UPPSALA
UNIVERSITET

UPTEC STS 22015
Examensarbete 30 hp
Maj 2022

Railway Fastener Fault Detection using YOLOv5

Alva Efraimsson
Elin Lemón

Civilingenjörsprogrammet i System i Teknik och Samhälle

UPPSALA
UNIVERSITET

Railway Fastener Fault Detection using YOLOv5

Alva Efraimsson
Elin Lemón

Abstract

The railway system is an important part of the sociotechnical society, as it enables efficient, reliable, and sustainable transportation of both people and goods. Despite increasing investments, the Swedish railway has encountered structural and technical problems due to worn-out infrastructure as a result of insufficient maintenance. Two important technical aspects of the rail are the stability and robustness. To prevent transversal and longitudinal deviations, the rail is attached to sleepers by fasteners. The fasteners' conditions are therefore crucial for the stability of the track and the safeness of the railway. Automatic fastener inspections enable efficient and objective inspections which are a prerequisite for a more adequate maintenance of the railway.

This master thesis aims to investigate how machine learning can be applied to the problem of automatic fastener fault detection. The master thesis includes the complete process of applying and evaluating machine learning algorithms to the given problem, including data gathering, data preprocessing, model training, and model evaluation. The chosen model was the state-of-the-art object detector YOLOv5s. To assess the model's performance and robustness to the given problem, different settings regarding both the dataset and the model's architecture in terms of transfer learning and hyperparameters were tested.

The results indicate that YOLOv5s is an appropriate machine learning algorithm for fastener fault detection. The models that achieved the highest performance reached an mAP[0.5:0.95] above 0.744 during training and 0.692 during testing. Furthermore, several combinations of different settings had a positive effect on the different models' performances.

In conclusion, YOLOv5s is in general a suitable model for detecting fasteners. By closer analysis of the result, the models failed when both fasteners and missing fasteners were partly visible in the lower and upper parts of the image. These cases were not annotated in the dataset and therefore resulted in misclassification. In production, the cropped fasteners can be reduced by accurately synchronizing the frequency of capturing data with the distance between the sleepers, in such a way that only one sleeper and corresponding fasteners are visible per image leading to more accurate results. To conclude, machine learning can be applied as an effective and robust technique to the problem of automatic fastener fault detection.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Utgivningsort Uppsala

Handledare: Amr Salem Ämnesgranskare: Carolina Wählby

Examinator: Elísabet Andrésdóttir

Populärvetenskaplig sammanfattning

En tillförlitlig och säker järnväg är viktigt för att upprätthålla en fungerande infrastruktur i det svenska samhället. Mellan år 2017 och 2040 förväntas tågtrafiken i Sverige öka med cirka 30% för persontrafik respektive 50% för godstrafik. Detta leder till högre krav på regelbunden och korrekt underhåll av järnvägen. Den naturliga miljön, kombinerat med slitage orsakat av kontaktfriktion och vibration mellan tåghjul och spår, påverkar infrastrukturen negativt. Därför är underhåll och inspektion av järnvägens tillstånd viktig för en tillförlitlig tågdrift. Järnvägen består av många komponenter såsom spår, signaler, växlar och elledningar, som alla måste samverka för att skapa ett tillförlitligt system. En viktig komponent är befästningselementen som fäster rälsen i sliprarna, och således förhindrar horisontella och longitudinella förskjutningar samt säkerställer en korrekt spårvidd. Saknade befästningselement innebär minskad stabilitet på järnvägen vilket kan leda till urspårningar. Noggrann inspektion och korrekt underhåll av befästningselementen är därför betydelsefullt för att upprätthålla en säker tågdrift. Kontroller av järnvägen sker idag genom manuell inspektion av komponenterna vilket är tidskrävande, riskfyllt och kostsamt. För att minska underhållskostnaderna och öka säkerheten, finns ett intresse att ersätta den nuvarande manuella inspektionen med ett system som möjliggör automatisk inspektion av befästningselementens tillstånd. Ett automatiskt kontrollsystem kan upprättas med hjälp av maskininlärningsalgoritmer som möjliggör en mer effektiv, objektiv och proaktiv inspektion av järnvägen. Utifrån detta undersöker denna mastersuppsats hur maskininlärning kan tillämpas för en automatisk detektering av befästningselementen. Uppsatsen avhandlar hela processen av praktisk tillämpning av maskininlärningsalgoritmer inom bildanalys, vilket inkluderar datainsamling, dataförbearbetning, modellträning och modellutvärdering. Modellen använder sig av objektdetektorn YOLOv5, en maskininlärningsalgoritm som kan tränas till att lokalisera och klassificera objekt i bilder.

Objektdetektering är den undergrupp inom maskininlärningstekniker som syftar till att upptäcka förekomsten av objekt med en specificerad klassificering i digitala bilder eller videor. Metoder för objektdetektering är vanligtvis baserade på neurala faltningnätverk (eng. Convolutional Neural Network) som automatiskt lär sig att utläsa information i bilderna för att kunna detektera objektet. För att kunna använda en inlärningsbaserad metod krävs en stor mängd data där objekten är annoterade för att uppnå goda resultat, samt ett varierat dataset. Annoteringen utgörs av manuell markering av den korrekta lokaliseringen av objektet i bilderna med hjälp av rektangulära boxar, samt en tilldelad klass till boxarna. I detta fall bör både befintliga och saknade befästningselement detekteras korrekt och därför insamlades bilder som innehöll båda fallen och annoteringen utgjordes av två klasser: befästningselement och saknade befästningselement. Sammanlagt bestod datasetet av 437 unika bilder tagna på befästningselementen uppifrån, där varje bild innehåller en enhet av två positioner för befästningselement, en på varje sida av järnvägsrälsen. För att skapa ännu mer variation i datasetet användes augmentationstekniker som utökade datasetet genom att konstruera bilder med skillnader i skärpa, inzoomning och ljusstyrka.

Inom maskininlärning är ett dataset traditionellt uppdelat i tre kategorier: träningsset, valideringsset och testset. Algoritmen använder träningssetet för att lära sig urskilja objekten genom att studera bildernas pixlar för att kunna urskilja mönster och konturer som utmärker objekten. Algoritmen gör sedan en ansats till en detektering i bilderna i valideringssetet, genom att själv markera objekten med en rektangulär box och tilldela en klass. Detekteringen jämförs sedan med den ursprungliga detekteringen för att avgöra vad som behöver justeras i algoritmens vikter, det vill säga den del av algoritmen som bedömer vad som utmärker objekten. På så vis tränas algoritmen att göra så korrekta ansatser som möjligt. Den slutgiltiga modellen används sedan för att detektera bilderna i testsetet, som består av, för algoritmen, en ny uppsättning bilder. Därefter görs en ny summering av modellens prestanda.

Det finns flera välkända algoritmer inom objektdetektering varav YOLOv5 är en. Fördelen med YOLOv5 är att algoritmen har hög prestanda avseende precision och tid samt lämpar sig för detektering av små objekt. För att utvärdera YOLOv5s potential inom detektering av befästningselementen, evaluerades olika modeller av YOLOv5 där skillnader i dataset, hyperparameterar och användningen av teknikens överföringsinlärning testades. Datasetet evaluerades genom att undersöka skillnader med och utan augmentationstekniker, samt med och utan så kallad bakgrundsbilder där inget objekt syns i bild och annoteringar därmed saknas. Hyperparametrar är parametrar som kan justeras i YOLOv5-algoritmen. Här evaluerades olika satsstorlekar och epokstorlekar. Överföringsinlärning är en teknik där förtränade vikter från en tidigare algoritm, tränad på ett större dataset, överförs till algoritmen. Vissa vikter är inte beroende av skillnader i objekt, utan tränas till att effektivt hitta skillnader i generisk information såsom färg, kanter och skepnader. Därför kan vikter som tränats mer på den typen av information i bilderna överföras till en algoritm som har ett mindre dataset att tränas på.

De bästa modellerna lyckades klassificera 100% av alla befästningselement som befästningselement, och 96% av de saknade befästningselementen som saknade befästningselement. Flera av de olika kombinationerna av modellinställningarna hade en positiv effekt på modellernas prestanda. Det gör att slutsatsen kan dras att YOLOv5-modellen är en robust och lämplig modell för att upptäcka befästningselementen. Testdata visar att modellen brister vid bilder där en del av nästkommande sliper är synlig i bild. Då kan modellen felaktigt detektera delar av slipern som ett befästningselement eller saknat befästningselement. Dessa fall kan dock reduceras genom att synkronisera frekvensen mellan bildtagningen av befästningselementen med avstånden mellan sliprarna, så att endast en sliper och tillhörande befästningselement är synlig i bild.

Acknowledgments

This master thesis has been carried out at Uppsala University in collaboration with Combine AB, a Gothenburg-based consultancy firm specializing in control systems, model-based design, and data science. First, we want to give our greatest thanks and utmost appreciation to Amr Salem, our supervisor at Combine AB for providing us with excellent help and support throughout the complete project. We would also like to thank all the other employees at the company who contributed to the project execution. In particular, the group manager Tobias Olsson, and the CEO Peter Karlsson, for allowing us to write our master thesis in collaboration with the company. At last, we want to express our sincere gratitude to our supervisor at Uppsala University, Carolina Wählby, professor in the division of Information Technology. Thank you for your professional, rigorous, and careful guidance and feedback, and for giving us the support we needed.

Gothenburg, May 2022

Alva Efraimsson

Elin Lemón

Table of Contents

1. Introduction

Rail transportation has emerged as a significant environmentally friendly alternative for the mobilization and transportation of people and commodities around the world. The number of people traveling by train has increased rapidly in the latest 25 years [1], and the workload is estimated to annually grow by 1% up to 2050. The transport alternative has great potential in solving societal challenges such as increasing energy costs, carbon emissions, and congestion of road and air traffic [2]. In 2020, 169 million passengers traveled on the Swedish railway, a decrease of 36% from 2019 due to the pandemic. 70 000 million kilogram goods were affreighted on the Swedish railway in 2020 [3].

However, the current state of the railway infrastructure requires more attention to be able to deliver and maintain available transportation that corresponds to the increasing request. The escalating workload on tracks and switches puts higher demands on the maintenance and reconstruction of the railway which is a crucial aspect of the availability [4]. The natural environment combined with damage caused by contact frictions and vibration's impact between train wheels and tracks challenges the infrastructure to remain intact. Consequently, the track is often deformed or even collapsed due to the gradual destruction that trains cause while operating across the railway day and night. Broken or missing components of the railway often lead to traffic disturbances, which induces delays and missing routes. Therefore, daily inspection and maintenance of the railway's condition are significant for delivering reliable train operations accordingly to the rising requests for train traffic [5].

Traditionally, maintenance of the track has been handled by track workers' manual visual inspection of the tracks' condition. It's a time-consuming technique leading to high manpower costs and is exposing the worker's' health considering the high-risk environment. Another, less risk-taking technique, is to gather images of the track and inspect the photos, which limits the physical presence at the railway. However, a manual inspection, in general, might be limited due to human errors such as fatigue or omission [6]. To increase the accuracy of manual inspection and reduce the workload and risks, automation techniques can be utilized to provide more efficient solutions. In recent years, fault detection techniques based on machine learning have received a lot of attention among researchers and industries with the ambition to solve the manual inspection problem. The aim is to provide an automatic solution for detecting faults of components of the railway proactively to enable precautions measures, so-called proactive maintenance [7].

The train system consists of many components such as the railway, signals, switches, and electricity, all of which must cooperate for a reliable system. Therefore, there are different types of maintenance inspections such as track geometry, track profiles, switches, comfort, and vibration. One important component of the railway construction is the fasteners, the component that fastens the rails onto the sleepers to prevent the transverse and longitudinal deviations of the rails from the sleeper, and to maintain the

gauge and rail geometry. The railway is the part of the system that is mainly exposed to damage caused by heavy train traffic. They are therefore crucial for the stability of the track and the safeness of the railway. Fastener defects can cause train derailment due to the horizontal deviation of rail from the sleeper and low lateral resistance. Therefore, daily inspection and maintenance of the fastener's conditions are significant for maintaining the safety and stability of train operations [8]. Recent studies have for this reason investigated machine learning's potential for detecting broken or missing fasteners based on image analysis of the fasteners [2, 8, 9, 10, 11].

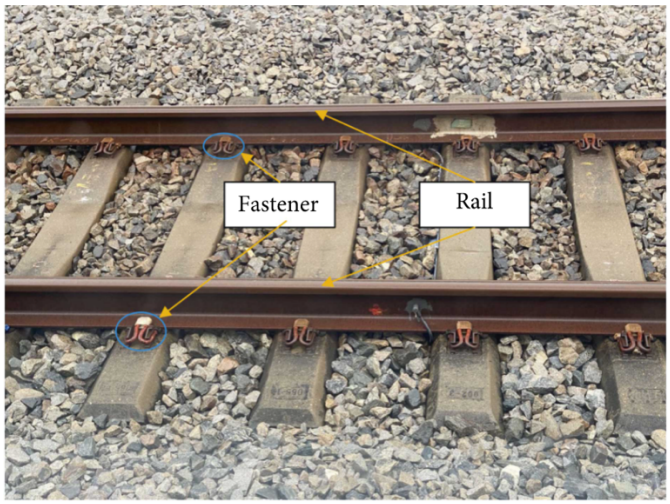

Figure 1. Railway track line [8].

In Sweden, the control of the railway relies on manual inspection based on inspection intervals defined by the track's' traffic load and the traffic's velocity. Despite increasing investments and allocations, the Swedish railway has encountered structural and technical problems due to deteriorated infrastructure caused by inadequate maintenance relative to the increasing workload [12, 13]. The traffic at the Swedish railway is expected to increase by 28% for public transportation and 51% for industrial transportation in the years 2017 to 2040. This will put even higher demands on accurate maintenance for reliable train operations. The maintenance cost in Sweden is highly related to the inspection time carried out along the railway, and component inspection on average accounts for 76.5% of the overall time and capital for track inspection. The major challenge within the railway industry is to reduce these maintenance costs simultaneously as augmenting the capacity of the rail network. To reduce maintenance costs, enhance safety and expand track capacity, railroad companies are interested in substituting the current manual inspection with a system that enables automatic fastener inspection [14].

With the increasing need for more adequate maintenance of the Swedish railway combined with an automatic fastener inspection system's ability for more efficient and objective inspections, this thesis aims to investigate how machine learning algorithms can be utilized at the Swedish railway in practice. Traditionally, automatic inspection systems have relied on model-based methods to be able to classify the status of the

fasteners [15]. Nowadays, learning-based methods have proven greater success in automatically extracting features of the images, localizing the object of interest, and classifying the status. Learning-based models such as R-CNN, SSD, and YOLO are constantly evolving and previous versions of YOLO have obtained great results when detecting fasteners [7, 8, 10]. This thesis emphasizes the latest version of the convolutional neural network YOLO, YOLOv5, released in October 2021, and investigates the model's ability to detect fasteners at the Swedish railway.

1.1 Purpose

The purpose of this thesis is to investigate how machine learning can be applied to the problem of automatic fastener fault detection. The thesis includes the complete process of applying machine learning algorithms to a given problem, including data gathering, data preprocessing, model training, and model evaluation. Rather than testing a large number of different machine learning approaches and types of object detection, the focus is on investigating the performance of state-of-the-art object detector YOLOv5 on the task of localizing and classifying e-clip fasteners.

1.2 Delimitations

Previous studies have different aims for fastener fault detection. Some focus on the status of the fasteners and classify if they are broken or not [8, 16, 17], and some do the binary detection of labeling the fasteners as missing or existing [18]. In the scope of this thesis, the detector is only trained on classifying the existence of the fasteners, and not the status. This is limited by the available dataset, where only the case of present and missing fasteners could be gathered. Other studies also focus on comparing different object detection models for investigating what models perform the best [10]. Recent studies conclude that a greater focus on the preprocessing and tuning of one model is more valuable than comparing different models since the evolution of object detection algorithms has resulted in well-performing models in general. Therefore, this thesis focuses on the model YOLOv5 exclusively. Finally, other studies detect different types of fasteners [7, 19]. The railway used for gathering the data is constructed by Pandrol e-clip, one of the most common types of fasteners used in Sweden. Therefore, this thesis is limited to only investigate how machine learning algorithms can be utilized for fault detection of Pandrol e-clip fasteners.

1.3 Disposition

This thesis initially introduces relevant information about the research field in the *Background section*. The information concerns the infrastructure of the railway, the current status and maintenance of the Swedish railway, and a summary of previous studies. The *Background* section is followed by the *Theory*. The section aims to present all technical aspects of the project from a theoretical point of view. The *Theory* section explains the field of object detection, convolutional neural networks, the final model,

YOLOv5, and its hyperparameters. It also introduces the techniques of transfer learning and model-based methods. Finally, the theory introduces the performance evaluation metrics used for measuring the models' performance. The *Method* section presents the procedure of collecting the data, preprocessing the data, implementing the model, and evaluating the models' performance. All relevant steps in the progress of creating a fastener fault detector are given and motivated in this section. The *Method* is followed by the *Result* section. Here are all the measurements of the models' performance presented, together with the inference on the test data. The result is further analyzed and explained in the *Discussion* section where the models' results are compared and evaluated in terms of its performance. The discussion ends with a review of YOLOv5 as a model for the given problem. All conclusions made connected to the purpose of the thesis are presented in the final section, *Conclusion*, together with future work.

2. Background

In this section, the history of the Swedish railway is presented to give a comprehensive understanding of the socio-technical aspect of the purpose. Furthermore, a more technical background about the components of the railway is given and the importance and the challenges of different types of railway maintenance in general, are described. This is followed by a presentation of the use of automatic inspection as a system for railway today and, at last, a summary of previous studies regarding automatic inspection systems focused on fasteners.

2.1 The Swedish Railway

The first documented railway in Sweden was constructed in Höganäs in 1798. From the middle of the 19th century, several small railways used for industrial purposes were built across the nation. In 1845 Adolf Eugène von Rosen got permission from the state to build privatized railways that would connect the major cities. Von Rosen's plans lead to a serious debate about the Swedish railway infrastructure. This resulted in a proposal adopted by the government in 1854 that implied that the state should build, own and maintain a mainline. The decision implied that all lines, except for the mainline, the railways would be built by private interest. This resulted in about twice as many railways being built by private stakeholders. For economic reasons, a third of the privately-owned railways were constructed as narrow-gauge railways. In 1939 the Swedish Riksdag determined a general nationalization of the private railways. The decision covered some exceptions, routes that later were closed or integrated with the government-owned public railway company, SJ AB [13].

After extensive closures during the years 1950 to 1970, the future was uncertain for the railway infrastructure. The growing vehicle industry and the intensified environmental debate led to a comprehensive political process that in 1988 resulted in the decision to split SJ AB into the state administrative authority called the Swedish Rail Administration (SV. Banverket) and the remaining SJ AB. The Swedish Rail Administration was responsible for the infrastructure and SJ AB administrated the train traffic of the most significant lines. The smaller county railways were organized by the counties of Sweden and were procured by SJ AB or competitive stakeholders. The deregulation aimed to facilitate feasible and more commercial conditions for the railway infrastructure toward other means of transportation [13].

The last step of the deregulation was taken in 2011 when SJ AB's monopoly of the mainline terminated [13]. In 2010 the Swedish Rail Administration was dissolved and replaced by the state administrative authority The Swedish Transport Administration (SV. Trafikverket). The Swedish Transport Administration is responsible for the overall long-term infrastructure planning involving sea, road, rail, and air transportation [20]. Within the railway area, this includes the responsibility for the management and maintenance of the railway, signal system, and electricity plants, traffic control, railway

guides, and managing the development of new railway lines. The Swedish Transport Administration is only responsible for the procurement of the surveillance, construction, or maintenance of the railway, which is pursued by external suppliers and contractors [13]. The Swedish Transport Agency (SV. Transportstyrelsen) examines and grants permission as well as supervises these external parts [21]. After the deregulation of the railway system, the traffic is pursued by several train companies. In 2022, SJ AB is the major operator for the conveyance of passengers and Green Cargo AB is the major operator for the carriage of goods. Apart from these, operators such as Transdev, Arravia Sverige AB manage passenger transportation, and Hector Rail AB, Railion Scandinavia, and Rush Rail manage goods [13].

