TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Can ERSÜ 212286 IASM

AUTOMATIC VISUAL TRAFFIC SIGN DAMAGE DETECTION USING DEEP LEARNING ALGORITHMS

Master's Thesis

Supervisor: Uljana Reinsalu
PhD

Co-supervisor: Karl Janson
PhD

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Can ERSÜ 212286 IASM

AUTOMAATNE VISUAALNE LIIKLUSMÄRKIDE KAHJUSTUSTE TUVASTAMINE KASUTADES SÜVAÕPPE ALGORITME

Magistritöö

Juhendaja: Uljana Reinsalu
PhD

Kaasjuhendaja: Karl Janson
PhD

Tallinn 2023

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Can ERSÜ

11.01.2023

Abstract

Traffic signs are a critical aspect of contemporary roadway infrastructure, guiding and controlling the movement of vehicles and pedestrians, as well as providing necessary information for road users. Even though they are vital, traffic signs are prone to damage from various factors like weather, vandalism and accidents, which can lead to confusion and dangerous situations on the road. Therefore, it is important to quickly fix or replace damaged traffic signs to guarantee that all road users have the information they need to stay safe. If damaged signs are not identified quickly, it could result in a serious of issues such as increased accident risks, making the maintenance of these signs imperative for a smooth flow of traffic and overall safety.

This thesis presents an approach for automatic visual detection of damage to traffic signs using YOLO, a real-time object detection algorithm and an autoencoder, a neural network for image compression and reconstruction based on GTSDB dataset. YOLO is used to detect and locate traffic signs within an image, followed by using the autoencoder to analyze an image of the detected sign for possible visual damage such as fading, graffiti, or physical damage. The autoencoder is pre-trained on a dataset of undamaged traffic signs to be able to identify anomalies on the image. Different image quality assessment metrics have been tested. Assessment metrics SSIM and RMSE performed the best on chosen dataset. Finally, detected traffic sign's condition is decided with respect to threshold as a result of statistical error distribution of related traffic sign class.

The proposed approach has the potential to improve the efficiency and reliability of traffic sign inspection by detecting the damaged traffic signs faster than currently used methods which leads to enhance the road safety.

The thesis is written in English and is 79 pages long, including 6 chapters, 33 figures and 18 tables.

Annotatsioon

AUTOMAATNE VISUAALNE LIIKLUSMÄRKIDE KAHJUSTUSTE TUVASTAMINE KASUTADES SÜVAÕPPE ALGORITME

Liiklusmärgid on kaasaegse teeinfrastruktuuri oluline osa, mis suunavad ja kontrollivad sõidukite ja jalakäijate liikumist ning annab vajalikku teavet liiklejatele. Kuigi liiklusmärgid on elutähtsad, võivad neid kahjustada mitmesugused tegurid, näiteks ilmastikuolud, vandalism ja õnnetused, mis võivad põhjustada segadust ja ohtlikke olukordi teel. Seetõttu on oluline kasjustatud liiklusmärgid kiiresti parandada või asendada, et tagada kõigile liiklejatele ohutuseks vajalik teave. Kui kahjustatud märke ei tuvastata kiiresti, võib see põhjustada tõsiseid probleeme, näiteks õnnetusohu suurenemist, mistõttu on märkide hooldamine hädavajalik, et tagada sujuv liiklusvoog ja üldine ohutus.

Käesolevas väitekirjas esitatakse lähenemisviis liiklusmärkide kahjustuste automaatseks visuaalseks tuvastamiseks, kasutades GTSDB andmekogumil õpetatud objektide tuvastamise algoritmi YOLO ja autoenkooderit, pildi tihendamise ja rekonstrueerimise neuronivõrku. Pakutud lahenduses kasutatakse YOLO-t liiklusmärkide tuvastamiseks ja leidmiseks pildil. Seejärel kasutatakse autoenkooderit tuvastatud märgi pildi analüüsimiseks võimalike visuaalsete kahjustuste, näiteks tuhmumise, graffiti või füüsiliste kahjustuste suhtes. Autoenkooder on eelnevalt treenitud kahjustamata liiklusmärkide andmekogumiga, et tuvastada pildi anomaaliaid. Testitud on erinevaid pildikvaliteedi hindamise mõõdikuid. Valitud andmekogumi puhul toimisid kõige paremini mõõdikud SSIM ja RMSE. Lõpuks hinnatakse liiklusmärgi seisunit kasutades eelmaiditud mõõdikuid lävendi suhtes, mis on tuletatud liiklusmärkide klassi statistilisest veajaotusest.

Väljakäidud lähenemisviis aitab parandada liiklusmärkide kontrollimise tõhusust ja usaldusväärsust, tuvastades kahjustatud liiklusmärgid kiiremini kui praegu kasutatavad meetodid seda võimaldavad. See aitab suurendada liiklusohutust.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 79 leheküljel, 6 peatükki, 33 joonist, 18 tabelit.

List of Abbreviations and Terms

ADAM	Adaptive Movement
ADAS	Advanced Driver Assistance Systems
API	Application Programming Interface
BTSDC	Belgian Traffic Sign Detection and Classification
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSF	Contrast Sensitivity Function
CSV	Comma Separated File
DCNN	Deep Convolutional Neural Network
FCNN	Fully Convolutional Neural Network
FPS	Frames Per Second
GTSRB	German Traffic Signs Recognition Benchmark
GTSDB	German Traffic Signs Detection Benchmark
HVS	Human Visual System
IDE	Integrated Development Environment
IJCNN	International Joint Conference on Neural Networks
IQA	Image Quality Assessment
ITS	iIntelligent Transportation Systems
LIDAR	Light Detection and Ranging
MLS	Mobile Laser Scanning
MRE	Mean Relative Error
MSE	Mean Squared Error
NMS	Non Maximum Suppression
PCA	Principal Component Analysis
PSNR	Peak to Signal Noise Ratio
RMSE	Root Mean Squared Error
RMSP	Root Mean Square Propagation
ROI	Region of Interest
SoC	System on Chip
SPP	Spatial Pyramid Pooling
SSIM	Structural Similarity Index Measurement
STSDC	Swiss Traffic Sign Detection and Classification
SVM	Support Vector Machine

TSR	Traffic Sign Recognition
VIF	Visual Image Fidelity
VSNR	Visual to Noise Ratio
YOLO	You Only Look Once

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Motivation

Traffic signs are essential part of the infrastructure of any modern road system. They are used to direct and regulate the flow of traffic, as well as to provide important information to drivers, pedestrians, and other road users about the rules of the road and the hazards ahead. Traffic signs can be found on streets, highways, and intersections, and come in a variety of shapes, sizes, and colors to convey different types of information. Some examples of common traffic signs include stop, yield, speed limit and warning signs.

Despite their importance, traffic signs are not immune to damage. Over time, traffic signs can become damaged due to various factors such as weather, vandalism, and accidents such as seen in Figure 1. When a traffic sign becomes damaged, it can be difficult to read or interpret, leading to confusion and potentially dangerous situations on the road. For example, a damaged stop sign might not be visible to drivers approaching an intersection, leading to accidents and injuries. Similarly, a damaged speed limit sign might not be visible to drivers, leading to speeding and other unsafe driving behaviors.

Regardless of the severity of the damage, it is important to repair or replace damaged traffic signs as soon as possible. This helps to ensure that all road users have the necessary information they need to stay safe and navigate the roads effectively. If traffic signs are not recognized rapidly, it can lead to a number of problems such as drivers may not have enough time to react to the sign and may not know how to proceed. This can lead to confusion and potentially dangerous situations on the road. Moreover traffic flow may also be disrupted, as drivers may not know how to proceed at intersections or may not be aware of construction or other obstacles in the road. Additionally, the risk of accidents may be increased, as drivers may not be aware of potential hazards or may not know how to react to certain situations.

(a) Fallen pole and folded sheet [1]

(b) Vandalized [2]

(c) Material damage [3]

(d) Folded sheet [2]

Figure 1. Examples of different types of damages

1.2 Problem Formulation

Traditionally, the detection and repair of damaged traffic signs has been a manual process, with workers physically inspecting traffic signs and identifying those that need repair. This process is time-consuming, labor-intensive, and prone to errors, and it is not practical to inspect all traffic signs on a regular basis. Therefore, there is a need for an automatic system that can detect damaged traffic signs and alert maintenance workers to their presence.

The goal of this thesis is to develop an approach for automatically detecting damaged traffic signs in images and video feeds. This approach will be able to classify different types of traffic signs, such as stop signs, yield signs, and speed limit signs, and will be able to identify a range of different types of damage, including fading, graffiti, dents, and other forms of physical damage. The system will be fully automated, requiring no human intervention, and will be able to operate continuously in the field.

To achieve this goal, it will need to be addressed several challenges. First, traffic signs can appear in a wide variety of shapes, sizes, and colors, and may be partially occluded or partially obscured by other objects in the scene. Therefore, it must robustly detect and classify traffic signs under such challenging conditions. Secondly, traffic signs can be damaged in various ways, and it must be accurate in identifying and detecting such damage. Additionally, it must function efficiently in different lighting and weather conditions, and adapt to changing conditions over time. Lastly, it must be reliable and robust, with a low rate of false positives and false negatives.

An automatic approach for detecting damaged traffic signs such as illustrated in Figure 2 would be a major advancement in transportation safety and would have practical applications in the maintenance of transportation infrastructure. By automating the detection process, the roads can become safer and more efficient, and the risk of accidents and injuries caused by damaged traffic signs can be reduced.

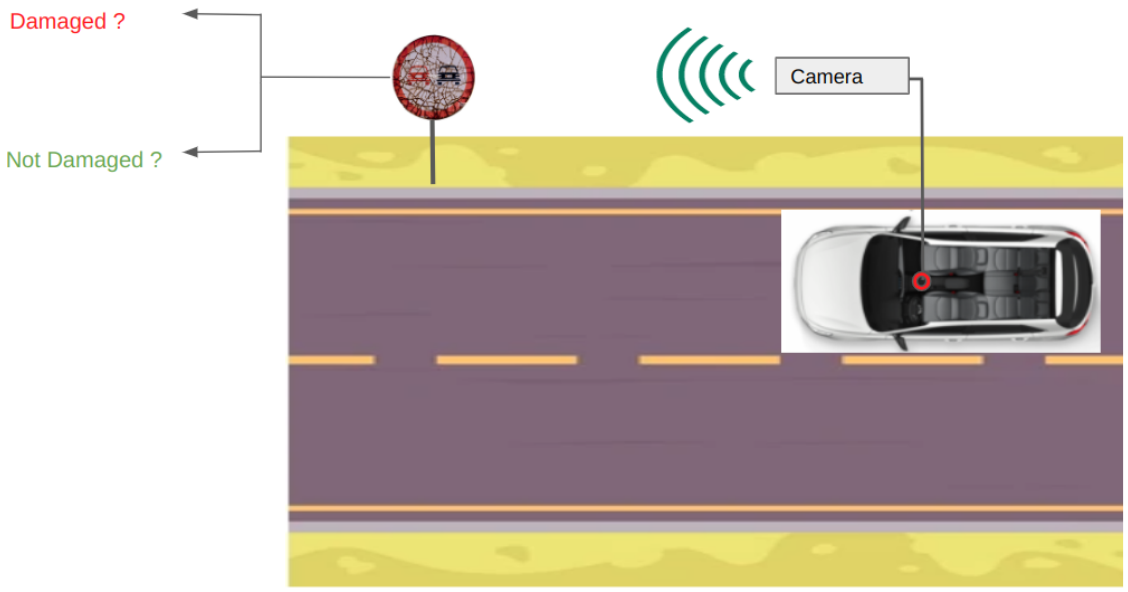


Figure 2. Damaged traffic sign detection overview

1.3 Contribution

One of the key challenge in damaged sign detection is the need to develop robust algorithms that can handle a wide range of damage scenarios and still accurately identify the intended meaning of the traffic sign. This requires the ability to handle variations in the degree and type of damage, as well as the ability to adapt to changing lighting and weather conditions.

Damaged traffic sign detection problem can be divided and examined into two main parts: Traffic sign detection and image anomaly detection.

- Traffic sign detection involves identifying the presence and location of traffic signs in an image or video stream. These techniques can be divided into two main parts: traditional methods and machine learning-based methods.
  Traditional methods for traffic sign detection typically rely on hand-crafted features and heuristics to identify the presence of traffic signs. These methods can include techniques such as color-based segmentation, shape-based matching, and edge detection. These approaches can be effective in certain scenarios, but they can be limited in their ability to handle variations in lighting, perspective, and other factors that can affect the appearance of traffic signs.
  Machine learning-based methods for traffic sign detection involve training a model on a large dataset of labeled traffic sign images. These models can be trained using a variety of techniques, including convolutional neural networks (CNNs) and support vector machines (SVMs). Due to machine learning based detection method works more accurate and better performance, this approach is chosen in this thesis.

- Image anomaly detection using autoencoder is a machine learning technique for identifying unusual or abnormal patterns in images. It is a form of unsupervised learning which means the algorithm don't have any information about the anomaly class. Autoencoder is a neural network architecture that is trained to reconstruct the input images that are fed into it. The training process allows the autoencoder to learn the features of normal images and create a compact representation of them in a bottleneck layer.
  Once the autoencoder is trained, it can be used to detect anomalies in new images by comparing the reconstructed image to the original input. When a new image is fed into the trained autoencoder, the output should be similar to the input image if it is a normal image but it will not if it has an anomaly. This difference can be used as a measure of whether or not an image contains an anomaly. By comparing the reconstruction error, the autoencoder can identify the image that deviate from the normal pattern. Anomaly detection using autoencoder is commonly used in applications such as surveillance, security and quality control. Damage detection is also in the scope of autoencoders, which suits well with the subject of this thesis.

There have been a number of researches conducted on detecting the damage on traffic/road signs, including state of the art computer vision and machine learning techniques on traffic sign detection [4] [5] [6] [7], detecting the damaged signs on road [8] [9], methods for extended analysis on detecting the shape and rotation based deforms on traffic sign sheet and stands [2] [10], and the use of technology to assist with particularly in the context of developing autonomous driving systems for identifying traffic signs in real-time. These studies have focused on a variety of approaches, including image processing techniques, machine learning algorithms, and light detection and ranging (lidar) based solutions.

This thesis aims to combine machine learning based traffic sign detection and image anomaly detection techniques to detect damages on traffic sign using image data from camera. This brings the following benefits: cost-effectiveness, light-weight software architecture and time-efficient solution which prone to improve safety on the roads by helping to ensure that drivers are able to accurately interpret and respond to traffic signs, even when they are damaged or in poor condition. Open source code and further experiments can be found on Github [11].

1.4 Research Questions

Traffic signs play a crucial role in ensuring the safety and efficient flow of traffic on roads. However, the ability to detect and identify traffic signs, especially in the context of Estonian roadways, is an ongoing challenge. In this thesis, the aim is to address the following questions:

- **RQ1:** What is the best existing traffic signs dataset suitable for detecting Estonian traffic signs?
- **RQ2:** What is the most suitable/effective method to detect traffic signs based on image data taking into account speed and accuracy?
- **RQ3:** What algorithm could be used to identify whether traffic sign is damaged or not based on image data?
- **RQ4:** How to numerically measure the damage of the traffic sign?
- **RQ5:** How to decide if a traffic sign is damaged or not based on chosen metric?

Through addressing these questions, the goal is to provide valuable insights that can aid in the development of more accurate and efficient traffic sign detection approach.

1.5 Thesis Organisation

The rest of thesis is organized as follows. Chapter 2 provides an overview of previous research and finding gaps in the literature that the current study aims to fill. In chapter 3, an extensive overview of the research process, the design, the procedures for obtaining data, and the plan for evaluating the data will be presented. Afterwards, Chapter 4 will make a mention of background information about popular traffic sign datasets, object detection algorithms and comparison of performance and accuracy analysis of proposed methods. Next, Chapter 5 will introduce a novel techniques about anomaly detection on images and artificial image generation techniques. Additionally traffic sign recognition dataset and data processing methods, autoencoder architecture, image comparison evaluation methods and experimental results of damaged traffic sign will be mentioned. Finally, Chapter 6 will conclude the thesis.

2. State of the Art

There have been a number of studies on the topic of damage detection of traffic signs. These studies typically focus on the development of algorithms and systems for automatically detecting and classifying damage to traffic signs, such as dents, scratches, graffiti, or fading. The goal of these studies is to improve the safety and efficiency of transportation systems by helping to identify the damaged traffic signs before they become a hazard to road users.

There have been a wide variety of studies conducted in literature that generally focus on the following priority concerns:

- An overview of the importance of traffic sign damage detection and the potential consequences of damaged or vandalized signs.
- A description of the different computer vision techniques and machine learning algorithms that are commonly used for traffic sign damage detection.
- A review of existing traffic sign damage detection systems and their performance in terms of accuracy and speed.
- A discussion of the challenges involved in developing and deploying traffic sign damage detection systems, such as the need for robust algorithms that can handle variations in lighting, weather, and other factors.
- Suggestions for future research directions and potential improvements to traffic sign damage detection systems.

One approach to damage detection of traffic signs is to use sensor-based methods, where damage is detected and classified using sensors such as cameras, lasers, or radar. These sensors may be mounted on vehicles or other platforms, and can be used to scan the traffic signs as they pass by. The collected data can then be analyzed to detect and classify damage in the traffic signs.

