F-score
YOLO18 Day	0.4194	0.3486	0.4024	0.3486
YOLO50 Day	0.2696	0	0.2317	0.1671
YOLO18 Night	0.5593	0.5973	0.5540	0.5669
YOLO50 Night	0.5165	0.5397	0.5016	0.5193
YOLO18 All conditions	0.4496	0.5423	0.5143	0.5021
YOLO50 All conditions	0.1181	0.1250	0	0.0810

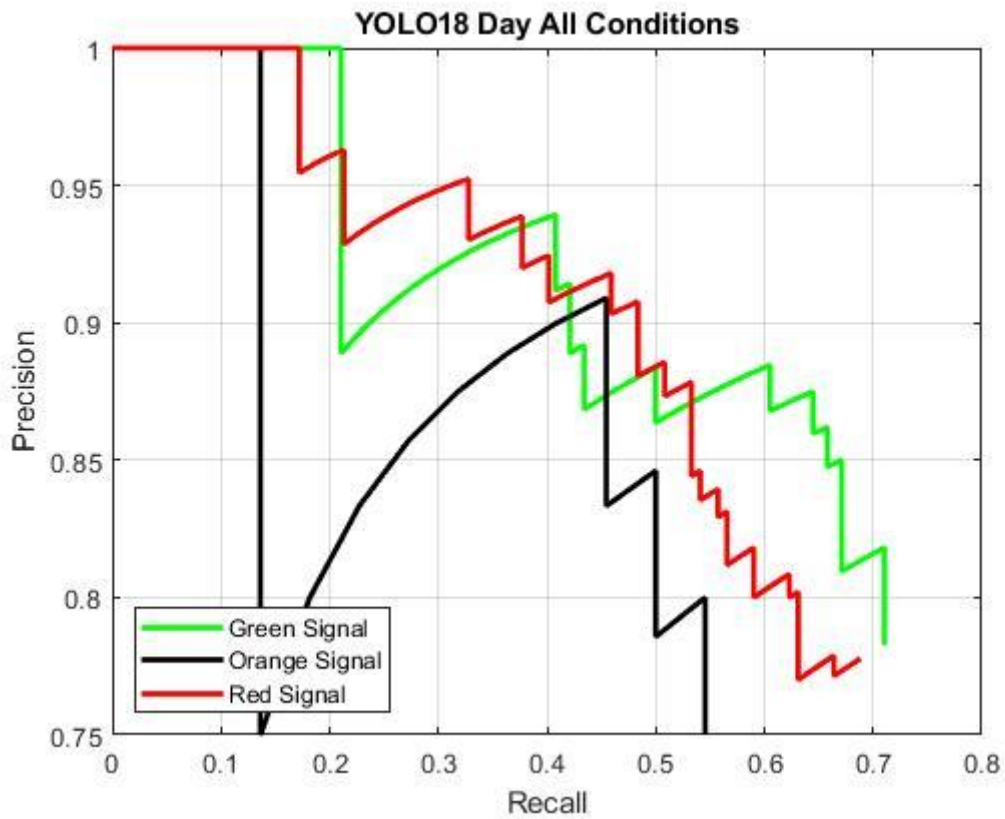


Figure 32: Daylight trained YOLO18 tested against the All conditions data

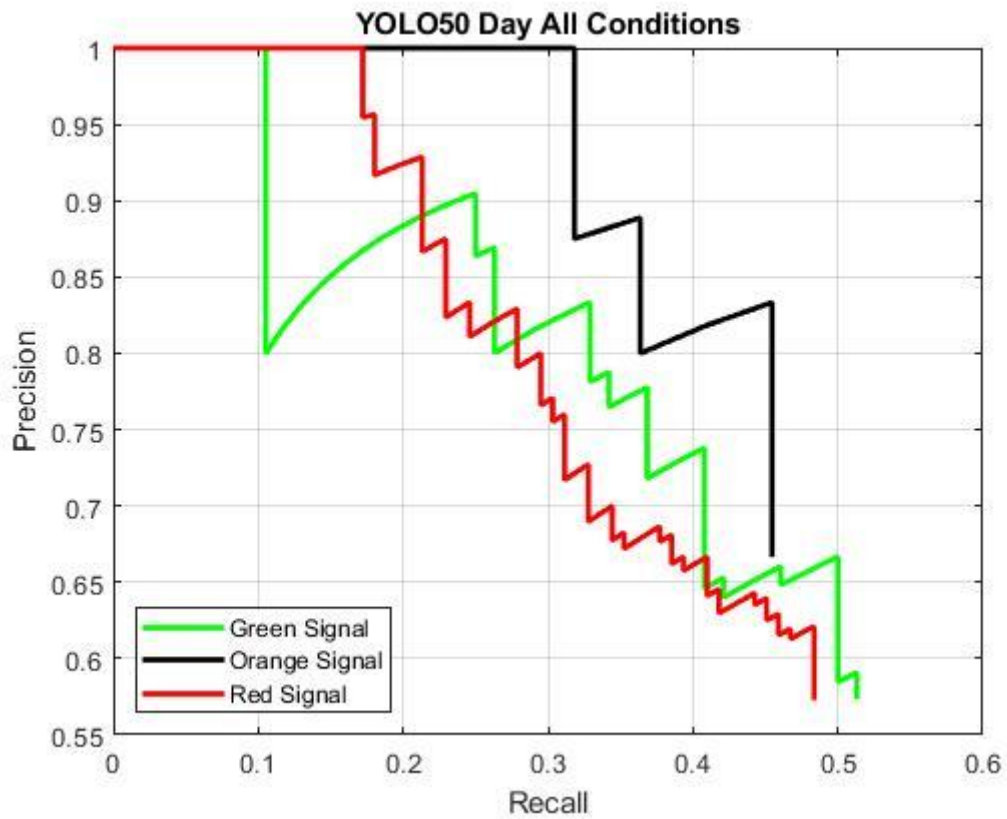


Figure 33: Daylight trained YOLO50 tested against the All conditions data

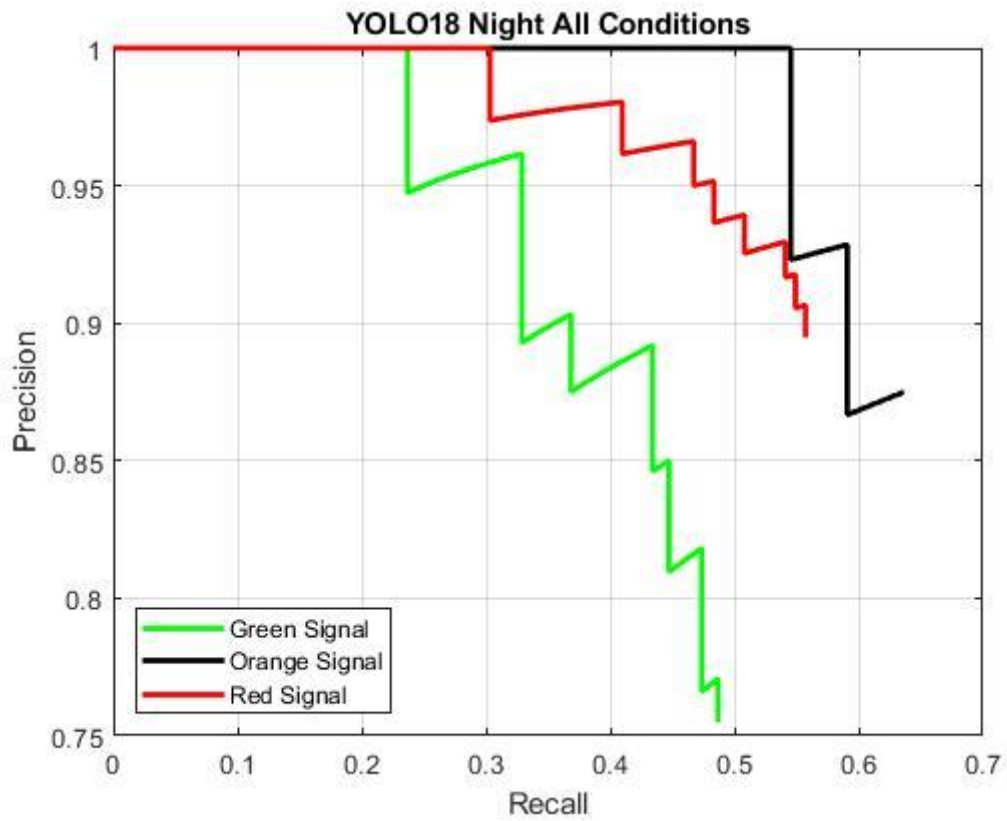


Figure 34: Night trained YOLO18 tested against the All conditions data

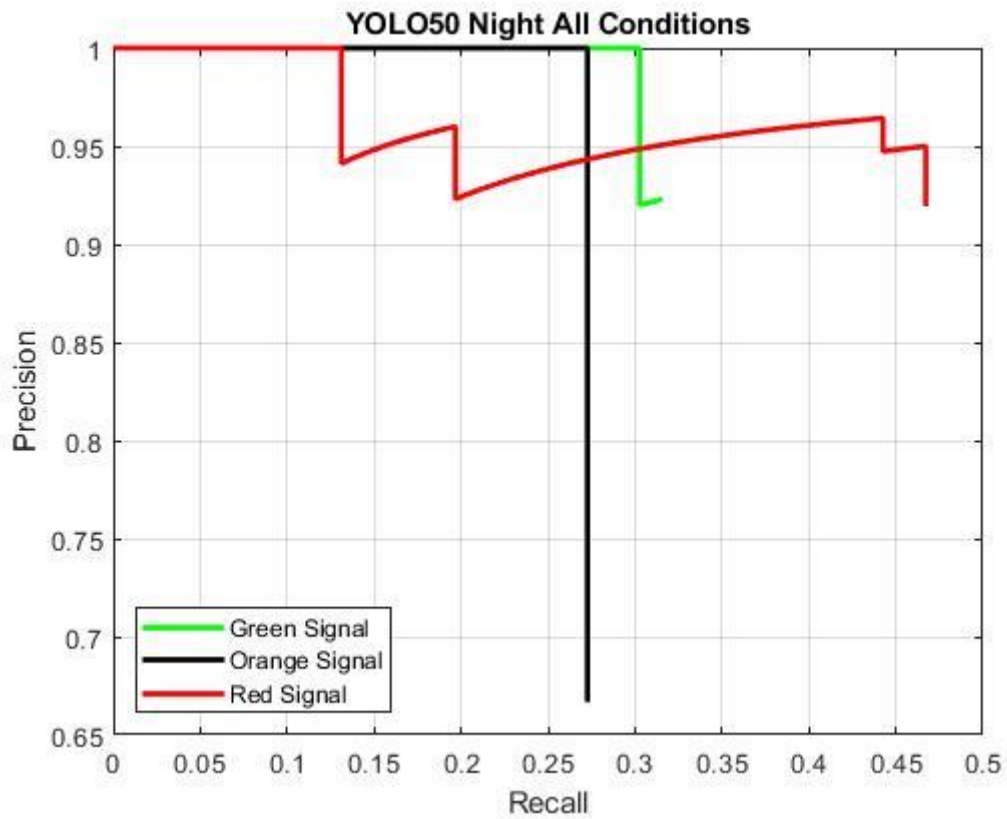


Figure 35: Night trained YOLO50 tested against the All conditions data

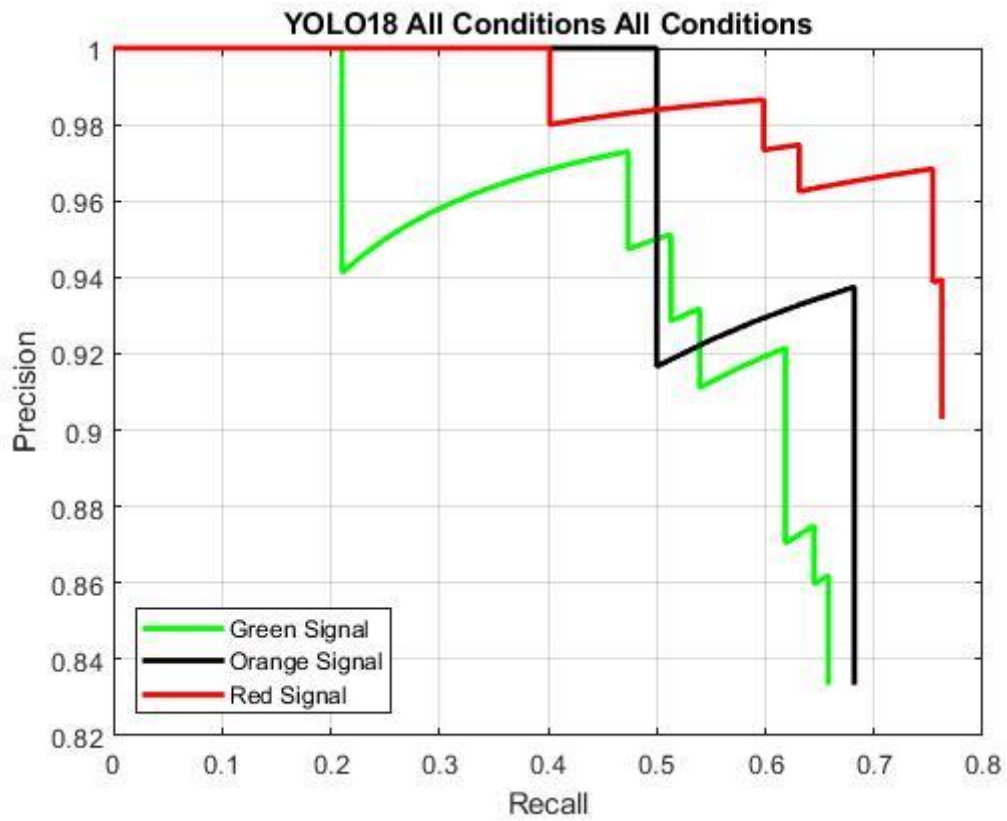


Figure 36: All conditions trained YOLO18 tested against the All conditions data

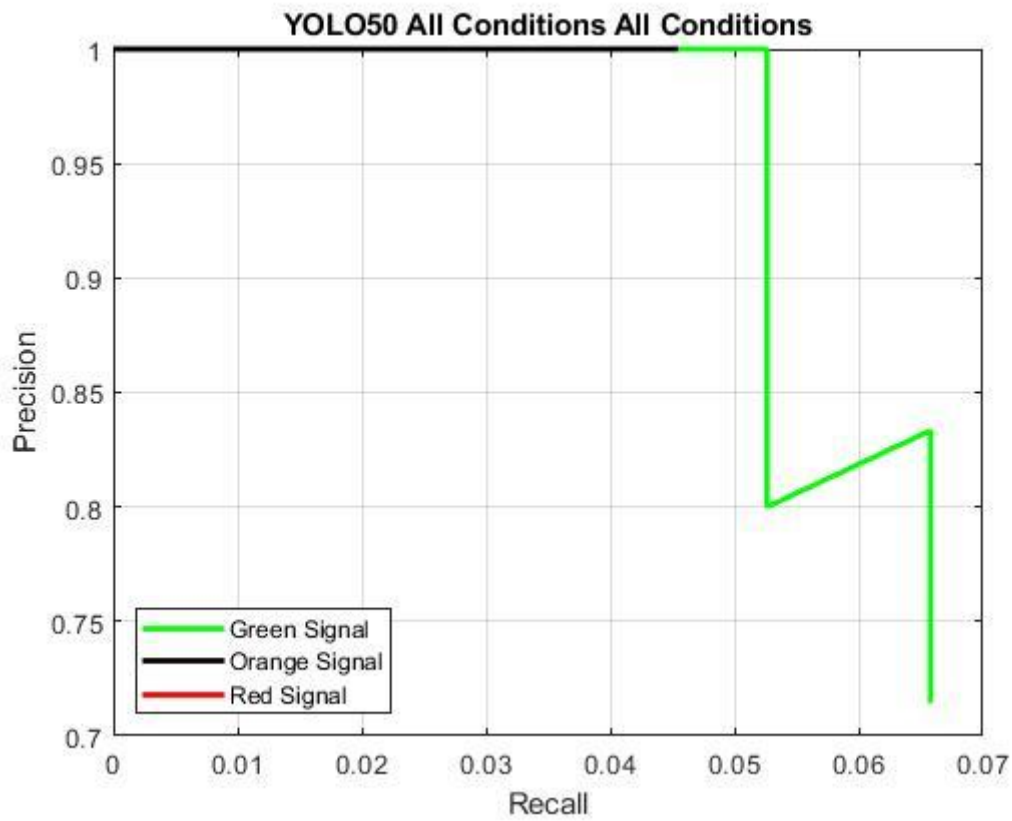


Figure 37: All conditions trained YOLO50 tested against the All conditions data

Table 10: All conditions F-score table

	Green Signal F-score	Orange Signal F-score	Red Signal F-score	Combined F-score
YOLO18 Day	0.5069	0.4148	0.4928	0.4715
YOLO50 Day	0.4089	0.3979	0.3910	0.3993
YOLO18 Night	0.4043	0.4731	0.4286	0.4353
YOLO50 Night	0.2752	0.2801	0.3643	0.3066
YOLO18 All conditions	0.4883	0.5057	0.5364	0.5104
YOLO50 All conditions	0.0743	0.0435	0	0.0393

## Chapter 6

### Analysis

#### 6.1 Training Data Analysis

As stated, it was expected that as the training progressed, the RMSE and LOSS values would trend toward zero with the difference between the mini-batch and validation being minimised. This expectation was roughly followed with only one egregious deviation, the training data for the YOLO50 day and night. While the differences between