Sweden has 15 600 kilometers of railway, where the Swedish Transport Administration manages 91% of the railway [22]. Today, the railway has a standard track gauge of 1435 millimeters and only one narrow-gauge railway is still used, Roslagsbanan in Stockholm [23, 13]. In 2020, 169 million passengers traveled on the Swedish railway, a decrease of 36% from 2019 due to the pandemic. 70 000 million kilogram goods were affreighted on the Swedish railway in 2020 [3].

The Swedish railway, despite increasing investments and allocations, has encountered structural and technical problems due to worn-out infrastructure as a result of insufficient maintenance. The primary reason for this is the increasing traffic on the railway relative to the maintenance, which is expected to continue increasing by 28% for public transportation and 51% for industrial transportation in the years 2017 to 2040 [12, 13]. The deregulation of the railway has also been widely discussed as a factor [13].

The Swedish Transport Administration, the responsible authority, has received criticism for its maintenance triage system. The system is structured so that the need for maintenance on the most frequently used railways in the metropolitan areas is prioritized to be eliminated and dimidiated on railways on major lines that interconnect metropolises and are of socioeconomic importance [13, 12]. The prioritization has resulted in the rest of the railway network continuing to deteriorate when the need for maintenance is increasing. The Swedish Transport Administration has therefore been forced to introduce temporary speed restrictions on long-distance routes. This affects both commuters and shipments that are dependent on safe and long-term sustainable transportation [13].

2.2 Railway Components

The railway has several important components, as displayed in Figure 2. The steel rails are placed on transverse sleepers. The sleepers are usually made of wood, concrete, or in some cases steel. The sleepers are used to secure the gauge of the rails and support the track on the ballast. Between the rail and the sleeper, a rail pad is placed to fix the rail to the sleeper by fasteners [24]. Moving material on the rail, changes in temperature and longitude changes of the rails position transfer forces from the rail to the ballast. The sizes of the forces vary depending on the extent of the traffic, type of rail, and

climate conditions and can act vertically, laterally, as well as along the rails. To counteract these forces, the fasteners need to generate a clamping force and neutralize some of the forces. Along with the development of railway traffic and rail technology the design of the fasteners has changed from simple nail brackets on wood sleepers to resilient fasteners on concrete sleepers. There are several different types of fasteners used at the Swedish railway, for example, Hayback fasteners, Hambo fasteners, Pandrol Fastclip, and Pandrol e-clips. In this thesis, the fastener of interest is Pandrol e-clips [25].

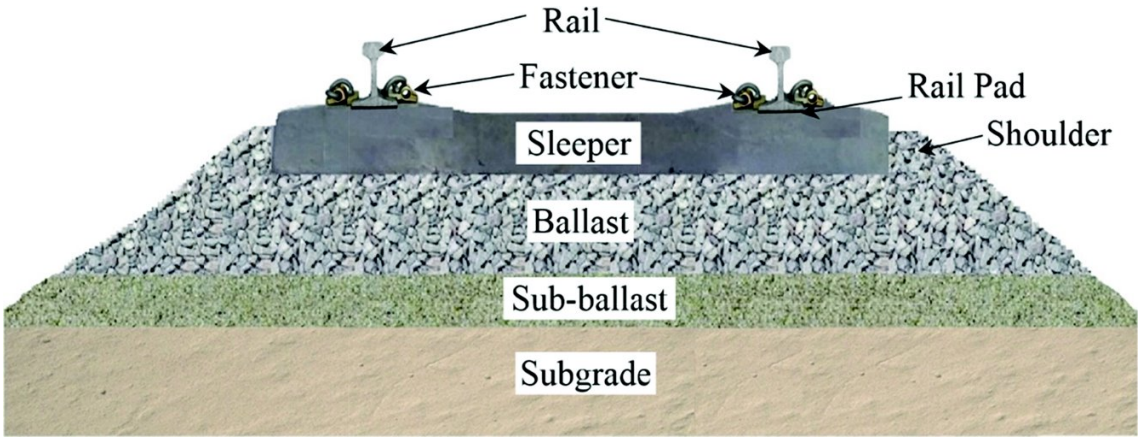

Figure 2. The structure of the general railway construction [26].

2.3 The Importance of Maintenance

One of the biggest expenses of the railway industry is the cost of the number of executed maintenance inspections. Railway maintenance plays a vital financial role in technical, administrative, and managerial decisions. Correct maintenance can prevent disturbances in traffic due to canceled or delayed routes. In 2020 the Swedish government decided to invest 500 million Swedish kronor yearly in the maintenance of the Swedish railway from 2021 to 2023. The purpose is to strengthen the railway's reliability and ability to provide climate-smart transportation of both goods and people. A reliable, robust, and long-term sustainable railway is a prerequisite for people and the industry to be able to live and act in Sweden [27]. In the Swedish maintenance strategy for 2021-2024, the Swedish Transport Administration accentuates the importance of data used to analyze and conduct efficient maintenance. New technologies accelerate the increase of automatized maintenance that can improve traffic safety, decrease the environmental impact, and increase capacity, which in turn will put higher demands on vehicles, information systems, and infrastructure [12].

According to Feng et al. [28], it is crucial to find a prominent maintenance strategy to optimize the efficiency of the industry and reduce costs. There are different approaches for detecting broken or missing components of the railway system that need to be recovered. One approach is to apply proactive maintenance which aims to prevent train derailments and other accidents from occurring by applying suitable measures in time.

The contradiction to proactive maintenance is reactive maintenance where maintenance is applied when any fault has occurred [12, 5]. This strategy is characterized by higher costs and shorter notice of planning. Proactive maintenance is performed in cycles based on an estimated time of functionality of the specified component. The time of functionality is the estimated time before the component requires maintenance [29]. Proactive maintenance is the most desirable approach owing to its ability to prevent the system from breaking which extends the lifetime of the system and therefore reduces costs by minimizing disturbances and accomplishing a more sustainable and reliable system [5]. When investigating the need for improving the railway track performance on Swedish railways, a long-term strategy and proactive solutions are required to reach the desired goals of improved safety, availability, and reliability [30].

Based on the scheduled maintenance, an inspection of the railway is executed to control the health condition of the railway infrastructure. In earlier days, the inspection was carried out by manually observing visible anomalies and technical deviations by walking along the track. This is a time-consuming technique that relies on the manpower's ability to carefully assess the status of the component and poses safety issues for maintenance staff. To reduce the risk, the railway can instead be recorded, and the inspection team can view the recorded files to give their assessment of the condition. However, this technique can lead to missing anomalies prone to human errors and is also inefficient and expensive, especially for long-term and large-scale development projects [6, 2]. Therefore, automatic inspection of the railway system (AIRS) is an important approach to achieve proactive maintenance of the railway. The recent increasing use of computer vision techniques has driven the development of AIRS. Automatic inspection can be utilized to detect rail track irregularities, essential components, and obstacles [11].

2.4 Automatic Inspection of the Railway System

Switzerland is currently using an automatic diagnostic rail system to increase the safety of rail transport by proactive maintenance. They used the Generative Analyzer Network (GAN) to detect 20 different fault types that are classified into five major categories: welding, joints, surface defects, and wheel slip. Specific "diagnostic trains" are equipped with high-resolution cameras and sensors that travel along the railway at a speed of 160km/h to collect necessary data to conduct a maintenance prognosis. Using deep learning techniques, they improve the traditional detection and classification technology, minimize the inspection carried out by manpower, and reduce the problem of misreporting. Canada has also implemented a full-scale rail inspection system that operates on a locomotive or modified vehicles. The system analyzes and measures the rail geometry such as can, alignment, curvature, etc., and detects defects on the tie, fastener, crossing, and rail surface damage [7]. These examples give evidence that an automatic inspection system is possible to implement in Sweden too.

2.5 Previous Studies

Automated rail inspection systems have recently received a lot of attention from researchers due to their potential of creating more efficient inspection of the track and its component. An automated rail inspection system can have various purposes such as rail profile measurement, rail surface defect detection, gauge measurements, and rail fastener detection. In the last two decades, automatic inspection systems focused on rail fastener detection have especially gained attention as they are important for clamping the rail to the sleepers and therefore are preventing transverse and longitudinal deviations of rails from the sleepers and maintaining the gauge. However, different machine learning techniques have been used over time for detecting the fasteners using computer vision [2]. The two most used techniques in previous research are model-based approaches and deep learning. The model-based methods have three aspects: locating and segmenting the fastener regions, extracting fasteners' features, and applying classification algorithms for detecting defects. The most commonly used feature extraction techniques used in the literature result in Haar-like features [28], DenseSIFT features [17], direction field features [18], edge features [31], HOG features [32], Gabor filter features [19], and Hough transform features [33]. After extracting features, classification models such as AdaBoost classifiers [34], support vector machines (SVM) [35, 32, 19], probabilistic graphical models (PGM) [17], and multilayered perception neural classifiers [36, 37] are used for classifying the state of the fasteners. However, these model-based methods are less robust to abnormal conditions since the extracted features are sensitive to the diversity of shapes and backgrounds and are therefore more likely to fail if the conditions change [8]. In recent years, deep-learning techniques have instead proven greater success in detecting defects on fasteners [15].

The rapid development of graphics processing units (GPU) has enabled the emergence of deep learning methods due to access to greater computing power and shorter training time. The learning-based methods, using deep-learning, automatically extract the features of images which makes them more robust to images containing noises [12]. In 2019, Wei et al. [9] constructed a model utilizing deep-learning techniques for fastener detection. They used Dense-SIFT features, spatial pyramid decomposition, and Bag of Visual Words (BOVW) techniques as preprocessing techniques before the use of the deep convolutional neural network VGG16. The model outperformed previous methods with a classification accuracy of 99.26% and the time for fastener detection decreased to one-tenth compared to other released models. Finally, they also introduced the use of Faster Region-based Convolutional Neural Networks (Faster R-CNN) which proved to improve the detection rate and time. Later that year, Lin et al. [7] also created a model using deep convolutional neural networks. They used You Only Look Once version 3 (YOLOv3) and obtained a precision rate of 89% and a recall rate of 95% when detecting multiple versions of fasteners along the railway of Taiwan. Wang et al. [10], 2020, evaluated the series of one-stage YOLO models and concluded that YOLOv2 performed the best with a performance of 93% mean Average Precision ($mAP_{[0.5]}$) for

Intersection over Union (IOU) threshold of 0.5. For a threshold in the interval between 0.5 and 0.95, they reached an $mAP_{[0.5:0.95]}$ of 0.5. In 2021, Liu et al. [16] constructed a convolutional neural network (D-CNN) supported by a U-Net-based model to generate defective fastener images and compared the result with other models such as YOLOv3. Their proposed method outperformed the other models with precision and recall of around 93-100% on different classes. The average F1 score for all classes reached 0.90. Chandran et al. [2] investigated the combined use of a model-based approach and a deep learning algorithm applied to images from inspection along the Borlänge-Avesta line in Sweden 2021. They concluded that image preprocessing improved fastener localization and the removal of redundant information from the fastener images. By using CNN and ResNet-50 algorithms they achieved a classification with 94% accuracy during testing. In 2021, Zheng et al. [8] used an improved YOLOv5-framework for multi-object detection and Mask R-CNN to detect surface defects of rail components on images collected from the railway line Shijazhuang-Taiyan in China. Their model achieved an $mAP_{[0.5]}$ of 99.68%, recall of 100%, and precision of 96.23% without specifying threshold.

Previous studies prove great results using object detection techniques on the task of creating a fastener fault detector. The research, therefore, gives evidence of following a similar approach to investigate how machine learning can be utilized in the case of fastener fault detection.

3. Theory

The theory section begins with introducing computer vision in general and two chapters describing the two approaches of creating a detector: using model-based methods and learning-based methods. This is followed by a section that in more detail explains convolutional neural networks used in learning-based methods. After that, the final model, YOLOv5, is presented in more detail followed by sections describing the technique transfer learning, the hyperparameters of YOLOv5, and different evaluation metrics techniques.

3.1 Computer Vision

Computer vision includes a set of machine learning technologies for extracting information from images and videos to understand and interpret our world [38]. Image classification, object localization, and object detection are all various types of object recognition techniques for extracting information in an image serving different purposes. Image classification simply assigns the image one class based on the object(s) inside the picture. Object localization, on the other hand, localizes the object in the image by drawing a bounding box around the object and outputting the classification of the object. Object detection is a combined technique of the two where multiple objects can be detected in an image given by an output containing several bounding boxes and multiple classes [39].

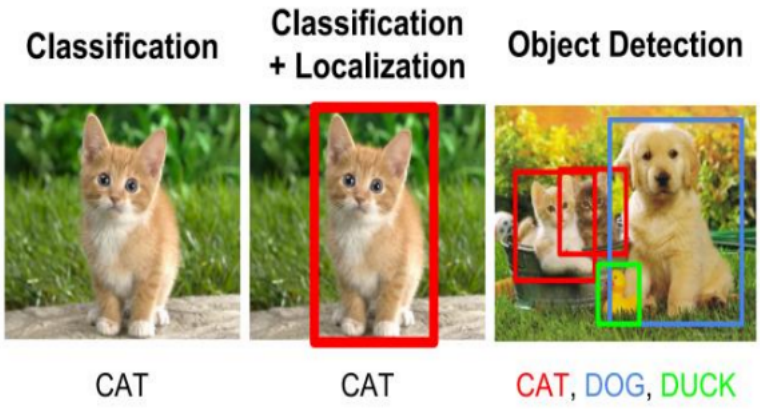

Figure 3. Object recognition techniques [39].

As described in previous studies, object detection has been utilized in the case of fastener detection. Recent studies take a more model-based approach where pre-knowledge about the images is used to apply correct image processing-based methods to extract information about the fastener's condition. Model-based techniques are used for extracting features in images using denoising, region growing, edge detection, etc. [15]. These techniques can be used for serving the complete purpose of identifying fasteners or combined with learning-based methods that manage the final classification of the object, such as image classification or object detection. The advantage of learning-based methods is that these techniques typically use deep learning algorithms

that automatically learn about the features in the images to be able to detect the object, instead of manually applying a suitable method based on pre-knowledge. Learning-based methods are singularly based on data as input, compared to model-based methods that rely on a mathematical model based on the developers' expertise and assessment of the images.

This project initially emphasized a model-based approach combined with image classification to detect the fasteners without any success. Instead, a learning-based method was used to successfully create a fastener detector. Both image classification and object detection can use convolutional neural networks as a classifier to recognize the object.

3.2 Model-based Methods

As mentioned before, model-based methods have previously been used to create a fastener fault detector before deep learning techniques gained more attention. The idea is to use pre-knowledge about important features of the images and filter out these features with some specialist filters to make the classification problems easier since the classifier does not have to work with irrelevant data. The new deep learning algorithms do not need the extraction phase since these manage to automatically learn about what details of the image are important if it has access to enough data [40]. The model-based methods, on the other hand, has the three following aspects: (1) locating and segmenting the fastener region; (2) extracting fastener features; (3) using a classification algorithm for fastener defect recognition [2].

For locating and segmenting the fastener region, the widely used techniques in this area include denoising, region growing, and edge detection. Gaussian smoothing is used for denoising the image as a preprocessing step. Region-growing partitions the image into different homogeneous areas with similar pixel intensity, and edge-based segmentation finds pixels that correspond to the boundaries in an image. One popular edge detection algorithm is Canny [41]. Another segmentation technique is the Hough transform. The Hough transform detects lines in the image by converting an image from Cartesian to Polar coordinates. It can therefore be used to find the intersection between the rail and the sleeper as it has a structure of vertical and horizontal lines [42]. Finally, K-means can also be utilized to group pixels by thresholding to group the image into different clusters. The disadvantage of model-based methods is the sensitivity to different shapes and backgrounds. They rely on the ability on differentiating between pixel colors in the image to detect shapes and other features [14].

3.3 Learning-based Methods

To use a learning-based method to detect the desired object, the model requires numerous annotated data to learn from to obtain good results. The annotation includes the label of the object and a bounding box around its ground-truth localization. Convolutional neural networks (CNN) are a class of artificial neural networks

applicable for image recognition and widely used in learning-based approaches [43]. Learning-based models can be divided into one-stage and two-stage methods. One-stage models directly predict the locations, sizes, and classes of the object by treating the detection as a regression problem. Two-stage models, on the other hand, first locate candidate object areas by a region proposal network, followed by identifying the categories of these areas with a fine-tuned network. Examples of such models are R-CNN and feature pyramid network (FPN). Two-stage object detection algorithms achieve high detection accuracy, but the inference time still makes it unapplicable in certain cases requiring real-time detection. It is also more challenging to optimize each component of the two-stage pipeline. To overcome these challenges, faster one-stage models such as single shot multibox detector (SSD), RetinaNet, and You Only Look Once (YOLO) are available [10, 44].

The YOLO model improves the speed of detection and inference by passing through the CNN once to accomplish the prediction. Thus, it is useful for real-time object detection. SSD uses FPN in CNN for efficient object detection of various sizes. Contradictory to YOLO, SSD uses anchors to predict bounding boxes, i.e., predefined bounding boxes which are more efficient than predicting bounding boxes from scratch. YOLO has been developed to a new enhanced version, YOLOv2, by also utilizing the advantages of anchors. YOLOv2 is acknowledged for its ability to adapt to various image sizes enabled by multiscale training, and to solve small object detection problems with many object classes as it combines high-resolution and low-resolution features. Furthermore, YOLOv3 was added to the YOLO series with an improved feature extractor and containing multiscale prediction output, thus a higher inference time due to an increase in the number of layers [10]. YOLOv4 added several features leading to an improvement of 10% in mAP. Some of the features are Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross Mini-Batch Normalization (CmBN), and mosaic data augmentation [45]. In this study, the most recent version of the YOLO series, the improved YOLOv5 is used. Compared with YOLOv4, YOLOv5 uses adaptive anchor box calculation to calculate the best anchor value depending on different training data sets. It also adds adaptive image scaling to improve the speed of object detection by adding a minimum of black borders when scaling the image. A more in detail explanation of YOLOv5 is followed [8].

3.4 Convolutional Neural Networks

Convolutional neural networks (CNN) are a class of artificial neural networks applicable for image recognition and widely used in learning-based approaches [43]. To understand a convolutional neural network, it is necessary to understand the underlying structure of a regular artificial neural network (ANN). ANNs are inspired by the biological network of neurons inside the human brain and therefore consist of so-called neurons that are organized in three types of layers: input layer, hidden layer, and output layer and is visualized in Figure 5. The task of a neuron is to calculate the dot product between an input and its internal weights and pass the results forward to a non-linear

activation function (sigmoid in the example below in Figure 4). The connections between the neurons are the weights that will be learned during the training and are therefore decided by the dataset [46].

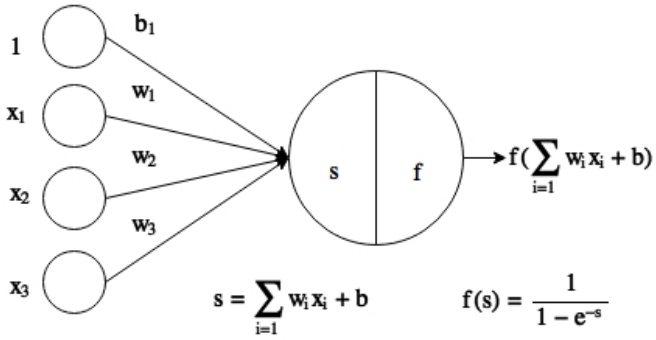

Figure 4. Illustration of a neuron using the sigmoid activation function [46].