Another commonly used approach is to use image-based methods, where damage is detected and classified by analyzing images of the traffic signs. This may involve the use of machine learning algorithms, such as CNNs, to automatically analyze the images and identify signs of damage. Other methods may involve the use of computer vision techniques, such as feature extraction and matching, to detect and classify damage in the images.

Hybrid solutions are also used to take the advantage of both approaches such as it is mentioned in [10]. Damage on the traffic sign is detected using camera image and mobile laser scanning (MLS) data according to You et al [10]. Traffic signs are detected from an image by using CNN, Fast R-CNN and detected traffic sign's pixel coordinates are found. After that point cloud data is analyzed on detected coordinates. By that approach, it is possible detect tilt, physical deformations like dents and local bends, fallen down or disappeared traffic signs. Example output of the study is shown in Figure 3.

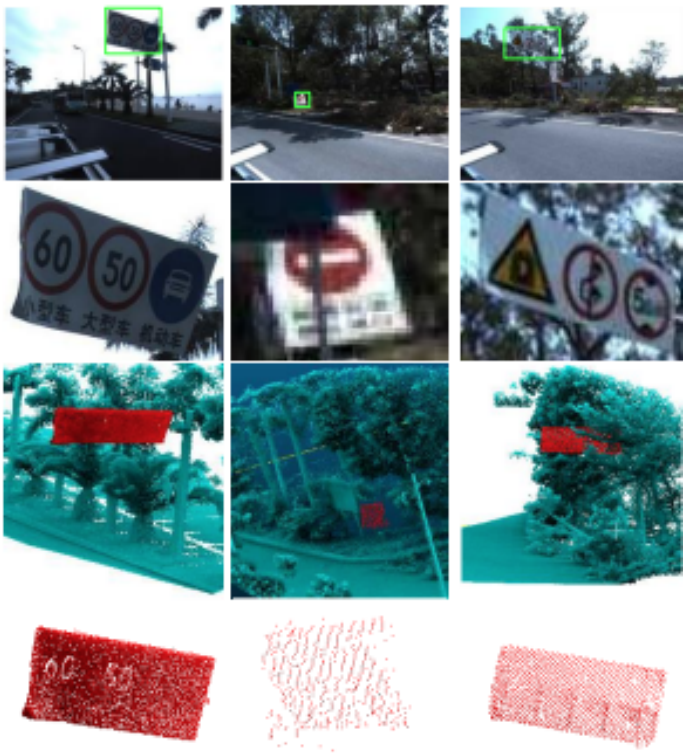


Figure 3. *Traffic sign damage analysis using image MLS data [10]*

Damage is detected as the paper [10] mentioned. Despite the fact of this study, damage scope is limited with bend, tilt and fallen down on ground. There might be damaged which can not effect the orientation and change the shape of the traffic sign such as vandalizing by paint or stickers, snow or mud or bird poop debris on the traffic sign and leafs surround etc.

Another study which belongs to Vokhidov et al.[9], worked with detecting the damages arrow signs on road. In that research only image processing and machine learning techniques are used. A custom dataset is created from several online sources which includes various directions of arrows (left, forward-left, forward, right, forward-right, forward-left-right). Dataset is trained with CNN based model. Methodology is shown in Figure 4. Damage detection is made by the accuracy of the detected sign on road. If the sign is not complete and/or have damage on it, the accuracy of detection gives lower result and the damage is detected. Writers claim that approach has a quite good accuracy due to not being effected by occlusions, illumination changes, shadows or perspective distortions.

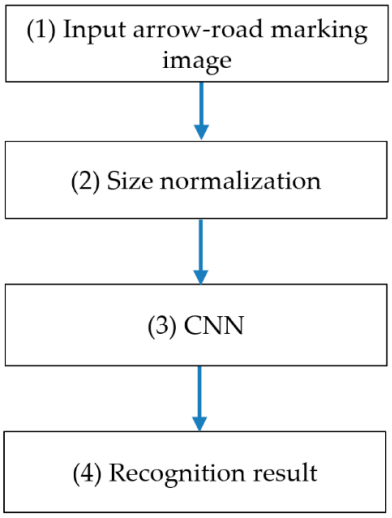


Figure 4. *Flowchart of road damage detection [9]*

The downside of this project is confusion between false positives/true negatives and damaged road sign both have the lower accuracy values. This causes different classes of road signs are predicted as damaged, or damaged road signs are predicted as not damaged other sign of road sign class.

One other approach is detecting the damage on the traffic sign by using radiometric information. Jorge et al. [2] claims that the damage on the traffic sign can be detected by analyzing the flatness of the traffic sign and the relative rotation angle with the pavement or ground. For this purpose optical profilers, ground penetrating radar and video cameras are used to support the radiometric data from laser scanner. Point cloud data is generated by the help of laser scanner and digital camera. Coating material on traffic sign acts like a high reflective surface which is detected by photoelectric detector of the laser scanner. As it is seen from the Figure 5b, there are high reflective points which indicates the position of traffic signs. Finally, highly reflected photons from the traffic sign are filtered to establish a 3D segmentation to analyze the shape of the metal sheet and find the angle with respect to the ground angle.

(a) Surveying process (b) 3D scanning data

Figure 5. Radiometric damage analysis on traffic sign [2]

This approach is accurate at detecting the rotational and bend related damages on traffic sign metal sheets. However there are several potential drawbacks to using this approach for damaged traffic sign detection. First of all, laser scanner sensors are expensive to purchase and maintain, especially if a large number of sensors are needed to cover a large area. In addition to that, laser data can be affected by weather conditions such as rain or snow, which can affect their accuracy and reliability. This can make it difficult to rely on laser sensors for consistent, reliable damaged traffic sign detection. Moreover, this approach is not be able to integrated on vehicles, which makes the system stationary and not be able detect the damaged traffic signs in short time. Finally this approach can be complex to set up and operate, requiring specialized training and expertise. This can increase the time and resources needed to deploy and maintain the damaged traffic sign detection system.

Another research is about the damage detection on roads [8] by Chanjun Chun and Seung-Ki Ryu. Traffic signs are not detected in this paper, however there are useful methods to consider. This paper that explains a method for detecting damage to road surfaces using a combination of fully convolutional neural networks (FCNNs) and semi-supervised learning. The main focus of the paper is on the use of FCNNs for image analysis, and how these networks can be trained to accurately identify and classify different types of road surface damage. Their approach to using semi-supervised learning is to improve the accuracy of the FCNN in detecting road surface damage. In this approach, the FCNN is trained using a combination of labeled and unlabeled data, which allows the model to learn from both examples of known road surface damage and more general patterns in the data.

The paper [12] investigates a deep learning approach for detecting pavement distress from non-ideal photographic images of the road. A. Tepljakov et al. note that due to inconsistent data quality, a significant challenge in this task is to produce training and validation data that bears coherent information sufficient for the task of successfully training a deep convolutional neural network (DCNN) that provides required detection performance.

They propose a method for detecting pavement distress and provide work-in-progress experimental results which are analyzed. The authors describe their motivation as the fact that in Estonia, daily temperature fluctuations around 0 degrees Celsius for more than five months a year can lead to expansion of cracks and other defects requiring frequent road inspections.

There are certain limitations and challenges that may need to be considered the research paper [8] and [12]. One potential downside of using FCNNs and DCNNs for road surface damage detection is that these networks require large amounts of labeled training data in order to be effective. This can be a challenge if the data is not readily available, or if it is difficult to label the data accurately. Moreover FCNNs and DCNNs may not be able to accurately detect all types of road surface damage, particularly if the damage is subtle or not easily visible in the images. In such cases, the model may not be able to accurately classify the damage, which could lead to false negatives or false positives.

In conclusion, there have been a number of studies conducted on the topic of damage detection of traffic signs in recent years. These studies generally focus on the development of algorithms and systems for automatically detecting and classifying damage to traffic signs, such as dents, scratches, graffiti, or fading. There are a wide variety of techniques and methods used in these studies, including sensor-based methods, image-based methods, and hybrid solutions. The goal of these studies is to improve the safety and efficiency of transportation systems by identifying damaged traffic signs before they become a hazard to road users.

However, There are still limitations and challenges such as detecting damage that does not affect the orientation or shape of the traffic sign, like vandalism by paint or stickers, snow, or debris. In addition, more research is needed to improve the performance and robustness of these systems and to address the challenges involved in deploying these systems in real-world settings. Future research directions may include the use of deep learning and more advanced computer vision techniques to improve the performance of traffic sign damage detection systems.

3. Methodology

The methodology for damaged traffic sign detection involves several steps that aim to accurately identify that the traffic sign is damaged or not. First, the traffic sign detection process must be done accurately, then damage should be investigated on the detected traffic sign. To detect traffic signs, firstly the necessary dependencies must be installed. This will ensure that all of the tools required for the task are properly configured and ready to use.

Next, a dataset of traffic sign images is gathered, which includes a large number of images and labels. These images and labels will be used to train and evaluate the performance of the model.

Before the data can be used to train the model, it must be preprocessed. This may include tasks such as resizing the images and generating the corresponding label files. Preprocessing the data in this way helps to ensure that it is in a suitable format for training and enables the model to more easily learn from the data.

Once the data is preprocessed, state-of-the-art object detection algorithm is used to train a model on the dataset. This involves specifying the model architecture and setting the training hyperparameters. The model is then trained by running the training process, which may take some time depending on the size of the dataset and the complexity of the model. Then, it can be used to perform traffic sign detection on new images by passing the images through the model and using the output to identify and label the traffic signs in the image. This allows the model to be applied to real-world situations illustrated as Figure 6 and enables it to help identify traffic signs in the field.

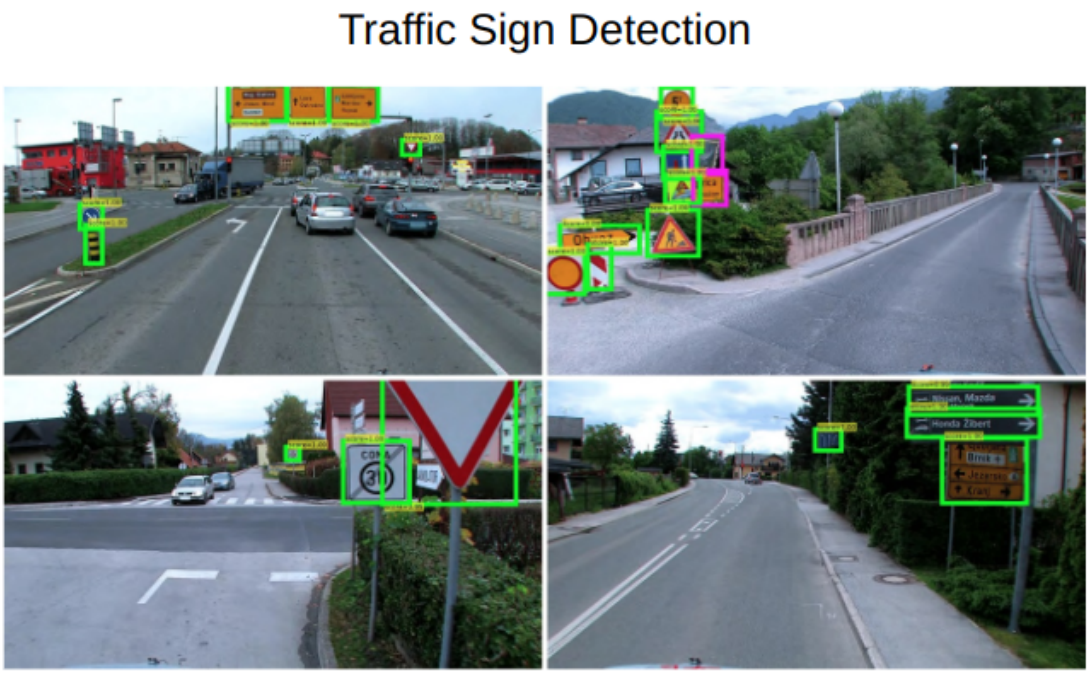


Figure 6. Traffic sign detection methodology [13]

To evaluate the model's performance, metrics such as accuracy and mean average precision are used. These metrics provide a measure of how well the model is able to identify and classify traffic signs in the images.

After traffic sign detection process finished, then damage detection on detected traffic sign will be analyzed. To detect the anomalies on images, autoencoder network architecture is used. The first step in training an autoencoder is to collect and preprocess a dataset of images. The images should be resized to a standard size to ensure that the autoencoder can learn to reconstruct them effectively. The autoencoder is a type of neural network that is trained by minimizing the difference between the original images and the ones it produces by encoding and then decoding them. Figure 7 shows an example process how autoencoder generates output image with respect to input image. Preprocessing the images in a consistent manner is essential for the autoencoder to learn effectively.

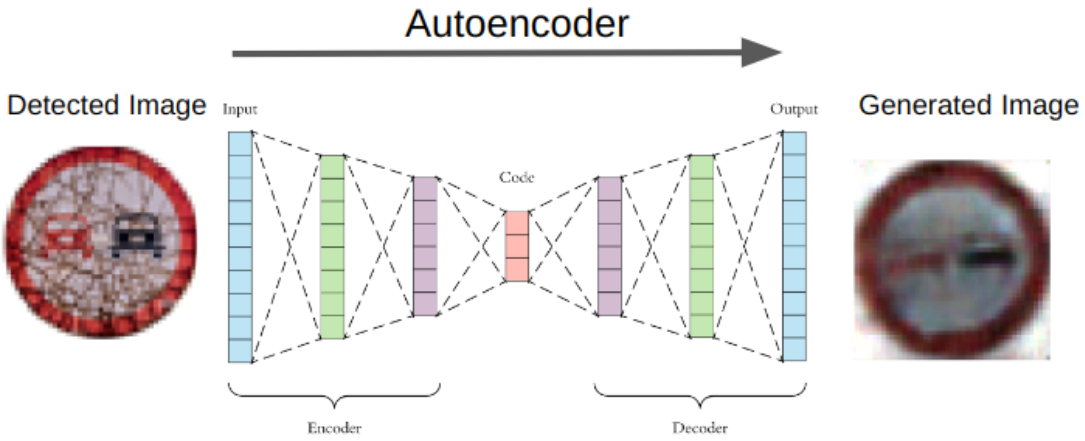


Figure 7. Traffic sign detection methodology [14]

Then dataset split into three smaller sets: a training set, a validation set, and a test set. The autoencoder is trained on the training set, and once it has been trained, it can be used to encode and decode images from the validation set. By comparing the reconstructed images to the original images, the reconstruction error for each image in the validation set can be calculated. This error can be used to evaluate the quality of the reconstructions. Finally, the autoencoder can be tested on the test set by calculating the reconstruction error for each image in that set.

Finally reconstructed image from autoencoder (artificial image) is compared with the detected traffic sign image. To determine the damage level, image quality assessments are applied in order to measure the distance between detected and regenerated traffic sign as it is shown in Figure 8.

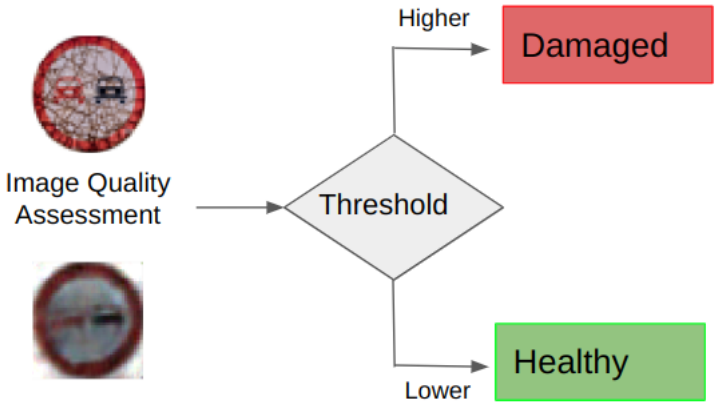


Figure 8. Decision methodolody

4. Traffic Sign Detection

Traffic sign detection algorithms are critical for ensuring the safety and efficiency of transportation systems. These algorithms are used to interpret and understand traffic signs in images or video, and can provide important information and instructions to drivers, pedestrians, and other road users. In autonomous vehicles, traffic sign detection algorithms are a crucial component for enabling the vehicle to navigate and make decisions. In driver assistance systems, such as advanced driver-assistance systems (ADAS) or intelligent transportation systems (ITS), traffic sign detection algorithms can provide drivers with warnings or recommendations based on the traffic sign information.

4.1 Selection of Dataset for Traffic Sign Detection

When it comes to traffic sign detection, having a dataset with a sufficient number of images is important in order to train and evaluate models effectively. The dataset should also include a diverse range of classes and variations in the signs in order to be representative of real-world conditions. Additionally, accurate annotations are necessary to provide the necessary information about the location and class of each traffic sign in the images.

Another important aspect of the dataset is the data collection from a variety of sources, including images taken under different lighting and weather conditions, as well as at different times of day. This helps to make it a more realistic and challenging dataset, as traffic signs in the real world can vary significantly in appearance. The dataset also includes images taken from different viewpoints, including front, rear, and side views, which can be useful for training and evaluating models that need to handle a wide range of viewing angles.