the mini-batch and validation values were the smallest of any training, it did not progress below one. This showed, without the need to test, that this algorithm will operate with significant error. The most likely cause of this was that the size of the training data was insufficient and the different data sets did not have enough data points for the larger neural network to learn from.

Other trends provided by the training data, is that the smaller algorithm does not require as much time to train. This was expected behaviour and gives a positive outlook toward any future work optimising the neural network. A surprise was that YOLO18 day and night algorithm had less difference between the Mini-Batch and Validation values than the individualised algorithms. This was not expected as the difference between the brightest daylight and darkest night images would cause some incorrect weightings between interactions and translate into errors in the validation pass, but the opposite has occurred. It would seem that the additional data has provided a significantly more balanced training and reduced the difference.

#### 6.2 Testing results Analysis

The test results tell an interesting story.

Overall, the smaller algorithm in every training configuration was better. There are many possible reasons for this. The most likely reason is that the training size was not large enough for the larger algorithm. Another theory as to why the larger algorithm did not perform is that its size worked against it. With the less varied background that comes with the older game engine, the thresholding and weighting that occurs in training could become counter productive the deeper the neural network.

An interesting comparison is between the separate day and night algorithms. The day algorithms worked well in the daylight tests as expected but also provided reasonable results in the night tests. This was not explicitly true for the reverse. A reason is that with the light faded in the night training images, the background only contains dark colours, leaving the algorithms to observe that the brighter colours are the primary feature of the detection. This worked against the night trained algorithms as they were not able to decipher the bright background of the daylight images. Inverting this reason explains why the day trained algorithms fared better when changing out of the trained environment. The algorithms 'learnt' that the signal was more than a bright colour on a dark background, it required a dark colour to surround the bright colour.

As expected from the training results, the YOLO50 day and night algorithm did not perform. It mainly produced a no pickup result. Surprisingly, it did manage to identify some signals, the adage of 'a broken clock will still tell the correct time twice a day' ringing true in this case.

### 6.3 The story told by the F-scores and precision-recall curves

While these results paint a similar picture to the previous graphs, they do provide unique insights that are worth looking at separately.

Starting with the F-scores, these general accuracy values give far more insight into how individual classes are handled. Each algorithms had its own weakest class and this poor performing class remained across all the tests. The inverse is also true. There is some variance to this trend, and this is attributed to the small size of the testing data. Table 11 has the most prominent example of this trend.

Table 11: All Daylight trained YOLO18 F-scores

