University of Birmingham

School of Engineering
Department of Mechanical Engineering

Individual Engineering Project

FINAL REPORT

BEng

Surname	Bouziotis
First Name(s)	Odysseas
ID number	2205473
Supervisor’s Name	Dr. Amir Hajiyavand
Project Title	A Comparison of YOLOv5 Models: Application to Recognition of Stair Lift Components

Table of Contents

List of Abbreviations

Abbreviation	**Definition**
YOLO	You Only Look Once
mAP	mean Average Precision
ANNs	Artificial Neural Networks
CNNs	Convolutional Neural Networks
RGB	Red Green Blue
SSD	Single Shot Detectors
Faster R-CNN	Faster Region-Based Convolutional Neural Network
IoU	Intersection over Union
GPU	Graphing Processing Unit

List of Figures

List of Tables

Acknowledgements

1 Dr. Amir Hajiyavand (Supervisor)

2 Jiaqi Ye (PHD Student)

Abstract

Target detection models are widely utilized to recognize desired components and features in an image. The computer vision field and object detection could thrive in the industrial environment by executing inspection tasks for automating the manufacturing process of medical devices. In this thesis, a comparison of YOLOv5n, YOLOv5s, YOLOv5m,YOLOv5l, and YOLOv5x for the recognition of stair lift components is performed. The pictures of the stair lift were captured using a low-quality mobile phone camera and were imported into Roboflow to prepare the data set appropriately. The training of the network took place in GitHub where all the different YOLOv5 architectures were implemented. All the trained models were compared against the mean Average Precision (mAP), precision, recall, and loss functions such as box loss, classification loss, and objectness loss. After they were all trained and evaluated on their respective test sets, they recorded accuracies above 97%. Meanwhile, the mAP at 0.5 for all the architectures did not drop below 98%. YOLOv5x was the best model in regards of accuracy but required the most computational time. Therefore, YOLOv5n offered a quality trade-off between accuracy and computational time. Limitations and future developments of Convolutional Neural Networks (CNNs) are also presented and discussed to provide a comprehensive understanding of the topic.

1 Introduction

1.1 Background

The classification and localisation of things in visual data is referred to as object detection, which is a key task in computer vision. The development of algorithms and techniques for processing, interpreting, and comprehending visual data from the real world is the focus of the interdisciplinary field of computer vision [1]. Object detection could help revolutionize the manufacturing processes of medical devices by increasing automation [2]. Due to the recent advancements in computer vision, the utilization of target detection models in industry would constitute a beneficial way of reducing potential costs as they would provide a reliable way of performing inspection tasks that are currently being done manually. By streamlining the production line of the product and by performing quality controls the overall productivity would be improved [2]. Finally, decreasing human intervention would drastically increase the output quality of the overall process.

Object detection techniques use Artificial Neural Networks (ANNs) to extract features from an image or a video. The architecture of Artificial Neural Networks was directly inspired by the mammalian visual cortex [1]. The human brain for example, has billions of neurons that are partially connected with each other but depending on the stimuli, the according ones are activated. The ANNs have been designed to follow and imitate this structure. A typical model is consisted of the feature extractor, fully connected layers, and classifier (Figure 1) to manipulate the input data. Although they have been proven able to identify the desired features and components in the visual data, they encounter numerous limitations due to the structure of their architecture. Those limitations include overfitting and lack of identification when the objects have different spatial relationships [3].

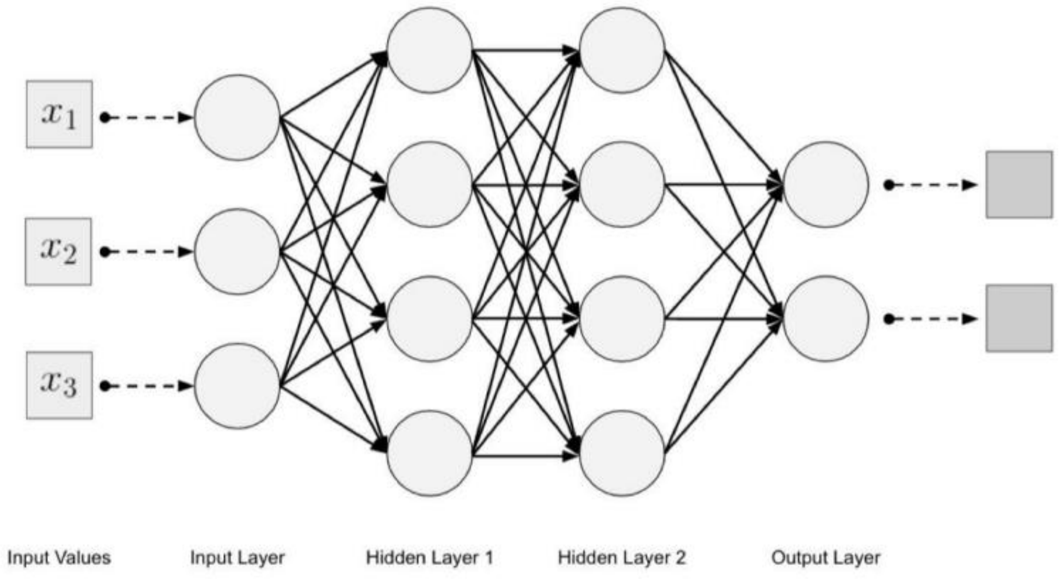


Figure 1: A Typical Neural Network Schematic [1]

Convolutional Neural Networks (CNNs) were introduced by LeCun Y, to account for spatial relationships and overfitting [4]. They address the former by applying different techniques such as incorporating receptive fields and parameter sharing [5]. Meanwhile to account for overfitting, regularization techniques such as dropout and data augmentation are implemented to reduce the parameters [6]. The main difference between traditional neural networks and convolutional neural networks is during the feature extractor region or hidden layers where the neurons of the latter are not fully connected to the next layer [1]. CNNs are using convolutional layers to perform the mathematical operation of convolution, pooling layers to reduce the dimensionality of the data and provide translational invariance. In the figure below the typical configuration of such a network is given. The hidden layers are followed by fully connected layers, and then a classifier which gives the final detections. In this thesis, the You Only Look Once (YOLO) algorithm and the Roboflow platform were implemented to create, train, and evaluate a data set which includes 160 stair lift images.

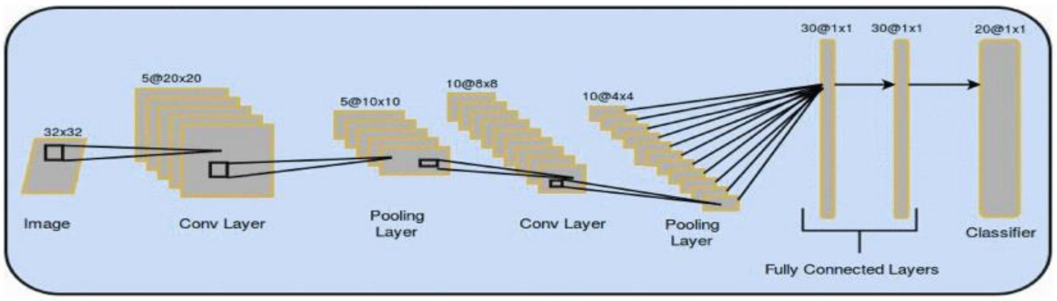


Figure 2: Typical Architecture of a Convolutional Neural Network [6]

1.2 Aim & Objectives

Aim:

- To develop a robust approach to compare the YOLOv5 architectures for recognizing stair lift components.

Objectives:

- To create a data set including images with different angles, orientations, and brightness levels.
- To train, validate, and test the data set.
- To evaluate each YOLOv5 sub-architecture against the performance metrics and loss functions.