There are several datasets that are commonly used for traffic sign detection and classification tasks. Commonly used ones are mentioned below:

- The German Traffic Sign Detection/Recognition Benchmark (GTSDB/GTSRB) [15], [16]: This dataset includes more than 50,000 images of German traffic signs, with detailed annotations for the location, orientation, and class of each sign.
- The KITTI Vision Benchmark Suite [17]: This dataset includes images and video from a moving vehicle, with annotations for the location and class of traffic signs.
- The Belgian Traffic Sign Detection and Classification (BTSDC) dataset [18]: This dataset includes more than 7,000 images of Belgian traffic signs, with annotations for the location and class of each sign.
- The Traffic Sign Recognition (TSR) dataset [19]: This dataset includes more than 1,000 images of UK traffic signs, with annotations for the location and class of each sign

GTSDB/GTSRB is widely used and well-known in the research community, which means that it has a large number of citations and is often used as a benchmark for comparison with other algorithms. This can make it a useful resource for researchers who want to compare their algorithms to state-of-the-art approaches or to see how their algorithms perform relative to other methods.

GTSRB is a dataset of more than 50,000 images of German traffic signs, including 43 different classes. The dataset includes both real-world images and synthetically generated images. It was created to evaluate the performance of traffic sign recognition models.

GTSDB is a dataset of more than 900 images of German traffic signs, with more than 43000 annotations. It include more complex situation, such as traffic signs occluded by other objects, traffic signs in the distance, and traffic signs with glare or shadows. It was created to evaluate the performance of traffic sign detection models.

One advantage of the GTSDB/GTSRB compared to other datasets is its size and diversity. GTSDB includes more than 50,000 images for GTSRB and 900 images for GTSDB, which is a large dataset that can provide a reliable evaluation of algorithms. It also includes a wide range of traffic signs, with a total of 43 classes, which allows for the evaluation of algorithms on a diverse set of signs.

Another advantage of GTSDB/GTSRB is its detailed annotations. Each image in the dataset includes precise annotations for the location, orientation, and class of each traffic sign. This allows for the precise evaluation of algorithms on specific tasks, such as localizing traffic signs in an image or identifying the class of a traffic sign, which is an answer of RQ1.

The shape, size and content of the traffic signs can be varied from country to country, however they are categorised into three main groups according to GTSDB: Prohibitory, Mandatory and Danger. The usage of prohibitory traffic signs are prohibiting certain types of manoeuvres or actions[20]. Road signs that specify the responsibilities of all traffic using a certain section of road are known as mandatory signs. Lastly, a sort of traffic sign known as a danger sign denotes a potential threat, obstruction, or condition that calls for extra care. In Figure 9, example of these categories are shown.

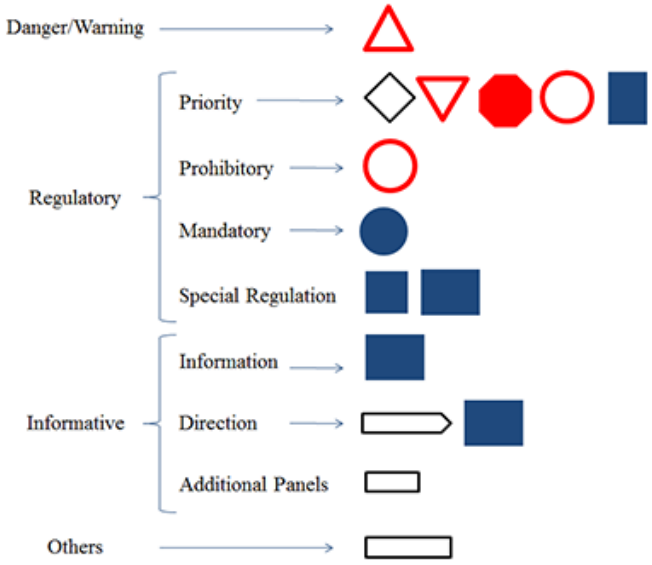


Figure 9. *Traffic sign general categories [21]*

Images are selected autonomously from training samples in this project and training process needs to consist of large amount of reliable and proven image set of traffic signs. Hence this is a research for Estonia, which is part of Europe, GTSDB fits perfectly to the needs of the project. Moreover GTSDB presented as a competition at IJCNN 2013 (International Joint Conference on Neural Networks)[22].

GTSDB is a publicly available dataset for evaluating the performance of traffic sign detection algorithms. It contains traffic signs images of from Germany, along with annotations indicating the location and class of each sign. The dataset is divided into a training set and a test set, and has been widely used for evaluating the accuracy and robustness of traffic sign detection algorithms.

The GTSDB evaluation benchmark is used to evaluate the performance of algorithms for detecting traffic signs in images. It provides a standard set of images for testing, along with ground truth labels indicating the locations and types of traffic signs in each image. The benchmark also includes a set of evaluation metrics for measuring the accuracy of traffic sign detection algorithms. This benchmark commonly used in the field of computer vision and machine learning for research and development of algorithms for traffic sign detection and recognition.

4.1.1 GTSDB Dataset Analysis

Each image in GTSDB dataset, contains zero to six traffic signs and all images are stored in .ppm format. Traffic sign sizes in an image is varied from 16x16 to 128x128 pixels and they can be appear in various perspectives and lighting condition. All classes are illustrated Figure 10.

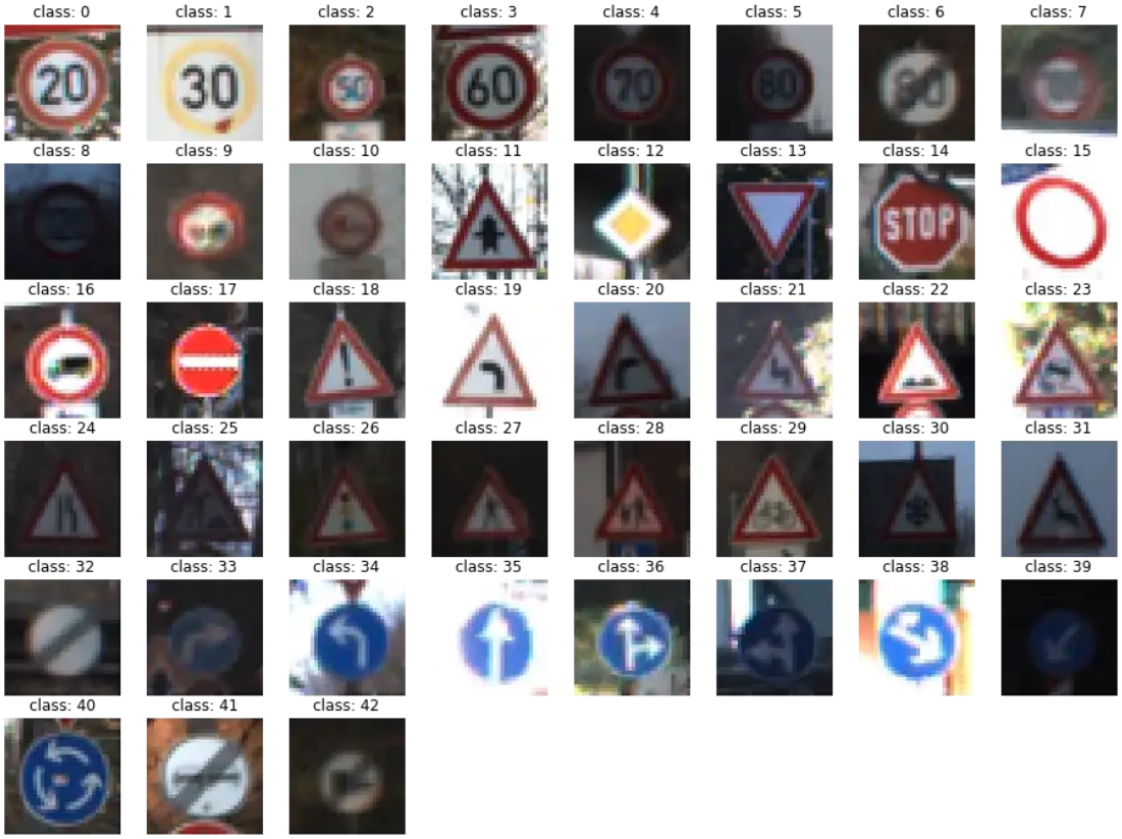


Figure 10. *GTSDB Classes [23]*

Number of instances per class are shown in Figure 11.

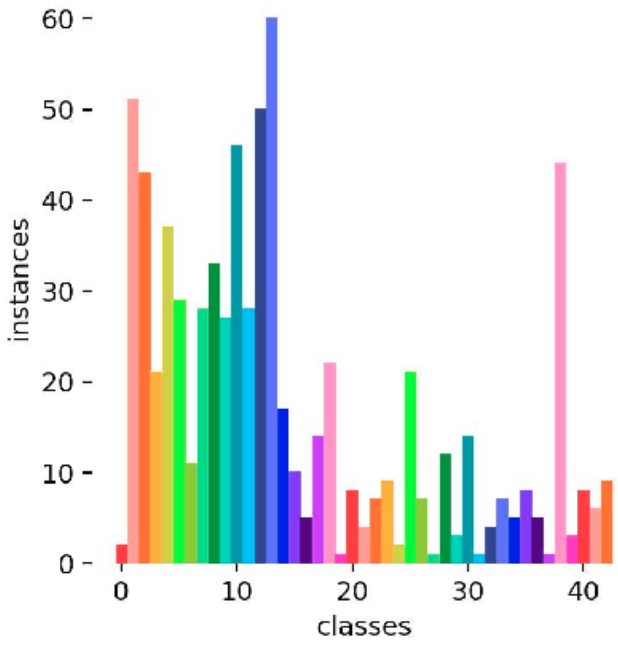


Figure 11. *GTSDB Class instances*

4.1.2 Description of GTSDB Dataset Annotation Format

Annotations are lined up in a comma separated file (CSV) and each property of an image separated with a semicolon (;). Each line contains the following information with an order:

- Filename: Filename of the image the annotations apply for
- Traffic sign's region of interest (ROI) in the image:
  - x_min -> Leftmost image column of the ROI
  - y_min -> Upmost image row of the ROI
  - x_max -> Rightmost image column of the ROI
  - y_max -> Downmost image row of the ROI
- ID providing the traffic sign's class

Example line of a CSV file is shown as below:
00027.ppm;102;375;157;430;2

However, annotation format is need to be converted to YOLO labeling format in order to obtain training processes. In the YOLO labeling format, directory hierarchy is needed to be set and the root directory needs be written in a .txt file with the same name is created for each image file in the same directory. The file hierarchy is shown in Figure 12.

Moreover the labeling format of YOLO is as:

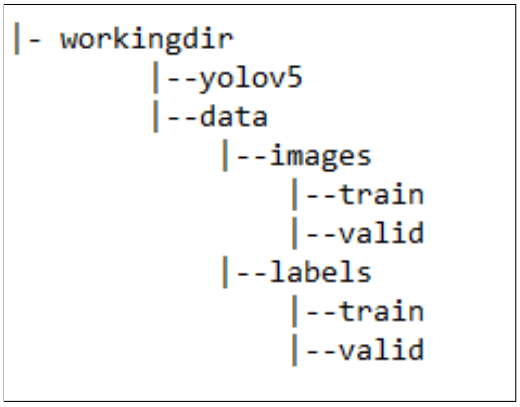


Figure 12. *YOLO directory hierarchy*

- ID providing the traffic sign's class
- X center of the bounding box
- Y center of the bounding box
- Width of the bounding box
- Height of the bounding box

Example line of a YOLO annotation format is shown as below:
5 0.47573529411764703 0.575625 0.01911764705882353 0.03375

To convert annotation formats from GTSDB to YOLO, below equations are used:

$$yolo\_center\_x = (gtsdb\_x\_max + gtsdb\_x\_min)/2$$

$$yolo\_center\_y = (gtsdb\_y\_max + gtsdb\_y\_min)/2$$

$$yolo\_width = gtsdb\_x\_max - gtsdb\_x\_min$$

$$yolo\_height = gtsdb\_y\_max - gtsdb\_y\_min$$

These equations are used to convert the bounding box coordinates of an object from the format used in the GTSDB dataset to the format used by the YOLO algorithm. The GTSDB dataset provides the coordinates of the bounding box in terms of the x and y coordinates of the top-left and bottom-right corner of the bounding box, whereas YOLO uses the x and y coordinates of the center of the bounding box, and the width and height of the bounding box.

4.2 Selection of Traffic Sign Detection Algorithm

Comparing the methods for traffic sign detection is difficult due to the extensive research of works done in different types of datasets and detection methods. According to [24], MS-COCO dataset is used for benchmark purpose because it provides ground truth for many different tasks with rich annotation content including object detection, segmentation, feature detection, image caption, multi-label image classification and object recognition. This approach is used for trying the best solutions on traffic sign detection problem. Top algorithms and best results on precision and performance are chosen to be tried on GTRDB dataset. The primary goal is to detect the traffic sign with high mean average precision, and then choose the optimum model by size, because it is also important to run that model on embedded computer or system on chip (SoC) specially.

There are several different algorithms that can be used for object detection, each with its own strengths and weaknesses. Some of the most common object detection algorithms include:

- R-CNN (Regional Convolutional Neural Network) [25]: R-CNN is a method that involves using a convolutional neural network (CNN) to classify individual regions of an image as containing an object of interest or not. R-CNN algorithms are generally accurate but can be slow to run.
- Fast R-CNN [26]: Fast R-CNN is an improvement on R-CNN that uses a single CNN to process the entire image, rather than individual regions. This makes Fast R-CNN faster than R-CNN.
- Faster R-CNN [27]: Faster R-CNN is another improvement on R-CNN that uses a "region proposal network" to generate a set of potential object locations in an image, which are then processed by a CNN to classify the objects. This method is faster than R-CNN and Fast R-CNN.
- YOLO (You Only Look Once): YOLO is a single-shot object detection algorithm that divides an image into a grid and uses a CNN to predict the class and location of objects within each grid cell. YOLO is much faster than R-CNN, Fast R-CNN, and Faster R-CNN and is generally considered to be one of the most accurate object detection algorithms.

Some advantages of YOLO over other methods are:

- YOLO does not require region proposals, which can save significant computational time.
- YOLO is a single pass algorithm, it scans the image only once, which makes it faster

than R-CNN and its variants that need multiple passes.

- YOLO is able to process images in real-time, making it suitable for video or live feed applications.
- YOLO has a simple structure and is easy to implement, compared to R-CNN and its variants.

To answer the RQ2, these advantages make YOLO as a solid traffic sign detection algorithm.

YOLO is a popular object detection algorithm that uses deep learning to identify and localize objects in images and videos. It was developed by Joseph Redmon and Ali Farhadi and has been improved and updated through several versions, with the latest being YOLOv7 [6].

The key concept of YOLO algorithm is Non-maximum suppression (NMS). NMS is a technique used to eliminate redundant or overlapping bounding boxes produced by an object detection algorithm. It involves selecting the bounding box with the highest confidence score and discarding any boxes that overlap with it by more than a certain threshold. This helps to improve the accuracy and efficiency of the model.

One of the main advantages of YOLO is its fast object detection performance, making it suitable for use in real-time applications such as self-driving cars, surveillance systems, and robotics. It is also relatively easy to implement and has a large community of developers and users, which makes it well-supported and constantly improving. However, like any machine learning model, the performance of YOLO can vary depending on the quality and diversity of the training data and the specific application it is being used for.

Using different versions of YOLO for traffic sign detection can allow to observe for better performance and improved accuracy. Each version of YOLO may have its own unique strengths and capabilities that make it more suitable for a particular task or dataset. For example, one version of YOLO may be faster and more efficient, while another version may have a higher accuracy rate. By comparing a variety of YOLO models, it is possible to leverage the strengths of each model and combine them to achieve the best possible results.

4.2.1 YOLOv4

YOLOv4 [5] is known for its fast object detection performance and good accuracy, making it a popular choice for many real-time applications such as self-driving cars, surveillance systems, and robotics. It works by dividing an image into a grid of cells and using a CNN to predict the bounding boxes and class probabilities for objects in each cell. YOLOv4 also uses anchor boxes, which are predefined boxes of different shapes and sizes, to help the model better detect objects of different sizes and aspect ratios.

YOLOv4 is a popular object detection algorithm that has several key advantages, including its real-time performance and ease of implementation. These characteristics make it well-suited for use in applications that require fast object detection. Additionally, YOLOv4 has a large and active developer community, which provides ongoing support and helps to continuously improve the algorithm.

MS-COCO dataset is commonly used for benchmarking the performance of object detection algorithms like YOLO because it provides a standardized and challenging test set for evaluating the performance of different models. MS-COCO contains a wide variety of objects in different contexts, accurate object annotations and a large scale. Furthermore, it is widely used by researchers and practitioners, allowing them to easily compare their results with the state of the art. The use of common dataset like MS-COCO facilitates the comparison and improvement of different models, allowing for the development of more accurate and efficient object detection algorithms.