	Green Signal F-score	Orange Signal F-score	Red Signal F-score	Combined F-score
YOLO18 Day	0.5402	0.4864	0.5398	0.5222
YOLO18 Day	0.4194	0.3486	0.4024	0.3486
YOLO18 Day	0.5069	0.4148	0.4928	0.4715

A point that is now underscored by the F-scores is that the smaller YOLO18 algorithms beat the larger YOLO50 algorithms comprehensively. This might speak to how the size of the training pool can effect how well the algorithms do. If additional time was available tests would have been completed with significantly larger training set sizes. This would better illustrate the effects of different sized training sets and would help in finding the point of diminishing returns. Another point of interest exposed by the F-scores that reinforces the Section 5.2 test data is the overall accuracy across the tests. Inspecting table 12, we can see that the most accurate across the testing was the All conditions trained YOLO18 with relatively little variance in accuracy across the tests. The accuracy is roughly equal when comparing the algorithms when operating outside of their training.



Table 12: Condensed F-score table

	Daylight F-score	Night F-score	All Conditions F-score
YOLO18 Day	0.5222	0.3486	0.4715
YOLO50 Day	0.4706	0.1671	0.3993
YOLO18 Night	0.3373	0.5669	0.4353
YOLO50 Night	0.1405	0.5193	0.3066
YOLO18 All conditions	0.5163	0.5021	0.5104
YOLO50 All conditions	0.0200	0.0810	0.0393

The precision-recall curve charts really highlight that as the sensitive goes up the precision falls away dramatically in for most of the tested configurations. All tested configurations have at least one class begin the decline before a recall value of 0.5 with the exception of night trained YOLO18 tested against the night data.