2 Methodology

2.1 The Model's Architecture

The YOLO architecture is based on Convolutional Neural Networks. In literature several variations of CNN's architecture can be found that constitute different models such as LeNet and GoogLeNet [5]. YOLO is a type of deep learning algorithm and as with all CNNs, is composed of hidden layers (or feature extractor region), fully connected layers and classifier. The input image is a square matrix in Red Green Blue (RGB) form, which gives an additional depth of three; This results in transforming the visual data in a 3-dimensional matrix before being fed forward to the convolutional layers [7]. To extract features and information from the input, filters or kernels are used to convolve around the input matrix. They simply perform the mathematical operation of convolution and combine two functions into one. Multiple filters are used throughout each layer to detect edges, corners, and small patterns. An example of the abstract features learnt by the filters is shown in Figure 3.

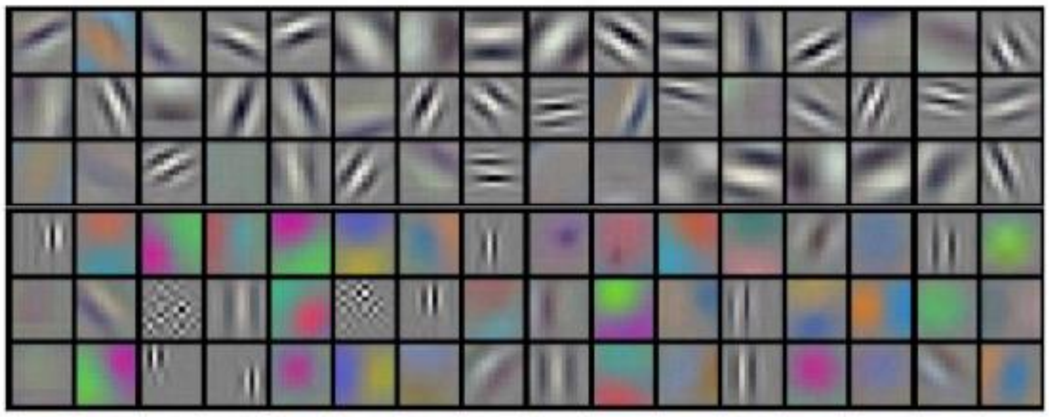


Figure 3: Example of Patterns Learnt during the Hidden Layers [1]

When the kernels convolute around the input data, they generate high numerical values at the place where there is a significant change in pixel intensity across bordering pixels, which interprets to a potential edge or corner. Therefore, as they progress through the hidden layers, the model is able to detect more abstract features and shapes such as the wheel of a car. The operation of convolution is shown in Figure 4 where the filters use a method called stride to slide across the input matrix. A stride of two shifts the filters two rows and two columns at a time after the completion of each computation. As the filter approaches the edges of the input matrix it could result in loss of information, so a technique called zero padding is utilized to prevent that by adding rows and columns of zeros to the input's dimensions (figure 4). Also, a bias term is introduced in each kernel and through an algorithm called backpropagation, the value of the bias is optimized. Backpropagation is defined as the gradient of the loss function. The value of the gradient computed is used to update the bias term in order to minimize the loss function and obtain better accuracy.

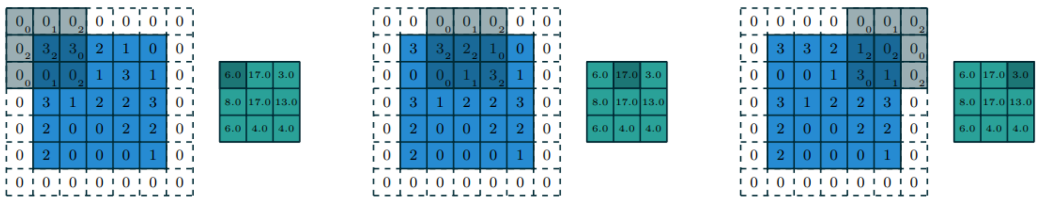


Figure 4: Convolution of a 3x3 Kernel with a Stride of two and Zero Padding [8]

The output of the filters is fed into an activation function to generate an activation map. Those functions serve the purpose of introducing non-linearity to the data; otherwise, the neural network would result in a linear regression model [9]. The activation function used in the YOLO algorithm is the Leaky ReLU due to its advantageous handling techniques when facing a plethora of parameters. If some of those parameters have a value less than zero, the function manages to keep them active and prevent the vanishing gradient problem or the gradient saturation [9]. The YOLO algorithm utilizes 3x3 kernels, a stride of two, zero padding, and the Leaky ReLU activation function to process the input image.

The convolutional layers are followed by pooling layers to downsample and reduce the magnitude of the data and parameters learnt, to account for overfitting. The pooling layers operate on the input activation maps from the previous convolutional layers. Several types of pooling exist such as average pooling, stochastic pooling, and max pooling that have slight variations in their respective output value [5]. The YOLO algorithm implements the max pooling to keep the pixels with the highest numerical value and intensity while reducing the size of the data. This whole process of convolutional layers followed by pooling layers is iterated and cascaded until it reaches the fully connected layers which marks the end of the feature extractor region.

Finally, as in all ANNs, each neuron of the fully connected layers connects to every other neuron in the next layer (see Figure 1); These layers are responsible for predicting and classifying the output of the hidden layers. During the training process, backpropagation and gradient descent update the weights and biases of the kernels and fully connected layers. As a result, the model can gradually increase its accuracy by learning to recognize various elements in the input images. A typical example of the final output with the corresponding detections for each class in YOLO can be observed in Figure 5.

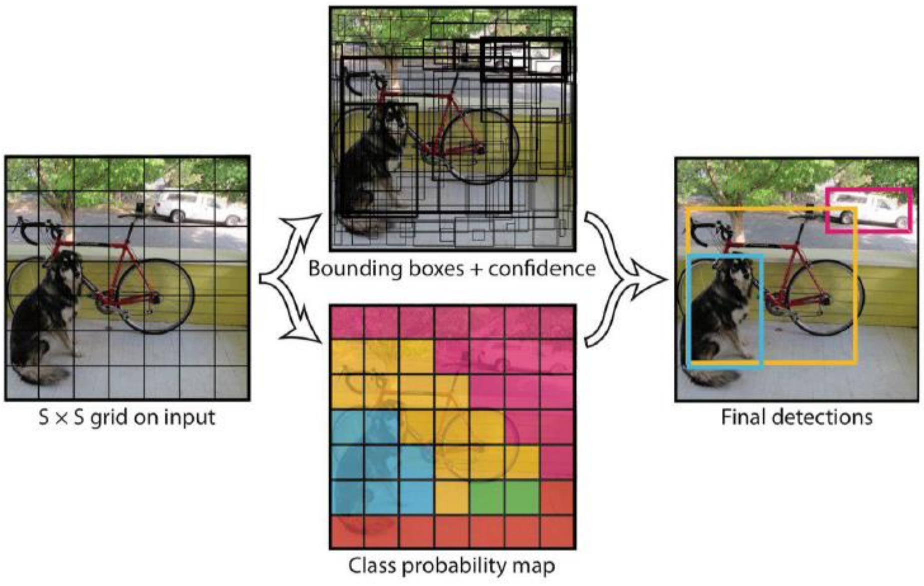


Figure 5: Object Detection in YOLO [1]

2.2 Data Set Acquisition

A set of 160 images were initially imported into Roboflow to apply image pre-processing and augmentation techniques. The visual data includes pictures of a medical device, in this case of a stair lift, and then were differentiated into four distinct classes such as batteries, motor, controller board, and the whole power pack. The stair lift components can be observed from figure 6.