Training of each model is done on Tesla V100 graphics card and results with various weights on MS-COCO dataset are shown in table 1:

4.2.1.1 Performance Results on MS-COCO Dataset

Table 1. *YOLOv4 performance analysis on MS-COCO Dataset*

Model	**#Params(M)**	**Size**	**FPS**	**mAP@0.5**	**AP**	**FLOPs**
yolov4-csp-x	99.75	640	70	0.662	0.475	223.0(G)
yolov4-csp-darknet-53	27.6	608	62	0.657	0.43	52(G)
yolov4-csp-darknet-50	20.6	512	62	0.606	0.424	31(G)

Table 1 lists the performance of different versions of YOLOv4, a object detection model, on the MS-COCO dataset. The performance measures include:

- #Params (M): Number of model parameters, measured in millions.
- Size: Size of input image, measured in pixels.
- FPS: Frames per second, a measure of the model's processing speed.
- mAP@0.5: Mean Average Precision at 0.5, a measure of the model's accuracy.
- AP: Average Precision measure of a model's accuracy that takes into account both the precision and recall of the model's predictions.
- FLOPs: Floating Point Operations, a measure of the model's computational complexity.

Table 1 shows that the yolov4-csp-x model has the highest accuracy (highest mAP@0.5 and AP), but it is also the largest and most computationally complex model (highest #Params and FLOPs). On the other hand, the yolov4-csp-darknet-50 model is smaller and faster, but has lower accuracy compared to the other models.

4.2.1.2 Experimental Results on GTSDB Dataset

Table 2. *Experimental results on GTSDB dataset using YOLOv4*

Model	#Params(M)	#Layers	GFLOPS	mAP@0.5	Weight File Size(MB)
yolov4-csp-x	**99.75**	**611**	**223.0**	**0.213**	**798.9**

Table 2 lists the performance of the YOLOv4 object detection model on the GTSDB dataset. The experiments were run on an NVIDIA GTX 1070 GPU. The performance measures include:

- #Params (M): Number of model parameters, measured in millions.
- #Layers: Number of layers in the model.
- GFLOPS: Giga Floating Point Operations per Second, a measure of the model's computational complexity.
- mAP@0.5: Mean Average Precision at 0.5, a measure of the model's accuracy.
- Weight File Size (MB): Size of the model's weight file, measured in megabytes.

Table 2 shows that the yolov4-csp-x model has a mAP@0.5 of 0.213 and a weight file size of 798.9 MB. It also has a high number of parameters and layers, and a high GFLOPS value, indicating that it is a complex and computationally intensive model.

4.2.2 YOLOv5

YOLOv5 by Ultralytics [28] has been designed to be even faster and more accurate than its predecessors, with a number of improvements and optimizations. It uses a hybrid architecture that combines the strengths of both anchor-based and anchor-free object detection approaches, and also includes a number of new features such as dynamic image sizes, automatic mixed precision training, and a new data augmentation technique called Mosaic data augmentation.

YOLOv5 has various network models to train. These models are YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x which are called P5 architecture. Moreover these models are extended by more efficient architecture and added '6' (i.e YOLOv5x6) at the of previous network models and named is as P6 architecture. The difference between P5 and P6 architectures are: P5 architecture has 3 output layers P3, P4, P5 at strides 8, 16, 32, trained at –img 640. However P6 architecture has 4 output layers P3, P4, P5, P6 at strides 8, 16, 32, 64 trained at –img 1280.

Training results on different models are shown in table 3 (MS-COCO dataset and Tesla V100 Graphics card):

4.2.2.1 Performance Results on MS-COCO Dataset

Table 3. *YOLOv5 performance analysis on MS-COCO Dataset*

Model	#Params(M)	Size	FPS	mAP@0.5	AP	FLOPs
yolov5n	1.9	640	159	0.457	0.28	4.5(G)
yolov5s	7.2	640	155	0.568	0.374	16.5(G)
yolov5m	21.2	640	122	0.641	0.454	49.0(G)
yolov5l	46.5	640	99	0.673	0.49	109.1(G)
yolov5x	86.7	640	83	0.689	0.507	205.7(G)
yolov5n6	3.2	1280	123	0.544	0.36	4.6(G)
yolov5s6	12.6	1280	122	0.637	0.448	16.8(G)
yolov5m6	35.7	1280	90	0.693	0.513	50.0(G)
yolov5l6	76.8	1280	63	0.713	0.537	111.4(G)
yolov5x6	140.7	1280	38	0.727	0.55	209.8(G)

Models with larger numbers of parameters and higher FLOP counts tend to have better accuracy, as indicated by higher values for the mAP@0.5 and AP columns. However, these models also tend to have lower FPS, indicating that they are slower to process images.

On the other hand, the models with smaller sizes and lower FLOP counts tend to have lower accuracy, as indicated by lower values for the mAP@0.5 and AP columns. However, these models also tend to have higher FPS, indicating that they are faster to process images.

It's worth noting that the size of the input images also seems to affect the model's performance. The models tested on smaller (640x640) images tend to have higher FPS but lower accuracy, while the models tested on larger (1280x1280) images tend to have lower FPS but higher accuracy.

4.2.2.2 Experimental Results on GTSDB Dataset

Table 4. *Experimental results on GTSDB dataset using YOLOv5*

Model	**#Params(M)**	**#Layers**	**GFLOPS**	**mAP@0.5**	**Weight File Size(MB)**
yolov5n	1.82	270	4.4	0.354	4.0
yolov5s	7.14	270	16.3	0.510	14.6
yolov5m	21.04	369	48.8	0.536	42.5
yolov5l	46.36	468	109.0	0.594	93.3
yolov5x	86.50	567	205.5	0.588	173.6
yolov5n6	3.18	355	4.5	0.298	6.7
yolov5s6	12.48	355	16.7	0.434	25.4
yolov5m6	35.52	481	49.8	0.582	71.6
yolov5l6	**76.49**	**607**	**111.2**	**0.597**	**153.6**
yolov5x6	140.44	733	209.7	0.550	281.7

Table 4 presents the results of a series of experiments conducted on the GTSDB dataset using the YOLOv5 object detection model. The experiments were run on an NVIDIA GTX 1070 GPU.

The results show that the model performance improves as the number of parameters and layers increases. The models with the suffix "6" appear to have slightly lower mAP values compared to their counterparts without the "6" suffix, but they have smaller weight file sizes. This suggests that these models have trade off some performance for reduced resource requirements.

The yolov5n model has the smallest number of parameters and the lowest GFLOPS, and it achieves an mAP of 0.354. As the number of parameters and GFLOPS increase, the mAP values also increase, with the yolov5l and yolov5x models achieving the highest mAP values of 0.594 and 0.588, respectively. However, it should be noted that the yolov5x model has a larger weight file size than the yolov5l model, which has implications for deployment and resource requirements.

In terms of resource requirements, the yolov5n model has the smallest weight file size, which makes it more suitable for deployment in scenarios where storage or memory is limited. On the other hand, the yolov5l and yolov5x models have larger weight file sizes but offers better performance.

4.2.3 YOLOR

YOLOR [4] means "You Learn Only One Representation". YOLOR has been proposed as unified network for encode implicit and explicit knowledge. YOLOR is different from YOLO versions in terms of authorship, architecture, and model infrastructure. YOLOR is especially designed for object detection as opposed to other machine learning use cases like object analysis or identification. This is thus because object detection is centered on the broad identifiers that place an object in a certain class or category. Other machine learning use cases, however, need for more precise procedures.

Results on MS-COCO dataset using Tesla V100 graphics card shown in table 5:

4.2.3.1 Performance Results on MS-COCO Dataset

Table 5. *YOLOR performance analysis on MS-COCO Dataset*

Model	#Params(M)	Size	FPS	mAP@0.5	AP	FLOPs
yolor-csp	56	640	106	0.712	0.528	121.0(G)
yolor-csp-x	100	640	87	0.731	0.548	223.0(G)
yolor-p6	37	1280	48	0.706	0.526	326.0(G)
yolor-w6	80	1280	44	0.72	0.541	454.0(G)
yolor-e6	116	1280	27	0.727	0.548	684.0(G)
yolor-d6	152	1280	22	0.733	0.554	937.0(G)

Table 5 presents the results of a series of experiments conducted on the MS-COCO dataset using the YOLOR object detection model.

The results show that the model performance improves as the number of parameters increases. The yolor-csp model has the smallest number of parameters and achieves an mAP of 0.712 and an AP of 0.528, while the yolor-d6 model has the largest number of parameters and achieves the highest mAP and AP values of 0.733 and 0.554, respectively. However, the larger models also have higher FLOPs, which have implications for computational requirements and performance.

The yolor-csp-x model has a slightly higher mAP and AP than the yolor-csp model but a lower FPS, indicating that it offers slightly better performance at the cost of reduced speed. The models with the suffix "6" (yolor-p6, yolor-w6, yolor-e6, and yolor-d6) tend to have higher mAP and AP values than the models without the "6" suffix, but they also have lower FPS and higher FLOPs.

A smaller model like yolor-csp, which has a higher FPS, is a good choice if processing speed is important. On the other hand, if the highest performance is the top priority, even at the expense of slower processing, a larger model like yolor-d6, which has higher mAP and AP values, is the better option.

4.2.3.2 Experimental Results on GTSDB Dataset

Table 6. *Experimental results on GTSDB dataset using YOLOR*

Model	#Params(M)	#Layers	GFLOPS	mAP@0.5	Weight File Size(MB)
yolor-csp	52.92	529	121.0	0.171	424.1
yolor-csp-x	**99.75**	**623**	**223.0**	**0.182**	**798.9**
yolor-p6	37.27	665	81.8	0.125	298.9
yolor-w6	79.87	665	113.8	0.139	639.9

Table 6 presents the results of a series of experiments conducted on the GTSDB dataset using the YOLOR object detection model. The experiments were run on an NVIDIA GTX 1070 GPU.

The yolor-csp model has the smallest number of parameters and the highest mAP value of 0.171, while the yolor-w6 model has the largest number of parameters and the second highest mAP value of 0.139. The yolor-csp-x model has a slightly higher number of parameters than the yolor-csp model but a lower mAP value, indicating that it may offer slightly worse performance despite having more parameters.

yolor-csp model has the highest GFLOPS and the largest weight file size, while the yolor-p6 model has the lowest GFLOPS and the second smallest weight file size. The yolor-csp-x and yolor-w6 models fall in between these two extremes.

Large model like yolor-csp is a good choice when high mAP value is desired, while a smaller model like yolor-p6 is preferred if reduced resource requirements are a concern. It should be noted that the mAP values in table 6 are relatively low, indicating that the models may not be performing well on the traffic sign detection task.

4.2.4 YOLOv7

YOLOv7 (also known as YOLOv3-SPP) is an improvement over earlier versions of YOLO, such as YOLOv3, and is designed to be faster and more accurate. It uses a combination of multiple CNNs and a new spatial pyramid pooling (SPP) layer to improve the performance of the model.

The theoretical background of YOLOv7 involves several key concepts from the fields of computer vision and machine learning, including CNNs, object detection, NMS and SPP which is a technique used to improve the performance of CNNs on tasks such as object detection. It involves dividing the feature map produced by a CNN into a pyramid of smaller regions and pooling the features within each region separately, allowing the model to maintain spatial information and improve its ability to detect objects at different scales.

"YOLOv7 is the best object detector available in terms of both speed and accuracy, ranging from 5 FPS to 160 FPS. It has the highest accuracy of 56.8 AP among all real-time object detectors that run at 30 FPS or higher on a GPU." as Wang Bochkovskiy and Liao said in the article [6]. In general, YOLOv7 offers a faster and more robust network architecture that offers a better feature integration approach, more precise object recognition performance, a more robust loss function, and an improved label assignment and model training efficiency.

Results on MS-COCO dataset using Tesla V100 graphics card shown in table 7:

4.2.4.1 Performance Results on MS-COCO Dataset

Table 7. *YOLOv7 performance analysis on MS-COCO Dataset*

Model	#Params(M)	Size	FPS	mAP@0.5	AP	FLOPs
yolov7	36.9	640	161	0.697	0.514	104.7(G)
yolov7-x	71.3	640	114	0.712	0.531	189.9(G)
yolov7-w6	70.4	1280	84	0.726	0.549	360.0(G)
yolov7-e6	97.2	1280	56	0.735	0.56	515.2(G)
yolov7-d6	154.7	1280	44	0.74	0.566	806.8(G)
yolov7-e6e	151.7	1280	36	0.744	0.568	843.2(G)

4.2.4.2 Experimental Results on GTSDB Dataset

Table 7 shows the results of experiments using the YOLOv7 object detection model on the MS-COCO dataset.

The model's performance improves as the number of parameters increases. The yolov7 model, which has the fewest parameters, has an mAP of 0.697 and an AP of 0.514, while the yolov7-d6 model, which has the most parameters, has the highest mAP and AP values of 0.74 and 0.566, respectively.

The yolov7 model has the highest FPS but the smallest input size, while the yolov7-d6 model has the lowest FPS but the largest input size. The other models fall in between these two extremes. The yolov7-x model has a slightly higher mAP and AP than the yolov7 model, but a lower FPS, indicating slightly better performance at the cost of reduced speed. Models with the "6" suffix (yolov7-w6, yolov7-e6, yolov7-d6, and yolov7-e6e) have higher

mAP and AP values than models without the "6" suffix, but also have lower FPS and higher FLOPs.

A smaller model with a higher FPS such as yolov7 is a suitable choice in case of the main concern is speed. However, yolov7-d6 is larger model with a higher mAP and AP, must be preferred when the highest possible performance is desired.

Table 8. *Experimental results on GTSDB dataset using YOLOv7*

Model	**#Params(M)**	**#Layers**	**GFLOPS**	**mAP@0.5**	**Weight File Size(MB)**
yolov7	37.42	415	104.7	0.165	75.2
yolov7-x	71.10	468	189.8	0.191	142.2
yolov7-w6	81.67	477	104.1	0.266	163.8
yolov7-e6	111.19	645	146.1	0.313	223.0
yolov7-d6	**153.86**	**733**	**200.5**	**0.326**	**308.4**
yolov7-e6e	165.7	1063	228.4	0.323	332.4

Table 8 presents the results of a series of experiments conducted on the GTSDB dataset using the YOLOv7 object detection model. The experiments were run on an NVIDIA GTX 1070 GPU.

The results indicate that increasing the number of parameters and layers leads to improved model performance. Specifically, the yolov7-d6 model, which has the largest number of parameters, achieved the highest mAP value of 0.326. The yolov7 model, which has the smallest number of parameters, had the lowest mAP value of 0.165. The yolov7-x model, which has more parameters than the yolov7 model, had a slightly higher mAP value, indicating slightly better performance.

In terms of computational requirements and model size, the yolov7 model has the lowest GFLOPS and the smallest weight file size, while the yolov7-d6 model has the highest GFLOPS and the largest weight file size. The remaining models have intermediate values for these measures.

A larger model like yolov7-d6 is a good choice when the goal is to achieve a high precision. On the other hand, if the focus is on reducing resource requirements, a smaller model like yolov7 is a better fit.

4.2.5 Traffic Sign Detection Model Selection

Based on the performance of YOLO network architectures trained on GTSDB, best of each method and their performances are listed in table 9:

Table 9. *Best of YOLO network architectures*

Model	**#Params(M)**	**#Layers**	**GFLOPS**	**mAP@0.5**	**Weight File Size(MB)**
yolov4-csp-x	99.75	611	223.0	0.213	798.9
yolov5l6	76.49	607	111.2	0.597	153.6
yolor-csp-x	99.75	623	223.0	0.182	798.9
yolov7-d6	153.86	733	200.5	0.326	308.4

According to table 9, the models with the highest accuracy in terms of mAP (mean average precision) when using a NVIDIA GTX 1070 graphics card are yolov5l6, with a mAP of 0.597, and yolov7-d6, with a mAP of 0.326.

In terms of other metrics, the yolov4-csp-x and yolor-csp-x models have a similar number of parameters (99.75 million) and layers (611 and 623, respectively), and similar GFLOPS values (223.0). However, the yolov4-csp-x model has a slightly higher mAP value (0.213) than the yolor-csp-x model (0.182).

The yolov5l6 model has a lower number of parameters (76.49 million) and layers (607), as well as a lower GFLOPS value (111.2) compared to the other models, but has the highest mAP value of all the models.

The yolov7-d6 model has the highest number of parameters (153.86 million) and layers (733), and the second highest GFLOPS value (200.5). It also has a relatively high mAP value (0.326).

It is worth noting that the NVIDIA GTX 1070 graphics card is a mid-range GPU that is capable of handling a variety of deep learning tasks. However, the performance of the models will also depend on the specific application and the quality of the training data. In general, models with a higher number of parameters and layers require more computational resources and is more computationally intensive to run, but also have a higher level of accuracy.

Overall, yolov5l6 model is the most accurate and efficient model among the ones listed, due to its high mAP value and relatively low number of parameters and layers. However, it is important to carefully consider the trade-off between model accuracy and computational efficiency when selecting a model for a traffic sign detection.

5. Image Anomaly Detection

5.1 Autoencoders

An autoencoder [29] is a type of neural network that is designed to learn a compressed representation of a dataset and then reconstruct the original data from this representation. It consists of two main components: an encoder and a decoder. The encoder takes the input data and maps it to a lower-dimensional representation, known as the latent code or bottleneck, while the decoder takes this latent code and maps it back to the original input data. Model learns the information of training data and generates an artificial image by applying the learned information on input image.