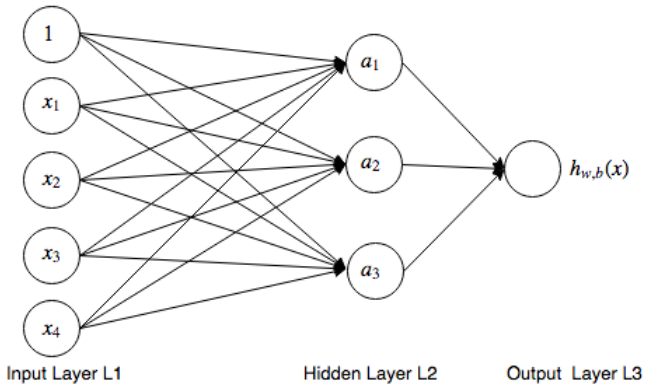

Figure 5. Illustration of a neural network [46].

The network learns by either addressing forward propagation or backward propagation. Forward propagation work through the network and evaluate the output by a loss function to tell how the network performed. Backward propagation goes backward through the network calculating the impact that each weight had on the final loss. Currently, backpropagation is most used and is in addition combined with a gradient-based optimizer that optimizes the weights to obtain the minimum loss [46].

Convolutional neural networks are artificial neural networks designed for images. The hidden layer consists of multiple convolution layers that filter the input layer to extract useful features of the input to the hidden units, as displayed in Figure 6. A convolution layer filters the input tensor, the image, in a tile-like fashion with a small window called a kernel. The kernel decides what to filter for in the given pixels and it is the kernel's weights that are learned in the training [46]. Multiple kernels operate over the pixels of the images and produce a two-dimensional feature map representing specific features such as edges, curves, corners, and points [47].

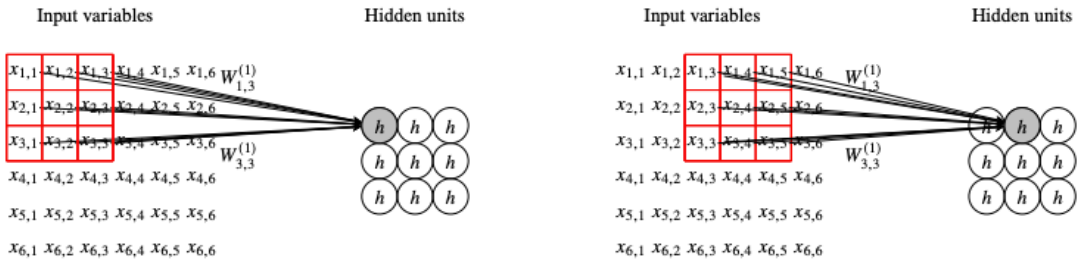

Figure 6. A convolution layer with filter size 3x3 [48].

The convolution layer uses the stride factor to control how many pixels the filter shifts over at each step. For example, the stride controls that not every pixel is considered when more convolution layers are added since the goal is to reduce the number of hidden units to only preserve the most prominent feature information. The information of the convolution layers is further summarized by the pooling layer. The pooling layer also acts on specific regions of pixels of the image and is used for down sampling the results from the convolution layers. Two common versions of pooling layers are average pooling and max pooling. The average pooling returns the average of all the units from the kernel space and the max-pooling returns the maximum value. One advantage of the pooling layer is that there are no parameters to learn [48].

Finally, a fully connected layer is applied to capture complex relationships between high-level features into a feature vector. Based on the feature vector, the classification layer in the output layer can classify the image [49].

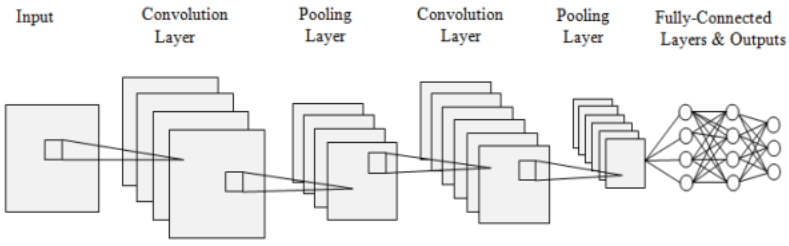

Figure 7. Generic example of A CNN architecture [50].

3.5 YOLOv5

YOLO is a state-of-the-art, real-time object detector, and YOLOv5 is the latest of the series of models YOLOv1-YOLOv5. Researchers have continuously improved the model by measuring its performance on the two official object detection datasets: Pascal VOC (visual object classes) and Microsoft COCO (common objects in context). Convolutional neural network systems usually consist of four parts as visualized in Figure 8: 1. backbone for extracting features, 2. neck for combining multi-level features, e.g., feature pyramid networks (FPN), 3. Region Proposal Network (RPN) for

generating region proposals (only in two-stage detectors) and at last 4. the head that gives the final classification and localization [51]. Generally, YOLOv5 uses the architecture of CSPDarknet53 with a Spatial Pyramid Pooling (SPP) layer as the backbone, PANet as Neck, and YOLO detection as the head.

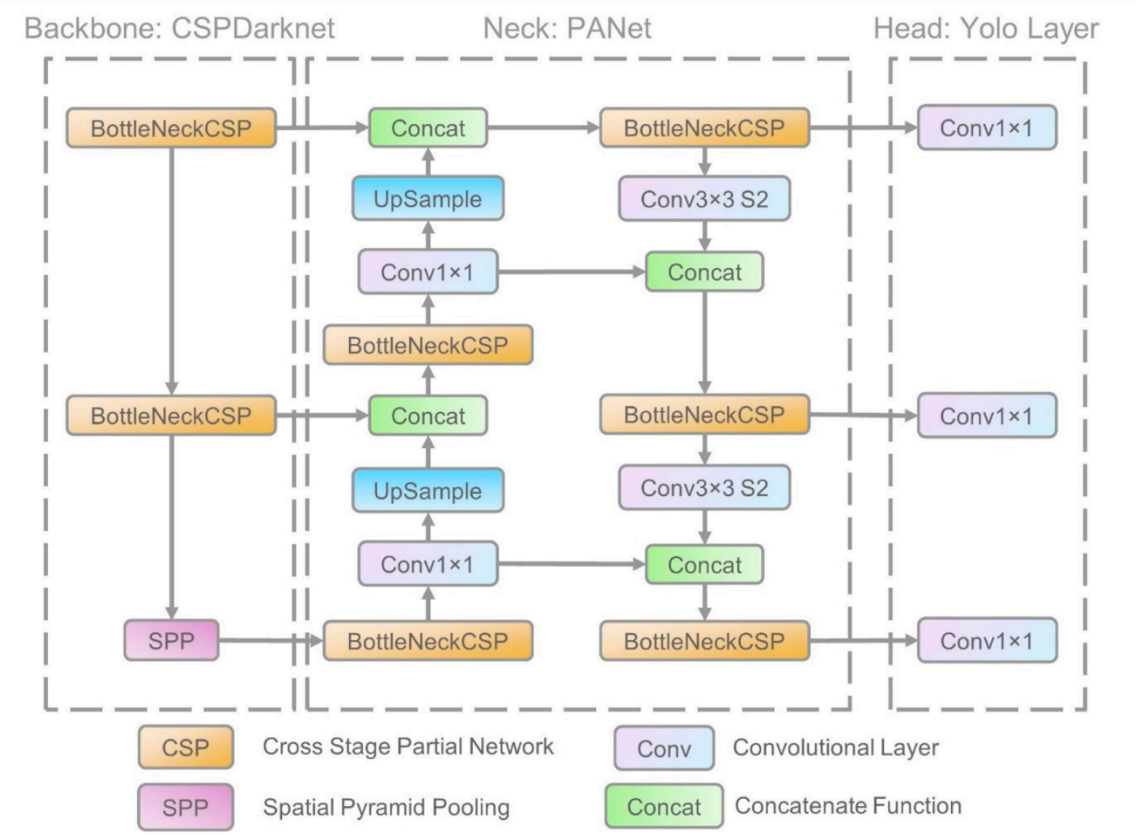

Figure 8. The architecture of the YOLOv5 network interest [44].

YOLOv5 creates the CSPDarknet as its backbone by incorporating the cross-stage partial network (CSPNet) into Darknet. The backbone increases the depth of the network and improves the ability to extract features. CSPNet efficiently decreases the parameters and FLOPS (floating-point operations per second) which improves inference speed and accuracy and reduces the model size. The Neck Network consists of a path aggregation network (PANet) which improves the problem of difficult propagation of low-level features of the original feature pyramid networks (FPN). PANet additionally more efficiently uses information about the accurate localization signals in lower layers to enhance the location accuracy of the object of interest [52]. The Head network is the same for YOLOv3 and YOLOv5 and consists of three components: bounding box loss, classification loss, and confidence loss. Binary cross-entropy is used for the loss function of the classification loss and the confidence loss, and Intersection over Union (IoU) loss is used as the loss function for the bounding box loss [8]. YOLOv5 has five different models including YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5n, and YOLOv5x. In this project, YOLOv5s is selected as it is the smallest and fastest model

available. The model is available open-sourced by the Ultralytics team and hosted on GitHub and is initialized with COCO pre-trained weights [53].

3.6 Transfer Learning

Transfer learning aims to extract knowledge from one or more source tasks to facilitate the training of a target task. In traditional machine learning techniques, the model is trained on the target data to learn the task, as visualized in Figure 9. Within transfer learning, knowledge is transferred from a previously trained source with a related problem, together with the target data, when training the target task, as displayed in Figure 10 [54]. This can especially be beneficial when the training data for the target task is limited since it avoids overfitting on the target training data [55]. By using transfer learning, the performance can be improved, and the training time can be decreased [54]. For reliable transfer learning, what to transfer, how to transfer, and when to transfer needs to be addressed [56].

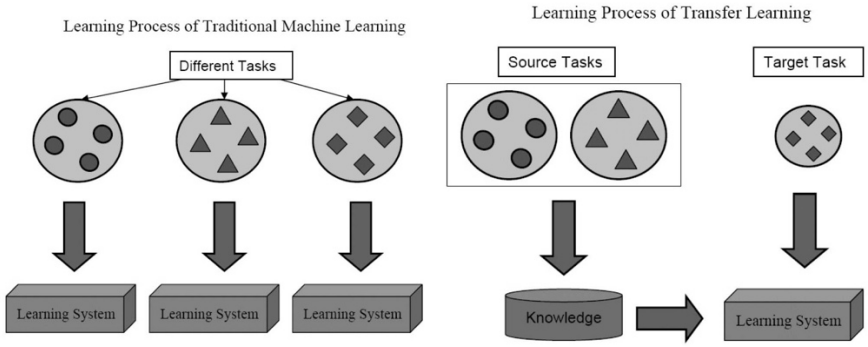

Figure 9. Learning processes over traditional machine learning and transfer learning [54].

When using pre-trained networks, transfer learning can be used to tune the network more efficiently for the target problem. This can be done by extracting the features from the pre-trained network, using the same architecture, or freezing some layers to train the others [55].

In image recognition, where the first layers typically extract generic features of the images such as colors, shapes, and edges, transfer learning can be utilized by freezing these layers. By doing so, these first pre-trained layers from one network trained on a big dataset, the base dataset, are applied to the target dataset. The performance of the layers trained on a bigger dataset is then utilized on a network trained on a smaller dataset to increase the overall performance. The remaining layers of the target network are responsible for extracting the specific features connected to the target dataset [55].

3.7 Hyperparameters

Hyperparameters are the parameters that can be manually set before training the model, based on properties corresponding to the characteristics of the data and the capacity of

the algorithm, to maximize the usefulness of the learning approach. When building an efficient model, one of the most important aspects is to tune the hyperparameters. The parameters tuned according to the dataset by the algorithm during training, weight, or model parameters, are not included in the hyperparameter segment [57].

Depending on the model, the hyperparameters vary. For convolutional neural networks, the hyperparameters are in general, among several, the number of nodes, layers, epochs, batch size, learning rate, momentum, and loss function. The network's ability to learn complex features corresponds to the number of layers, also described as the depth of the network. Each layer has several nodes, the first and the last hidden layer prerequire to be equal to the number of input features and classes to predict. The batch size defines the number of data samples to go through before updating the weights and calculating the gradient. The batches can be the size of the training set or smaller. When the batch size is smaller than the training set, the algorithm iterates over the batches before making the predictions. A too-small batch size makes the predictions fluctuate and a bigger batch size opens for a larger learning rate but can cause memory errors. The number of epochs determines how many times the algorithm will iterate over the whole training dataset. The loss function is chosen based on the output, to evaluate the performance of the weight corresponding to the desired output [57, 46].

Defining the architecture of a neural network is one of the most challenging tasks since the network is complex and trying all the combinations of hyperparameters is computationally expensive and time-consuming. YOLOv5 has about 30 hyperparameters used for various training settings. As a default, the hyperparameters are set to optimize the model when training on the COCO dataset.

3.8 Performance Evaluation

To evaluate the performance of an object detection algorithm the importance of suitable elevation measurements is crucial. Since the purpose of an object detection model is binary; both localizing and classifying the object, the overall performance is measured as a combination of the two. Detection of an object is represented by three attributes: the object class/label, the bounding box, and the confidence score. The confidence score is a value between 0 and 1 and describes how confident the model is about the prediction. To assess the prediction, the detection is compared to the ground-truth bounding box and the label given from the annotation. This section describes how the information about the ground-truth bounding box and label, and the predicted bounding box and label can be utilized to assess the model by different performance measurements.

3.8.1 Intersection over Union

Intersection over Union (IOU) is used to measure how close the predicted bounding box is to the ground-truth bounding box. The method calculates how much of the area of the predicted bounding box, B_p, overlaps with the ground-truth bounding box, B_{gt}, by dividing the area of overlap with the area of union, i.e., the total area of the ground-truth

box, and the predicted box. The result is a percentage of how much of the ground-truth box the model succeeded to predict, where a score of 1 means that the predicted box is the ground-truth box, visualized in Figure 10 and given in Equation 1 [58].

$$IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \quad (1)$$

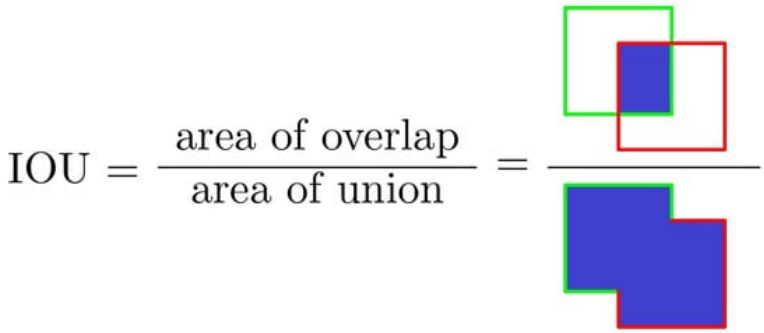

Figure 10. Illustration of intersection over union (IOU) [58]

In Figure 11 an example of a poor, good, and excellent IOU is presented.

Figure 11. Illustration of IOU performance [58]

3.8.2 Precision and Recall

By applying an IOU threshold, one can determine if the localization is correct or not. The threshold is typically 50%, 75% or between 50%-95% of IOU and if the predicted bounding box overcomes the threshold, and has the correct classification, it is considered a correct detection. It is based on these measurements one can determine if the detection is correct, i.e., the IOU overcomes the threshold combined with a correct classification. Detections with the wrong classification label are not considered. Based on the IOU, the results can be divided into three categories. If the IOU threshold is 0.5, the three categories are:

- If IOU ≥ 0.5 categorize the object detection as **True Positive (TP)**
- If IOU < 0.5, then it is a wrong detection and categorized as **False Positive (FP)**

- When a ground truth object is present in the image and the model failed to detect the object, categorize it as **False Negative (FN)**

Based on the three categories, precision and recall can be calculated. Precision describes how well the model can identify relevant objects and answer the question: *when the model guesses how often does it guess correctly?* It is the percentage of correct positive detections by dividing the number of true positive cases by the sum of all true positive cases and false-positive cases, as given in equation 2.

$$Precision = \frac{\Sigma\, True\ positive}{\Sigma\, True\ positive + \Sigma\, False\ positive} \tag{2}$$

Recall describes the model's ability to detect the ground truth cases by calculating the percentage of all ground truth cases that were found, i.e., the division of the true positive cases and the sum of the true positive and the false negative, as given in equation 3. It answers the question, *has the model guessed every time that it should have guessed* [58]?

$$Recall = \frac{\Sigma\ True\ positive}{\Sigma\ True\ positive + \Sigma\, False\ negative} \tag{3}$$

The prediction scores TP, FN, and FP are typically presented in a confusion matrix. The confusion matrix summarizes the number of correct and incorrect predictions and is broken down by the different classes. For a binary classifier, four cells of the confusion matrix quantify the frequency of every combination of a predicted class and the actual class. In the case of fastener detection, the 'class' background image will also be encountered in the confusion matrix to measure if parts of the background in images are predicted as missing or fasteners leading to false positives [59].

Table 1. The confusion matrix.

	Predicted positive	Predictive negative
Actual positive	True positive (TP)	False negative (FN)
Actual negative	False positive (FP)	True negative (TN)

3.8.3 F1 - Score

Analyzing the recall, precision, and confusion matrix is a good start for inspecting the false negatives and positives and getting an understanding of the misclassification rates. However, for an imbalanced problem where the detector succeeds to detect all the cases, leading to a precision of 1, but with a less desirable recall of 0.0001, a trade-off of the two can be useful. It can also sometimes be useful to, contradictory to a confusion matrix, summarize the results in one score. For that purpose, the F1 score summarizes the precision and recall by their harmonic mean, as is given in equation 4.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (4)$$

The F1 score is a number between 0 and 1 where a higher number is better since it is the optimal harmonic mean between precision and recall. Like the receiver operating characteristic (ROC) curve, the F1 score can be mapped into a curve depending on different thresholds. The curve gives an indication of the trade-off between precision and recall given different confidence thresholds [48].

3.8.4 Mean Average Precision

As described above, the output of an object detection model is represented by a bounding box for localization, a class/label for classification, and a confidence interval. Precision and recall are depending on the IOU level as detections with IOU larger or smaller than the IOU threshold is considered positive or negative detections respectively. By taking the IOU threshold, τ, into account, one can rewrite the precision and recall equations above as dependent on the threshold τ as given in equation 5 and 6:

$$Precision\ (\tau) = \frac{\Sigma\ True\ positive\ (\tau)}{\Sigma\ True\ positive\ (\tau) + \Sigma\ False\ positive\ (\tau)} \quad (5)$$

$$Recall\ (\tau) = \frac{\Sigma\ True\ positive\ (\tau)}{\Sigma\ True\ positive\ (\tau) + \Sigma\ False\ negative\ (\tau)} \quad (6)$$

When increasing the threshold τ the number of positive detections reduces as the requirement for being a correct detection is much higher. Therefore, both TP(τ) and FP(τ) are decreasing as a function of τ. Contradictory, FN(τ) is an increasing function of τ since a reduction of positive detections implies an increase of negative detections. As FN(τ) is increasing, the recall function is a decreasing function of τ, but as both TP(τ) and FP(τ) are decreasing functions of τ nothing can be said about the precision function depending on τ. Displaying precision and recall as a function of the threshold typically exhibit a zig-zag behavior, and is called the ROC curve [58].

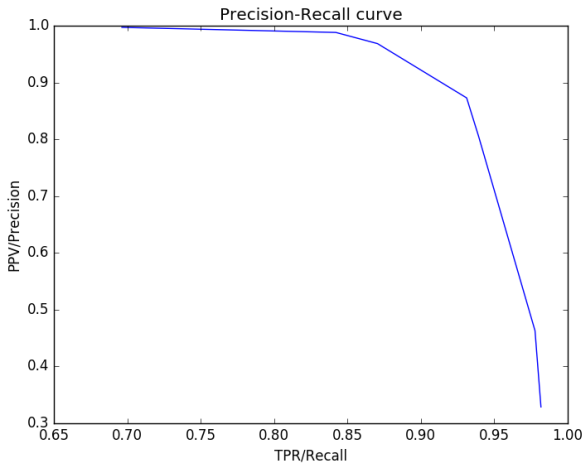

Figure 12. The ROC curve [60].