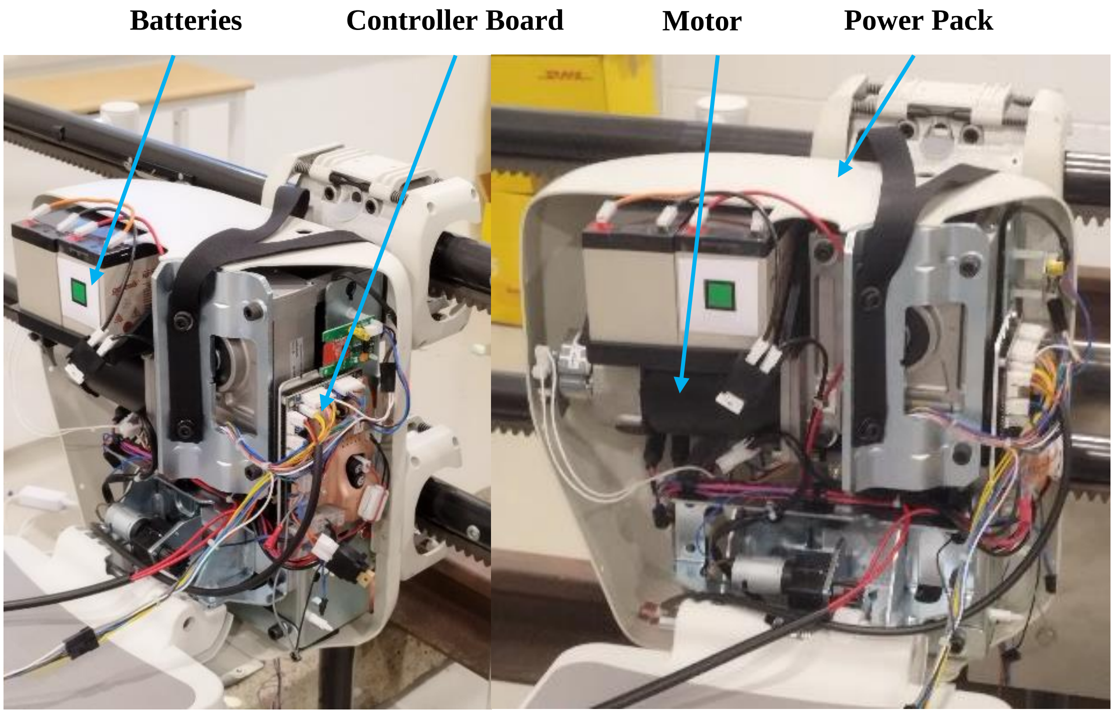


Figure 6: Examples of the Visual Data Set

Roboflow splits the images into three subsets; training set, validation set, and test set. Those subsets had 80%, 10% and 10% respectively out of the total 160 captured images. However, before being split into those subcategories all the images were annotated, and bounding boxes were placed as closely as possible to each respective class. Thus, during training hyperparameters such as neurons in each layer, number of hidden layers, learning rate, batch size, and activation functions, were optimized by the model to maximize its accuracy. The main scope of the training set is to be able to generalize when encountering new unseen images. Some classes could be shifted, rotated, and have different spatial relationships, but when training is done correctly identification and localization is accomplished.

The validation set is responsible for evaluating the model's performance during training after each epoch or in other words after each complete pass through the images. It is also in charge of selecting the appropriate hyperparameters, a process called hyperparameter tuning [10]. By selecting appropriate hyperparameters, overfitting gets prevented and the model is able to generalize and not focus on the data too closely. Once the validation is completed the model utilizes the test set to make predictions and to calculate performance metrics such as mean Average Precision (mAP), precision, and recall. The visual data set after arranging it into those subsets needs further processing before training in YOLOv5 takes place.

Inside the Roboflow platform the pre-processing techniques that were followed started by resizing the images to 640x640 in order to be in a multiple of 32. The background idea behind changing the resolution is due to the downsampling factor of 32 that YOLO has integrated into its architecture as it divides the image into a grid of 32x32 cells. Therefore, by resizing the images in a multiple of 32 all the cells are equally distributed while occupying the same amount of area to achieve efficient processing. Finally, when the down-sampling procedures take place the dimensions of the input image are evenly decreased without having any inconsistencies [10]. Auto orient and grayscale filters were also applied during the pre-processing techniques. Their selection was based on automatically applying the correct orientation that the image was supposed to be displayed and on removing the colour to reduce the unnecessary information that it might add. The last step before training in YOLO was the implementation of the augmented pictures into the data set. After augmenting the data set it reached a number of 500 images; The pictures that were added to the original number of 160 were automatically distributed in the training set rather than the validation or test set to improve the quality and robustness of the model. The features that those images had were blur, noise, and different brightness levels. After the completion of augmentation, Roboflow has prepared the visual data set to be imported to the YOLO algorithm in GitHub.

2.3 Training in YOLOv5

In literature there are multiple ways of performing object detection such as Single Shot Detecotrs (SSD), and Faster R-CNN but the YOLO algorithm was selected due to better solving speed and accuracy [11]. Also, YOLOv1 is one of the first models that were launched and since, there have been numerous upgrades with the latest version being YOLOv8. In this thesis, YOLOv5 was implemented to train the visual data set created on Roboflow. There are five smaller sub versions of YOLOv5 that indicate the size of the backbone of the convolutional layers. Specifically, YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x are the pretrained models. The different subscripts represent nano, small, medium, large, and extralarge versions of YOLOv5 respectively. For instance, the backbone of the nano version has the least connections between each neuron leading to fewer parameters learnt during training and for that reason is the fastest and requires the least computational power. All models were utilized to train the data set, and their performance was compared to understand which is the optimum for this application.

In the GitHub environment the batch size describes how many samples or data points a model can process at once and essentially divides the data set into fewer parts. For this visual data set it was specified to be 16. When choosing an incorrect batch size, the updated weights could initiate noise and slow down the convergence rate or even completely prevent the model to solve [12]. As it was discussed previously the appropriate image size was chosen to be in 640x640 square form as per the YOLO requirements of being in a multiple of 32. To accomplish effective model performance 150 epochs were specified giving an ideal number of complete passes through the data. Otherwise, the model might experience underfitting and not have enough parameters to learn from or overfitting where it continues to learn from the noise. There was an option for the user to specify and initialize weights, but it was not recommended from the YOLO algorithm and GitHub thus it was left at its predefined settings. The data were already in the correct format from the Roboflow platform. Finally, the model was ready to proceed to training and the visualization of the results is presented in the next section.

3 Results

In this section, evaluation of all the different versions of YOLOv5 was conducted and recorded through the diagrams and tables presented below. For each version, the same methodology was implemented including the same visual data set and annotations to perform a valid comparison. After the completion of each simulation, metric graphs were provided by GitHub for each individual performance. As in every object detection application, the most important parameters to track and assess the model's effectiveness are namely the accuracy, precision, and recall. Additional graphs of box loss, classification loss, and objectness loss were given to obtain a broader understanding of the model's resilience. All the generated graphs presented below concern the YOLOv5n architecture while the other results for YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x are documented in the appendix.

The mean Average Precision (mAP) of the nano architecture approached a value of 0.75 (Figure 7 b)); This metric identifies how confidently the trained model is able of detecting the different classes as it is an average of the maximum precision over the various values of recall. It could be observed that there is another graph of mAP given in figure 7 a) below. It represents the mAP where the Intersection over Union (IoU) is above 0.5 or in other words the overlap between the bounding boxes over the whole area of union exceeds 50%. While in Figure 7 b) it computes the accuracy over different IoU values falling inside a range of 0.5-0.95. From the precision graph (Figure 7 c)) the derived magnitude of 0.99 was obtained and represents the true positive over the sum of the true and false positives. From the recall diagram (Figure 7 d)) a metric value of 0.97 was obtained. This performance metric shows how many classes the model was able to recognize over the overall classes appeared in the images.

The additional loss function graphs of box loss, classification loss, and objectness loss that were computed during training are depicted in figure 8. The box loss recorded a value less than 0.023 and its purpose is to indicate how well centralized the bounding box is relative to the detected object as well as how adequately it encircles it (Figure 8 a)). The model during training uses the test set to predict and label each class in an image. The classification loss computes the performance and accuracy of the predictions and label tasks. In figure 8 b) the final value of 0.0045 for the classification loss function was obtained. Finally, from figure 8 c) the objectness loss function can be observed. The result acquired from the graph was 0.018 where the function concentrates at a place of interest and calculates the possibility of the class existing in that area. All the performance metrics and all the loss function values for each respective version of YOLOv5 were recorded in Table 1 and Table 2 below to aid the reader visualize and comprehend the data. Figure 9 shows the final bounding boxes with additional confidence scores for each class that the model was able to recognize and localize while going through the test set. In the appendix the final predictions for the other architectures can be observed in figures 13, 16, 19, and 22.