# Chapter 7

## Discussion

### 7.1 Results Discussion

Beyond the statistical and trend analysis of the results, what do they mean? The results answer two of the three research questions, it also answers the overarching question of can current machine vision techniques capture and interpret rail signals with control automation the logical step afterwards.

The results also provide clarity on how algorithms interact with different neural network that also react differently depending on the training provided. If this was only a checkpoint in a multi-year project, these results indicate 2 important lessons that should be taken aboard.

First is the importance of large and varied training data. Too little data can lead to ineffective training and a narrow band of data can significantly limit the flexibility of the algorithm. Second, starting small with the neural networks. Many hours were dedicated to the training of the algorithms with the YOLO50 variants taking the majority of the time. This fact stings as these variants were all lacking compared to their comparable YOLO18 algorithm.

### 7.2 Consequences of Implementation

With all advancements in automation, there are valid concerns that technology will replace humans, this project is no different. A standard freight train has two-person crew, the conductor, and the engineer(driver). Development of the proposed research area will lead to a shift in jobs. Early implementation would not see many jobs move as this technology will still be in its infancy. As confidence in the technology improves, that is where the job shift will occur as the technology will make the engineer role mostly redundant on the vast majority of railroad. Engineers will still be required in the railyards or when the technology has a fault that cannot be rectified before the train is required to depart.

The flip side is that once this technology has matured, it will provide a far safer environment than having a driver in direct control of a train and will greatly increase efficiency. A system that is always

vigilant, never needs to rest, coupled with existing ATP systems, it could bring safety and automation to the countries that cannot afford large infrastructure overhauls. Bringing an aftermarket product for rail automation will breathe new life into aging locomotives and rollingstock.

# Chapter 8

## Conclusions

### 8.1 Future work

There are two fields of future work that this dissertation has spawned, first is the practical work of physical implementation, second is refinement of the algorithm and associated neural network/s.

The practical work is seemingly straight forward, repeat this work but with data collected from relevant train lines. This would require industry connections to complete such a task. Another section in this area is the determination of the physical hardware and the mounting techniques.

There is still much work to be done regarding the algorithms and neural networks. This paper only looked at a YOLOv2 as the algorithm with an attempt at using Faster-RCNN. As YOLO has proven to be capable, using a classifier algorithm like Faster-RCNN is not necessary in future work beyond the conformation that regression algorithms are the better choice when speed and accuracy are required. Use of more advanced versions of YOLO would yield useful data as the incremental changes could have unintended effects. The testing of other regression algorithms that are created would be interesting as only two distinctive algorithms exist at the time of writing, the two being YOLO and Single Shot Detection.

Only two neural networks were tested, and these were prebuilt. There are so many variables inside just one neural network that it could be an entire dissertation. Optimisation of network complexity, where the algorithm is meshed with the neural network, training amounts, hardware compatibility. All have tremendous depth that was barely scratched in this work. Building a custom neural network optimised to be used with object detection algorithms, much like Tesla and their self-driving cars. The mesh points that were used may be completely incorrect and another node might provide superior performance. Anchor box size may restrict what the detector can identify accurately. There is significant work to be done to improve the understanding of both neural networks and their associated integration with machine vision algorithms.

## 8.2 Conclusion

To conclude, the 5 objectives were set in the introduction:

- Explore the current state of train control and railway infrastructure.  
- Accomplished in **chapter 2**
- Highlight the unique challenges of railway automation.  
- Accomplished in **chapter 3**
- Review the advancements in machine vision.  
- Accomplished in **chapter 3**
- Investigate possible system/algorithms for implementation  
- Accomplished in **chapter 3**
- Develop a prototype algorithm that detects rail signals as a proof of concept.  
- Accomplished in **chapter 4 -> 7**

To reiterate, advanced machine vision techniques have matured to the point where implementation onto locomotives is a possibility. There is still much work to be done before implementation is to occur on mass. Optimisation of the neural networks and algorithms are still required along with determination of physical hardware to make the system functional.

## References

Shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, Kazuya Takeda, Tsuyoshi Hamada, 2015, "An Open Approach To Autonomous Vehicles", viewed June 2021, <<http://cs.furman.edu/~tallen/csc271/source/openAppr.pdf>>

Dickmanns, 2002, "keynote", viewed May 2021 <[http://www.dyna-vision.de/main/Text\\_PDF/2002%20Dickmanns-Keynote%20IV'02.pdf](http://www.dyna-vision.de/main/Text_PDF/2002%20Dickmanns-Keynote%20IV'02.pdf)>

Transport Technology Center, 2014, "research machine vision inspection", viewed June 2021 <[http://www.railway-research.org/IMG/pdf/aar\\_ttc\\_i\\_research\\_machine\\_vision\\_inspection\\_jan\\_2014.pdf](http://www.railway-research.org/IMG/pdf/aar_ttc_i_research_machine_vision_inspection_jan_2014.pdf)>

RailSystem, unknown, "Communications-Based Train Control (CBTC)", viewed March 2021 <http://www.railsystem.net/communications-based-train-control-cbtc/>

Wolfgang Pree, Oliver Gebauer, Burkhard Stadlmann, 2012, "Autonomously Driving Trains on Open Tracks — Concepts, System Architecture and Implementation Aspects", viewed December 2020 [https://www.researchgate.net/publication/272537732\\_Autonomously\\_Driving\\_Trains\\_on\\_Open\\_Tracks\\_-\\_Concepts\\_System\\_Architecture\\_and\\_Implementation\\_Aspects](https://www.researchgate.net/publication/272537732_Autonomously_Driving_Trains_on_Open_Tracks_-_Concepts_System_Architecture_and_Implementation_Aspects)