The theoretical background of autoencoders involves several key concepts from the field of machine learning, including:

- Neural networks: Autoencoders are a type of neural network, which are algorithms that are inspired by the structure and function of the human brain. They consist of multiple layers of artificial neurons that are connected by weighted edges and are trained to recognize patterns and features in data using an iterative optimization process.
- Encoding and decoding: The encoder and decoder components of an autoencoder are designed to transform the input data into a lower-dimensional representation and then reconstruct the original data from this representation, respectively. This process of encoding and decoding allows the autoencoder to learn a compact, abstract representation of the data that captures the most important features and patterns.
- Unsupervised learning: Autoencoders are typically used for unsupervised learning, which means that they do not require labeled training examples. Instead, they learn to recognize patterns and features in the data by minimizing a reconstruction loss function, which measures the difference between the input data and the reconstructed data produced by the decoder.

- Dimensionality reduction: One of the main motivations for using autoencoders is to perform dimensionality reduction, which is the process of reducing the number of features or dimensions in a dataset. By learning a lower-dimensional representation of the data, autoencoders can capture the most important patterns and features while discarding noise and unnecessary details. This can be useful for tasks such as data visualization and data compression.

According to Hinton [30], dimensionality reduction makes high-dimensional and complex data easier to classify, visualize, communicate and store. Dataset's directions with the highest variance are identified and each data point is then represented by its coordinates along each of these directions by principal component analysis (PCA). PCA is a method of reducing the number of variables in a dataset by transforming a set of correlated variables into a set of new, independent variables known as principal components. However PCA is a linear approach which is not sufficient enough to construct and represent the data with higher complexity as it is shown in Figure 13. That's why a non-linear a multi-layer adaptive network is used to transform high dimensional data into low dimensional code by encoder layer and recover the data from the code by decoder layer. In other words, general purpose of using autoencoder is capturing the most important parts of lower dimensional input features from a higher dimensional data. This process is called the encoding part. After encoding the input features, autoencoder reconstructs a new set of features by the captured data from training set which is called decoding process.

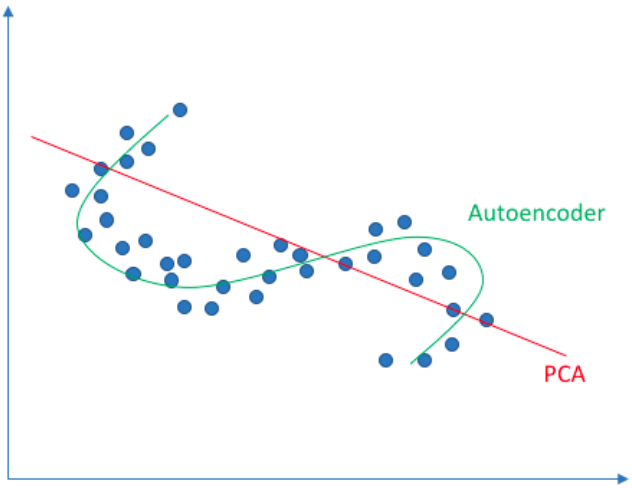


Figure 13. *Linear vs Nonlinear Dimensionality Reduction [30]*

To answer RQ3, autoencoders are used to detect damage on traffic signs due to they are a type of neural network that can be trained to reconstruct input data, such as images of traffic signs. When an autoencoder is trained on a dataset of undamaged traffic signs, it learns to encode the salient features of the signs into a compact, hidden representation, and then decode this representation back into an approximation of the original image. If a damaged traffic sign is then presented to the trained autoencoder, it will likely produce a reconstruction that is different from the original undamaged image, indicating that damage has occurred. Because autoencoders are able to learn the underlying structure of the input data, they are well suited for detecting subtle changes or variations in the images of traffic signs.

5.1.1 Dataset Creation for Anomaly Detection

In this thesis, autoencoder method is applied on German Traffic Sign Recognition Benchmark (GTSRB) dataset. GTSRB has the same class of traffic signs with GTSDB which is mentioned in chapter 4. The main difference between GTSDB and GTSRB is; GTSDB has 900 images in total which can includes multiple or none of a traffic sign, GTSRB has 39209 images in training set and 12630 in test set separately and each image are raw RGB and size varies from 15x15 to 250x250 pixels. Figure 14 illustrates the distribution of class instances.

Processing raw images (image file that has not been processed or edited by any software) caused accuracy problems due to the irrelevant pixels outside the traffic sign region of interest (ROI). GTSRB dataset provides a CSV file which includes the corner points, class ID and name of the file. Format of CSV file is includes items below:

- Width : Number of pixels of the image width.
- Height : Number of pixels of the image height.
- Roi.X1 : Top left X-coordinate of the ROI bounding box.
- Roi.Y1 : Top left Y-coordinate of the ROI bounding box.
- Roi.X2 : Bottom right X-coordinate of ROI the bounding box.
- Roi.Y2 : Bottom right Y-coordinate of ROI the bounding box.
- ClassId : Class label of the image. It is an Integer between 0 and 42.
- Path : Path where the image is present in the Train folder. Its format is "/Train/ClassId/image_name.png".

Example line of a CSV file is shown as below:
35,36,6,6,30,31,17,Test/05694.png

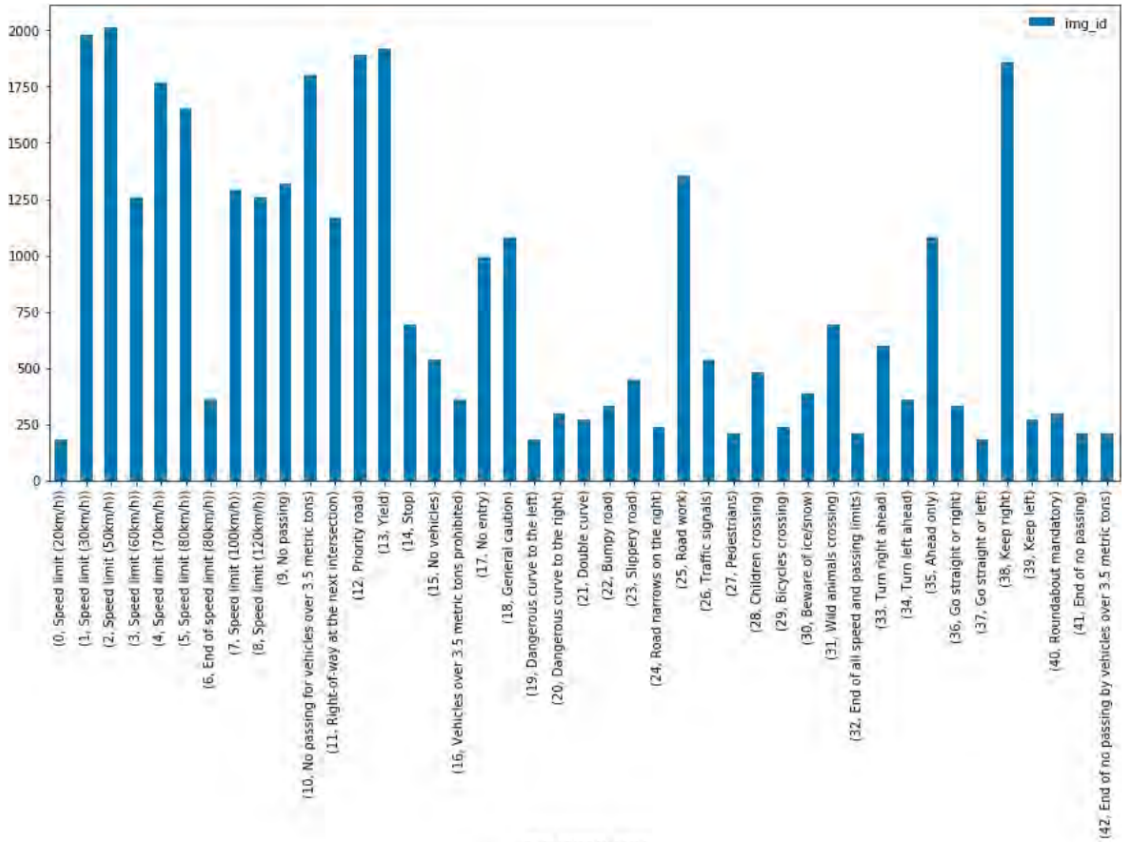


Figure 14. *Distribution of GTSRB training set[31]*

cropRoi function (which is shown in Appendix 3) takes the path, corner coordinates, traffic sign id, crops and saves the cropped ROI into the destination directory. I created a root directory named cropped_train and saved all the cropped ROI images inside it regarding the sub-directories named of each traffic sign id.

GTSRB presents a separate test images which we have worked with it while observing the results of the network model. The ROI of test images are also cropped using the function in Appendix 3.

5.1.2 Autoencoder Model for Traffic Sign Damage Detection

Model architecture plays a significant role in machine learning problems. Number of nodes, layers and parameters determines the size of network and the capacity of the model. If the model size is too big (or deep), it will be too chunky and inaccuracy problem can be seen due to memorizing each and every data of the training set. In contradiction, if the model size is too small (or shallow) results will be also inaccurate due to lack of data are learned from training set. That is why finding the right balance between model size and capacity is an important consideration when training autoencoders.

In this thesis, convolutional neural network (CNN) based autoencoder model architecture is used to train the network. Autoencoder architecture consists of 3 parts: Encoder, bottleneck and decoder. Encoder is responsible for compressing the train/validate/test set of input data into encoded form which consist orders of magnitudes smaller than input data. Bottleneck contains the compressed information of representation which are extracted from input data. Decoder network decompresses that representation information and reconstructs the data back and it has same size and architecture with the reversed of encoder network. General overview of the autoencoder is show in Figure 15.

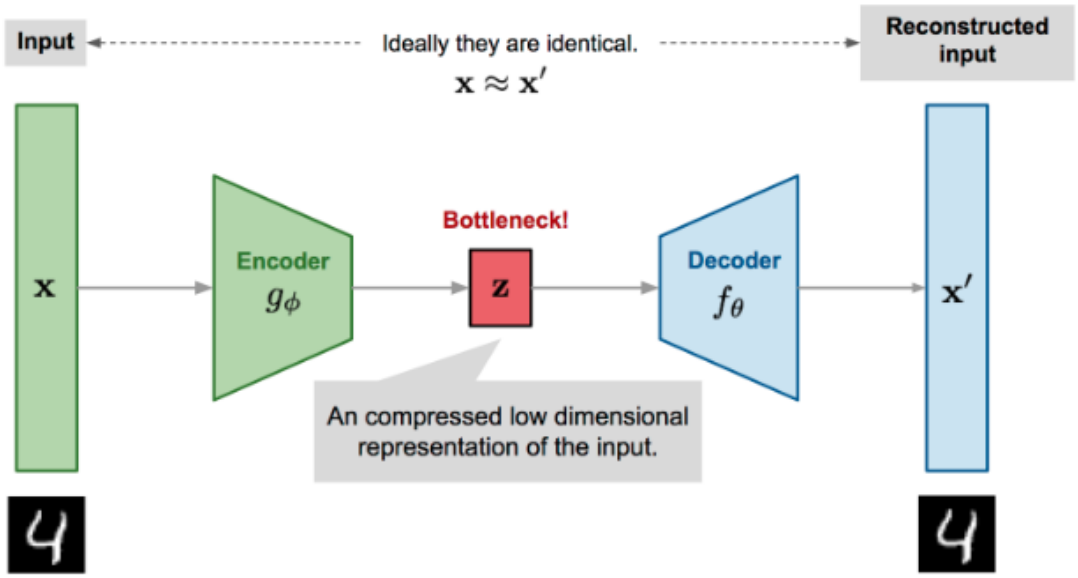


Figure 15. *Overview of Autoencoder Model Architecture [32]*

Model size is specified with 3 main hyperparameters. These are number of layers, number of nodes per layer and number of bottleneck layers. To create the model, 4 encoder layers, 4 bottleneck layers and 4 decoder layers are used. Detailed view of the autoencoder network is shown in Appendix 2.

5.1.3 Autoencoder Network Training

To train the autoencoder network, all the images must be resized and normalized before passing the network model. After cropping the ROI in dataset creation phase, image dimensions are resized to 48x48x3 and each pixel values in each channel are normalized the range between 0 and 1. These processes prepare all the images to feed into the model.

Images need to be collected as batches and separated into training and validation sets before training process. Hence autoencoder is a type of unsupervised learning, labeling is not required for this application. However to observe the training accuracy on a set of images, validation set needs to be created.

Lastly number of batches, epochs, loss function and optimizer need to be specified. According to Minhas and Zelek's approach [33], loss function is selected as L2 or mean squared error (MSE) and Adam optimizer selected.

L2 loss is the squared difference between a prediction and the actual value of evaluated for each set of input features in a dataset as shown in equation 5.1.

$$MSE = \frac{\sum_{i=1}^{D}(actual_i - predicted_i)^2}{number\ of\ observations} \tag{5.1}$$

Adaptive Movement Estimation (Adam) is a optimizing technique based on gradient descent technique. This optimizer is pretty efficient especially using with large-scale high-dimensional machine learning problems as the paper[34] indicates. Adam is a combination of momentum and root mean square propagation (RMSP) algorithms. The purpose of usage the momentum algorithm is to accelerate gradient descent by taking into consideration the exponential weighted average, which provides faster converge to minima. Mathematical approach of momentum is shown in formulas 5.2 and 5.3:

$$m_t = \beta m_{t-1} + (1 - \beta) * [\frac{\delta L}{\delta W_t}] \tag{5.2}$$

$$W_{t+1} = W_t - \alpha m_t \tag{5.3}$$

where:
m_t = aggregate of gradients at time t [current] ($m_t = 0$, $t = 0$)
m_{t-1} = aggregate of gradients at time $t - 1$ [previous]
W_t = weights at time t
W_{t+1} = weights at time $t + 1$
αt = learning rate at time t
δL = derivative of Loss Function
δW_t = derivative of weights at time t
β = Moving average parameter

And RMSP is expressed with the equations of 5.4 and 5.5 :

$$W_{t+1} = W_t - \frac{\alpha t}{(V_t + \epsilon)^{1/2}} * [\frac{\delta L}{\delta W_t}] \tag{5.4}$$

$$V_t = \beta V_{t-1} + (1 - \beta) * [\frac{\delta L}{\delta W_t}]^2 \tag{5.5}$$

where:
V_t = sum of square of past gradients (Vt = 0, t=0)
ϵ = A small positive constant

Combination of the 2 methods gives the Adam algorithm which is formulated in equation 5.6:

$$m_t = \beta m_{t-1} + (1 - \beta_1) * [\frac{\delta L}{\delta W_t}], \quad V_t = \beta V_{t-1} + (1 - \beta_2) * [\frac{\delta L}{\delta W_t}]^2 \tag{5.6}$$

By calculating "bias-corrected" M_t and V_t, this optimizer resolves the issue. In order to avoid large oscillations when close to the global minimum, this is also done to manage the weights as they approach it. The formulas are shown in equation 5.7.

$$\hat{m}_t = \frac{V_t}{1 - \beta_2^t}, \quad W_{t+1} = W_t - \hat{m}_t(\frac{\alpha}{\sqrt{\hat{V}_t} + \epsilon}) \tag{5.7}$$

where:
β_1&β_2 = decay rates of average of gradients in the above two methods.
α = Step size parameter / learning rate

Gradient descent approach is applied based on the formula above which provides controlled and unbiased throughout the process.

There is no one size fits all answer to the question of what the good values are for batch size, epoch number, optimizer, and learning rate when training an autoencoder model. The best values for these hyperparameters will depend on the specific characteristics of the dataset and the task at hand.

With respect to the general guidelines [35] and [36] that can be followed when choosing hyperparameters below:

- Batch size: A common rule of thumb is to set the batch size to a power of 2, such

as 32, 64, 128, etc. However, the optimal batch size will depend on the size of the dataset and the memory and computational resources available.

- Epoch number: The number of epochs to train for can vary widely, depending on the complexity of the task and the quality of the model. A good starting point is to train for a few epochs and then gradually increase the number if the model is not yet performing well.
- Learning rate: The learning rate determines the size of the step that the optimizer takes when updating the model's weights. A good starting point is to use a moderate learning rate, such as 0.001 or 0.0001. If the model is not converging or is oscillating, you can try decreasing the learning rate. If the model is converging too slowly, you can try increasing the learning rate

With respect to train and validation sets, loss function, network model and optimizer selection, model is trained with moving average β_1 and β_2 are selected 0.9 and 0.999 respectively as it is mentioned in paper [34], and learning rate is chosen as 0.0005. With these parameters, the network is trained for 100 epochs and batch size is chosen as 32.

5.1.4 Image Quality Metrics

Image Quality Assessment (IQA) is regarded as an image's defining characteristic. Assessment of picture quality measures how well images are viewed. Degradation is typically calculated in comparison to a reference image, which is an idea image. This approach can be expressed in several categories which reflects different aspects of metrics with respect to usage of construction or intended use. It is hard to separate the boundaries of image quality measurement in terms of metrics, however Pedersen et. al. [37] divided into four groups based on the usage of information:

- mathematics-based metrics: Applied on intensity and distortion of pixel values on image such as root mean squared error (RMSE), mean relative error (MRE) and peak to signal noise ratio (PSNR).
- Low-level metrics: Applied on visibility of distortions like contrast sensitivity function (CSF)
- High-level metrics: Applied on human visual system (HVS) to extract the information and structure from the image such as structural similarity index (SSIM) or visual image fidelity (VIF).
- Other metrics: Applied on other strategies or combination two or more of above groups such as visual to noise ratio (VSNR) which is combination of low and mid level visual properties.