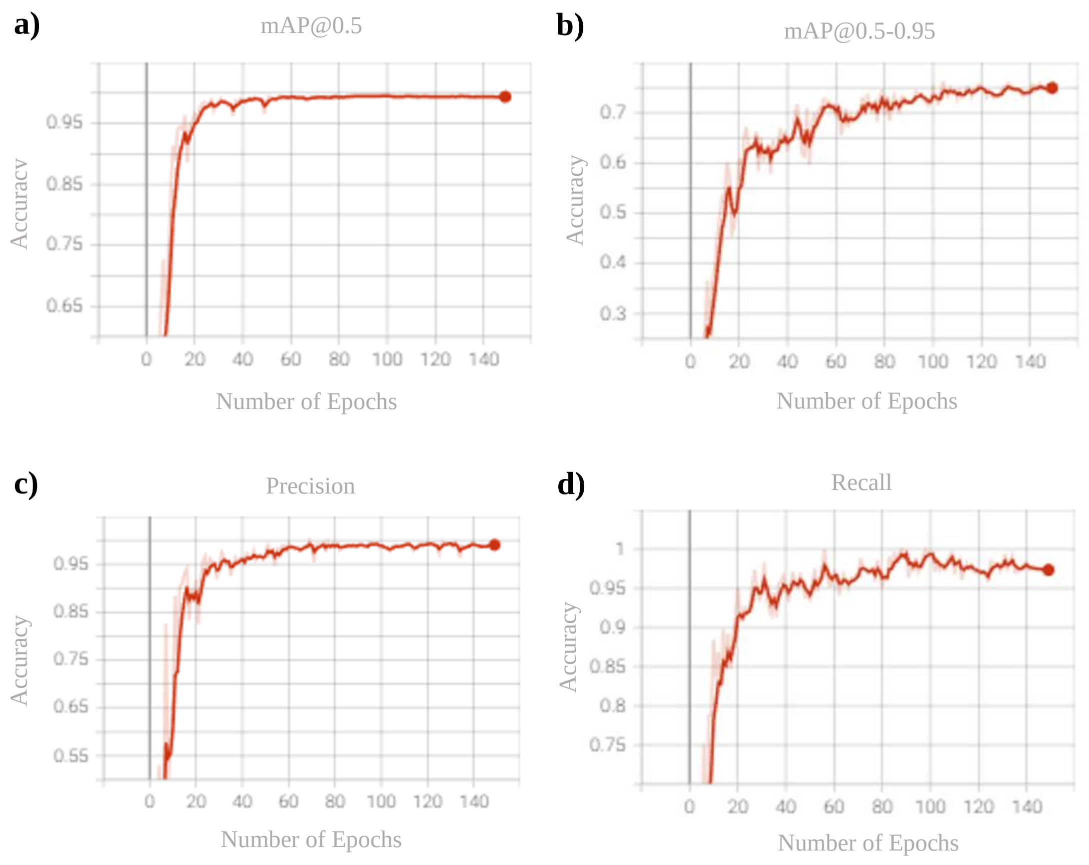
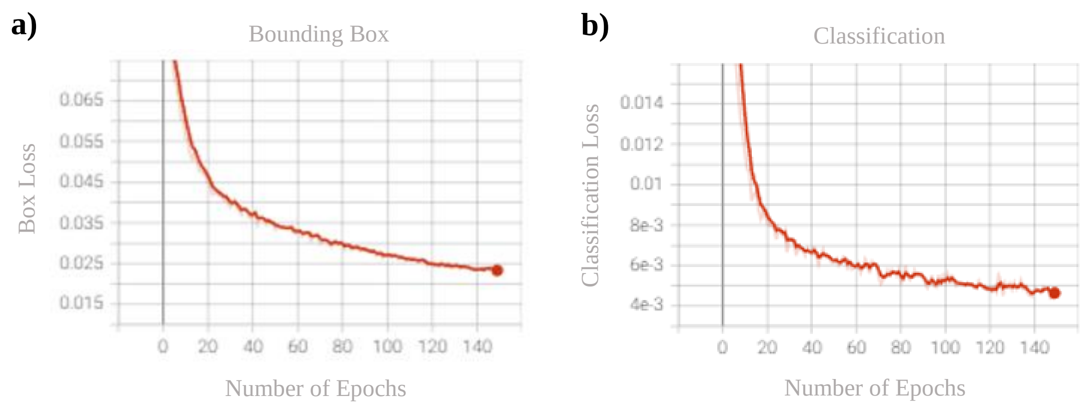


Figure 7: Performance Metrics Graphs



c)

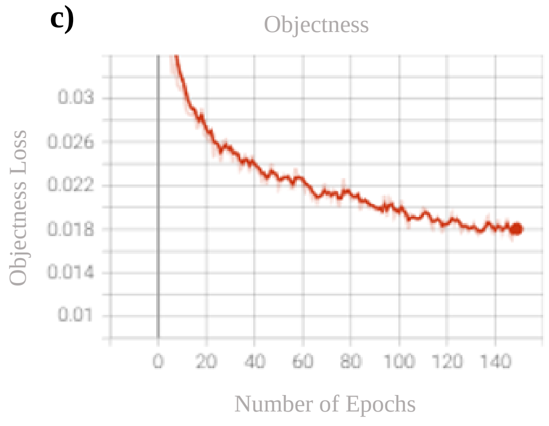


Figure 8: Loss Function Graphs

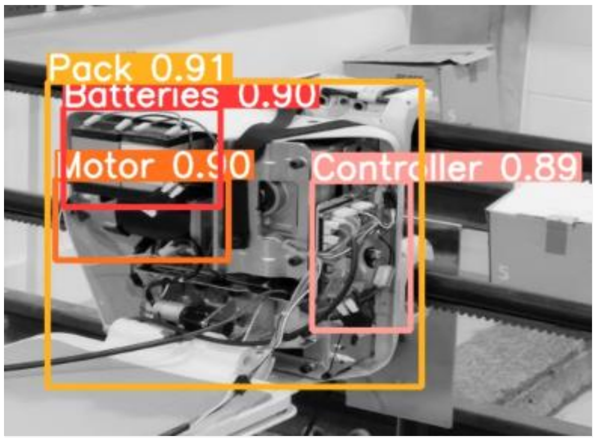


Figure 9: YOLOv5n Final Output after the Detection was Completed

Table 1: Performance Metrics Calculations for all YOLOv5 Architectures

Model	mAP_0.5	mAP_0.5:0.95	Precision	Recall
YOLOv5n	0.98	0.75	0.99	0.97
YOLOv5s	0.99	0.74	0.99	0.96
YOLOv5m	0.99	0.75	0.99	0.99
YOLOv5l	0.98	0.74	0.99	0.98
YOLOv5x	0.99	0.75	0.99	0.99

Due to the different architecture of each model, while increasing up in size, the neurons inside the convolutional layers have more connections between them resulting in requiring more computational time during training. In Table 2 the computational time recorded for each simulation is presented accompanied by the values of each loss function.

Table 2: Loss Functions Results and Required Computational Time for each Model's Training

Model	Box Loss	Class Loss	Object Loss	Comp. Time
YOLOv5n	0.023	0.0045	0.018	14.4 mins
YOLOv5s	0.019	0.004	0.0145	19.05 mins
YOLOv5m	0.015	0.0038	0.011	25.02 mins
YOLOv5l	0.014	0.0036	0.009	36.96 mins
YOLOv5x	0.0125	0.0035	0.0085	68.04 mins

The YOLOv5n architecture was reasonably the fastest as it required the least computational time of less than a quarter of an hour due to the fewer neuron connections. The increase in computational time saw an exponential growth from the smaller up to the most advanced architecture where for YOLOv5x it exceeded an hour for it to solve.

4 Discussion

In this section a critical analysis of the results obtained from the YOLOv5 simulations is conducted to inform the reader which architecture would be best suited for applications with a small data set. Also, to comprehend and comment on existing limitations that object detection models experience that affect their performance and accuracy. By reviewing those limitations additional areas of improvement were identified in the current methodology and future developments were proposed to account for that.