Felicity Caldwell, 2021, "Commuters at risk as train drivers increasingly run red lights", viewed august 2021 <https://www.brisbanetimes.com.au/politics/queensland/commuters-at-risk-as-train-drivers-increasingly-run-red-lights-20210603-p57xtw.html>

International Association of Public Transport, 2012 "A global bid for automation: UITP Observatory of Automated Metros confirms sustained growth rates for the coming years", viewed July 2021 [https://metroautomation.org/wp-content/uploads/2012/12/Automated\\_metros\\_Atlas\\_General\\_Public\\_2012.pdf](https://metroautomation.org/wp-content/uploads/2012/12/Automated_metros_Atlas_General_Public_2012.pdf)

Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, 2014, "Rich feature hierarchies for accurate object detection and semantic segmentation", viewed September 2021 <https://arxiv.org/pdf/1311.2524.pdf>

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, 2016, "You Only Look Once: Unified, Real-Time Object Detection" viewed September 2021, < <https://arxiv.org/pdf/1506.02640.pdf>>

University of Waterloo, 2020, "Intrinsic plasticity improves learning in deep neural networks and provides a plausible explanation of how brains operate at a local neuronal level" viewed august 2020 < <https://cs.uwaterloo.ca/news/intrinsic-plasticity-improves-learning-in-dnns-and-provides-plausible-explanation-how-brains-operate-at-local-level>>

# Appendix A – Project Specification

ENG4111/4112 Research Project

Project Specification

For: Anthony Wallin

Title: Application for machine vision for autonomous locomotive operation

Major: Mechatronics

Supervisors: Dr Tobias Low  
Dr Derek Long

Enrollment: ENG4111 – DIR S1, 2021  
ENG4112 – DIR S2, 2021

Project Aim: To investigate existing machine vision algorithms then prototype a program capable of operating a simulated locomotive as a proof of concept for real world applications.

Programme: Version 1.2, 5<sup>th</sup> April 2021

1. Review available regions for testing, make selection then collect information about infrastructure in the region.
2. Review machine vision algorithms used in similar applications and identify candidate approach/es.
3. Testing methodology development. Design a testing matrix for the evaluation of each section of coding. Use test footage gathered to test machine vision code.
4. Build program for the recognition of signals using MATLAB as the prototyping environment.
5. Extend program to encompass trackside signage. This will mainly encompass the speed signs.
6. Build a program capable of interfacing with the simulator.
7. Transfer the prototyped machine vision algorithms from MATLAB to the control program for complete system testing.
8. Complete system evaluation. Once working, the system will be marked against a criterion of accuracy, speed, and reliability. The accuracy of data collection and interpretation. The speed at which the data is collected and interpreted. And the overall stability of the system in endurance testing.

If time and resources permit:

9. Design and acquisition of hardware.

10. Physical implementation and testing.



## Appendix B – Code

### Neural Network creation

```
%{
YOLO V2 Object Detection Creator
Author: Anthony Wallin

Description:
Code to create a YOLO V2 Object Detector
%}

inputSize = [640 640 3];
numClasses = 3;
featureExtractionNetwork = resnet18;
featureLayer = 'res4a_relu';
anchorBoxes = [
    32 32
    16 64
    64 16
    ];
lgraph =
yolov2Layers(inputSize,numClasses,anchorBoxes,featureExtractionNetwork,featureLayer);

save YOLOV218
```

### Algorithm training

```
Prototype training
Author: Anthony Wallin

Description:
Code for training an YOLO V2 object detector
%}
%Set 0 for initial Training
%Set 1 for further Training
%Set 2 for Testing
Train = 0

%extracting the ground truth table
data = load('SignalDataIV.mat');
SignalDataset = data.gTruth;

% Display first few rows of the data set.
SignalDataset(1:4,:)

% Add the fullpath to the local data folder.
SignalDataset.imageFilename = fullfile(SignalDataset.imageFilename);

%Image shuffling and distribution
```

```

rng(0);
shuffledIndices = randperm(height(SignalDataset));
idx = floor(0.6 * length(shuffledIndices) );

trainingIdx = 1:idx;
trainingDataTbl = SignalDataset(shuffledIndices(trainingIdx),:);

validationIdx = idx+1 : idx + 1 + floor(0.1 * length(shuffledIndices) );
validationDataTbl = SignalDataset(shuffledIndices(validationIdx),:);

testIdx = validationIdx(end)+1 : length(shuffledIndices);
testDataTbl = SignalDataset(shuffledIndices(testIdx),:);

imdsTrain = imageDatastore(trainingDataTbl{:, 'imageFilename'});
bldsTrain = boxLabelDatastore(trainingDataTbl(:,2:4));

imdsValidation = imageDatastore(validationDataTbl{:, 'imageFilename'});
bldsValidation = boxLabelDatastore(validationDataTbl(:,2:4));

imdsTest = imageDatastore(testDataTbl{:, 'imageFilename'});
bldsTest = boxLabelDatastore(testDataTbl(:,2:4));

trainingData = combine(imdsTrain,bldsTrain);
validationData = combine(imdsValidation,bldsValidation);
testData = combine(imdsTest,bldsTest);

%{
Data Verification
data = read(trainingData);
I = data{1};
bbox = data{2};
annotatedImage = insertShape(I, 'Rectangle', bbox);
annotatedImage = imresize(annotatedImage, 2);
figure
imshow(annotatedImage)
%}
if Train == 0;
%Data Augmentation and preprocessing
%Aug = imageDataAugmenter('RandXTranslation', [-5,5], 'RandYTranslation', [-5,5]);
inputSize = [640,640];
PTD =
transform(trainingData, @(data) preprocessData(data, inputSize)); %Preprocessed
Training
PVD =
transform(validationData, @(data) preprocessData(data, inputSize)); %Preprocessed
Validation

%{
%Data Verification
data = read(PTD);
I = data{1};
bbox = data{2};
annotatedImage = insertShape(I, 'Rectangle', bbox);
annotatedImage = imresize(annotatedImage, 2);
figure
imshow(annotatedImage)
%}

%Detector Training
options = trainingOptions('adam', ...
    'MiniBatchSize', 10, ....