Based on precision and recall, a detector is considered good if all ground-truth objects are found, i.e., FN = 0 which leads to a high recall, and if only the relevant objects are identified, i.e., FP = 0 which implies a high precision. The goal is therefore to remain a high precision as the recall increases if the IOU threshold decreases. If such behavior, the area under the roc curve is large leading to another measurement of performance – the area under the curve (AUC). Average precision (AP) summarizes the precision-recall trade-off carried out by τ(k) IOU thresholds. For every threshold k, one can extract a pair of the recall and precision values from the roc curve, $\big(Pr\big(\tau(k)\big), Rc\big(\tau(k)\big)\big)$Before calculating the average of these values, the precision and recall pairs must interpolate such that the resulting precision and recall curve is monotonic. The interpolated curve is a continuous function $Pr_{interp}(R)$, where R is a real value contained in the interval [0,1] defined as in equation 7:

$$Pr_{interp}(R) = \max_{k|Rc(\tau(k))\geq R} \{\Pr\,(\,\tau(k))\} \tag{7}$$

The interpolated value of the precision at recall R is the maximum value of the precision where the corresponding recall value is greater than or equal to R. The AP can now be calculated as the area under the roc curve calculated by a Riemann integral of $Pr_{interp}(R)$ using K recall values from the set of recall sampling points $R_r(k)$, see equation 8:

$$AP = \sum_{k=0}^{K}\big(R_r(k) - R_r(k+1)\big)Pr_{interp}\big(R_r(k)\big) \tag{8}$$

Finally, one can calculate the mean average precision (mAP) which simply take the mean value of all classes' average precision. Considering C classes, mAP can be calculated as equation 9 [58]:

$$mAP = \frac{1}{C}\sum_{i=1}^{C} AP_i \tag{9}$$

The mAP can be visualized in a graph depending on different thresholds. One typical metric is $mAP_{[0.5:0.95]}$ where the mAP score depending on IOU thresholds in the interval between 0.5 to 0.95 is calculated.

3.8.5 Loss Functions

The YOLO loss function composes of three components: the classification loss, the localization loss, and the objectless/confidence loss. All loss functions are Mean-Squared error losses calculated based on some scalar meta-parameter of IOU score between the prediction and ground-truth. The algorithm predicts the bounding boxes by dividing the image into grid cells [61].

Figure 13. Illustration of the grid cells [61].

To modulate the loss based on the presence of an object on a particular cell in the grid (i, j) one introduces the 1_{ij}^{obj} member. If an object is detected, the classification loss at each cell is the squared error of the class conditional probabilities for each class, given in equation 10:

$$Classification\ loss = \sum_{i=0}^{s^2} 1_{ij}^{obj} \sum_{c \in classes} \left(p_i(c) - {p'}_i(c)\right)^2 \qquad (10)$$

where

- $1_{ij}^{obj} = 1$ if box j and cell i match, otherwise 0.
- ${p'}_i(c)$ denotes the conditional class probability for class c in cell i [62] [62].

The localization loss measures the error of the predicted boundary box in terms of location and size in comparison with the ground-truth box and is given by equation 11:

$$Localization\ loss =$$

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} ((x_i - x'_i)^2 + \left(y_i - y'_i\right)^2) +$$
$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} ((\sqrt{w_i} - \sqrt{w'}_i)^2 + \left(\sqrt{h_i} - \sqrt{h'}_i\right)^2) \quad (11)$$

where

- $1_{ij}^{obj} = 1$ if the *j* the boundary box in cell *i* is responsible for detecting the object, otherwise 0.
- λ_{coord} increase the weight for the loss in the boundary box coordinates.

YOLO predicts the square root of the bounding box width and height to give large and small box errors equal impact on the final loss. To put more weight on the boundary box accuracy the loss is multiplied with λ_{coord} which by default is set to 5 [63].

The objectness/confidence loss measures the error related to the confidence threshold for each bounding box predictor and is given by equation 12:

$$Confidence\ loss =$$

$$\sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - C'_i)^2 + \lambda_{no\ obj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{no\ obj} (C_i - C'_i)^2 \quad (12)$$

were

- C is the confidence score and C' is the IOU of the predicted bounding box with the ground-truth box.
- $\lambda_{no\ obj}$ is by default set to 0.5 to empathize that the loss careless about the confidence when there is no object [64] is by default set to 0.5 to empathize that the loss careless about the confidence when there is no object [64].

The final loss of YOLO is a summation of the classification, localization, and, confidence loss and hence is given by equation 13:

$$Loss = Classification\ loss + Localization\ loss + Confidence\ loss =$$

$$= \sum_{i=0}^{s^2} 1_{ij}^{obj} \sum_{c \in classes} \left(p_i(c) - p'_i(c)\right)^2 +$$

$$+ \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} ((x_i - x'_i)^2 + (y_i - y'_i)^2) +$$

$$+ \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} ((\sqrt{w_i} - \sqrt{w'}_i)^2 + (\sqrt{h_i} - \sqrt{h'}_i)^2) +$$

$$+ \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - C_i)^2 + \lambda_{no\ obj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{no\ obj} (C_i - C_i)^2 \text{ [61]} \qquad (13)$$

4. Methodology

This thesis aims to investigate how machine learning can be applied to the problem of automatic fastener fault detection. To address the problem, a detector based on the YOLOv5s model is applied to a custom dataset representing the problem. This section aims to describe the procedure and decisions made when constructing the fastener detector model from collecting the data to evaluating the results. The practical part of the method can be divided into three stages: data preprocessing, model development, and model evaluation. An overview of the steps followed in the method is visualized in Figure 14 and will be further explained in this section.

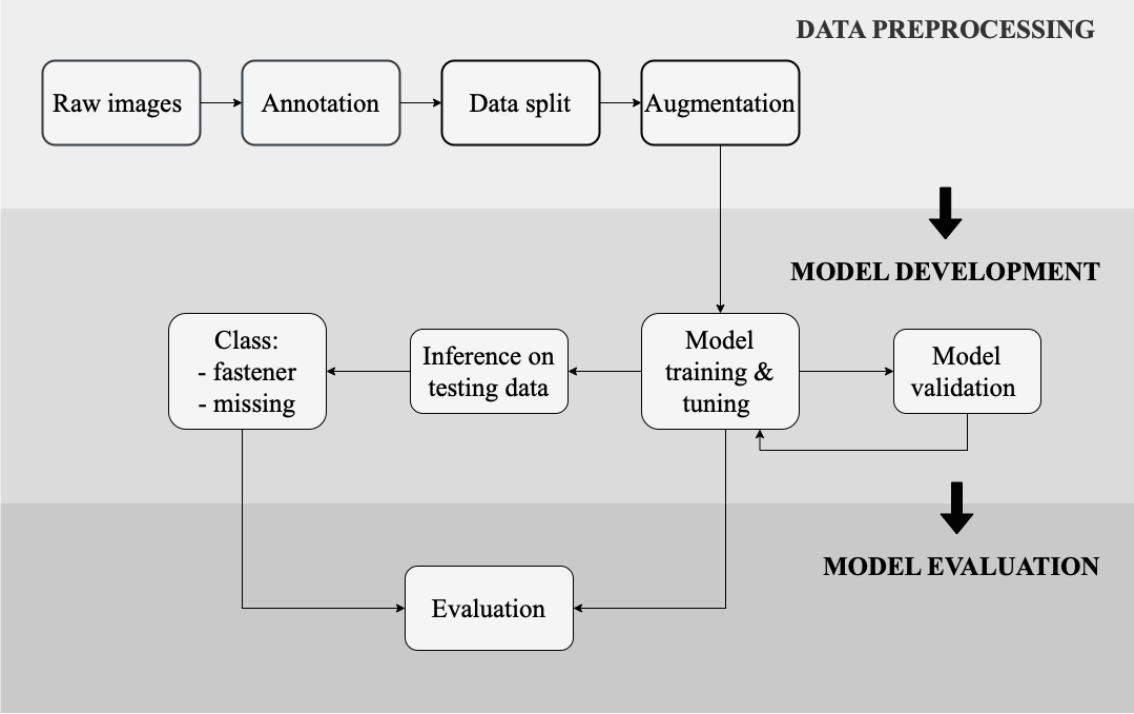

Figure 14. The structure of the process involves data preprocessing, model development, and model evaluation.

4.1 The Dataset

The dataset used is collected specifically for the project. The dataset consisted of images and videos displaying both the left and the right rail from above. The data was gathered from a 30-meter-long test railway located outside Gothenburg, with a GoPro 8 hero mounted on a mobile trolly placed on the rails with a fixed height of 35.5 cm, displayed in Figure 15. The trolly were manually pushed at different speeds to create variation in the dataset and mimic the actual environment.

Figure 15. GoPro mounted on the trolly.

To capture different scenarios, some fasteners were manually removed. The different scenarios are displayed in Figure 16 and are the following: both fasteners missing, both fasteners in place, right fastener missing, and left fastener missing. To create a natural variation in the data set, stones were randomly placed on some of the sleepers and the fasteners. The detached fasteners were placed near the rails, still visible in some of the images displaying the missing fastener cases. This was done to create variation in the dataset and recreate real scenarios.

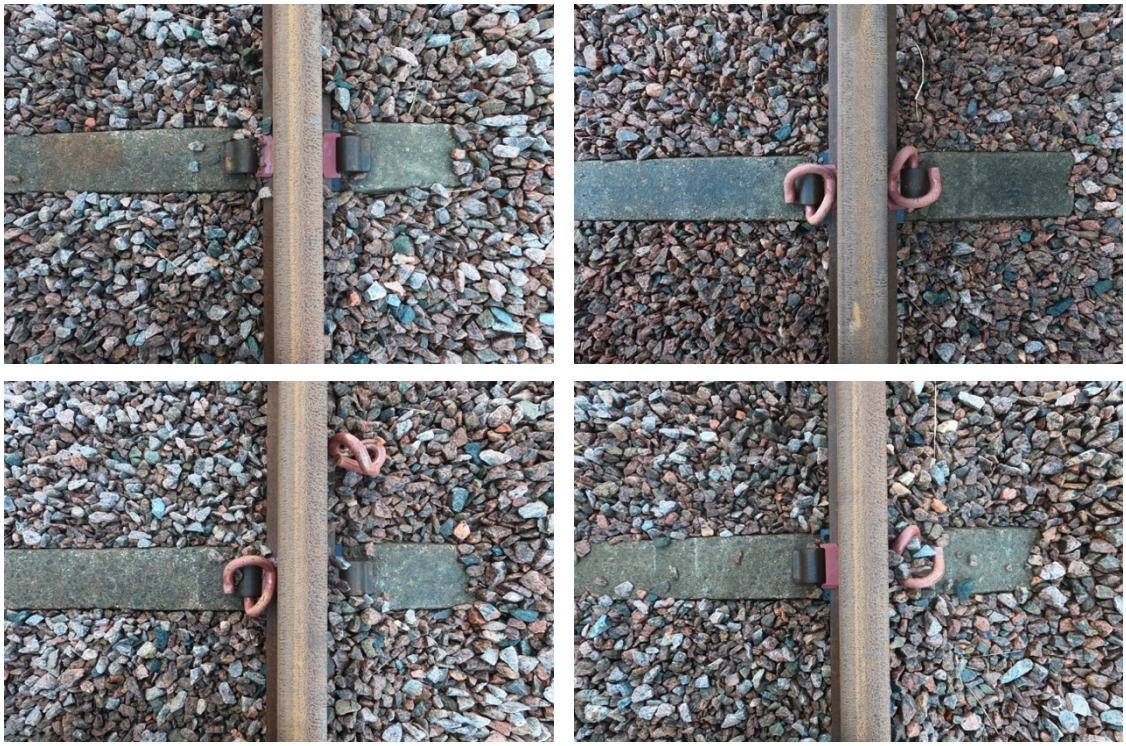

Figure 16. The scenarios captured in the data gathering. From top left: both fasteners missing, both fasteners in place, right fastener missing, left fastener missing.

The data collection involved some delimitations in the data. The test railway only contained one type of fasteners, sleepers, and ballast. Another aspect is the weather condition. Since the railway in Sweden is used during the entire year, both day and night, the environment and lighting change tremendously. The data, however, were gathered during one event, in only one type of environment. No snow, grass, or similar environmental changes were therefore captured.

All images were used in the dataset. From the videos, frames displaying the different cases were manually extracted to create a bigger dataset and a more equal division among the classes. In total 437 images were used. All images were also auto-oriented and resized to 416x416 to fit the YOLO model.

4.1.1 Annotation

The data was annotated by manually applying bounding boxes around the position of fasteners and missing fasteners and adding labels depending on two classes: *fastener* and *missing*. The bounding boxes were placed as close to the fasteners or missing fasteners as possible but with a part of the sleeper and rail visible. This was done to annotate the correct position of the different cases. The images used for the object detection were not cropped. Therefore, the number of sleepers, and by that, the number of fasteners, in the images varied. The fasteners and position of the missing fasteners were therefore only annotated when the full position was displayed. Approximately 7% of the data, 30 images in total, consisted of images without any sleepers, which were used as non-label cases, so-called background images. In total, 910 annotations were created in the dataset of 437 images. Out of 910 annotations, 658 annotations were labeled as a *fastener* and 252 annotations were labeled as *missing*. The distribution of the different annotations can be found in Table 2.

Table 2. Distribution of labels.

Total number of annotations	Fastener	Missing
910	658	252
100%	72%	28%

4.1.2 Dataset Split

A dataset is traditionally divided into three categories for image analysis purposes: training, validation, and testing. The network uses the training data for training, meaning that the network uses the images' pixels to learn about patterns and to construct a model whose weights can categorize and localize the fasteners. The validation data is used during training to validate the performance of the current weights of the network after every batch. The network uses the result to retrain and learn how to increase the

performance even more. Finally, the best weights from the training phase are applied for inference on test data to evaluate the final model's performance on an unseen portion of the dataset. Since the test data is separated from the training data, it enables a performance test of the model that resembles a test in production.

The dataset is split randomly into 60% training data, 20% validation data, and 20% test data. The ratio between the different sets is good practice for smaller datasets [65]. The division among the classes in the different datasets is manually inspected to represent all the classes in every dataset. The distribution and number of images in the different datasets are displayed in Table 3.

Table 3. Images are split into training, validation, and testing.

Total number of images	Training	Validation	Test
437	261	88	88
100%	60%	20%	20%

4.1.3 Data Augmentation

Data augmentation is a technique used to increase the amount of data by adding slightly modified copies of already existing data. The purpose of the technique is to reduce overfitting by increasing the dataset and to mimic variations of the real-world environment by creating images with modified features compared to images collected during the data gathering process. In addition to the 261 original training images another 522 images were created by augmentation leading to a total number of 783 training images. For each image, a maximum of three augmented versions were generated by randomly applying blur (up to 4px), resizing (cropping from 19% minimum zoom to 67% maximum zoom) and changes in brightness (variations of 35%), examples of these are displayed in Figure 17. The decisions of applying blur, resizing, and brightness were based on the natural environment's impact on the image quality in production. In production, the images might be affected by motion blur as images might be taken during speed along the railway. Therefore, blur was added to the images as the resolution of the images will be affected by the motion. The camera might also not be mounted exactly 35.5 cm above the track as in this case, which resizing enables to simulate. The deviation in brightness catches the deviation of sunlight during the day as the dataset was collected during daylight, which might not always be the case in production. In many cases, horizontal mirroring and flips are applied to images as augmentation techniques. In the case of the fastener images, the images are symmetric around the rail if both fasteners exist or are missing. The camera will also always be mounted in the same direction as the movement of the train, leading to no purpose of applying these augmentation techniques to fastener images since a horizontal mirroring or flips will not cover a new scenario in the real environment.

Figure 17. Examples of augmentation. From the top left; 19% crop, 54% crop, +35% brightness, -35% brightness, 0px blur, 4px blur.

Images are further augmented during training by YOLOv5. For each training batch, YOLOv5 passes the training data through a data loader which augments the data. The data loader applies three different augmentation techniques: scaling, color shape adjustments, and mosaic augmentation. Mosaic augmentation, displayed in Figure 18, is a novel technique that combines four images into four tiles of random ratio. The techniques improve the model's ability to detect small objects which for the COCO dataset has been proven to not be detected as accurately as larger objects. Mosaic augmentation can be visualized in Figure 19. Data augmentation is finally applied to the test data with so-called test time augmentation. The idea is to try the model's robustness by applying augmentation to test images during inference. The images are then left-right flipped and processed at three different resolutions.

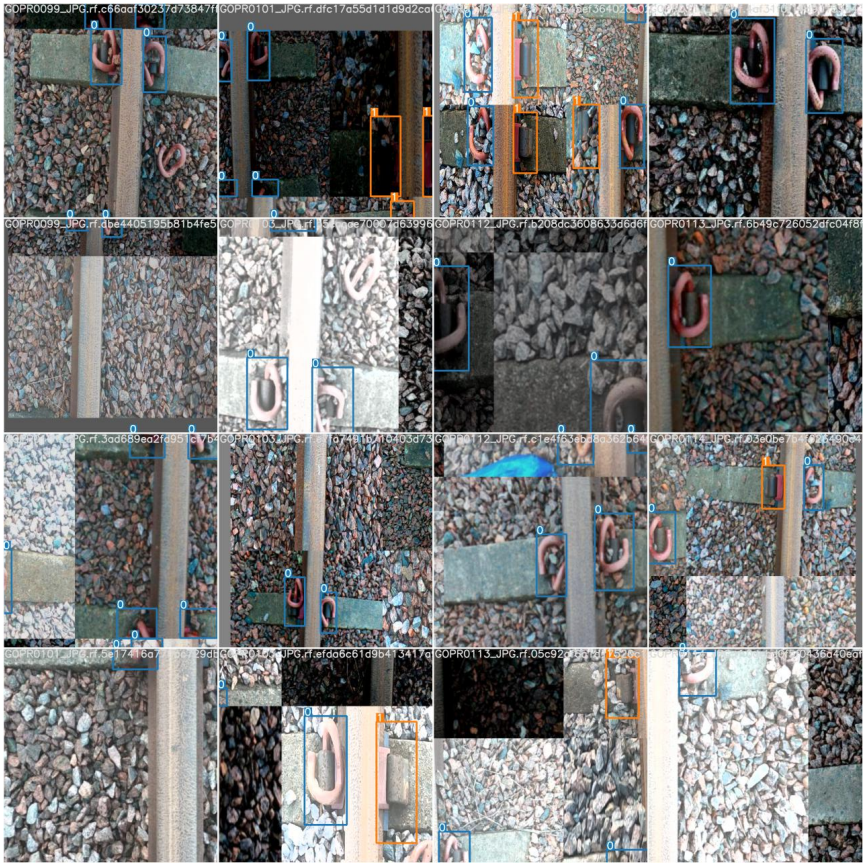

Figure 18. Mosaic augmentation in YOLOv5 on the dataset.

4.2 Model Selection

In the initial stage of the process, image classification was implemented to classify the images into the different scenarios; both fasteners in place, left fasteners missing, and right fastener missing, and both fasteners missing. The image classification method using the VGG16 network required a thorough preprocessing of the data. The preprocessing involves automatically cropping the images at the right segment so that the model can classify the images based on the area of interest. This can be accomplished by finding the intersection between the sleeper and the rail using a model-based approach and cropping the image based on the location of the intersection. Since the sleeper and the rail are structured as straight lines the Hough transform, where shapes are found using a voting procedure, was implemented to extract these features. The implementation of the Hough transform failed due to too much blur in the images caused by the bank gravel in the background. The shape of the rail was determined but the shape of the sleeper was not captured. Since the dataset is relatively small, cropping could be done manually but will limit the scalability of the model when using bigger datasets or in production, and is therefore not executed. In addition to the Hough transform, other segmentation techniques such as edge detection, contour detection, and K-means were tested with the same results. Figure 19 displays the challenge of segmenting the fastener region due to noise caused by the gravel.