4.1 Critical Analysis

It can be observed from Figure 7 that all the graphs for the performance metrics had an initial sharp increase and eventually converged to their final value with some low magnitude fluctuations. For the mean Average Precision with Intersection over Union above 0.5 the respective graph depicts a steep rise followed by an almost flat line until settling down to the final value (Figure 7 a)). On the other hand, the graph of mAP with IoU between 0.5 and 0.95 had a significant jump in the first 10 epochs and then experienced a gradual growth with constant oscillations up until the end (Figure 7 b)). The graph profile of precision and recall metrics was similar as they both encountered a low magnitude volatility and a constant upsurge throughout, while they recorded peak values of 0.99 and 0.97 respectively (Figure 7 c), d)). From Table 1 the values for each parameter were tabulated for all the respective YOLOv5 models. It is noticeable from the results obtained that there was no significant difference in the values but rather a 1-3% change. This is indicative that YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x can equally produce reliable results for the given data set. The model that stands out regarding the accuracy was the YOLOv5x which coincides to have the most connections between its neurons in the convolutional layers. According to the model, during training the YOLOv5l and YOLOv5x were completed in less epochs than the assigned ones as there were no additional features to be learned. The simulation did not execute the full 150 epochs as it was a precaution measure to prevent overfitting. Therefore, when implementing the bigger architectures to small data sets there is a high chance of generating unreliable results.

From figure 8 the loss functions play a significant role in evaluating the error and hence the model's accuracy. It is visible from Table 2 that YOLOv5n achieved the highest values regarding the loss functions. However, the loss functions compute the difference between the actual and the true output or in other words obtaining high values translates to poor performance. On the other hand, YOLOv5x attained the best performance by having the lowest magnitude for each corresponding loss function. Hence, it was better in predicting, localizing, and centralizing compared to its counterparts. Despite the high accuracy the trade-off between that and the computational time was substantial and needed to be taken under consideration. It is apparent from Table 2 that as increasing in size more computational time is consumed for the execution of the simulations. YOLOv5n took almost a quarter of an hour to finalize while YOLOv5x exceeded an hour for it to converge. The difference in the metric results is almost negligible while the time that the simulation was performed increased by 372.5% from the smaller to the bigger architecture. Since all models were implemented to the same data set and had insignificant change in accuracy, the increase in computational time was considered as the main factor of selecting the model with the best trade-off. Therefore, for the current and further applications of similar magnitude, YOLOv5n is the ideal architecture to be utilized.

4.2 Limitations of Convolutional Neural Networks (CNNs)

There is a plethora of aspects that can have a negative influence in the effectiveness of an object detection architecture and lead to generating biased and false results. However, depending on the application, different constraints are expected. Therefore, in this section the controlled environment of a production line was taken under consideration to analyze and present the expected limitations, where the model is utilized to perform inspection tasks for the automating manufacturing process of a stair lift.

The computational time required for each inspection poses the most significant issue; When the model performs instantaneous detections combined with high accuracies it can eventually boost the overall productivity and effectiveness of the production line. Thus, to meet the high-speed requirements of the inspection tasks the deployment of more advanced hardware is necessary. The implementation of more powerful Graphics Processing Units (GPUs) can increase the computational power capabilities of the model and eventually reduce the time of the detections [13]. To account for high accuracies the equipment of cameras that are utilized need to provide quality images with higher resolutions to minimize the potential noise initiated.

In scenarios where the inspection tasks need to check the product from different viewpoints the model is required to be flexible and adaptable. During the generation of the data set the images captured need to represent the classes from the range of different angles and orientations that they are expected to be seen in the production line. Therefore, for a robust model translational invariance is necessary as image recognition must take place regardless of the position and rotation of the object. CNNs provide limited flexibility of translational invariance but resolve that issue during training by applying receptive fields and a technique called parameter sharing [5].

Another challenge that object detection architectures encounter is the spatial relationships which refer to the correlation of objects and their location in three-dimensional space [5]. The various spatial relationships between the objects need to be depicted and learnt through plenty of images for the model to understand how the detected object is related to the other classes in terms of location. Otherwise, the output results might include misclassifications or false positives and false negatives. Diversity and variety in the visual data set helps define spatial relationships better and provides resilience.

Further limitations encountered in the application of computer vision in inspection tasks is the detection of small objects. The YOLO algorithm finds it challenging to identify and recognize fine details and relatively small components which could lead to the approval of a low-quality product with missing parts [14]. In the case of automating the manufacturing process of medical devices small objects would be bolts and nuts that are crucial for the product's functionality. Therefore, for the inspection of small components data augmentation techniques such as cropping and scaling need to be applied to mitigate the limitation.

In environments where there is significant visual complexity and cluttered scenes the performance of the model is affected. It is noticeable from the disassembly of the stair lift presented in Figure 10 that the existence of small cables and features scattered around lead to crowded scenes. As a result, when encountering cluttered scenes there is a possibility of producing false positives and false negatives. The former falsely detects a class as present in an image and the bounding box encircles a region with no important information or encircles an object but recognizes it as a different class from the assigned one. On the other hand, false negatives ignore the existence of a depicted class or completely misclassifying it which introduces error and lowers the accuracy. Both false positives and negatives could be alleviated by adjusting the confidence value thresholds. To further mitigate the cluttered scenes the background of the production line during inspection should minimize features that distract and make the model interact with the surroundings.

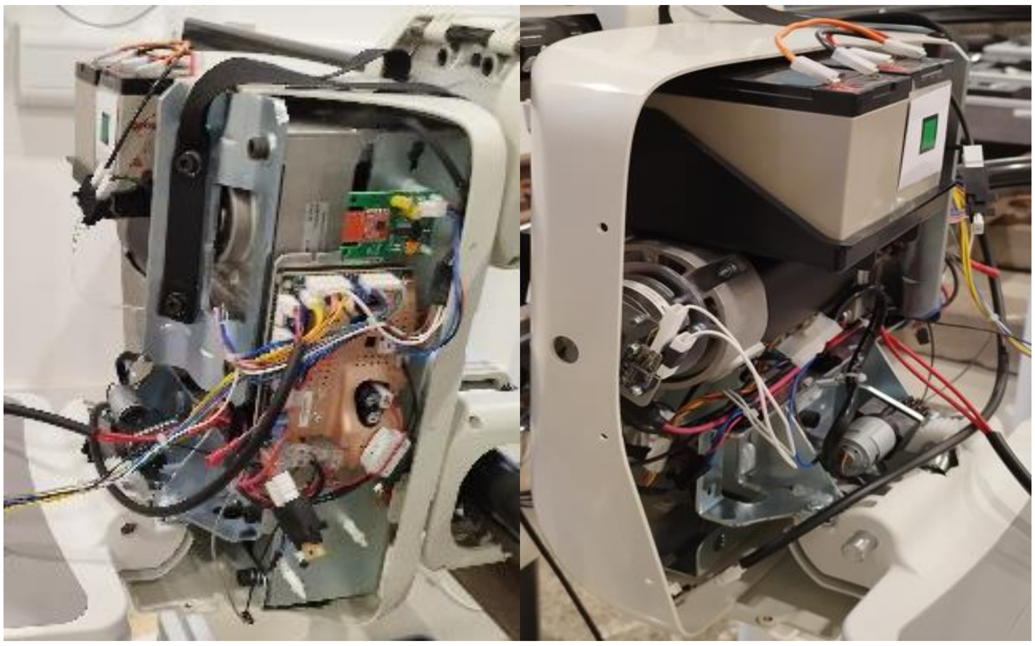


Figure 10: Cluttered Scene

4.3 Future Developments

After reviewing the methodology that was followed for the comparison of the YOLOv5 architectures certain areas of improvement were identified. The data set utilized after the augmentation techniques gathered a total of 500 images while in literature typical numbers of images reach up to 10,000 [10]. To obtain a robust model the first step would be to increase its size and include a variety of different viewpoints of the detected components to prevent overfitting. Proposed by Sergey loffe and Christian Szegedy batch normalization is a beneficial method of improving the model's resilience by normalizing the input of the layers [12]. In other words, the input values get rescaled and re-centred which can result in faster and more accurate computations. According to 'Peiyuan Jiang' by applying batch normalization techniques the mean Average Precision could increase by 2% as the convergence rate would also be faster [15].