```

```

        'InitialLearnRate',1e-3, ...
        'MaxEpochs',50, ...
        'CheckpointPath',tempdir, ...
        'ValidationData',PVD,...
        'plots','training-progress',...
        'shuffle','every-epoch');

SignalDetectorXVI = load('YOLOV250.mat');
lgraph = SignalDetectorXVI.lgraph;

[SignalDetectorXVI,info] = trainYOLOv2ObjectDetector(PTD,lgraph,options)

elseif Train == 1;
    %extracting the ground truth table
    data = load('SignalDataIV.mat');
    SignalDataset = data.gTruth;

    % Display first few rows of the data set.
    SignalDataset(1:4,:)

    % Add the fullpath to the local data folder.
    SignalDataset.imageFilename = fullfile(SignalDataset.imageFilename);

    %Image shuffling and distribution
    rng(0);
    shuffledIndices = randperm(height(SignalDataset));
    idx = floor(0.6 * length(shuffledIndices) );

    trainingIdx = 1:idx;
    trainingDataTbl = SignalDataset(shuffledIndices(trainingIdx),:);

    validationIdx = idx+1 : idx + 1 + floor(0.1 * length(shuffledIndices) );
    validationDataTbl = SignalDataset(shuffledIndices(validationIdx),:);

    testIdx = validationIdx(end)+1 : length(shuffledIndices);
    testDataTbl = SignalDataset(shuffledIndices(testIdx),:);

    imdsTrain = imageDatastore(trainingDataTbl{:,'imageFilename'});
    bldsTrain = boxLabelDatastore(trainingDataTbl(:,2:4));

    imdsValidation = imageDatastore(validationDataTbl{:,'imageFilename'});
    bldsValidation = boxLabelDatastore(validationDataTbl(:,2:4));

    imdsTest = imageDatastore(testDataTbl{:,'imageFilename'});
    bldsTest = boxLabelDatastore(testDataTbl(:,2:4));

    trainingData = combine(imdsTrain,bldsTrain);
    validationData = combine(imdsValidation,bldsValidation);
    testData = combine(imdsTest,bldsTest);

    %Data Augmentation and preprocessing
    %Aug = imageDataAugmenter('RandXTranslation',[-
5,5],'RandYTranslation',[-5,5]);
    inputSize = [640,640];
    PTD =
transform(trainingData,@(data)preprocessData(data,inputSize));%Preprocessed
Training
    PVD =
transform(validationData,@(data)preprocessData(data,inputSize));%Preprocessed
Validation

```

```

%Detector Training
options = trainingOptions('adam', ...
'MiniBatchSize',10, ...
'InitialLearnRate',1e-3, ...
'MaxEpochs',50, ...
'CheckpointPath',tempdir, ...
'ValidationData',PVD,...
'plots','training-progress',...
'shuffle','every-epoch');

SignalDetectorV = load('SignalDetectorV.mat');
lgraph = SignalDetectorV;

[SignalDetectorXI,info] = trainYOLOv2ObjectDetector(PTD,lgraph,options)
else
load('SignalDetectorIII.mat')
inputSize = [640,640];
I = imread('SS16.jpg');
I = imresize(I,inputSize(1:2));
[bboxes,scores,label] = detect(SignalDetectorIII,I)
I = insertObjectAnnotation(I,'rectangle',bboxes,scores);
figure
imshow(I)

end

```

## Algorithm Testing

```

%{
Train Signal Detection function
Author: Anthony Wallin

Description:
Used to capture simulator screen, apply a machine vision algorithm to the
capture, then output requested signal status
Version 1.0
%}

% Take screen capture
robot = java.awt.Robot();
pos = [0 0 1920 800]; % [left top width height]
rect = java.awt.Rectangle(pos(1),pos(2),pos(3),pos(4));
cap = robot.createScreenCapture(rect);

% Convert to an RGB image
rgb =
typecast(cap.getRGB(0,0,cap.getWidth,cap.getHeight,[],0,cap.getWidth),'uint8');
imgData = zeros(cap.getHeight,cap.getWidth,3,'uint8');
imgData(:, :, 1) = reshape(rgb(3:4:end),cap.getWidth,[]);
imgData(:, :, 2) = reshape(rgb(2:4:end),cap.getWidth,[]);
imgData(:, :, 3) = reshape(rgb(1:4:end),cap.getWidth,[]);

% Load Trained Detector
SignalDetector = load('SignalDetectorV.mat','SignalDetectorV');
Detector = SignalDetector.SignalDetectorV;