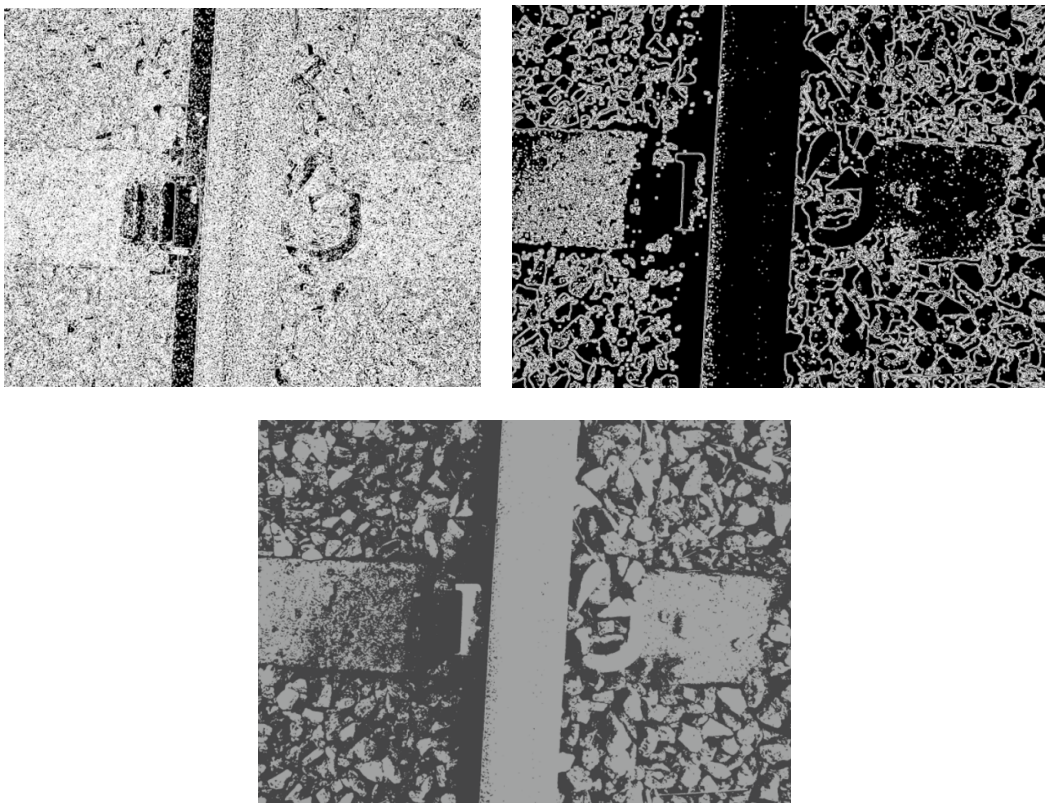

Figure 19. Edge detection using Canny (a), Contour detection (b), and K-means (c)

To keep the model more scalable and robust, a learning-based approach was chosen for detecting the fasteners. Further research also shows the advantages and great performance made by using deep learning for detecting features. There are several well-known algorithms within deep learning such as SSD, RetinaNet, R-FCN, and FPN. The series of YOLO models have the advantage of higher performance in terms of accuracy and speed. Most importantly, YOLOv5 is specifically good at detecting small objects, which fasteners can be categorized as. Different state-of-the-art object detection models have been compared multiple times and YOLO has proven to be the optimal method for the recognition of small objects in terms of speed and accuracy. Previous studies have used YOLO mode on the task of fastener fault detection with great results which gives evidence for using YOLO in this thesis [7, 10, 16, 8]. YOLOv5 is the latest version in the series of constantly improved models. YOLOv5s is the version of YOLOv5 selected as it is the smallest and fastest model available [66].

4.3 Technical Architecture

The YOLOv5 algorithm is provided by Ultralytics and is based on the PyTorch library. It is coded in Python and is hosted by the developers on the open-source platform Github. Pytorch is an open-source machine learning framework for deep learning developed by Google Brain, a research team at Google AI. The development is implemented using Combine's internal server Metallica in Jupyter Notebooks. Jupyter Notebook is a free software for interactive coding, visualizations, and computational outputs displayed with explanatory text supporting all programming languages.

The dataset was annotated in the YOLO annotation format and augmented using the platform Roboflow. Roboflow is a platform used to upload, organize, annotate and augment computer vision datasets. The dataset is split into a training set, validation set, and test set in Roboflow and exported as a YOLOv5 PyTorch text file to the Jupyter Notebook.

To track the results of different runs and visually compare them to each other, Weights & Biases are used to evaluate the results. Weights & Biases is a platform used to track, compare, and visualize machine learning models. The collaborative dashboards enable experiment tracking, hyperparameter optimization, and interactive data visualization.

4.4 Model Optimization

YOLOv5 is a model developed to be practical for many purposes. It is pretrained on the COCO dataset consisting of over 330 000 images displaying 1.5 million objects. To assess the model's performance, different settings to both the dataset and the model's architecture were tested. The different settings of the model were divided into three categories: dataset settings, transfer learning settings, and hyperparameter settings. The process of evaluating and asses YOLOv5 as a model was therefore divided into these three categories followed by a comparison and final evaluation of the best performing models. For every test case, the maximum $mAP_{[0.5:0.95]}$ was extracted, and the belonging weights were stored. The weights were then used for inference on the test data. The $mAP_{[0.5:0.95]}$ from testing was stored as well together with other metrics such as the confusion matrix and the F1-curve. The weights were also used on test data together with test time augmentation and visibly inspected to investigate where the model failed to detect the missing or existing fastener. The default settings of a model, if nothing else is specified, have the properties presented in Table 4.

Table 4. Default settings of the tested models.

Default settings
Augmented images
Background images
Random weights
Batch size 16
100 epochs

4.4.1 **Dataset Evaluation**

When implementing YOLOv5 on a new dataset, it is important to provide a dataset large enough for the model to be retrained on. It is also important to have consistent annotation and a dataset that well represents the object of interest. In the case of fastener

detection, the dataset was adapted to the situation but to further evaluate different aspects of the dataset, runs were executed both with and without augmentation and with and without background images. The overall goal of object detection is to detect the objects of interest in an image, but it is also important to not detect the object if the object is not present in the image. To simulate the cases where the object is not present in the images, background images were added to the dataset. Background images are images without any object and therefore no bounding boxes. Increasing the size of the dataset and creating more variation in the images with augmentation has proven to improve the model's performance. To be convinced by the impact, YOLOv5 was tested on the original raw dataset as well in this phase. The optimal settings regarding the dataset were chosen for the next phases of investigating transfer learning and hyperparameters.

4.4.2 Transfer Learning Evaluation

Apart from the dataset, it is important to evaluate different settings for the YOLOv5 algorithm. In this phase, the impact of transfer learning were tested. The Ultralytics team, the developers behind YOLOv5, recommend initially using the default settings of YOLOv5. However, the model's robustness to changes to the default settings was investigated. The use of transfer learning was tested by using pre-trained weights with and without freezing some or all the layers. By freezing layers, the frozen layers are prevented from training on the target dataset by freezing the initial weights. YOLOv5 is pretrained on the COCO dataset but all weights are normally continuously tuned depending on the features of the target dataset. By freezing the initial weights, only the remaining weights are tuned on the new dataset while training the network. This requires fewer resources than normal training and allows for faster training time, but it may result in a negative transfer, i.e., a reduction of final trained accuracy.

4.4.3 Hyperparameter Evaluation

Finally, different values of the hyperparameters *number of epochs*, and *batch size* were tested and evaluated. By increasing the number of epochs, one allows the algorithm to continue to learn and, in some cases, improve its performance. If a higher number of epochs results in greater performance, it is of interest to keep the number of epochs high. If the performance converges at lower epochs, it is more efficient in terms of GPU power and training time to keep the number of epochs low. Both 100 epochs and 300 epochs were tested in this case. The batch size is recommended to be as high as the GPU power allows. To test the batch size impact on the performance, both the default setting of a batch size of 16 was tested, followed by test cases with 32 and 64 in batch size.

4.4.4 Model Comparison

In the dataset, transfer learning, and hyperparameter evaluation, different settings were tested on the model in different runs, each presented in Table 5. Every run corresponds

to the training and evaluation of the model. For every epoch, measurements such as $mAP_{[0.5:0.95]}$, precision, recall, and different loss function values are given. After the run, a summary of the model's maximum performance in detecting the correct object in the evaluation dataset is given. These metrics give a strong indication of the performance of the model in total. The model's maximum $mAP_{[0.5:0.95]}$ during training was stored for comparing the different cases' performance. Furthermore, the weights corresponding to the epoch with the maximum $mAP_{[0.5:0.95]}$ were used for inference on the test data. Again, the $mAP_{[0.5:0.95]}$ score and other metrics were registered to be able to compare the different model's performances. The $mAP_{[0.5:0.95]}$ based on the IOU thresholds in the interval 0.5:0.95 is used as the main metric for comparing the different models between each other. The reason for this is that the $mAP_{[0.5:0.95]}$ score showed the most variation in performance of the different models and is therefore convenient for comparison. In the stage of evaluating the models, it is of great interest and importance to understand where the models fail in terms of wrong class predictions or misclassification of the object. To ensure an adequate inspection, the test result was visibly printed in addition to the result of evaluation metrics. In the case of fastener detection, it is of greater interest to classify the object correctly compared to localizing the object with identically bounding boxes as the annotation. Therefore, greater focus was invested in correct classification loss compared to localization loss.

Since the performance on test data is of greater importance than the training and validation data, the result was also visibly inspected to investigate where the model failed to detect the missing or existing fastener. The testing was executed with a threshold of 0.75 and together with test time augmentation to challenge the models' robustness.

Table 5. An overview of the tested models.

Evaluation goal	Model name
Dataset evaluation	yolov5s_random
	yolov5s_random_no_augmentation
	yolov5s_random_no_background_images
Transfer learning evaluation	yolov5s_pretrained
	yolov5s_pretrained_freeze_10
	yolov5s_pretrained_freeze_all
Hyperparameter evaluation	yolov5s_random_300_epochs
	yolov5s_random_batchsize_32
	yolov5s_random_batchsize_64

5. Result

The result section begins with the $mAP_{[0.5:0.95]}$ results from all the models described in section 4.4.4 Model Comparison. The $mAP_{[0.5:0.95]}$ scores initially display the final $mAP_{[0.5:0.95]}$ score from training and testing, followed by the $mAP_{[0.5:0.95]}$ curves from the dataset evaluation, transfer learning evaluation, and hyperparameter evaluation, as well as the GPU utilization. Continuously, results from some models' performance on test data are presented in terms of confusion matrices, F1-curves, and examples of correct and incorrect predictions.

5.1 Mean Average Precision

This section aims to describe the results given by the models proposed in Section *4.4.4 Model Comparison*. The default settings of the different models are batch size 16, 100 epochs, random weights, image augmentation, and background images. This model is referred to as *yolov5s_random*, and all the other models differ from the base model given by its name.

Table 6. $mAP_{[0.5:0.95]}$ for the models during training and testing.

Model name	Train $mAP_{[0.5:0.95]}$	Test $mAP_{[0.5:0.95]}$
yolov5s_random	0.743	0.696
yolov5s_random_no_augmentation	0.715	0.658
yolov5s_random_no_background_images	0.741	0.690
yolov5s_pretrained	0.747	0.719
yolov5s_pretrained_freeze_10	0.744	0.688
yolov5s_pretrained_freeze_all	0.351	0.293
yolov5s_random_300_epochs	0.747	0.693
yolov5s_random_batchsize_32	0.750	0.692
2yolov5s_random_batchsize_64	0.748	0.693

Table 6 displays the result from the best $mAP_{[0.5:0.95]}$ obtained during the training phase and the $mAP_{[0.5:0.95]}$ obtained when testing on the test data. The model with 32 in batch size has the best training $mAP_{[0.5:0.95]}$ and the model using pre-trained weights has the best testing $mAP_{[0.5:0.95]}$.

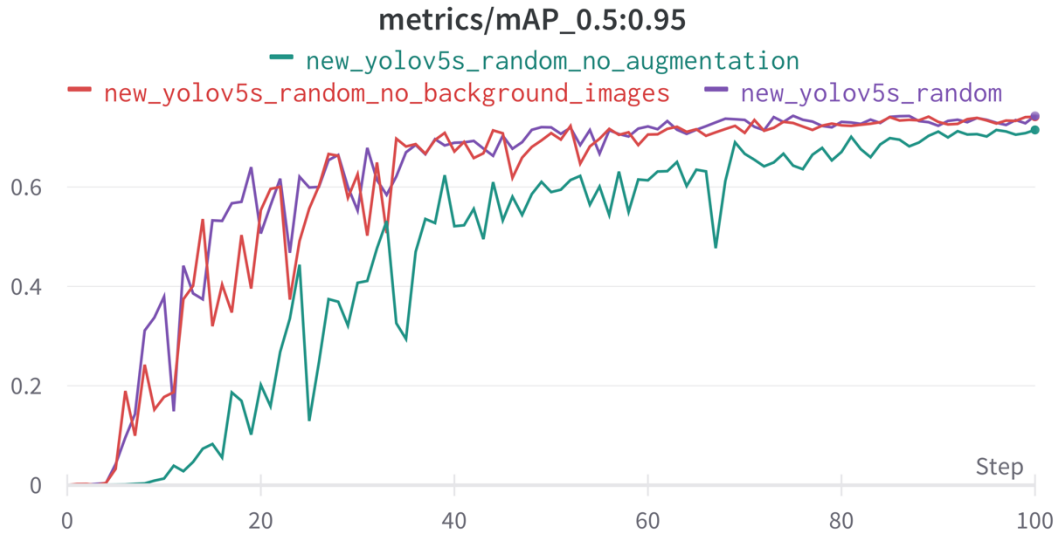

Figure 20. Training $mAP_{[0.5:0.95]}$ curve for models during dataset evaluation.

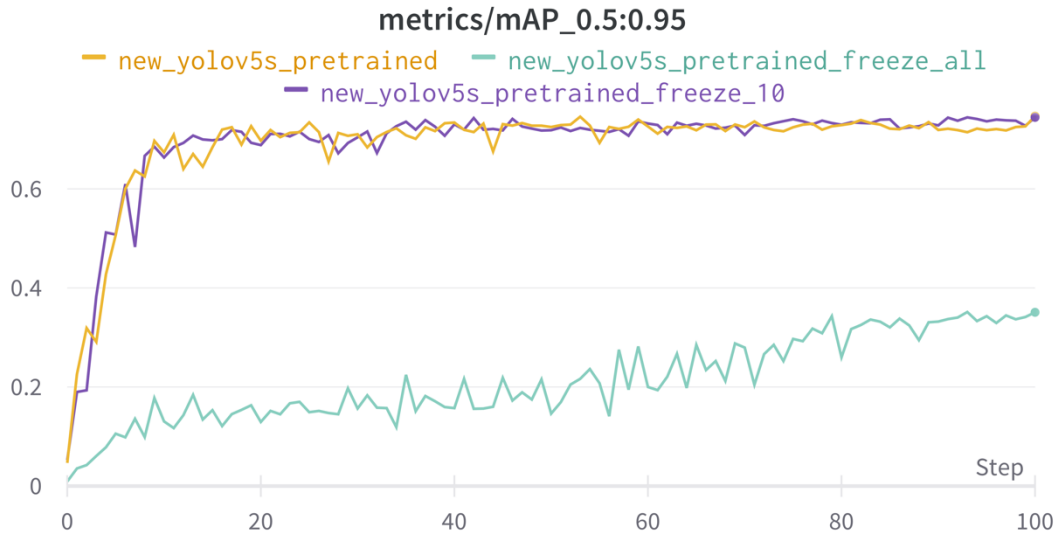

Figure 21. Training $mAP_{[0.5:0.95]}$ curve for models during transfer learning evaluation.

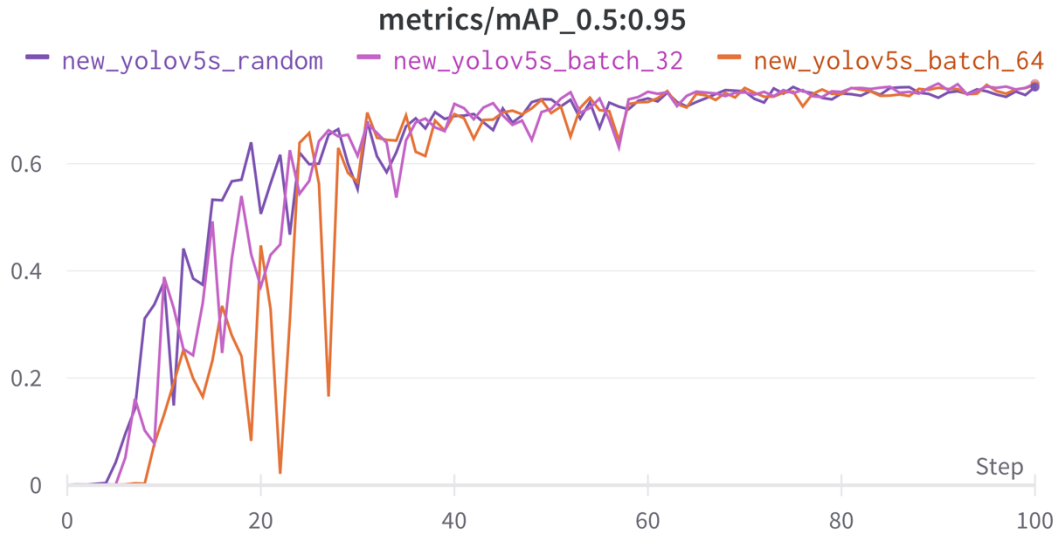

Figure 22. Training $mAP_{[0.5:0.95]}$ curve for models during the evaluation of the batch size.

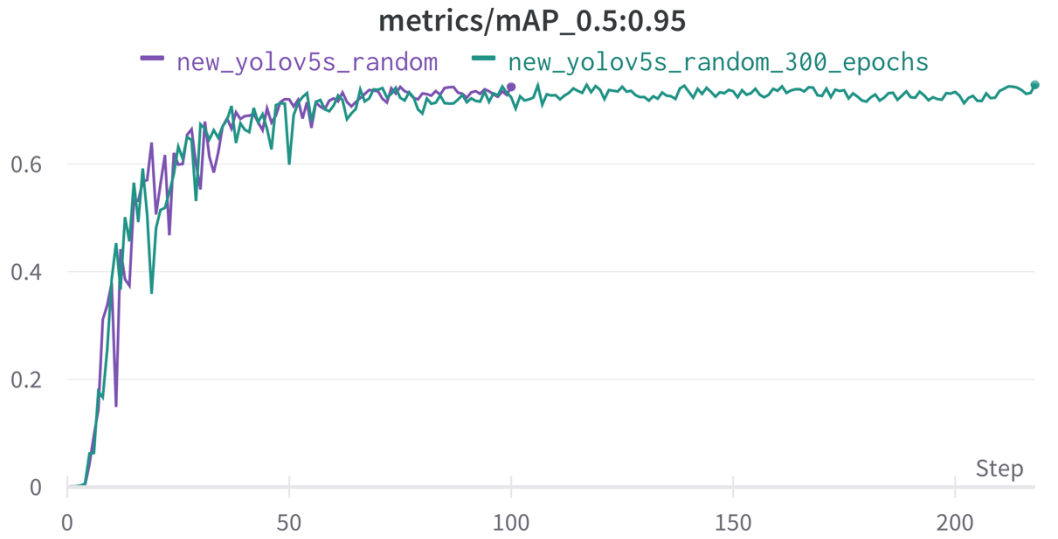

Figure 23. Training $mAP_{[0.5:0.95]}$ curve for models during the evaluation of a number of epochs.