In addition to that, multi-scale training is another approach for future developments where the data set includes images with different resolutions [15]. The background idea of resizing the images prior to being fed for training, is to provide adaptability so when deployed to different circumstances the model is not familiar with just a specific scale but rather with many. However, depending on the resizing of the images the training speed and accuracy might be affected. When the input has larger dimensions, it results in increased training time but better accuracy while smaller size interprets an opposite outcome obtaining a balanced trade-off between those two parameters.

5 Conclusion

The YOLOv5 algorithm is considered as a state-of-the-art architecture due to its speed and simplicity in computing high accuracy results. In this thesis, the targeted objectives were accomplished in order to perform the comparison of the YOLOv5 sub-architectures. The phone camera was utilized to create a data set of 160 images and by using the Roboflow platform it was split into training, validation, and test sets with 80%, 10% and 10% of the total images respectively. The training took place in the GitHub environment implementing all the YOLOv5 sub-architectures and providing all the necessary performance metrics to conduct a valid assessment.

The main findings of this comparison indicated that all models accomplished an adequate accuracy of 97% and above with a difference between 1-3% and were able to generalize when encountering new unseen images. YOLOv5x had the highest performance metrics when compared to the other architectures. However, it exceeded an hour for it to converge and solve and since it has the most neuron connections it also requires the most computational power. As the time increased by 372.5% from YOLOv5n to YOLOv5x the former provides an optimum balance between all parameters and is recommended for future applications of similar magnitude.

The convolutional neural networks need further development as they face numerous limitations and challenges that need to be addressed. Translational invariance and spatial relationships are existing issues just to name a few. In further applications the implementation of batch normalization and multi-scale training can facilitate an answer for those limitations and result in a more robust model. The field of computer vision and object detection was recently introduced and is still under development but seems to be very promising for applications such as inspecting the automating manufacturing process of medical devices.

References

1. Caschili D. Optimization of CNN-Based Object Detection Algorithms for Embedded Systems.

2. Ge C, Wang J, Wang J, Qi Q, Sun H, Liao J. Towards automatic visual inspection: A weakly supervised learning method for industrial applicable object detection. Computers in Industry. 2020 Oct 1;121:103232.

3. Yin C, Rosendahl L, Luo Z. Methods to improve prediction performance of ANN models. Simulation Modelling Practice and Theory. 2003 Jul;11(3-4):211–22.

4. LeCun Y, Bengio Y, Hinton G. Deep Learning. Nature. 2015 May;521(7553):436–44.

5. Ajit A, Acharya K, Samanta A. A Review of Convolutional Neural Networks. 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE). 2020 Feb;

6. Aloysius N, Geetha M. A review on deep convolutional neural networks. 2017 International Conference on Communication and Signal Processing (ICCSP). 2017 Apr;

7. Albawi S, Mohammed TA, Al-Zawi S. Understanding of a Convolutional Neural Network. 2017 International Conference on Engineering and Technology (ICET). 2017 Aug;1–6.

8. Murphy J. An Overview of Convolutional Neural Network Architectures for Deep Learning.

9. IMAGE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS [Internet]. [cited 2023 May 15]. Available from: https://www.theseus.fi/bitstream/handle/10024/276486/thesis%20-%20Hung%20Dao%20-%20final.pdf?sequence=2&isAllowed=y

10. PhD LGA. The practical guide for Object Detection with YOLOv5 algorithm [Internet]. Medium. 2022. Available from: https://towardsdatascience.com/the-practical-guide-for-object-detection-with-yolov5-algorithm-74c04aac4843

11. Kim J, Sung JY, Park S. Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition. 2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia). 2020 Nov 1;

12. Sergey Ioffe, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. PMLR. 2015 Jun;448–56.

13. Sood S, Singh H, Malarvel M, Ahuja R. Significance and Limitations of Deep Neural Networks for Image Classification and Object Detection. IEEE Xplore. 2021. p. 1453–60.

14. Yang X, Yang J, Yan J, Zhang Y, Zhang T, Guo Z, et al. SCRDet: Towards More Robust Detection for Small, Cluttered and Rotated Objects.

15. Jiang P, Ergu D, Liu F, Cai Y, Ma B. A Review of Yolo Algorithm Developments. Procedia Computer Science. 2022 Jan 1;199:1066–73.

Appendix

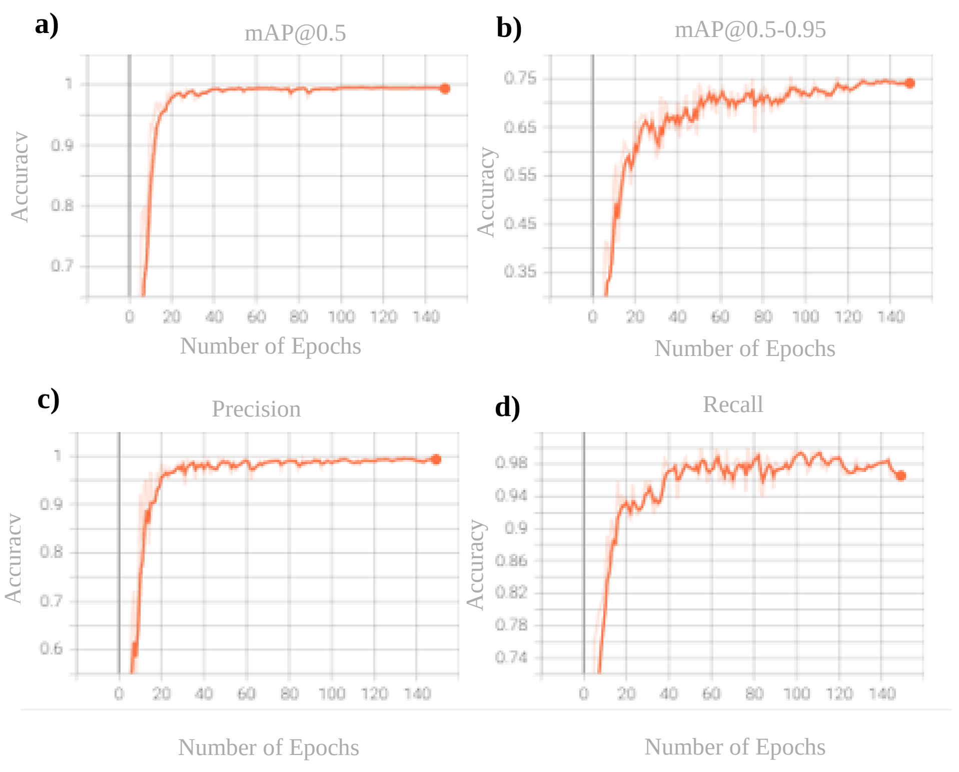


Figure 11: Performance Metrics Graphs for YOLOv5s

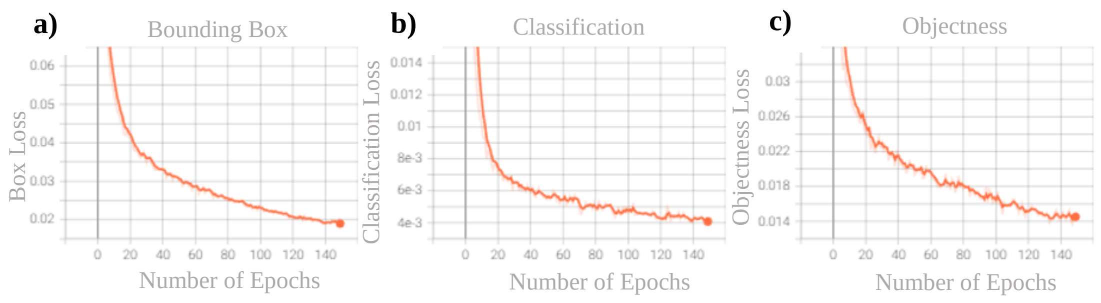


Figure 12: Loss Function Graphs for YOLOv5s

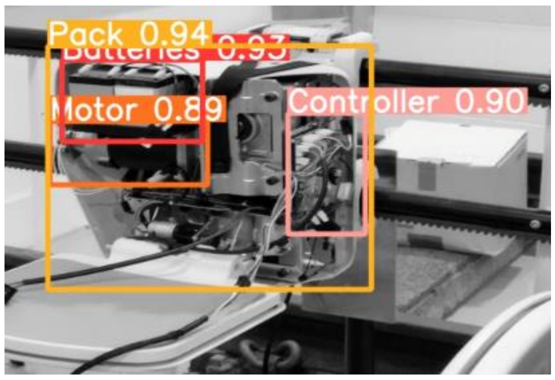


Figure 13: YOLOv5s Final Output after the Detection was Completed.