First group (mathematics-based metrics) is very popular due to ease of implementation, analytical tractability and convenient for optimization with respect to Avcibas et. al. [37] mentioned in their study. Distortion is measured by taking the difference of original image and reproduction image. The mathematical formula of generalized form given in equation5.8:

$$E(x, y) = I_O(x, y) - I_R(x, y) \tag{5.8}$$

where I_O is the original image, I_R is the reproduction image, x and y is the pixel coordinates.

The most appropriate image quality assessment metric depends on the specific characteristics of the images being evaluated and the goals of the assessment. In this thesis, it will be compared the output image of autoencoder and the raw input image using RMSE, SSIM, MRE and PSNR as it is mentioned in [37].

Based on mathematics-based metrics, RMSE calculates the cumulative squared error between original and distorted images. RMSE is expressed as:

$$RMSE = \sqrt{\frac{1}{MN} \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} [E(x, y)^2]} \tag{5.9}$$

where x and y are the pixel coordinates and M and N are width and height of the image. RMSE is varies between 0 and ∞. Images are getting more identical when the RMSE value closer to 0. RMSE is also used as training loss function which has similar mathematical background.

MRE is also a term of basic mathematical image quality method. Different from RMSE, MRE value is calculated with absolute value instead of taking the square of the error. MRE is expressed as:

$$MRE = \frac{1}{MN} \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} [|E(x, y)|] \tag{5.10}$$

PSNR is used to measure peak error between original and reproduction image. This technique is usually used to measure the quality of compressed or distorted images. PSNR is useful metric for autoencoder networks due to an artificial image is generated from an original image which both images have highly correlation. Mathematical presentation of PSNR is shown in equation 5.11:

$$PSNR = 20 \, . \, log10(\frac{MAX_I}{\sqrt{MSE}}) \tag{5.11}$$

where the MAX_I is the maximum intensity value of the image and MSE needs to be calculated as it is shown in equation 5.9. The higher value of PSNR denotes the better quality of generated artificial image.

Last but not least, structural similarity index measurement (SSIM) is one of the essential metric to measure the image quality between original and reproduced as it is included into high level metrics. HVS is highly capable of identifying structural differences between a reference and sample images. To better formulate the difference, image comparison performed based on 3 key features: luminance, contrast and structure. Figure 16 shows the flow of structural similarity measurement system as expressed in Wang et al. [38].

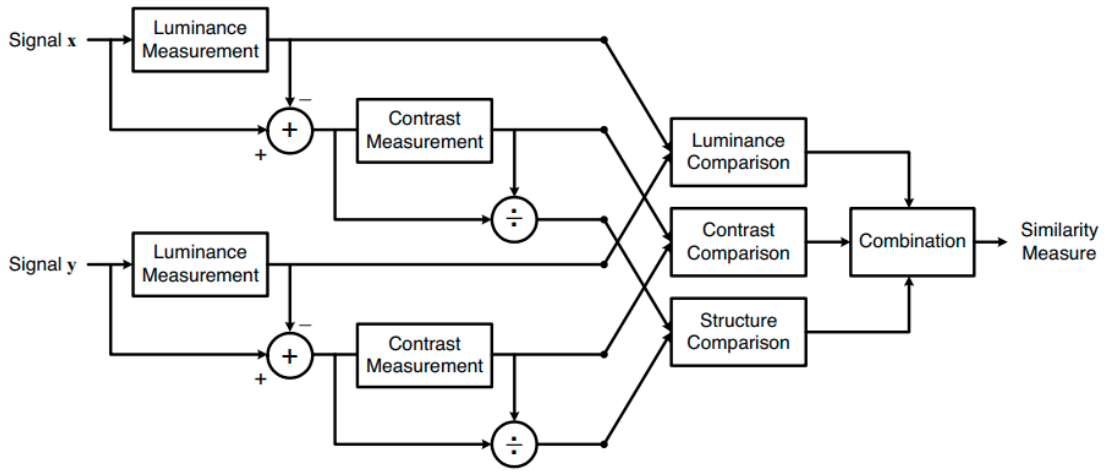


Figure 16. *Diagram of the SSIM measurement system[38]*

Overall similarity measurement is expressed as:

$$S(x, y) = f(l(x, y), c(x, y), s(x, y)) \tag{5.12}$$

Luminance of each signal (assuming they are discrete) is estimated as the mean intensity:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{5.13}$$

And the standard deviation is used for contrast estimation. An unbiased form of the standard deviation is given by formula:

$$\sigma_x = (\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2)^{\frac{1}{2}} \tag{5.14}$$

Luminance comparison $l(x, y)$ is a function of comparison μ_x and μ_y, and contrast comparison $c(x, y)$ is a function of comparison σ_x and σ_y. Structure comparison $s(x, y)$ is conducted on normalized signals $(x - \mu_x)/\sigma_x$ and $(y - \mu_y)/\sigma_y$.

Luminance comparison is defined as:

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{5.15}$$

where C_1 is used to avoid instabilities in case $\mu_x^2 + \mu_y^2$ is close to zero and $C_1 = (K_1 L)^2$. L denotes the dynamic range of pixel values (for 8 bit images, $0 \leq L \leq 255$) and K_1 is a small constant.

Contrast comparison:

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{5.16}$$

where $C_2 = (K_2 L)^2$. Similarly K_2 is a small constant to avoid instabilities when $\sigma_x^2 + \sigma_y^2$ is close to zero.

Finally structure comparison is expressed as:

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3} \tag{5.17}$$

where $C_3 = (K_3 L)^2$, and

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)(y_i - \mu_y) \tag{5.18}$$

To combine $l(x,y), c(x,y)$ and $s(x,y)$, final equation of SSIM is as follows:

$$SSIM(x,y) = \frac{(2\mu_x \mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{5.19}$$

In conclusion, image quality assessment is an important task in the field of image and video processing, and it is particularly relevant in the context of autoencoder-based image restoration. Image quality assessment metrics such as SSIM, RMSE, PSNR, and MRE can be used to evaluate the effectiveness of this reconstruction, as an answer for RQ4. These metrics allow to quantitatively compare the performance of different autoencoder models and to determine which model is most effective at restoring the original image. In general, image quality assessment metrics are useful tools for evaluating the visual fidelity of an image and for comparing the performance of different image processing and machine learning algorithms such as autoencoders.

5.2 Experiments on Traffic Sign Damage Detection

5.2.1 Region of Interest Analysis

GTSRB dataset is used to train and test the autoencoder network. GTSRB has 43 different traffic signs as it is discussed earlier with different shapes and textures. Firstly, this study is focused on one class to observe the network model by visual and image quality assessment metrics.

No entry sign is chosen to work with due to the ease of finding the damaged signs online and the number of data in the training set. No entry sign class id is 17 in GTSRB and has 1110 images in train set.

To detect the anomalies in the image, an artificial image is generated using autoencoder network on no-entry traffic sign. Original image is resized with 48x48 WxH with 3 RGB channels to feed the autoencoder and output (artificial) image is also 48x48x3 which gives the error metric as a sum of RGB channels. Figure 17 illustrates the original (Figure 17a) and artificial (Figure 17b) image.

(a) Original image

(b) Generated (artificial) image

Figure 17. Autoencoder image generation

Artificial image has the following differences that can be recognized by visual inspection:

- Artificial image is more blurry
- Background colors of artificial image looks faded
- Shape of the traffic sign is smoother in artificial image (outer border looks more ideal circle)
- Artificial image has less contrast where the color changes
- Yellow part under the traffic sign (red circle) of the original image is not seen in the same location on artificial image
- Some features are lost at background in artificial image

As it is seen from the visual comparison, there are textures and pixel differences outside of the traffic sign, which contains the irrelevant background pixels. To avoid that problem, region of interest (ROI) must be cropped out from the original image. The original size of the image is 35x36 which is shown in Figure 18a and the ROI coordinates are specified in train.csv file provided with GTSRB. The ROI coordinates are 6,6,30,31 Xmin, Ymin, Xmax,Ymax respectively. Cropped image is shown in Figure 18b.

(a) Full scale image (b) Cropped ROI

Figure 18. Region of interest

Cropping the images definitely removes the irrelevant background information from the data. However generating an artificial cropped test image from the un-cropped train set cause a new problem. In artificial image, the size of traffic sign is reduced and the background noise is added as it is shown in Figure 19.

(a) Cropped ROI (b) Generated Image from ROI

Figure 19. Artificial Image from Cropped ROI

To improve the accuracy of generated image, all images from the training set are cropped due to the ROI x,y coordinates. Another model is trained with cropped train set and the same image looks like as the Figure 20.

(a) Cropped ROI (b) Generated Image from ROI

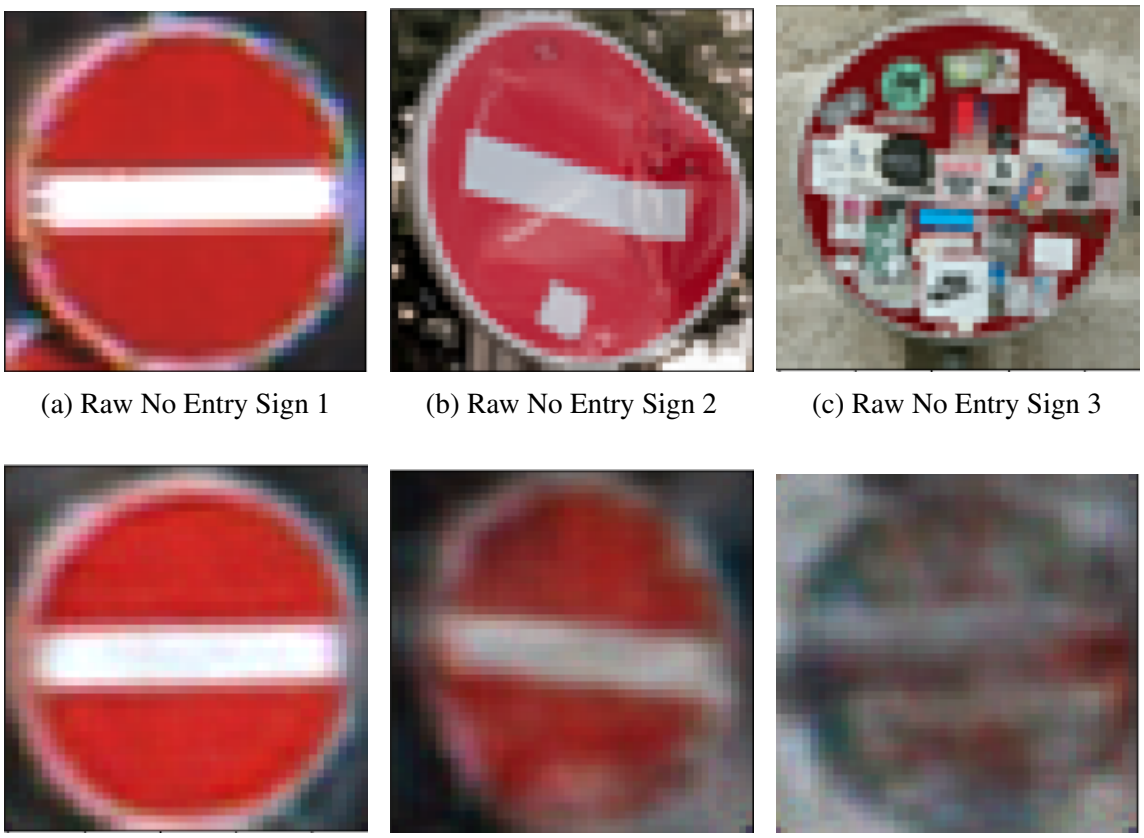
Figure 20. Artificial Image from Cropped ROI Using Cropped Train Set

In this set of experiments, each class is trained separately thus the output models are saved according to their classes in tensorflow *.h5 extension. Each saved model file size is 134.9 MB. When working with traffic sign class, the model file that correspond to the class of the sign must be loaded to generate artificial image and measure the image quality.

5.2.2 Image Quality Assessment on Artificial Images

In this thesis, four techniques are used to measure the image quality between the reference and artificial images. These techniques are SSIM, PSNR, RMSE and MRE. SSIM and RMSE vary between 0 and 1 however MRE has a minimum value of 0 and PSNR has 24.065 but they do not have upper boundary. In SSIM, RMSE and MRE, lower values indicates the better quality in other words it will indicate less damage inside the borders of this thesis. On the other hand, higher PSNR values indicate less damage unlike other techniques. Purpose of this study is to compare the accuracy of error metrics.

In Figure 21, there are 3 raw no entry sign images and their generated versions by the result of autoencoder network are presented. Image 21a is taken from the test set of GTSRB, which looks visually good. Second image (Figure 21b) is bent and has a white square at bottom part however it is definitely damaged. Lastly, the third image (Figure 21c) has multitude of stickers on it makes it almost impossible to be recognized due to loss of features.



(a) Raw No Entry Sign 1 (b) Raw No Entry Sign 2 (c) Raw No Entry Sign 3

(d) Generated No Entry Sign 1 (e) Generated No Entry Sign 2 (f) Generated No Entry Sign 3

Figure 21. Visual Image Quality Analysis on No Entry Sign

After the visual inspection it is needed to proven with error metrics in order to automate damage detection process. Image quality assessment metrics are applied to measure the error between raw and generated images and the results are presented in table 10:

Table 10. *Image quality assessment metrics on no entry sign*

Images/IQA Metrics	**SSIM**	**PSNR**	**RMSE**	**MRE**
No Entry Sign 1	0.158	68.791	0.093	0.228
No Entry Sign 2	0.671	61.583	0.213	0.729
No Entry Sign 3	0.934	60.37	0.244	1.222

Table 10 is comparing the performance of three different images labeled "No Entry Sign" using four different image quality assessment (IQA) metrics. The SSIM values range from 0.158 to 0.934, with lower values indicating better image quality. The PSNR values range from 60.37 to 68.791, with higher values indicating better image quality. The RMSE values range from 0.093 to 0.244, with lower values indicating better image quality. The MRE values range from 0.228 to 1.222, with lower values indicating better image quality.

"No Entry Sign 1" has the highest image quality based on these metrics, followed by "No Entry Sign 2" and then "No Entry Sign 3".

Next, it is going to be investigated speed limit 30 sign. This sign contains various shapes and colors inside. Moreover, it allows to observe the numbers "3" and "0" in various damage conditions. Working with a different traffic sign in a similar manner, it is needed to load the specific model of that sign in order to generate the artificial image of the raw image with learned data from it's training set. First image (Figure 22a) is a healthy sign which has no damage on it. Second image (Figure 22b) has a red sticker under the number 30 and surrounding red circle is interrupted by a white paint at bottom. Moreover there are some letters at top of the red circle. Third image (Figure 22c) is rusted and getting yellow-ish which started to interfere with the number 30. Hence the contrast has lost, the sign is harder to be recognized with respect to the second image.

(a) Raw Limit 30 Sign 1 (b) Raw Limit 30 Sign 2 (c) Raw Limit 30 Sign 3

(d) Generated Limit 30 Sign 1 (e) Generated Limit 30 Sign 2 (f) Generated Limit 30 Sign 3

Figure 22. Visual Image Quality Analysis on Speed Limit 30 Sign

In order to compare the images using numerical methods, previous error metrics values are shown in table 11:

Table 11. *Image quality assessment metrics on speed limit 30 sign*

Images/IQA Metrics	**SSIM**	**PSNR**	**RMSE**	**MRE**
Speed Limit(30) Sign 1	0.082	72.95	0.057	0.172
Speed Limit(30) Sign 2	0.312	67.34	0.11	0.278
Speed Limit(30) Sign 3	0.64	61.352	0.218	0.532

Table 11 compares the performance of three different images labeled "Speed Limit(30) Sign" using image quality assessment (IQA) metrics.

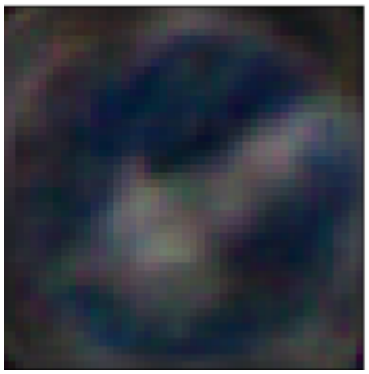
According to table 11, "Sign 1" has the lowest SSIM value, followed by "Sign 2" and then "Sign 3". "Sign 1" also has the highest PSNR value, followed by "Sign 2" and then "Sign 3". In terms of RMSE, "Sign 1" has the lowest value, followed by "Sign 2" and then "Sign 3". Finally, "Sign 1" has the lowest MRE value, followed by "Sign 2" and then "Sign 3". That indicates "Sign 1" has the best quality and "Sign 3" has the worst of all.

With similar approach, several artificial images are generated from a raw no entry sign image. Figure 23 illustrates the generation artificial images from different models which are trained for certain classes. Figure 23a is the source image of no entry traffic sign class which has defect on it. Figure 23b is generated from the model which belongs no entry sign class. Figure 23c, Figure 23d, Figure 23e and Figure 23f are generated from the same source image with using the models speed limit 30, turn left down, priority road and road construction traffic sign classes.