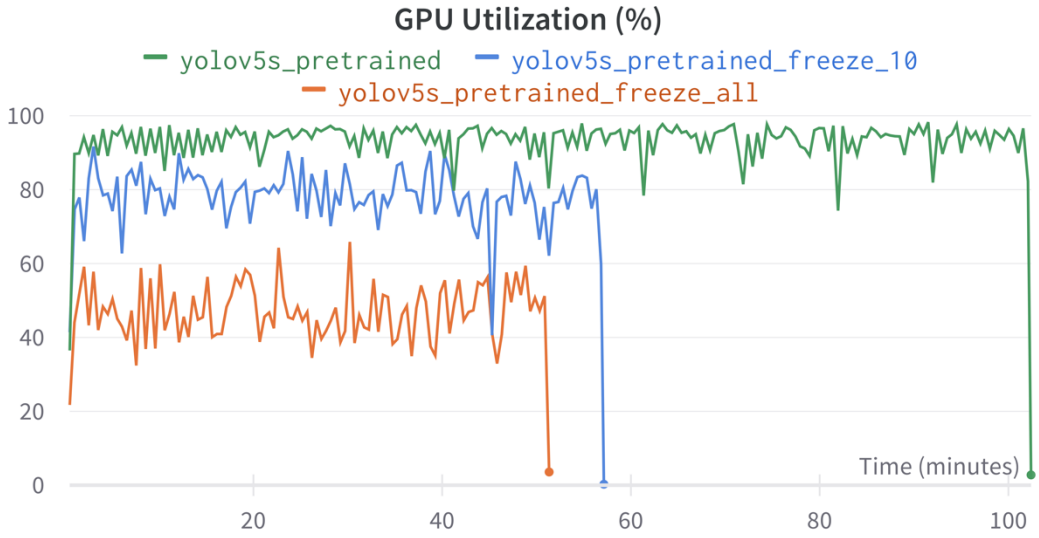

Figure 24. GPU utilization (%) and training time for models during transfer learning evaluation.

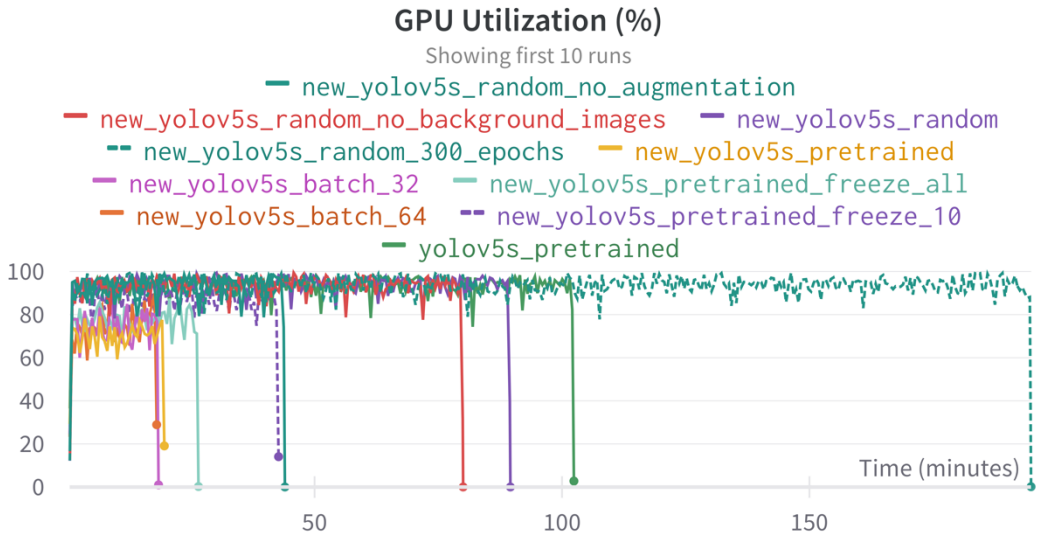

Figure 25. GPU utilization (%) and training time for all the models evaluated.

Figures 20-23 display the different $mAP_{[0.5:0.95]}$ curves obtained when training the different models when evaluating different datasets, transfer learning, and hyperparameters. Figure 25 displays the GPU utilization and training time for all the models evaluated and Figure 24 presents the GPU utilization and training time for the transfer learning evaluation specifically.

Table 7. $mAP_{[0.5]}$, precision and recall for some models during training.

Model name	Train $mAP_{[0.5]}$	Precision	Recall
yolov5s_pretrained	0.995	0.993	1
yolov5s_pretrained_freeze_10	0.991	0.976	0.988
yolov5s_random_batchsize_32	0.995	0.996	1

Table 7 displays precision, recall and train $mAP_{[0.5]}$ for some models whose settings had a positive impact on the results.

5.2 Confusion Matrix

This section presents the confusion matrices of the models with the highest performance in terms of $mAP_{[0.5]}$ and GPU usage**,** namely the models with batch size 32 *yolov5s_random_batch_32*, pre-trained weights *yolov5s_pretrained,* and pre-trained weights with 10 frozen layers *yolov5s_pretrained_freeze_10.*

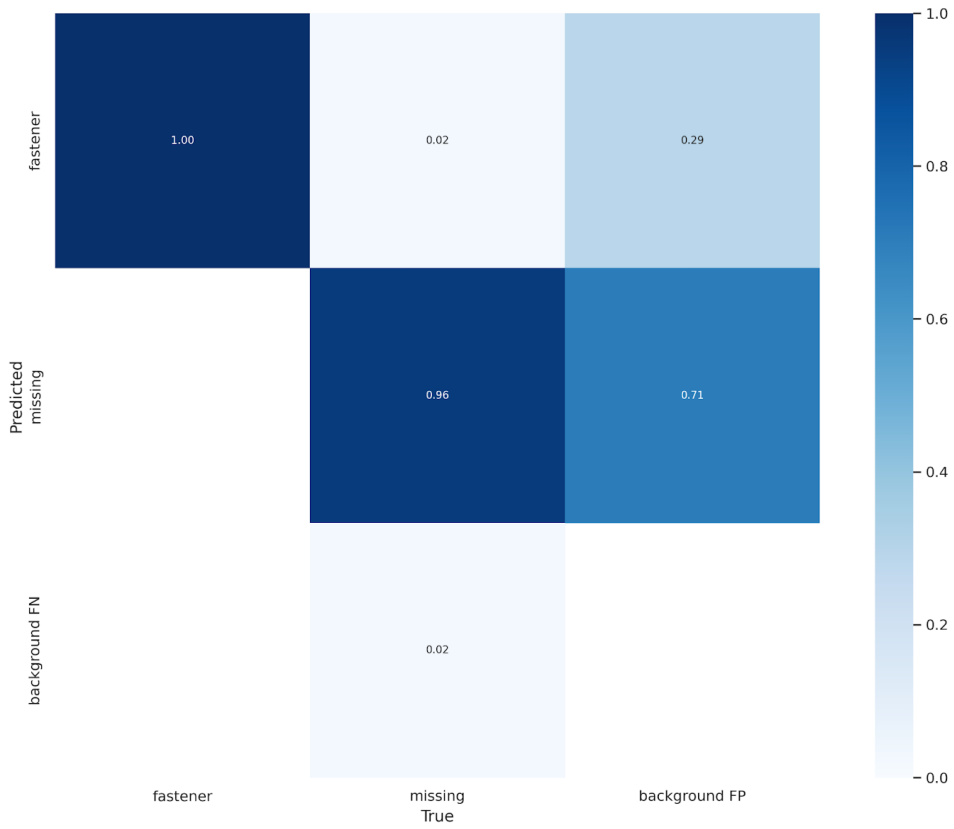

Figure 26. Confusion matrix of yolov5s_random_batch_32 with IOU threshold at 0.6.

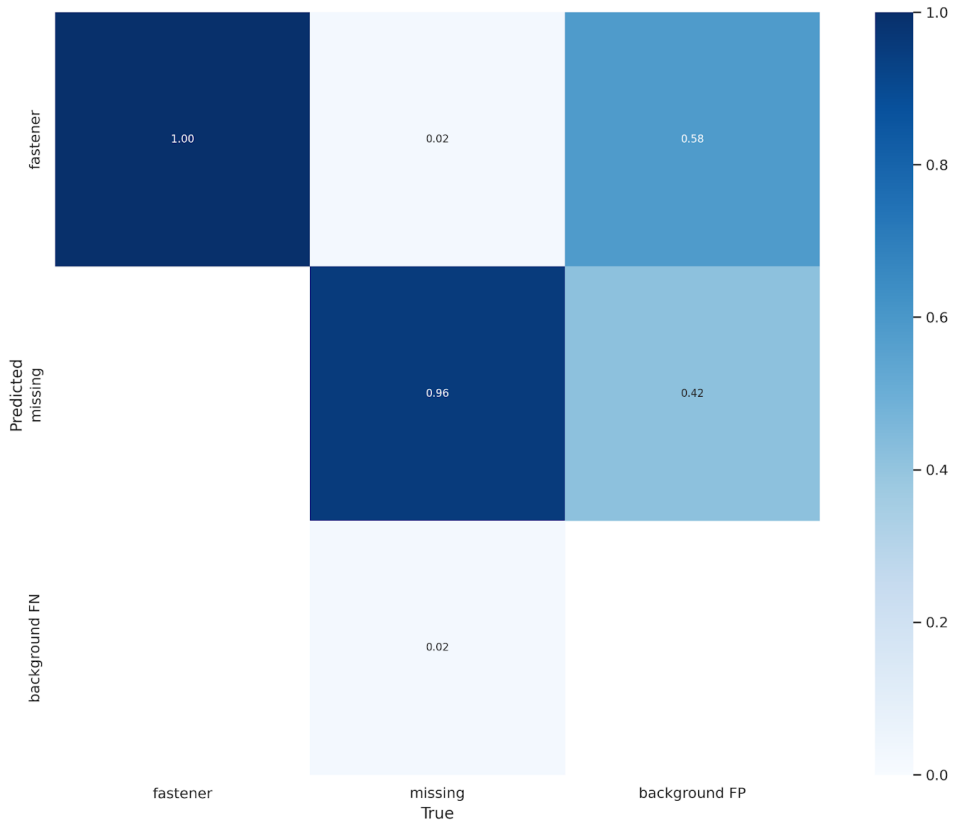

Figure 27. Confusion matrix of yolov5s_pretrained with IOU threshold at 0.6.

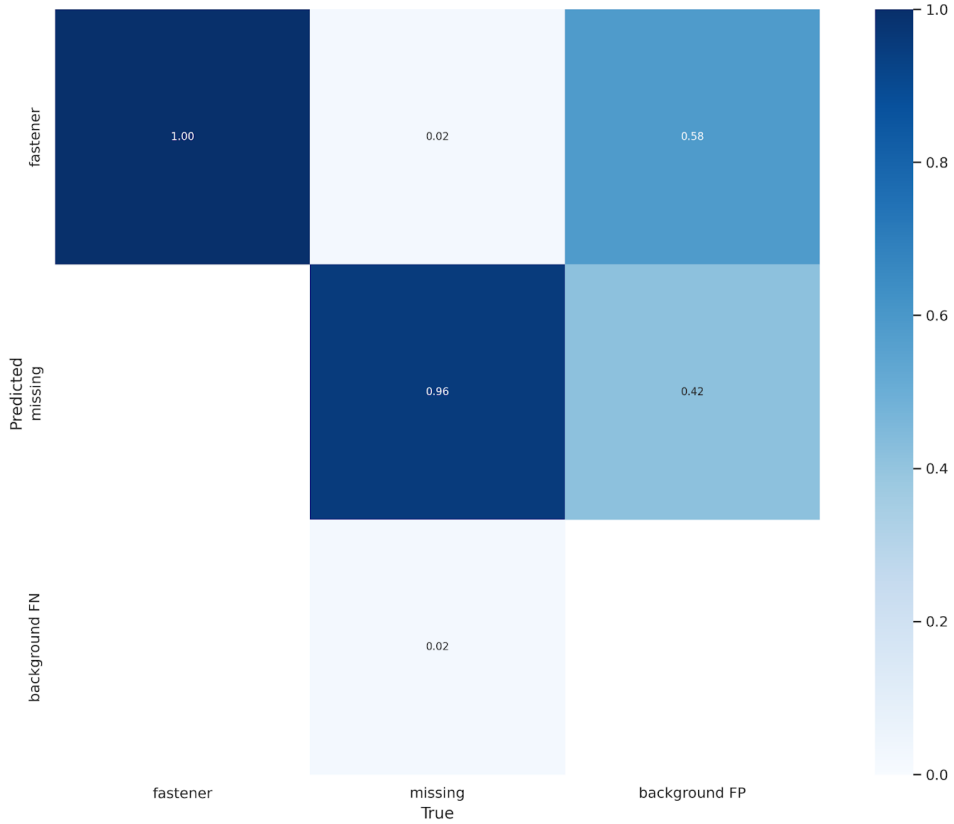

Figure 28. Confusion matrix of yolov5s_pretrained_freeze_10 with IOU threshold at 0.6.

All three models succeed to classify all true fasteners as fasteners and 96% of the true missing fasteners as missing. The only difference in the result is the predictions of the background images, and false predictions of fasteners and missing fasteners.

5.3 F1-curve

This section presents the F1-curves belonging to *yolov5s_random_batch_32, yolov5s_pretrained* and, *yolov5s_pretrained_freeze_10*, presented in Figure 29-31.

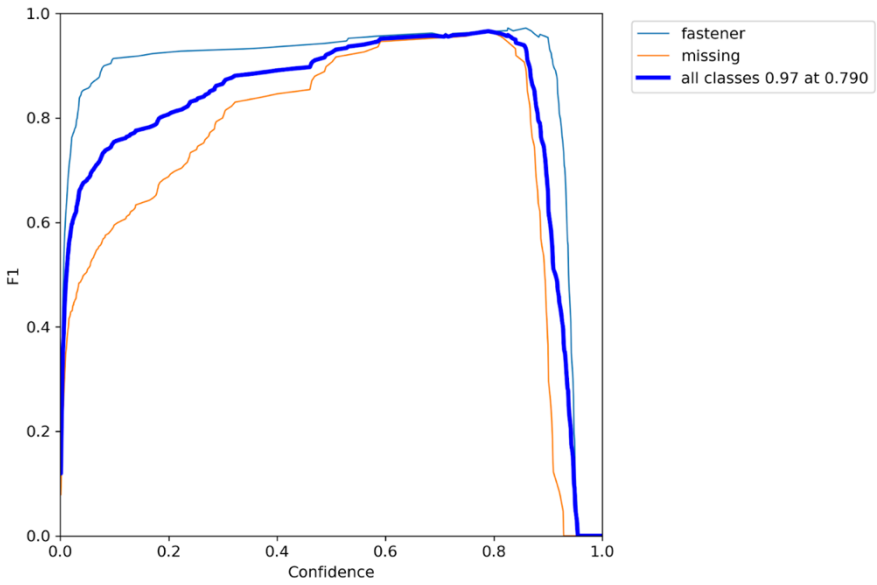

Figure 29. F1-curve for yolov5s_random_batch_32 with IOU threshold at 0.6.

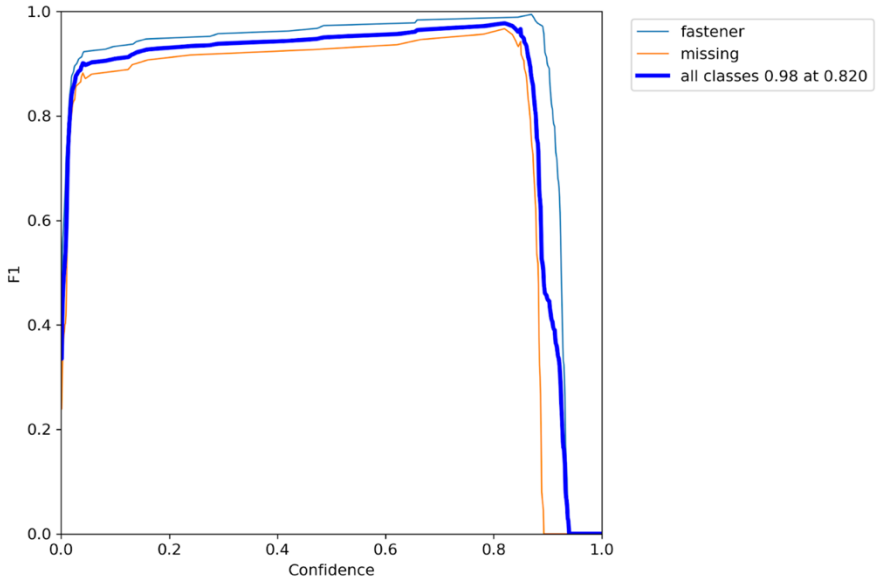

Figure 30. F1-curve for yolov5s_pretrained with IOU threshold at 0.6.

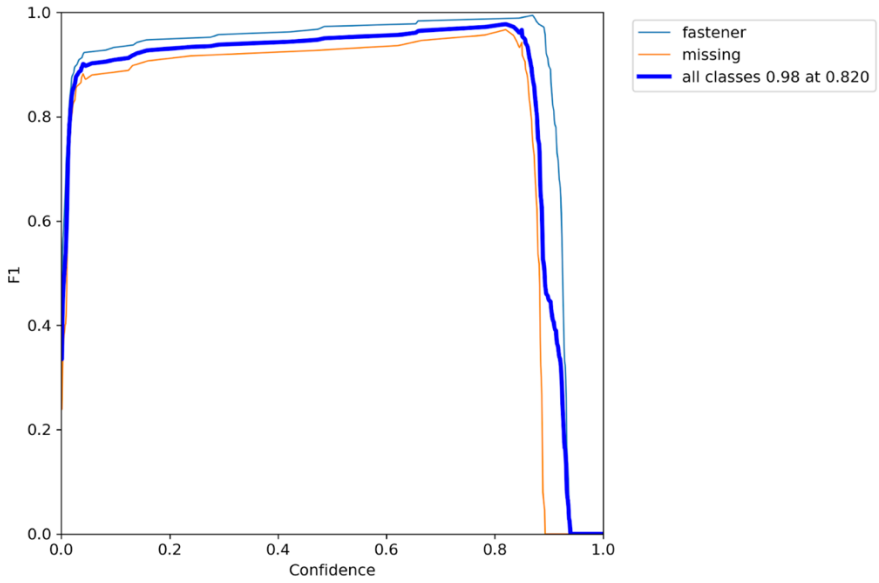

Figure 31. F1-curve for yolov5s_pretrained_freeze_10 with IOU threshold at 0.6.

5.4 Test Images

This section presents a subset of the result when inferencing on the test images. Initially, examples of different images where the detectors succeed to detect the objects are presented in Figures 32-36. This is followed by examples of cases where the detectors fail on the task of detecting the correct objects, displayed in Figures 37-40.

Figure 32. The detectors succeed to detect fasteners on both sides. From left; yolov5s_random_batchsize_32, yolov5s_pretrained, yolov5s_pretrained_freeze_10. IOU threshold at 0.75.

Figure 33. The detectors succeed to detect one fastener and one missing fastener. From left; yolov5s_random_batchsize_32, yolov5s_pretrained, yolov5s_pretrained_freeze_10. IOU threshold at 0.75.v

Figure 34. The detectors succeed to detect the missing fasteners on both sides. From left; yolov5s_random_batchsize_32, yolov5s_pretrained, yolov5s_pretrained_freeze_10. IOU threshold at 0.75.

Figure 35. The detectors succeed in not detecting any objects in background images. From left; yolov5s_random_batchsize_32, yolov5s_pretrained, yolov5s_pretrained_freeze_10. IOU threshold at 0.75.

Figure 36. The detectors succeed in not classifying the detached fastener next to the rail. From left; yolov5s_random_batchsize_32, yolov5s_pretrained, yolov5s_pretrained_freeze_10. IOU threshold at 0.75.