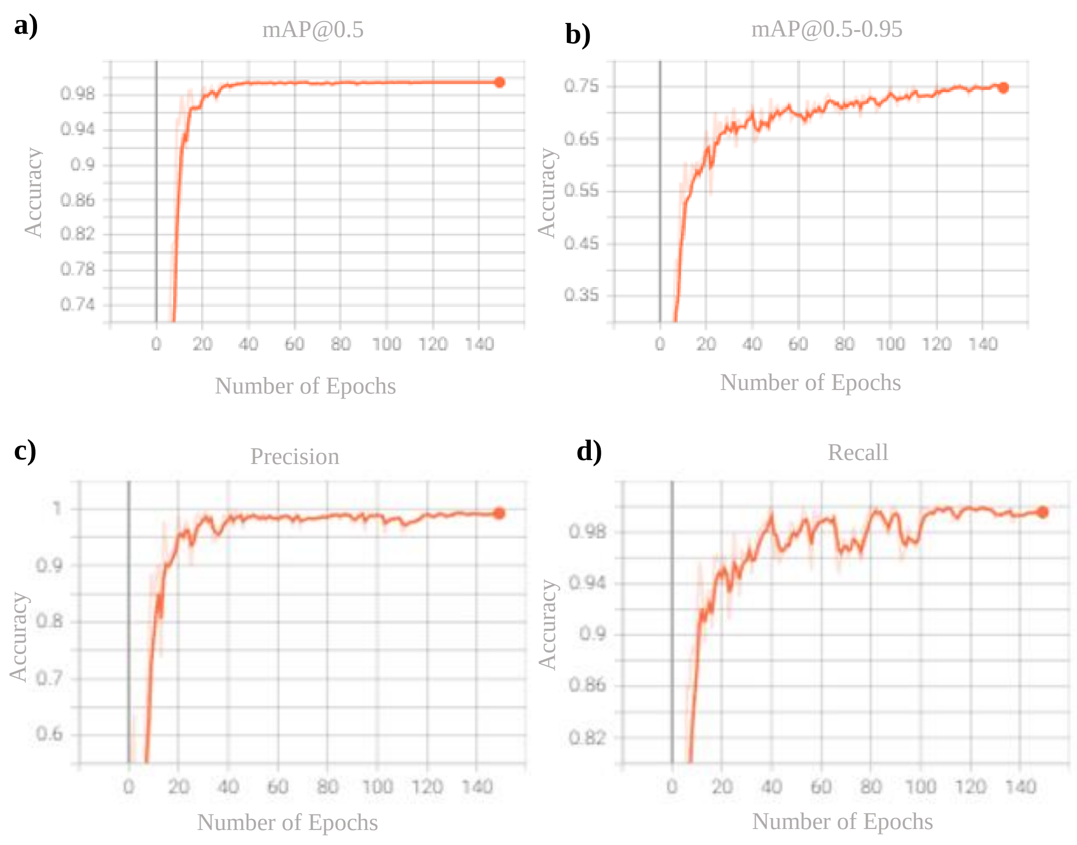


Figure 14: Performance Metrics Graphs for YOLOv5m

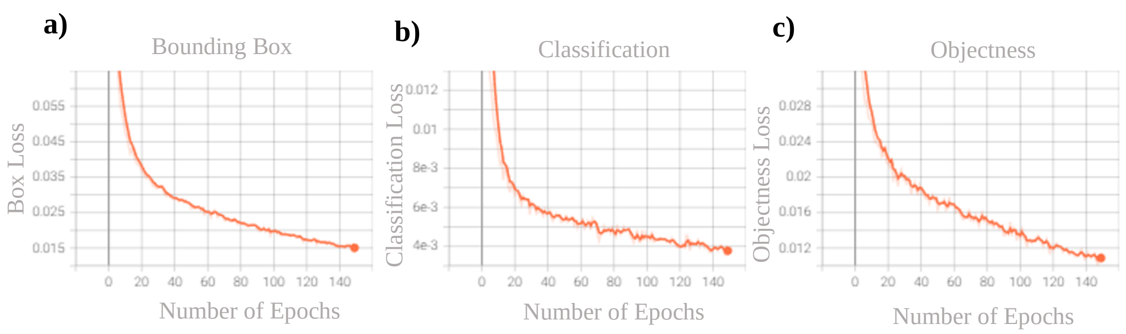


Figure 15: Loss Function Graphs for YOLOv5m

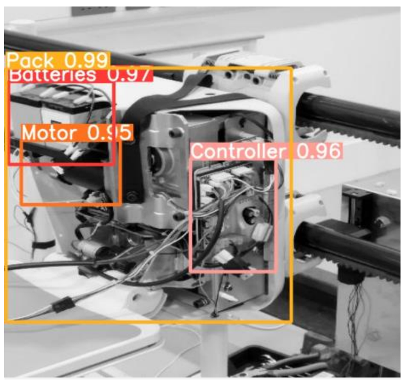


Figure 16: YOLOv5m Final Output after the Detection was Completed.