```

```

%Resize Image
inputSize = [640,640];
I = imgData;
I = imresize(I,inputSize(1:2));

%Run Signal Detection
[bboxes,scores,label] = detect(Detector,I);
I = insertObjectAnnotation(I,'rectangle',bboxes,label);
figure
imshow(I)

% Organisation of collected data
label = double(label);
combined = [bboxes label];
combined = sortrows(combined);

%Signal logic
n = 0;
Buffer = zeros(1,5);
SignalMatrix = zeros(3,4);
while combined ~= 0
    n = n+1;
    Signal = combined(combined(:,1)<(combined(1,1)+30),:);
    Signal = sortrows(Signal,2);
    Signal = [Signal;Buffer];
    combined = combined(combined(:,1)>(combined(1,1)+30),:);
    k = 1;
    while Signal(k,5) ~=0
        SignalMatrix(k,n) = Signal(k,5);
        k = k+1;
    end
end
%Output
SignalMatrix

```

## Creation of F-score

```

%{
F-score
Author: Anthony Wallin

Description:
create F-score graphs
%}

data = load('SignalDataVI.mat');
SignalDataset = data.gTruth;

imds = imageDatastore(SignalDataset.imageFilename(51:75));
blbs = boxLabelDatastore(SignalDataset(51:75,2:end));

SignalDetector = load('SignalDetectorXI.mat','SignalDetectorXI');
Detector = SignalDetector.SignalDetectorXI;

results = detect(Detector,imds);

[ap, recall, precision] = evaluateDetectionPrecision(results, blbs);

```

```

figure
plot(recall{1},
precision{1}, 'green', recall{2}, precision{2}, 'black', recall{3}, precision{3}, 'red'
, 'LineWidth', 2);
legend('Green Signal', 'Orange Signal', 'Red Signal', 'location', 'southwest');
xlabel('Recall'), ylabel('Precision');
grid on;
title('YOLO50 All Conditions Night');

fg = 2*(recall{1}.*precision{1})./(recall{1}+precision{1});
fo = 2*(recall{2}.*precision{2})./(recall{2}+precision{2});
fr = 2*(recall{3}.*precision{3})./(recall{3}+precision{3});
f(1) = mean(fg);
f(2) = mean(fo, 'omitnan');
f(3) = mean(fr)

fA = mean(f, 'omitnan')

```

## Appendix C – Raw data

test image	YOLO18 Day	YOLO50 Day	YOLO18 All Conditions	YOLO50 All Conditions	YOLO18 Night	YOLO50 Night
1	1	1	1	4	1	4
2	1	1	1	4	4	4
3	1	2	1	4	2	2
4	1	1	3	4	1	2
5	3	4	4	4	4	4
6	4	4	4	4	4	4
7	3	4	4	4	4	4
8	2	2	2	4	4	4
9	2	1	1	4	2	4
10	1	1	1	2	2	4
11	4	4	4	4	4	4
12	2	1	1	4	2	4
13	1	1	1	4	1	1
14	2	2	4	4	4	4
15	1	1	1	4	4	4
16	2	4	1	4	2	4
17	1	1	1	4	1	4
18	1	1	1	4	4	4
19	1	1	2	4	4	4
20	1	1	1	4	1	4
21	1	1	1	4	1	4
22	1	1	2	4	1	4
23	4	4	4	4	4	4
24	1	1	1	4	2	4
25	2	2	2	4	4	4
26	1	1	1	4	1	4
27	2	2	2	4	1	4
28	1	1	1	1	1	1
29	1	1	1	4	4	4
30	1	1	1	4	1	1
31	2	1	2	4	4	4
32	1	1	1	2	1	2
33	2	2	2	4	2	4
34	1	1	1	4	1	2

35	1	1	1	4	1	4
36	1	1	1	4	1	2
37	1	1	1	4	1	4
38	1	1	1	4	1	1
39	1	2	2	4	2	4
40	1	1	1	4	2	4
41	1	1	1	4	2	4
42	2	2	3	4	4	4
43	1	1	1	4	2	2
44	2	2	4	4	4	4
45	1	1	1	4	2	4
46	3	3	4	4	4	4
47	1	2	2	4	2	4
48	1	1	1	4	1	1
49	1	1	2	4	4	4
50	1	1	1	4	2	2
51	1	1	1	4	1	1
52	4	4	3	4	4	4
53	2	2	2	4	2	2
54	1	1	1	4	1	1
55	2	2	2	4	2	4
56	1	1	1	4	1	1
57	4	4	4	4	4	4
58	1	1	1	4	1	1
59	1	1	1	4	1	1
60	1	1	1	1	1	1
61	1	1	1	4	1	1
62	1	1	1	4	1	1
63	1	1	1	1	1	1
64	2	2	2	4	2	4
65	1	2	1	4	1	1
66	1	1	1	4	1	1
67	4	4	4	4	4	4
68	1	2	1	4	1	1
69	2	2	1	2	1	1
70	1	1	1	4	1	1
71	2	2	1	4	1	1
72	1	1	1	2	1	1
73	1	1	1	4	1	1
74	2	2	1	2	1	1
75	1	2	1	2	1	1
Complete	49	46	49	3	37	24
Partial	17	19	13	6	17	8
False positive	3	1	3	0	0	0
No pickup	6	9	10	66	21	43



Accuracy	65.33333333	61.33333333	65.33333333	4	49.33333333	
Day						
Complete	33	33	30	1	18	5
Partial	11	10	10	2	14	7
False positive	3	1	2	0	0	0
No pickup	3	6	8	47	18	38
Night						
Complete	16	13	19	2	19	19
Partial	6	9	3	4	3	1
False positive	0	0	1	0	0	0
No pickup	3	3	2	19	3	5