The following images, in Figures 37-41, are examples of when the detectors fail to classify the correct objects in the images. The corresponding annotated image is presented to the right, relative to the classified images.

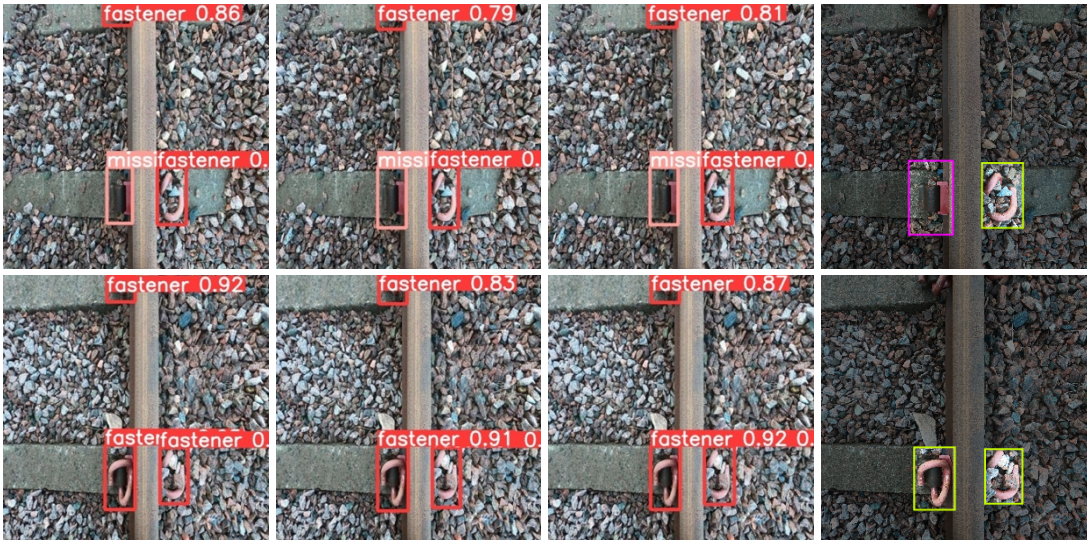

Figure 37. The detectors classify parts of the upper sleepers' fastener as a fastener. From left; yolov5s_random_batchsize_32, yolov5s_pretrained, yolov5s_pretrained_freeze_10 and the corresponding annotation. IOU threshold at 0.75.

Figure 38. The detectors classify parts of the lower sleepers' missing fasteners as missing fasteners. From left; yolov5s_random_batchsize_32, yolov5s_pretrained, yolov5s_pretrained_freeze_10, and the corresponding annotation. IOU threshold at 0.75.

Figure 39. From left; yolov5s_random_batchsize_32, yolov5s_pretrained, yolov5s_pretrained_freeze_10, and the corresponding annotation. yolov5s_random_batchsize_32 and yolov5s_pretrained_freeze_10 classifies part of a fastener in a background image as a fastener. IOU threshold at 0.75.

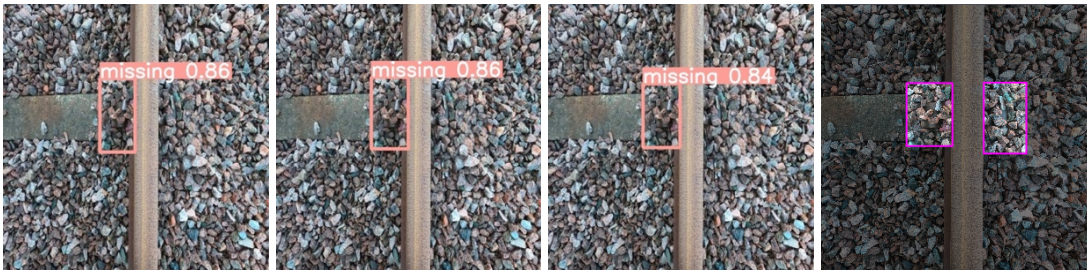

Figure 40. A missing fastener is hidden under stones on the right side of the track. The detectors do not classify it. From left; yolov5s_random_batchsize_32, yolov5s_pretrained, yolov5s_pretrained_freeze_10, and the corresponding annotation. IOU threshold at 0.75.

6. Discussion

The results presented in section 5 is discussed and analyzed in this section. The discussion begins with comparing the different models with each other in terms of settings and performance by analyzing the train $mAP_{[0.5:0.95]}$ and test $mAP_{[0.5:0.95]}$. Further, three models whose settings had a positive impact on the performance are discussed by analyzing the corresponding confusion matrices, F1-curves, and inference on the test data. The section is finished by discussing YOLOv5 performance as a model for the given problem by comparing with previous studies' results.

6.1 Comparison of Result and Model Optimization

The objective of this thesis is to investigate how machine learning can be applied to the problem of automatic fastener fault detection, and the selected model for the purpose is YOLOv5s. To investigate how YOLOv5s can be applied to the problem, different settings to the YOLOv5s model in terms of the input (dataset), weights (transfer learning), and hyperparameters were compared. The main metric to evaluate the impact of the different settings was $mAP_{[0.5:0.95]}$.The $mAP_{[0.5:0.95]}$ score was calculated during training and testing and is used to compare the different models with each other. The test $mAP_{[0.5:0.95]}$ is in general lower than the train $mAP_{[0.5:0.95]}$ which can be explained by the fact that the test images are the unseen portion of the dataset and therefore slightly differ from the dataset used when training. This section aims to discuss how the different settings of the dataset, transfer learning, and hyperparameters affected the model. The $mAP_{[0.5:0.95]}$ values given in this section are therefore singularly used to comment on the different settings' performance in comparison with each other, and not used to comment on the performance in general. Parts of the results in Table 6 in the *Result* section is re-displayed in Table 8.

Table 8. $mAP_{[0.5:0.95]}$ for the dataset models during training and testing.

Model name	Train $mAP_{[0.5:0.95]}$	Test $mAP_{[0.5:0.95]}$
yolov5s_random	0.743	0.696
yolov5s_random_no_augmentation	0.715	0.658
yolov5s_random_no_background_images	0.741	0.690

When investigating how machine learning can be applied to automatic fastener fault detection it is important to evaluate the dataset for the given problem. The dataset is the core of the training and should as close as possible reflect the real-world problem, therefore augmentation and the addition of background images were added to the dataset to create variation. To evaluate the effect of image augmentation and background images, three different models relying on three different datasets were tested during the model evaluation. The first model, *yolov5s_random,* consisted of augmented images and background images, and random refers to the use of random

weights. The two other datasets, *yolov5_no_augmentation,* and *yolov5_no_background_images* differ from *yolov5s_random* by the reduction of augmentation and background images respectively. The datasets were trained by the YOLOv5 algorithm with randomly initialized weights, batch size 16, and over 100 epochs. The results in Table 6, also displayed in Table 8, and Figure 20 show that the dataset without augmentation had an $mAP_{[0.5:0.95]}$ of 0.715 during training and an $mAP_{[0.5:0.95]}$ of 0.658 on the test dataset, and the dataset with augmentation had an $mAP_{[0.5:0.95]}$ of 0.743 during training and an $mAP_{[0.5:0.95]}$ of 0.693 on the test dataset. This indicates that image augmentation improves the model's performance. The addition of background images is proven in the literature to increase performance in general. A marginal improvement was shown during evaluation as the $mAP_{[0.5:0.95]}$ without background images was 0.741 during testing and 0.690 on the test dataset. In conclusion, the optimal dataset was the one with both image augmentation and background images, i.e. *yolov5_random* as image augmentation and the addition of background images improved the performance. The model *yolov5s_random* was further evaluated by changing the default settings to the hyperparameters batch size and epochs, and by using transfer learning. Here some of the results, presented in Table 6 in the *Result* section, is re-displayed Table 9.

Table 9. $mAP_{[0.5:0.95]}$ for the transfer learning models during training and testing.

Model name	Train $mAP_{[0.5:0.95]}$	Test $mAP_{[0.5:0.95]}$
yolov5s_pretrained	0.747	0.719
yolov5s_pretrained_freeze_10	0.744	0.688
yolov5s_pretrained_freeze_all	0.351	0.293

Training a neural network is a complex process that requires large datasets and GPU power, transfer learning can therefore be used to faster improve the performance and decrease the training time. By using pre-trained weights, trained on the COCO dataset, knowledge will be extracted and used from the start. Three different models with three different settings were compared with each other: *yolov5s_pretrained* with pre-trained weights, *yolov5s_pretrained_freeze_10* with pre-trained weights with 10 frozen layers, and *yolov5s_pretrained_freeze_all* with pre-trained weights with all the layers frozen. By freezing the layers, the weights will not be tuned based on the new dataset. From Table 9, also displayed in Table 6, and Figure 21, one can conclude that freezing all layers has a negative impact on the model's performance. The model *yolov5s_pretrained_freeze_all* achieved an $mAP_{[0.5:0.95]}$ of 0.351 and 0.293 during training and testing respectively, while the model using pre-trained weights without freezing any layers achieved an $mAP_{[0.5:0.95]}$ of 0.747 and 0.719. The result also shows that freezing the first 10 layers does not have a considerable impact on the model's performance. *Yolov5s_pretrained_freeze_10* performed almost as well as *yolov5s_pretrained* with an $mAP_{[0.5:0.95]}$ of 0.744 and 0.688 during training and testing.

One can therefore conclude that the use of transfer learning, in general, does not improve the result rapidly compared to when using random weights. However, a small improvement can be visible in the results as the model yolov5s_pretrained performed better on test data compared to *yolov5s_random* as the $mAP_{[0.5:0.95]}$ increased from 0.696 to 0.719.

The fact that using transfer learning does not improve the results rapidly, witnesses that the model can achieve equally great results without pre-trained weights and therefore succeed to learn about the features of the images even on a smaller dataset. The result is expected since a fastener is an object with static features, compared to other objects that vary a lot in shapes and color but still belong to the same class of objects. However, the model with all layer's frozen witnesses that the single use of pre-trained weights without training on the custom dataset does not result in accurate recognition of fasteners. The behavior was expected since the COCO dataset does not include fasteners as objects.

When using pre-trained weights, the result shows that freezing 10 layers does not have a significant effect on the model's performance in terms of $mAP_{[0.5:0.95]}$. However, better performance in terms of GPU and training time can be achieved when freezing layers. Figure 24 displays the GPU utilization and training time used when applying transfer learning. One can conclude that by freezing 10 layers, a lot of GPU power can be saved. This can be useful information for future research, as the dataset might increase leading to a more complex model requiring smarter solutions for saving GPU power and time. Figure 25 also displays all the models' GPU utilization and training time. One can conclude that the models do not differ significantly in GPU power except when freezing layers. The number of epochs has the greatest impact on the training time, where the model using 300 epochs differs a lot from all other models' training time. To further investigate number of epoch's and batch size's impact on the results, some of the results in Table 6 in the *Result* section is re-displayed in Table 10.

Table 10. $mAP_{[0.5:0.95]}$ for the hyperparameter models during training and testing.

Model name	Train $mAP_{[0.5:0.95]}$	Test $mAP_{[0.5:0.95]}$
yolov5s_random_300_epochs	0.747	0.693
yolov5s_random_batchsize_32	0.750	0.692
yolov5s_random_batchsize_64	0.748	0.693

Finally, the performance of YOLOv5 was evaluated by investigating a selection of hyperparameters' impact on the model. The hyperparameters that were compared were the number of epochs and batch size. The rule of thumb is to use the maximum number of batches as the GPU resources allow. The default value of YOLOv5 is 16 but one GPU also manages a batch size of 32 and 64. The tested models are therefore *yolov5s_random_batch_32* and model *yolov5s_random_batch_64,* which use random

weights and compared to model *yolov5s_random* does not have the default batch size of 16. The results presented in Table 10, also displayed in Table 6, and Figure 22, show a limited effect on the performance of the enlarged batch size compared to *yolov5s_random*. However, the model *yolov5s_random_batch_32* resulted in the highest train $mAP_{[0.5:0.95]}$ of all three batch sizes with a testing $mAP_{[0.5:0.95]}$ score of 0.750. In fact, that is the highest training $mAP_{[0.5:0.95]}$ of all models. The test $mAP_{[0.5:0.95]}$ of the three different batch sizes only differs between 0.692 and 0.693 and thus, no significant difference can be seen. Using a higher batch size also affects the model's ability to converge to the optimal $mAP_{[0.5:0.95]}$ as figure 16 shows a more fluctuating behavior for a higher batch size. One can therefore conclude that the batch size does not have an important effect on the YOLOv5 model in the case of fastener fault detection.

To evaluate the effect of the number of epochs, the number of epochs was increased to 300 leading to the model *yolov5s_random_epoch_300.* The model can be compared with the default value of 100 epochs for the model *yolov5s_random.* The results, displayed in Table 10, also displayed in Table 6, and Figure 23, show that a limited effect of the enlarged number of epochs could be seen on the model's performance. 300 epochs did not improve the results and the test $mAP_{[0.5:0.95]}$ was even lower than the test $mAP_{[0.5:0.95]}$ using 100 epochs. Since the model with 300 epochs only has a longer training time, one can conclude that 100 epochs are enough in this case.

6.2 Model Performance

When developing any machine learning model, the process is driven towards achieving great results on unseen data. The inference on the test data determines how well the model performs for the given problem. The result obtained from the inference on the test data is presented in section 5.2 Confusion Matrix, 5.3 F1-curve, and finally visually presented through test images in section 5.4 Test Images. The results presented are given by extracting and using the weights from the models evaluated with the best performance, and are the following: *yolov5s_random_batchsize_32*, *yolov5s_pretrained*, *yolov5s_pretrained_freeze_10*. The confusion matrices, displayed in Figures 26-28, show that all three models succeed to predict and classify all true fasteners as fasteners and 96% of all true missing fasteners as missing fasteners. Since the missing fasteners are underrepresented in the dataset it was expected that the models are better at detecting the fasteners than the missing fasteners. A small fraction of the missing fasteners is predicted as fasteners and background. The confusion matrices also show that the models predict part of the background as fasteners or missing fasteners leading to false positives.

The F1 curves give an indication of the trade-off between precision and recall given different confidence levels. Figures 29-31 display that all three models perform well in terms of F1 scores since the F1 score is high given many confidence levels. However, the F1-curve in Figure 30 displaying the model using pre-trained weights has a slightly better shape for lower confidence scores compared to random weights. The models

using pre-trained weights, therefore, seem to perform better in terms of precision and recall for all possible confidence levels.

When visually inspecting the test images, one can conclude that all models in general succeed to the detect the objects well. The test images display the three models' inference on the test data. Figures 32-36 show examples of where the objects of interest are adequately classified and localized in the images. Figure 36 also demonstrates how the models successfully do not recognize the detached fasteners lying next to the rail, which implies that the model is localizing that the objects are located next to the rail and onto the sleeper. This also indicates the model's robustness to different scenarios, in this case where a fastener can be placed in the image but not attached to the rail and therefore not classified. Figures 37-40 however, demonstrate examples where one or multiple models are misclassifying objects as fasteners or missing fasteners in the images, and are compared to the annotated images. In Figures 37-39, this occurs when part of the sleepers is visible in the upper or lower part of the image and cropped by the edge of the image. Since the requirement for annotating the objects of interest was defined as when the complete object was visible in the image, the detection of cropped objects results in a false positive prediction. This is represented in the confusion matrices as the off-diagonal values where the model predicts background as fastener or missing fastener. Since the objects only were annotated when the complete object was visible it was expected that the detectors would behave the same. On the other hand, the images were not properly cropped which causes the misclassification. To prevent this from happening, the images can be collected at a distance and with a frequency that ensures that the images only display the region where the fasteners are expected to be located. This would most likely result in higher precision and consequently in a higher $mAP_{[0.5:0.95]}$ and F1-score.

Further investigation of the confusion matrices in Figures 26-28 shows that 2% of the true missing fasteners are classified as fasteners, leading to another off-diagonal value. However, this case could not be found when manually inspecting the test images after inference. The situation can most probably be explained by the fact that different IOU thresholds are used when for the confusion matrices compared to inference on test data since 0.6 is used for the confusion matrices and 0.75 is used for inference on test images. The model, therefore, seems to be more accurate with a higher threshold. However, this was not further investigated and no conclusions about the performance in relation to the thresholds can be drawn.

Figure 40 exemplifies where a missing fastener is covered by stones or gravel, and the detectors fail to locate the object. When reviewing this image, one would probably not classify the object as a missing fastener either since the sleeper is not present, even if it is annotated as missing. In the confusion matrices, the scenario is represented as a missing fastener predicted as background. However, this is an example of how the detector would behave during these circumstances. From this result, one can expect similar cases in a natural environment where missing and existing fasteners are covered

by grass, stones, lumber, or other objects. When manually assessing the predictions of the test images, one can conclude that the models only fail in the case of cropped sleepers in the upper or lower part of the images, leading to background predicted as fastener or missing, or in the case of Figure 40 where the missing fastener and corresponding sleeper are completely covered by stones and gravel, and it is therefore predicted as background. If the cases with cropped sleepers would be reduced by adding a requirement limiting predictions located next to the images' edges, or by bounding the data set to only contain images that visualize one sleeper, the performance of the YOLOv5 models, in this case, would increase rapidly. In fact, only 1 prediction (Figure 40) out of 128 predictions was misclassified when excluding these cases, leading to an accuracy of 99%.

6.3 YOLOv5 as a Model

This thesis aimed to evaluate and investigate the performance of the state-of-the-art detector YOLOv5 on the task of localizing and classifying e-clip fasteners in the image collection. The series of YOLO models have the advantage of higher performance in terms of accuracy and speed compared to other successful object detection models. More importantly, YOLO has proven to be the optimal model for the recognition of small objects. YOLOv5 is the latest version of constantly improved YOLO models, and YOLOv5s is the smallest and fastest model available. This aims to compare the results with previous studies and comment of the performance of YOLOv5 as a model in general.

The main performance metric used in this thesis is $mAP_{[0.5:0.95]}$ where the mAP score depending on IOU thresholds in the interval between 0.5 to 0.95 is calculated. Several combinations of different settings had a positive effect on the different models' performance as all models had a training and testing $mAP_{[0.5:0.95]}$ between 0.69 and 0.75, except *yolov5s_pretrained_freeze_all* with all layers frozen. To give a broader understanding of the performance of the YOLOv5s model in terms of mAP it is important to compare the results with previous studies. One important notice in previous studies is that the mAP score is calculated with the IOU threshold of 0.5 or simply presented without a given threshold. For example, Zheng et al [8], achieved an mAP of 0.9968 using YOLOv5 without specifying the threshold. Wang et al [10] achieved an $mAP_{[0.5:0.95]}$ of 0.5, and an $mAP_{[0.5]}$ of 0.93 using different YOLO models. In comparison with the results of Wang et al, one can argue that the result of this thesis of an $mAP_{[0.5:0.95]}$ between 0.69-0.75 is good. Table 7 also displays the $mAP_{[0.5]}$ of three models: *yolov5s_pretrained*, *yolov5s_pretrained_freeze_10* and, *yolov5s_random_batch_32*. All models achieved an $mAP_{[0.5]}$ of 0.991-0.995 which is better than the results of Wang et al. with an $mAP_{[0.5]}$ of 0.93. Most probably, did Zheng et al. use the same or similar threshold as they reached an mAP of 0.9968. The result of this thesis is therefore similar to the result of Zheng et al. The reason for the lower number of the $mAP_{[0.5:0.95]}$ score is that the threshold puts a higher demand on accurate IOU. There is always an inconsistency when annotating the data as human errors are

present in the data. The model's prediction relies on the training based on the annotations and will therefore most probably not reach 100% IOU with the ground-truth bounding box. Increasing the threshold will therefore lead to fewer predictions and decrease the recall of the model.