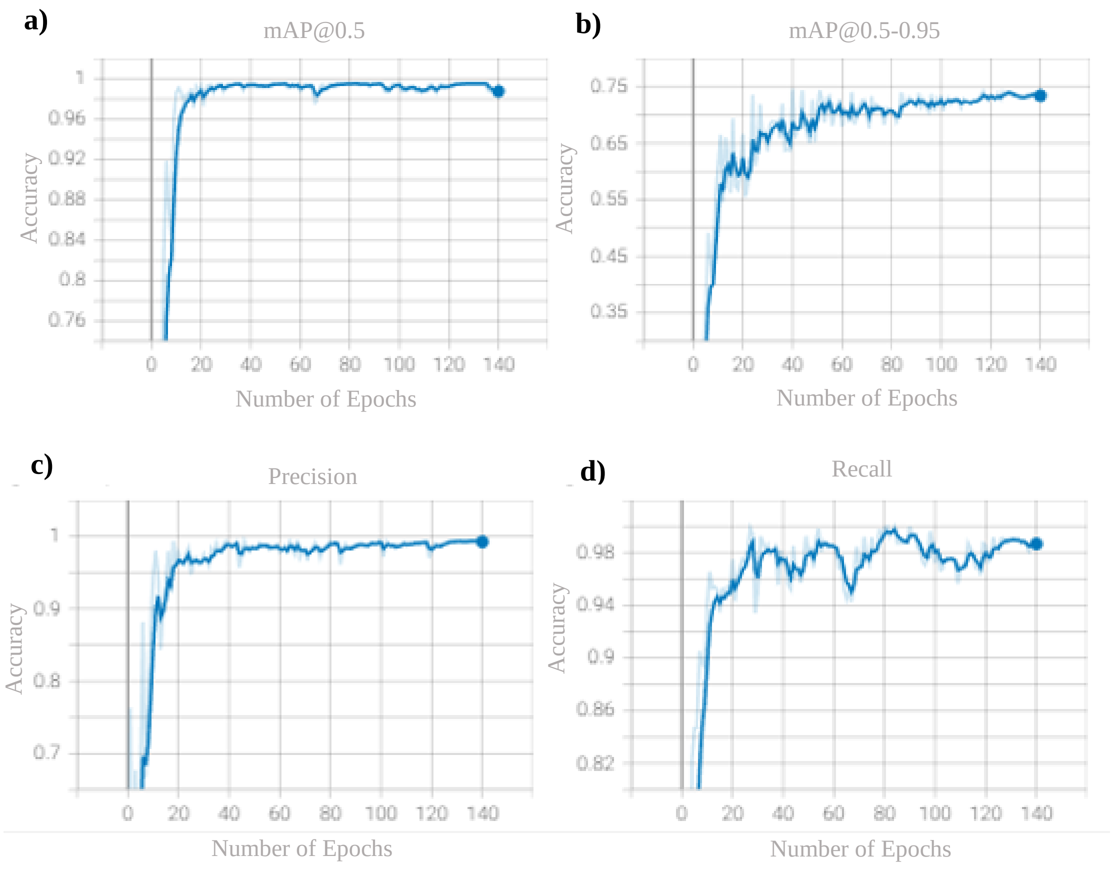


Figure 17: Performance Metrics Graphs for YOLOv5l

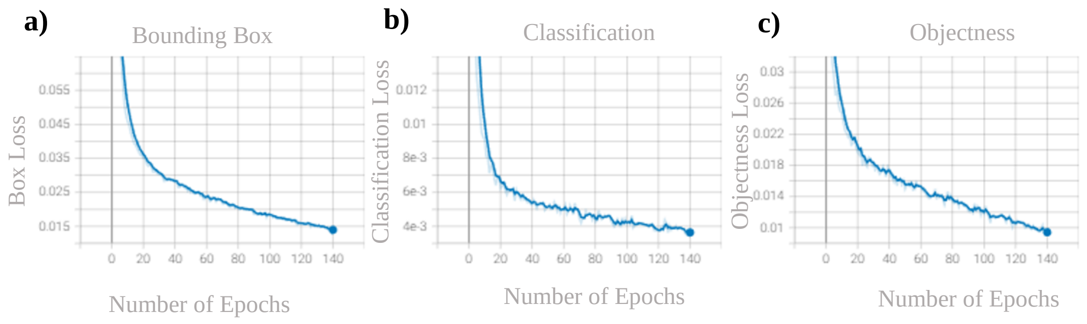


Figure 18: Loss Function Graphs for YOLOv5l

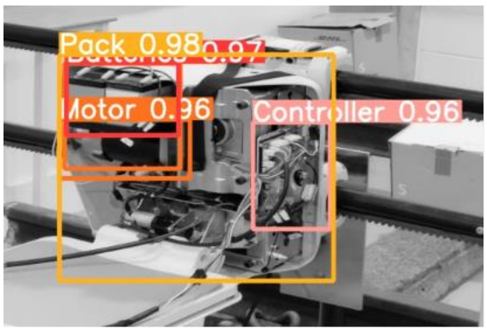


Figure 19: YOLOv5l Final Output after the Detection was Completed.

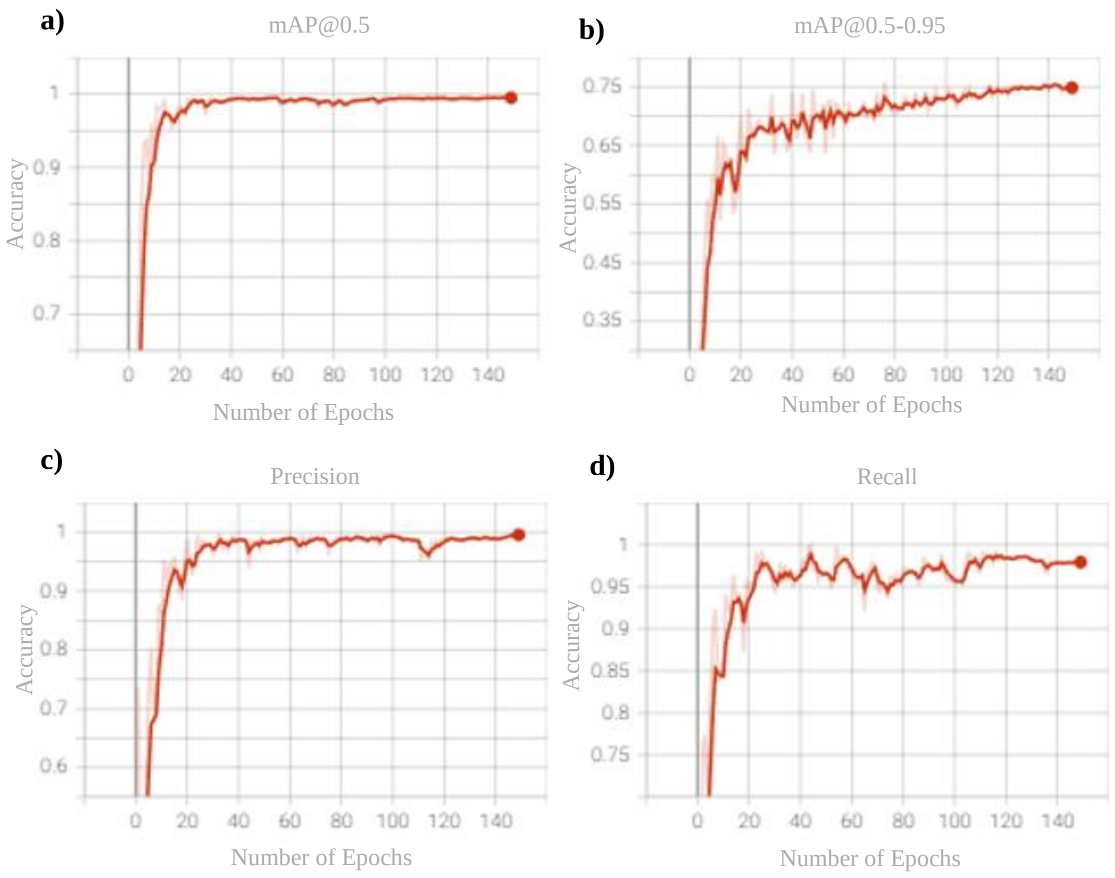


Figure 20: Performance Metrics Graphs for YOLOv5x

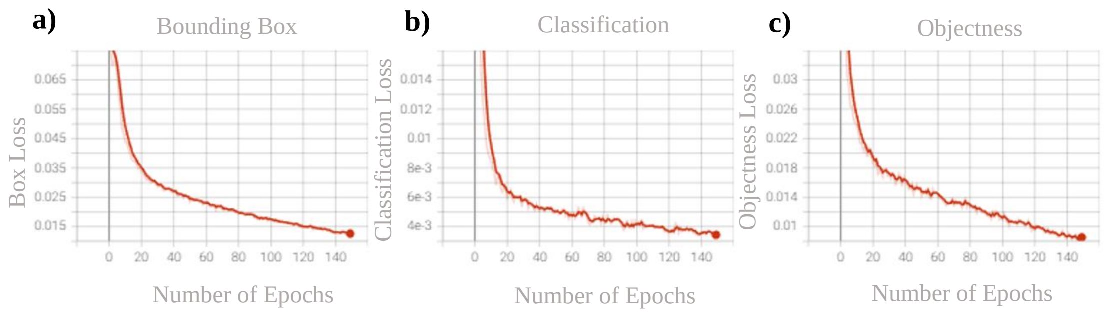


Figure 21: Loss Function Graphs for YOLOv5x

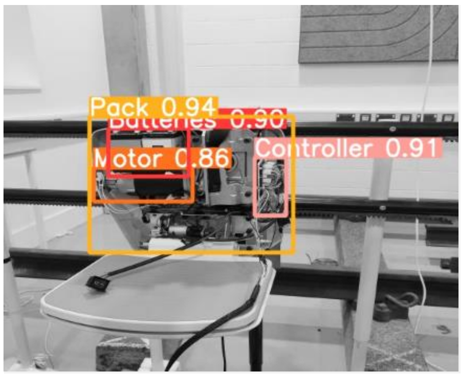


Figure 22: YOLOv5x Final Output after the Detection was Completed.