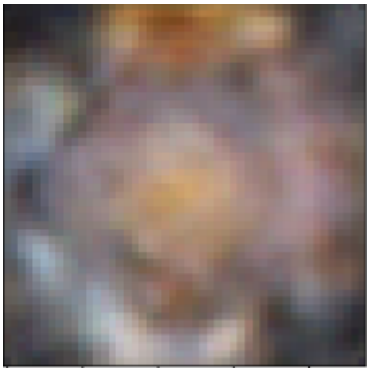
(a) Raw No Entry Sign

(b) Generated No Entry Sign Using No Entry Sign Model

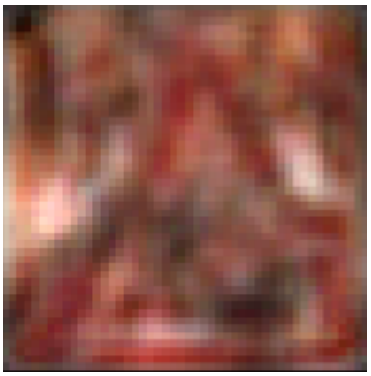
(c) Generated No Entry Sign Using Speed Limit 30 Sign Model



(d) Generated No Entry Sign Using Turn Left Down Sign Model



(e) Generated No Entry Sign Using Priority Road Sign Model



(f) Generated No Entry Sign Using Road Construction Sign Model

Figure 23. Generation of No Entry Sign From Different Models

Results of image quality metrics are shown in table 12:

Table 12. *Image quality assessment metrics on no entry sign with different models*

Images/IQA Metrics	**SSIM**	**PSNR**	**RMSE**	**MRE**
Generated No Entry Model	0.176	70.042	0.08	0.207
Generated Speed Limit 30 Model	0.83	59.466	0.271	0.529
Generated Turn Left Model	0.846	55.277	0.439	0.556
Generated Priority Road Model	0.811	59.225	0.279	0.693
Generated Road Construction Model	0.847	58.967	0.287	0.496

Based on the table 12, we can make the following observations. SSIM values are relatively high for all generated models, with the highest value being 0.847. This suggests that there may be some distortion or degradation in the generated images. PSNR values are also relatively low, with the highest value being 70.042. This indicates that there may be a significant difference between the original and generated images. RMSE values are generally higher for the generated models, with the lowest value being 0.08. This suggests that the generated images may not be as similar to the original images as desired. MRE values are relatively high for all generated models, with the lowest value being 0.207. This indicates that there may be significant errors in the generated images.

5.2.3 Combined Weight Files

After data preparation and validation of the model with artificial image generation techniques and image quality metrics, detection method and measured the performance are combined. A video file from youtube [39] includes the dashboard camera video records that are showing streets and traffic in Germany. Traffic signs are also visible and rich quantity which makes it suitable video source to work with. Video was downloaded and first one minute is taken for test purpose. A simple script counted the frame numbers. There are 1373 frames in the cropped video clip.

All tests were conducted using an Nvidia GTX 1070 GPU. This GPU has 8 GB of memory, 1920 cuda cores, and a frequency of 1506 MHz. The performance of the methods was evaluated using frame rate, which is often measured in frames per second (fps). This refers to the number of individual frames or images displayed in a video per second. A higher frame rate means that more frames are displayed per second, resulting in a smoother and more fluid video. A lower frame rate, on the other hand, can result in a choppy video.

Hence there are 43 traffic sign classes and each class has it's own autoencoder model file, the unique model file need to be uploaded when YOLO algorithm detect a new traffic sign class. Due to large model file sizes (134.9 MB), program ran too slow to process all frames. It took 535 seconds to complete 1373 frames, 2.57 fps, which is far slower than what is required for.

A new solution was needed to make the pipeline faster. That is why all images in training set (which is a combination of all 43 classes) are trained with the same network architecture. The weights are saved into a file. The size of the weight file is 134.9 MB, which is the same size with separate classes weight files. Hence there will be one weight file for each and every traffic sign classes, only one time loading will be required. Accuracy and visuals are presented in the Figure 24 and Figure 25.

(a) Raw No Entry Sign 1 (b) Raw No Entry Sign 2 (c) Raw No Entry Sign 3

(d) Generated No Entry Sign 1 (e) Generated No Entry Sign 2 (f) Generated No Entry Sign 3

Figure 24. Visual Image Quality Analysis on No Entry Sign Using Combined Weight File

Figure 24f is generated different than no entry sign. Due to the new weight file includes all information from each and every classes, too much damaged traffic signs can be generated with different classes. However the rest of all generated in correct classes and all the image quality assessment metrics are presented in table 13:

Table 13. *Image quality assessment metrics on no entry sign with combined model*

Images/IQA Metrics	SSIM	PSNR	RMSE	MRE
No Entry Sign 1	0.103	72.311	0.062	0.145
No Entry Sign 2	0.490	64.495	0.144	0.626
No Entry Sign 3	0.727	63.249	0.175	1.117

Table 13 shows the results of image quality assessment on three different images using the combined model. The SSIM values are relatively low for the No Entry Sign 1 and No Entry Sign 2. The highest value is 0.727, which indicates that there are distortion or degradation in No Entry Sign 3. The PSNR values are also generally lower for the three images, with the lowest value being 63.249, suggesting that there is a significant difference between the original and generated images. The RMSE values are relatively low, with the highest value being 0.175, indicating that the images are generally similar to the original images. However, the MRE values are relatively high, with the lowest value being 1.117, suggesting that there may be significant errors in the same images.

(a) Raw Limit 30 Sign 1 (b) Raw Limit 30 Sign 2 (c) Raw Limit 30 Sign 3

(d) Generated Limit 30 Sign 1 (e) Generated Limit 30 Sign 2 (f) Generated Limit 30 Sign 3

Figure 25. Visual Image Quality Analysis on Speed Limit 30 Sign Using Combined Weight File

Similarly Figure 25f is not generated with correct class information. Similarly, all the image quality assessment metrics of speed limit 30 are presented in table 14:

Table 14. *Image quality assessment metrics on speed limit 30 sign with combined model*

Images/IQA Metrics	**SSIM**	**PSNR**	**RMSE**	**MRE**
Speed Limit(30) Sign 1	0.048	76.247	0.039	0.117
Speed Limit(30) Sign 2	0.259	68.60	0.095	0.209
Speed Limit(30) Sign 3	0.521	64.796	0.147	0.479

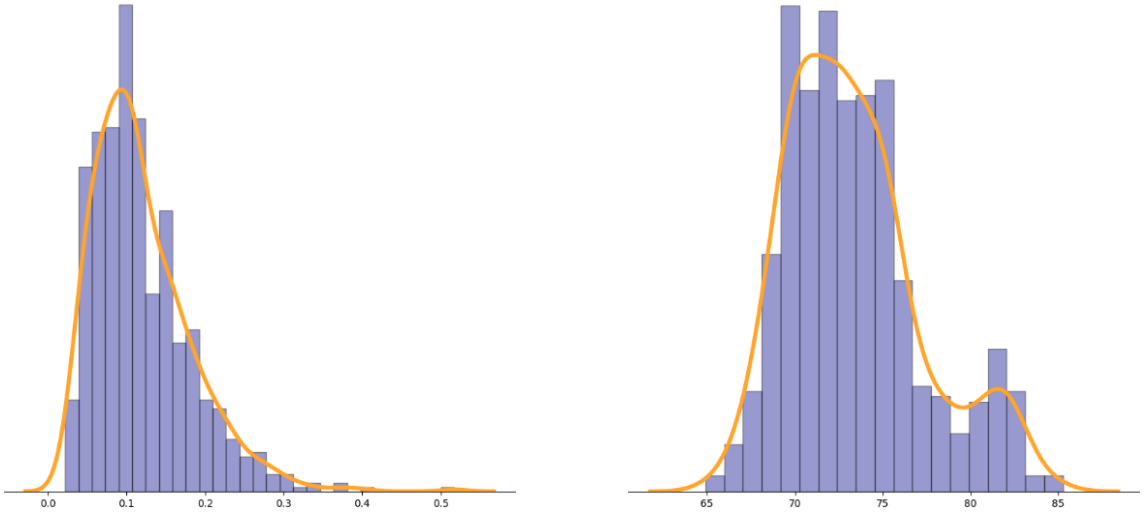
Based on SSIM, the first image of the speed limit sign has the highest perceived quality, with an SSIM value of 0.048. The second image has a slightly lower perceived quality, with an SSIM value of 0.259, and the third image has the lowest perceived quality, with an SSIM value of 0.521. And based on RMSE, the first image has the highest perceived quality, with an RMSE value of 0.039. The second image has a slightly lower perceived quality, with an RMSE value of 0.095, and the third image has the lowest perceived quality, with an RMSE value of 0.147. Both metrics indicates that first image has the least damaged and third image has the most damaged ones.

With decent damages on traffic signs, single weight file can be used. By using single weight file, there is only one file needed to load to program on run time. This reduces the processing time and improves the performance significant. With that modification, 1373 frames processing took only 52 seconds, in other words pipeline works at the speed of 26.4 fps.

5.2.4 Further Analysis on GTSRB Test Set

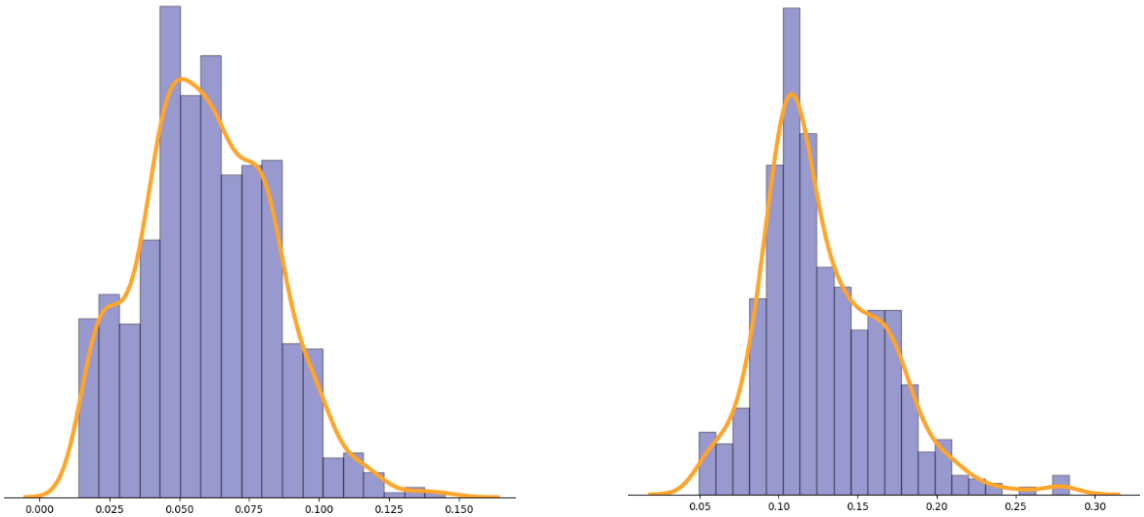
In this section, it is going to be investigated each image of each class by measuring the image quality assessment metrics. The purpose of this study is to analyze the values of each class, observe the error distribution and analyze the threshold candidate points using mean (μ) and standard deviation (σ).

Regarding to GTSRB, there are 720 test images for speed limit 30 and Figure 26 shows the error distribution of four different metrics of speed limit 30 sign. Test images do not contain damaged traffic signs, only contains different illumination, perspective and contrast variations.



(a) SSIM Distance

(b) PSNR Distance

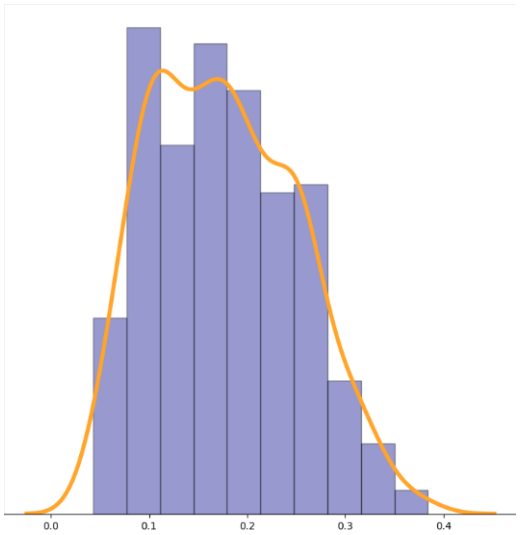


(c) RMSE Distance

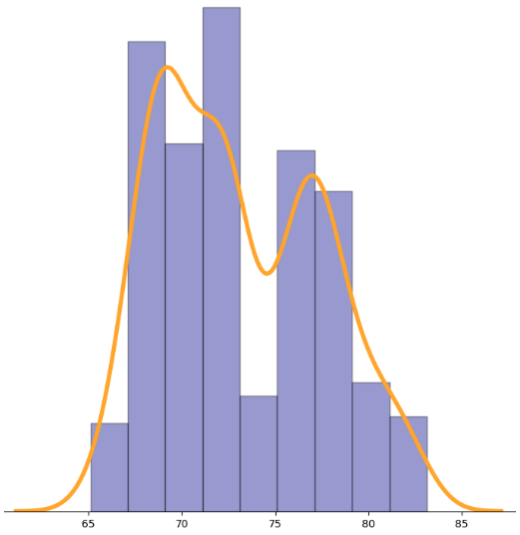
(d) MRE Distance

Figure 26. Distance between raw and artificial images on test set of speed limit 30 sign

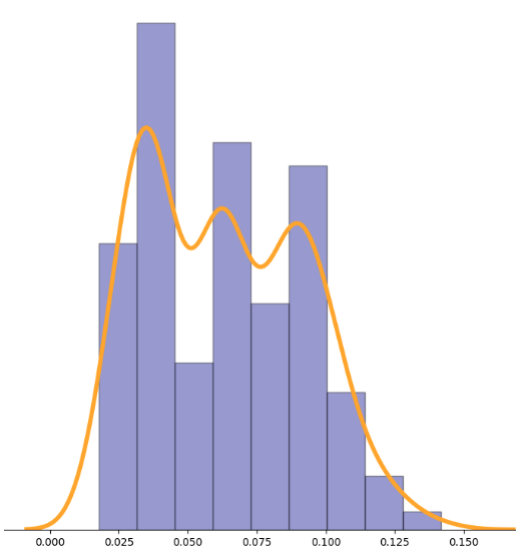
There are different error distribution graphs and numbers with respect to each class. To illustrate the different traffic sign class, no entry sign images are analyzed next. Total number of images in test set are 360 according to GTSRB. Figure 27 shows the error distribution of no entry sign.



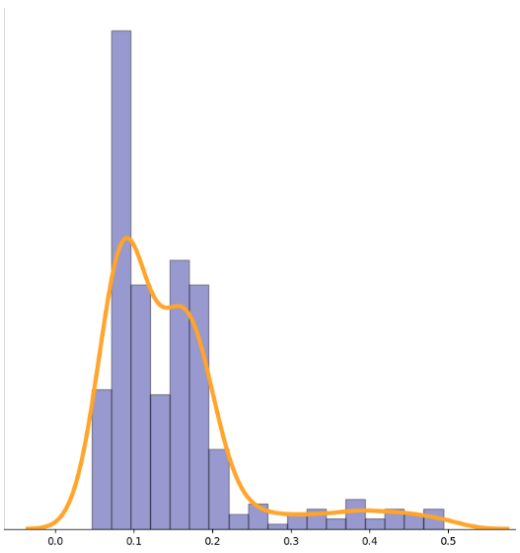
(a) SSIM Distance



(b) PSNR Distance



(c) RMSE Distance



(d) MRE Distance

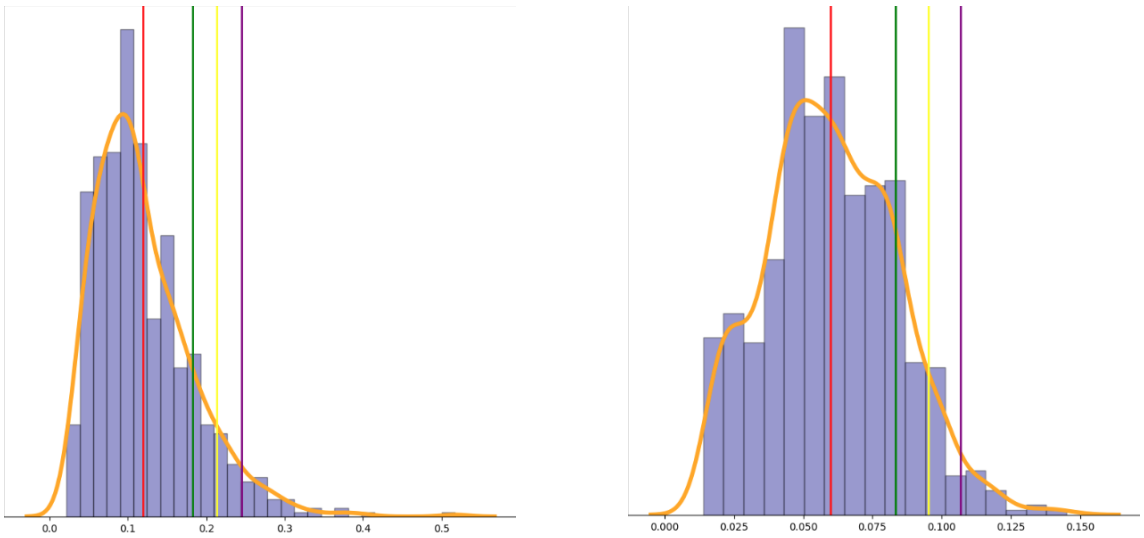
Figure 27. Distance between raw and artificial images on test set of no entry sign

As it can be seen from the distributions, each traffic sign has different set of error ranges with respect to the image quality assessment. Error distribution graphs can be used for finding the threshold to decide if the traffic sign is damaged or not. To analyze those graphs deeper, mean and standard deviation will give meaningful results. Hence the SSIM and RMSE have a certain error range 0-1, error distribution graphs will be based on those metrics.