Liu et al [16] also constructed a fastener fault detector using YOLOv3 and achieved precision and recall of around 93-100% on different classes. Lin et al [7] present their results of the YOLO detector in terms of precision and recall as well and reached a precision rate of 89% and recall rate of 95%. Table 7 displays the precision and recall values for the three models *yolov5s_pretrained*, *yolov5s_pretrained_freeze_10*, and *yolov5s_random_batch_32*, where the precision is between 0.976-0.996 and the recall is between 0.988-1. In conclusion, YOLOv5s perform well on the task of detecting the fasteners in terms of precision and recall. Liu et al. [16] present the average F1 score for all classes without specifying the confidence level and reached an F1 score of 0.9. Figure 29-31 shows that *yolov5s_random_batch_32* reached an F1 score of 0.97 at a confidence level 0.79, and *yolov5s_pretrained* and *yolov5s_pretrained_freeze_10* reached an F1 score of 0.98 at a confidence level 0.82. Compared to the results of Liu et al., the YOLOv5s models of this thesis do not outperform their model in terms of the F1 score. On the other hand, they do not resent the confidence score for the F1 score which makes the results difficult to compare.

7. Conclusion

The conclusion summarizes the results and discussion based on the purpose of this master thesis. The section ends with different aspects and further investigations when it comes to future work for the given problem.

This thesis aimed to evaluate and investigate the performance of the state-of-the-art detector YOLOv5 on the task of localizing and classifying e-clip fasteners in the image collection. The series of YOLO models have the advantage of higher performance in terms of accuracy and speed compared to other successful object detection models. More importantly, YOLO has proven to be the optimal model for the recognition of small objects. YOLOv5 is the latest version of constantly improved YOLO models, and YOLOv5s is the smallest and fastest model available.

The results demonstrate that YOLOv5s perform well in detecting the existing and missing fasteners. The training $mAP_{[0.5:0.95]}$ for all models, despite when freezing all layers, was over 0.71 and the test $mAP_{[0.5:0.95]}$ was over 0.658. The highest achieved training $mAP_{[0.5:0.95]}$ was 0.750 for the model with random weights and batch size 32. The model with pretrained weights gained the highest test $mAP_{[0.5:0.95]}$ score of 0.719. Using both augmentation techniques and adding background images resulted in the dataset with the most variation and thus the best results. Using transfer learning does not improve the results rapidly as the results do not significantly differ from the results using random weights. It witnesses that the case of fastener detection is not challenging for the YOLOv5s model to detect in terms of shapes, colors, and edges as the model still can achieve equally great results without pre-trained weights and therefore succeed in learning about the images' features even on smaller datasets. However, using transfer learning decreased the use of GPU power and can therefore be useful for a more efficient model in terms of speed and power. Finally, different batch sizes did not have an important effect on the YOLOv5s model and 100 epochs where enough as 300 epochs required more training time but did not improve the results. Since several combinations of different settings had a positive effect on the model's performance, one can conclude that the YOLOv5s model in general is a suitable model for detecting fasteners.

The confusion matrices show that all three models succeed to predict and classify all true fasteners as fasteners and 96% of all true missing fasteners as missing fasteners. The test images witnesses that the models fail when parts of a sleeper are visible in the upper or lower part of the images, leading to false positives shown in the off-diagonal of the confusion matrices where background is predicted as fastener/missing. However, a manual assessment of the test images shows that this is the only case where the models fail except when the fastener region is covered by stones and gravel leading to a challenging case for both humans and algorithms to detect. In production, the cropped sleepers can be reduced by accurately synchronizing the frequency of capturing data with the distance between the sleeper, such that only one sleeper is visible per image

leading to more accurate results. Compared to previous studies, the results of over 0.658 in $mAP_{[0.5:0.95]}$, and over 0.991 in $mAP_{[0.5]}$, can be considered good results as previously studies obtained similar results or reached lower mAP values. To conclude, the YOLOv5s model is a well-performing model for fastener fault detection.

In addition, when analyzing the purpose of object detection for automating the maintenance of fastener fault detection, it must detect all the objects to securely prevent derailments or other accidents. Therefore, it is preferred that the detector overestimate the number of objects that will result in false positives, rather than underestimating the number of objects resulting in false negatives. The misclassification of fasteners and missing fasteners, giving false positives, could therefore be seen as a positive result in terms of railway safety. When it comes to predicting the different classes, it is more important that the detector classifies the missing fasteners correctly rather than the fasteners since missing fasteners are more crucial for maintenance purposes.

By evaluating both the results and YOLOv5 as a model in general one can conclude that machine learning can be applied as an effective and robust technique to the problem of automatic fastener fault detection. As for all models, based on data, it is important to understand the dataset and its differences in comparison to the real-world environment, resulting in limitations when applying and using the model in practice, which needs to be identified and evaluated.

7.1 Future Work

Several important perspectives are interesting for further investigation. First, the robustness of the YOLOv5s model concerning to different environments, lighting, type of rails, sleepers, and fasteners can be evaluated. This can be done by extending the dataset with more diverse data that represent different scenarios and rail constructions. Another aspect is the scalability of the model and its implementation, to extend the dataset and train the algorithm on a bigger dataset the scalability of the development tools needs to be investigated in terms of needed GPU power, hyperparameter optimization, annotation management, and extraction of the weights. Finally, an analysis of what factors need to be considered when implementing the automatic fastener fault detection for maintenance in terms of stakeholders, technical difficulties, and tracking the number of missing fasteners can be evaluated.

References

[1] Trafikverket, ”Järnkoll på tågresor,” Trafikverket, 10 February 2021. [Online]. Available: https://www.trafikverket.se/resa-och-trafik/jarnvag/jarnkoll--fakta-om-den-svenska-jarnvagen/jarnkollpa-tagresor/. [Använd 14 March 2022].

[2] P. Chandran, J. Asber, F. Thiery, J. Odelius och M. Rantatalo, ”An Investigation of Railway Fastener Detection Using Image Processing and Augmented Deep Learning.,” vol. 13, nr 21, 2021.

[3] F. Lindberg och F. Söderbaum, ”Bantrafik 2020,” Trafikanalys, Stockholm, 2021.

[4] Trafikverket, ”Underhåll av järnvägssystem,” Trafikverket, 3 februari 2020. [Online]. Available: https://www.trafikverket.se/resa-och-trafik/underhall-av-vag-och-jarnvag/Sa-skoter-vi-jarnvagar/Underhall-av-jarnvagssystemet/. [Använd 7 mars 2022].

[5] X. Jiawei, J. Huang, C. Zeng, S.-H. Jiang och N. C. Podlich, ”Systematic Literature Review on Data-Driven Models for Predictive Maintenance of Railway Track: Implications in Geotechnical Engineering,” *Geosciences 10,* nr no. 11: 425, 2020.

[6] Q. Song, Y. Guo, J. Jiang, C. Liu och M. Hu, ”High-speed Railway Fastener Detection and Localization Method based on convolutional neural network.,” *arXiv preprint arXiv:,* nr 1907.01141., 2019.

[7] Y.-W. Lin, C.-C. Hsieh, W.-H. Huang, S.-L. Hsieh och W.-H. Hung, ”Railway Track Fastners Fault Detection using Deep Learning,” IEEE, Taipai City, Taiwan, 2019.

[8] D. Zheng, L. Li, S. Zheng, X. Chai, S. Z. Zhao, Q. Tong, J. Wang och L. Gua, ”A Defect Detection Method for Rail Surface and Fasteners Based on Deep Convolutional Neural Network,” *Computational Intelligence and Neuroscience,* p. 15, 2 August 2021.

[9] D. Wei, X. Wei, Y. Liu, L. Jia och W. Zhang, ”The Identification and Assessment of Rail Corrugation Based on Computer Vision,” *Applied sciences,* vol. 9, nr 3913, 2019.

[10] T. Wang, Y. Fangfang och K.-L. Tsui, ”Real-Time Detection of Railway Track Component via One-Stage Deep Learning Networks,” *Sensors,* vol. 20, nr 4325, 2020.

[11] T. Wang, Z. Zhang, F. Yang and K.-L. Tsui, "Automatic Rail Component Detection Based on AttnConv-Net," *IEEE Sensors Journal,* vol. 22, no. 3, pp. 2379-2388, 2022.

[12] U. Honauer och S. Ödeen, ”Underhållsplan 2021-2024,” Trafikverket, Borlänge, Sweden, 2021.

[13] C. Engström, L. O. Karlsson och L. Arméen, ”Järnväg,” Nationalencyklopedin, [Online]. Available: http://www-ne-se.ezproxy.its.uu.se/uppslagsverk/encyklopedi/lång/järnväg. [Använd 08 March 2022].

[14] P. Chandran, M. Rantatalo, J. Odelius, H. Lind och S. M. Famurewa, ”Train-based differential eddy current sensor system for rail fastener detection,” *Measurement Science and Technology,* vol. 30, nr 12, 19 September 2019.

[15] H. A. Leopold, A. Singh, S. Sourya och V. Lakshminarauanan, ”State of the Art in Neural Networks and their Applications,” i *Chapter 12 - Deep learning for ophthalmology using optical coherence tomography*, Academic press, 2021, pp. 250-253.

[16] J. Liu, Y. Teng, X. Ni och H. Liu, ”A Fastener Inspection Method Based on Defective Sample Generation and Deep Convolutional Neural Network,” *IEEE Sensors Journal,* vol. 21, nr 10, pp. 12179-12188, 2021.

[17] X. Wei, Z. Yang, Y. Liu, D. Wei, L. Jia och Y. Li, ”Railway track fastener defect detection based on image processing and deep learning techniques: a comparative study,” *Engineering Applications of Artificial Intelligence,* vol. 80, pp. 61-81, 2019.

[18] J. Yang, W. Tao, M. Liu, Y. Zhang, H. Zhang och H. Zhao, ”An efficient direction field-based method for the detection of fasteners on high-speed railways,” *Sensos,* vol. 11, nr 8, pp. 7364-7381, 2011.

[19] X. Gibert, V. M. Patel och R. Chellappa, ”Robust fastener detection for autonomous visual railway track inspection,” i *Proceedings of the 2015 IEEE Winter Conference on Applications of Computer Vision*, Waikoloa, HI, USA, 2015.

[20] Trafikverket, ”Trafikverkets uppdrag,” Trafikverket, 27 April 2020. [Online]. Available: https://www.trafikverket.se/om-oss/var-verksamhet/trafikverkets-uppdrag/. [Använd 09 march 2022].

[21] Transportstyrelsen, ”About us,” Transportstyrelsen, 09 January 2014. [Online]. Available: https://www.transportstyrelsen.se/en/About-us/. [Använd 09 March 2022].

[22] Trafikverket, ”Järnkoll på spåret,” Trafikverket, 11 februari 2021. [Online]. Available: https://www.trafikverket.se/resa-och-trafik/jarnvag/jarnkoll--fakta-om-den-svenska-jarnvagen/jarnkoll-pa-sparen/. [Använd 7 mars 2022].

[23] Region Stockholm, ”Roslagsbanan,” Region Stockholm, [Online]. Available: https://www.regionstockholm.se/roslagsbanan. [Använd 09 March 2022].

[24] University of Birmingham and Network Rail, *Railway Lexicon,* Birmingham, England: University of Birmingham and Network Rail, 2011.

[25] A. Hammar, ”Spårkomonenter Sliper och befästning,” Trafikverket, 2020.

[26] M. A. Sayeed och M. A. Shahin, ”Design of ballasted railway track foundations using numerical modelling. Part 1: Development,” *Canadian Geotechnical Journal,* vol. 55, nr 3, pp. 353-368, 2017.

[27] Infrastrukturdepartementet, ”Regeringskansliet,” 14 september 2020. [Online]. Available: https://www.regeringen.se/pressmeddelanden/2020/09/satsning-pa-okat-underhall-av-jarnvag/. [Använd 07 03 2022].

[28] H. Feng, Z. Jiang, F. Xie, P. Yang, J. Shi och L. Chen, ”Automatic fastener classification and defect detection in vision-based railway inspection systems,” *IEEE Transactions on Instrumentation and Measurement,* vol. 63, nr 4, pp. 877-888, 2014.

[29] V. Narayan, Effective maintenance management: risk and reliability strategies for optimizing performance., New York: Industrial Press Inc., 2004, pp. 150-151.

[30] Y. K. Al-Douri, P. Tretten och R. Karim, ”Improvement of railway performance: a study of Swedish railway infrastructure,” *Journal of Modern Transportation (J Mod Transport),* vol. 24, nr 1, pp. 22-37, 2016.

[31] P. Dollár och C. L. Zitnich, ”Fast edge detection using structured forests,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 37, nr 8, pp. 1558-1570, 2015.

[32] E. Resendiz, J. M. Hart och N. Ahuja, ”Automated visual inspection of railroad tracks,” *IEEE Transactions on Intelligent Transportation Systems,* vol. 14, nr 2, pp. 751-760, 2014.

[33] Y. Li, C. Otto, N. Haas och Y. Fujiki, ”Component-based track inspection using machine-vision technology,” i *Proceedings of the 1st ACM International Conference on Multimedia Retrieval,*, Treno, Italy, 2011.

[34] H. Trinh, N. Haas, Y. Li, C. Otto och S. Pankanti, ”Rail Component Detection, Optimization and Assessment for Automatic Rail Track Inspection,” i

Proceedings of the 2012 IEEE Workshop on the Applications of Computer Vision, Breckenridge, CO, USA, 2012.

[35] L. Liu, F. Zhou och Y. He, ”Automated status inspection of fastening bolts on freight trains using a machine vision approach,” *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit,* vol. 230, nr 7, pp. 1629-1641, 2016.

[36] F. Marino, A. Distante, P. L. Mazzeo och E. Stella, ”A realtime visual inspection system for railway maintenance: automatic hexagonal-headed bolts detection,” *EEE Transactions on Systems, Man, and Cybernetics,* vol. 37, nr 7, pp. 418-428, 2007.

[37] P. De Ruvo, A. Distante, E. Stella och F. Marino, ”A GPU based vision system for real time detection of fastening elements in railway inspection,” i *Proceedings of the 2009 16th IEEE International Conference on Image Processing*, Cairo, Egypt, 2010.

[38] Q. Ji, ”Probabilistic Graphical Models for Computer Vision,” i *In Computer Vision and Pattern Recognition*, Academic Press, 2020, pp. 191-192.

[39] A. Vahab, M. Naik, P. G. Raikar och S. R. Prasad, ”Applications of Object Detection System,” *International Research Journal of Engineering and Technology (IRJET),* April 2019.

[40] I. Zafar, G. Tzanidou, R. Burton, N. Patel och L. Araujo, ”Old versus new Ml,” i *Hands-On Convolutional Neural Networks with TensorFlow*, Birgmingham, Mumbai, Packt Publishing Limited , 2018.

[41] H. A. Leopold, A. Singh, S. Sengupta och V. Lakshminarayanan, ”Chapter 12. Deep learning for ophthalmology using optical coherence tomography,” i *State of the Art in Neural Networks and their Applications*, Waterloo, 2021.

[42] E. Davies, ”Chapter 9. Line Detection,” i *Machine Vision (Third Edition) Theory, Algorithms, Practicalities*, 2005.

[43] J. Du, ”Understanding of Object Detection Based on CNN Family and YOLO,” *Journal of Physics: Conference Series,* 2018.

[44] B. Dong, Q. Li, J. Wang, W. Huang, P. Dai och S. Wang, ”An End-to-End Abnormal Fastener Detection Method Based on Data Synthesis,” 2019.

[45] A. Bochkovskiy, C.-Y. Wang och H.-Y. M. Liao, ”YOLOv4: Optimal Speed and Accuracy of Object Detection,” 2020.

[46] I. Zafar, G. Tzanidou, R. Burton, N. Patel och L. Araujo, ”Deep Learning And Convolutional Neural Networks,” i *Hands-On Convolutional Neural*

Networks with TensorFlow, Birgmingham, Mumbai, Packt Publishing Limited, 2018.

[47] U. Michelucci, Advanced Applied Deep Learning, Berkeley, CA: Apress, 2019, pp. 79-123.

[48] A. Lindholm, N. Wahlström, F. Lindsten och T. B. Schön, Machine Learning - A First Course for Engineers and Scientists, Cambridge University Press, 2022, pp. 124-129.

[49] A. Cruz-Roa, A. Basavanhally, F. González, H. Gilmore, M. Feldman, S. Ganesan, N. Shih, J. Tomaszewski och A. Madabhushi, ”Automatic detection of invasive ductal carcinoma in whole slide images with Convolutional Neural Networks,” *Progress in Biomedical Optics and Imaging - Proceedings of SPIE,* 2014.

[50] H. Hakim och A. Fadhil, ”Survey: Convolution Neural networks in Object Detection,” *Journal of Physics: Conference Series,* 2021.

[51] J. Guo, K. Han, Y. Wang, C. Zhang, Z. Yang Han Wu, X. Chen och C. Xu, ”Hit-Detector: Hierarchical Trinity Architecture Search for Object Detection,” i *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[52] R. Xu, H. Lin, K. Lu, L. Cao och Y. Liu, ”A Forest Fire Detection System Based on Ensemble Learning,” *Forests,* vol. 12, nr 217, 2021.

[53] G. Jocher, ”Train Custom Data,” Ultralytics, [Online]. Available: https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data.

[54] S. J. Pan och Q. Yang, ”A Survey on Transfer Learning,” *IEEE Transactions on Knowledge and Data Engineering,* vol. 22, nr 10, pp. 1345-1259, 2010.

[55] M. Umberto, Advanced CNNs and Transfer Learning. In: Advanced Applied Deep Learning. Apress, Berkeley, CA: Apress, 2019.

[56] W. Zhang, L. Deng, L. Zhang och D. Wu, ”A Survey on Negative Transfer,” *IEEE Transactions on Neural Networks and Learning Systems,* pp. 1-13, 2021.

[57] T. Agrawal, Hyperparameter Optimization in Machine Learning, Bangalore, Karnataka, India: Apress Standard, 2021.

[58] R. Padilla, W. L. Passo, T. L. B. Dias, S. L. Netto och E. A. B. da Silva, ”A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit,” vol. 10, nr 3, 2021.

[59] C. Cao, D. Chicco och M. M. Hoffman, ”The MCC-F1 curve: a performance evaluation technique for binary classification,” *arXiv:2006.11278,* 2020.

[60] I. Zafar, G. Tzanidou, R. Burton, N. Patel och L. Araujo, ”Evaluation metrics,” i *Hands-On Convolutional Neural Networks with TensorFlow*, Birgmingham, Mumbai, Packt Publishing Limited, 2022.

[61] I. Zafar, G. Tzanidou, R. Burton, N. Patel och L. Araujo, ”Detector Loss function (YOLO),” i *Hands-On Convolutional Neural Networks with TensorFlow*, Birmingham, Mumbai, Packt Publishing Limited, 2018.

[62] I. Zafar, G. Tzanidou, R. Burton, N. Patel och L. Araujo, ”Loss Part 3,” i *Hands-On Convolutional Neural Networks with TensorFlow* , Birmingham, Mumbai, Packt Publishing Limited, 2018.

[63] I. Zafar, G. Tzanidou, R. Burton, N. Patel och L. Araujo, ”Loss Part 1,” i *Hands-On Convolutional Neural Networks with TensorFlow*, Birgminham, Mumbai, Packt Publishing Limited, 2018.

[64] I. Zafar, G. Tzanidou, R. Burton, N. Patel och L. Araujo, ”Loss Part 2,” i *Hands-On Convolutional Neural Networks with TensorFlow* , 2018, Packt Publishing Limited, 2018.

[65] I. Zafar, G. Tzanidou, R. Burton, N. Patel och L. Araujo, ”Machine Learning Best Practices And Troubleshooting,” i *Hands-On Convolutional Neural Networks with TensorFlow*, Birgmingham, Mumbai, Packt Publishing Limited, 2018.

[66] V. Kharchenko och I. Chyrka, ”Detection of Airplanes on the Ground Using YOLO Neural Network,” i *IEEE 17th International Conference on Mathematical Methods in Electromagnetic Theory (MMET)*, 2018.