Mean and standard deviation values are calculated based on each traffic sign test set images and error distribution values. Vertical colored lines in figures indicate the following:

- Red line: Mean
- Green line: Mean + standard deviation
- Yellow line: Mean + (1.5*standard deviation)
- Purple line: Mean + (2*standard deviation)

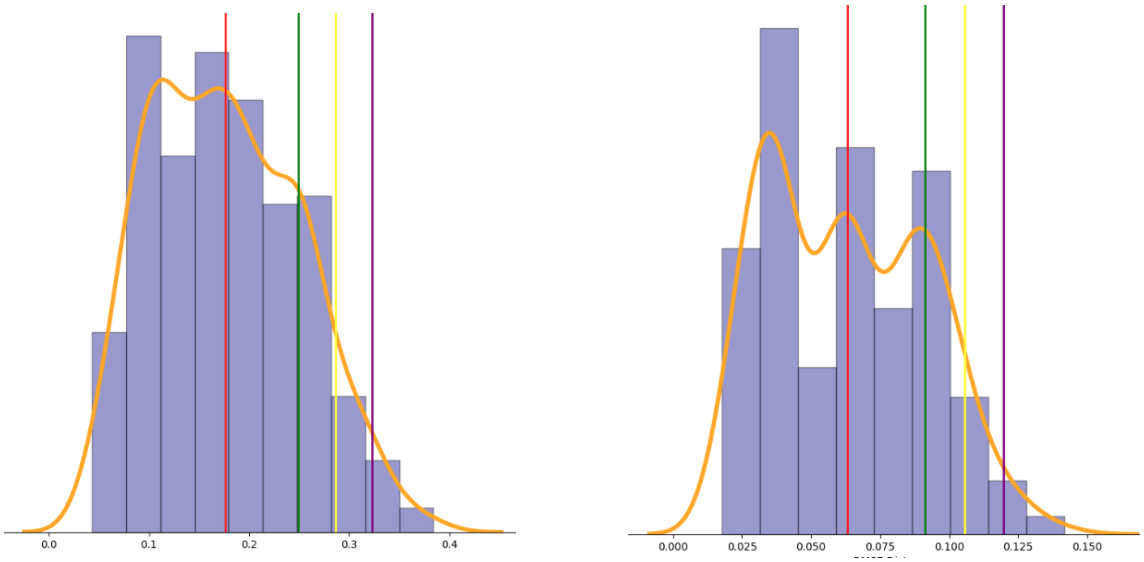
Regarding the distributions on speed limit 30 sign, mean value for SSIM is 0.119 and standard deviation is 0.063. RMSE mean 0.06 and standard deviation is 0.023 as it is shown in Figure 28.



(a) SSIM (b) RMSE

Figure 28. Mean and standard deviation values on speed limit 30 sign test set

Similarly, the distributions for no entry sign, mean value for SSIM is 0.176 and standard deviation is 0.073. RMSE mean 0.063 and standard deviation is 0.028 as it is shown in Figure 29.



(a) SSIM

(b) RMSE

Figure 29. Mean and standard deviation values on no entry sign test set

Table 15 below shows SSIM values of traffic sign mean and standard deviation values of selected traffic sign classes test sets.

Table 15. *SSIM mean and standard deviation on sample traffic classes*

Traffic sign	**#Samples**	μ	$\mu+\sigma$	$\mu+1.5*\sigma$	$\mu+2*\sigma$
Speed Limit(30) Sign	720	0.119	0.182	0.214	0.245
No Overtaking Sign	480	0.12	0.193	0.23	0.267
Stop Sign	270	0.183	0.277	0.325	0.372
No Entry Sign	360	0.176	0.249	0.286	0.323
Road Construction Sign	480	0.226	0.321	0.369	0.417
Keep Left Sign	90	0.187	0.261	0.299	0.336

In table 15 provided, six different traffic sign classes are being analyzed using the SSIM metric. The number of samples of each traffic sign class is also provided.

For each traffic sign class, the mean (μ) and standard deviation (σ) of the SSIM values are given, as well as several values derived from these: $\mu+\sigma$, $\mu+1.5\sigma$, $\mu+2\sigma$. These values represent the mean SSIM value plus an increasing number of standard deviations.

The "Road Construction Sign" class has the highest mean SSIM value, indicating lower perceived quality according to this metric. The "Stop Sign" and "Keep Left Sign" classes have relatively high mean SSIM value, while the "Speed Limit (30) Sign" and "No Overtaking Sign" classes have somewhat lower mean SSIM values. The "No Entry Sign" class has a similar mean SSIM value to the "Speed Limit (30) Sign" class.

Similarly, RMSE values are shown in table 16:

Table 16. *RMSE mean and standard deviation on sample traffic classes*

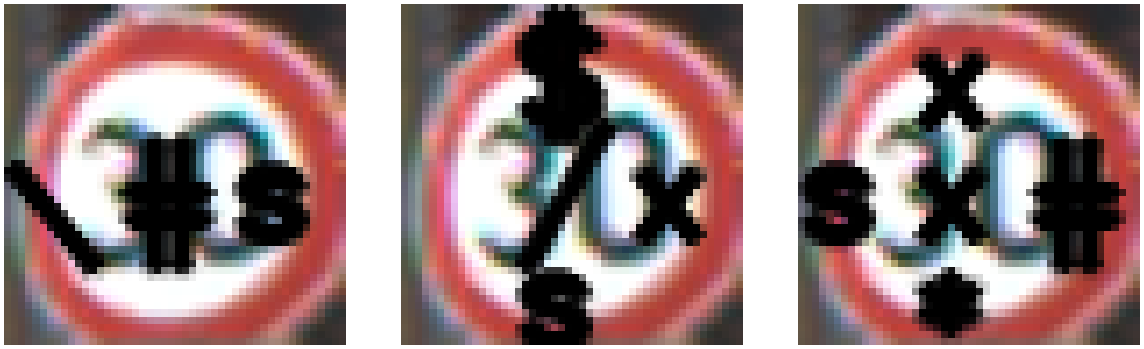
Traffic sign	**#Samples**	μ	$\mu + \sigma$	$\mu + 1.5 * \sigma$	$\mu + 2 * \sigma$
Speed Limit(30) Sign	720	0.06	0.083	0.095	0.107
No Overtaking Sign	480	0.042	0.066	0.077	0.089
Stop Sign	270	0.065	0.087	0.098	0.109
No Entry Sign	360	0.063	0.091	0.106	0.12
Road Construction Sign	480	0.082	0.115	0.131	0.147
Keep Left Sign	90	0.063	0.096	0.113	0.13

RMSE mean values of the same class of traffic signs are much smaller than SSIM metric. However there is a correlation between two metrics. For example No Overtaking Sign has the least value and Road Construction Sign has the biggest value in both metrics.

5.2.5 Artificial Damage on Test Set

There are limited sources for copyright free images of damaged traffic signs on internet. That is why synthetic damage is created on GTSRB test set images for testing the precision of autoencoder model. Computer vision techniques are applied to each image by dividing the image 3x3 grids and drawing random characters on left, right, top, bottom and the center parts. Random characters are printed on random grid parts. Damage level codes are assigned to the set of images with respect to characters they have. For example, if an image includes one character damage, it means damage level is one on that image. Figure 30 shows all levels of synthetic damages.

(a) Damage Level 0 (b) Damage Level 1 (c) Damage Level 2



(d) Damage Level 3 (e) Damage Level 4 (f) Damage Level 5

Figure 30. Synthetic damage levels on speed limit 30 sign

Mean values of the traffic sign classes are increased with respect to the damage levels. Table 17 shows SSIM mean values of damage levels on selected traffic sign classes:

Table 17. *SSIM mean values with damage levels on sample traffic classes*

Traffic sign	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**
Speed Limit(30) Sign	0.237	0.343	0.423	0.489	0.546
No Overtaking Sign	0.217	0.309	0.386	0.445	0.5
Stop Sign	0.287	0.365	0.433	0.494	0.539
No Entry Sign	0.31	0.419	0.502	0.565	0.616
Road Construction Sign	0.309	0.382	0.454	0.511	0.559
Keep Left Sign	0.282	0.366	0.428	0.47	0.521

According to table 17, the SSIM is a quality assessment measure for images, and it ranges from 0 (best quality) to 1 (worst quality). Table 17 suggests that as the damage level increases, the mean SSIM value increases for all traffic sign categories. This indicates that the quality of the images of the traffic signs decreases as the damage level increases. In general, the traffic sign categories with lower SSIM values at each damage level have better image quality compared to those with higher values. For example, at damage level 1, the "No Overtaking Sign" category has a lower mean SSIM value (0.217) compared to the other levels of the same sign.

Table 18 shows RMSE mean values of damage levels on selected traffic sign classes:

Table 18. *RMSE mean values with damage levels on sample traffic classes*

Traffic sign	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**
Speed Limit(30) Sign	0.096	0.12	0.138	0.153	0.165
No Overtaking Sign	0.064	0.079	0.092	0.1	0.109
Stop Sign	0.087	0.101	0.113	0.123	0.131
No Entry Sign	0.102	0.129	0.149	0.164	0.179
Road Construction Sign	0.105	0.122	0.136	0.149	0.159
Keep Left Sign	0.99	0.123	0.141	0.154	0.167

The table 18 shows the mean values RMSE metric for different traffic sign categories at various damage levels. The RMSE is a measure of the difference between two images, and it is typically used to assess the quality of image reconstruction or image comparison algorithms. The lower the RMSE, the better the quality of the images. Table 18 suggests that as the damage level increases, the mean RMSE value increases for all traffic sign categories. This indicates that the difference between the original images and the reconstructed or compared images increases as the damage level increases. In general, the traffic sign categories with lower RMSE values at each damage level have better image quality compared to those with higher values.

Table 17 and table 18 shows the correlation between damage level and error values. Hence each class of traffic sign has different mean error value, there can not be only one threshold value which covers all the traffic signs. That is why threshold values must be specified for each traffic sign class. This can answer RQ5, however further tests could be done as further analysis.

In conclusion, autoencoders can be used for damaged traffic sign detection by training the model on a dataset of undamaged traffic sign images and then using it to reconstruct new images of traffic signs. If the reconstruction error for a new image is significantly higher than the error for the undamaged images, it can be considered a damaged traffic sign. Autoencoders can also be used to identify specific types of damage within a traffic sign image by using class labels as part of the training process. However, it is important to carefully preprocess and curate the training dataset to ensure that the autoencoder accurately learns the features of undamaged traffic sign images and is able to effectively identify damaged images.

6. Conclusion and Discussion

In conclusion, the development of a damaged traffic sign detection system using machine learning techniques, such as YOLO and autoencoders, represents a major achievement in the field of transportation safety. By combining the power of these technologies, this system was able to accurately identify damaged traffic signs in a variety of real-world scenarios, with high levels of accuracy.

To answer RQ1, GTSDB is diverse, large and contains high-quality and reliable annotations, making it an ideal dataset for training, evaluating, and benchmarking object detection models for German traffic signs. Majority of GTSDB traffic signs are also used in Estonia. Additionally, the dataset contains images of traffic signs in real-world context, taken from various sources such as web cameras, dashboard cameras and mobile phones, which makes it more representative of real-world scenarios. Furthermore, GTSDB dataset is widely used and well-established in the research community, it serves as a good reference to compare and evaluate different models and methodologies. Among other datasets, GTSDB stands out for the quality and quantity of its data which makes it a robust benchmark for the traffic sign detection task.

Traffic sign detection is one of the major section within this thesis borders. When it comes to detecting traffic signs based on image data as mentioned in RQ2, one of the most suitable and effective methods is the YOLO algorithm. YOLO is a single-shot object detection algorithm that divides an image into a grid and uses a convolutional neural network (CNN) to predict the class and location of objects within each grid cell. One of the key advantages of YOLO is its speed and real-time processing capability, as it can process images very quickly and is suitable for video or live feed applications. Additionally, YOLO has a simple structure and is easy to implement, compared to other object detection algorithms like R-CNN and its variants. YOLO also has the ability to run in real-time while still maintaining high accuracy, which makes it ideal for traffic sign detection tasks where both speed and accuracy are important. Furthermore, YOLO is robust against occlusions, deformations, and it can detect small objects which makes it more suitable for detecting traffic signs. In general, YOLO is considered one of the most suitable and effective methods to detect traffic signs based on image data, due to its high accuracy, real-time performance, and flexibility.

As an answer of RQ3, autoencoders are neural networks that can be employed to determine if a traffic sign has been damaged by analyzing its image data. They are made up of two parts, an encoder and a decoder. The encoder condenses the input image into a compact, lower-dimensional representation, also known as a bottleneck, which contains the vital characteristics of the input. The decoder then rebuilds the input image using this bottleneck representation. By teaching the autoencoder to recreate the initial image from the input, it learns to identify the most crucial features in the image data.

Measuring the damage of a traffic sign numerically can be achieved by using image quality metrics. One common metric is RMSE which calculates the average difference in pixel values between the original image and the reconstructed image. A lower RMSE value indicates better reconstruction and less damage. Another widely used metric is SSIM, which compares the structural information of the original and reconstructed image, and gives a score between 0 and 1, where 0 represents a perfect match and 1 no similarity. PSNR calculates the ratio of the maximum signal power to the power of the noise in dB, indicating a higher PSNR means less noise and less damage. MRE calculates the mean of the absolute difference between the original image and the reconstructed image. These metrics can be used to provide a numerical assessment of the damage. SSIM and RMSE are used for further damage analysis on traffic signs used in RQ4.

To answer RQ5, deciding whether a traffic sign is damaged or not based on a chosen metric such as SSIM or RMSE, one approach would be to set a threshold value that separates the damaged signs from the undamaged ones. For example, if using SSIM as the metric, a threshold value of 0.6 could be chosen, meaning that any signs with an SSIM value greater than 0.6 would be classified as damaged, and any signs with an SSIM value below 0.6 would be classified as undamaged. Similarly, for RMSE, a threshold value of 0.5 could be chosen, signs with an RMSE value above 0.5 would be classified as damaged, and signs with an RMSE value below 0.5 would be classified as undamaged.

Threshold values are determined experimentally by testing the performance of the model on a test set of traffic signs. Statistical analysis on error distributions are used for each traffic sign class to find a more suitable threshold value that maximizes the performance of the model. Based on image quality assessment error distributions, synthetic damages are created on test set images to test the accuracy and efficiency of the proposed method. However real world test could be done with damaged traffic sign images to measure the performance and accuracy of this approach as a further analysis.

References

[1] *Help Your Community Replace Damaged Traffic Signs*. https://www.tssco.com/news/help-your-community-replace-damaged-traffic-signs/. Accessed: 2022-04-11.

[2] Higinio González-Jorge et al. "Geometric Evaluation of Road Signs using Radiometric Information from Laser Scanning Data". In: *28th International Symposium on Automation and Robotics in Construction. 1007-1012.* (2011).

[3] *Bullet holes on the speed limit road sign.* https://stock.adobe.com/de/images/bullet-holes-on-the-speed-limit-road-sign-damaged-road-sign/181611567. Accessed: 2018-11-16.

[4] Chien-Yao Wang et al. "You Only Learn One Representation: Unified Network for Multiple Tasks". In: *arXiv preprint arXiv:2105.04206* (2021).

[5] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: *arXiv* (2020).

[6] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag of freebies sets new state-of-the-art for real-time object detectors". In: *arXiv:2207.02696* (2022).

[7] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Scaled-YOLOv4: Scaling Cross Stage Partial Network". In: *arXiv* (2021).

[8] Chun Chanjun and Seung-Ki Ryu. "Road Surface Damage Detection Using Fully Convolutional Neural Networks and Semi-Supervised Learning". In: *Sensors 19, no. 24: 5501* (2019).

[9] Husan Vokhidov et al. "Recognition of Damaged Arrow-Road Markings by Visible Light Camera Sensor Based on Convolutional Neural Network". In: *Sensors 2016, 16, 2160. https://doi.org/10.3390/s16122160* (Dec. 2016).

[10] Changbin You et al. "Rapid Traffic Sign Damage Inspection In Natural Scenes Using Mobile Laser Scanning Data". In: *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (July 2007).

[11] Can Ersu. *Damaged Traffic Sign Detection*. https://github.com/canersu/damaged_ts_detection. 2023.

[12] Aleksei Tepljakov et al. "Deep Learning for Detection of Pavement Distress Using Nonideal Photographic Images". In: *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)* (2019).

[13] *Traffic Sign Detection*. https://www.vicos.si/research/traffic-sign-detection/. Accessed: 2008-04-01.

[14] *Applied Deep Learning - Part 3: Autoencoders*. https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798. Accessed: 2017-10-03.

[15] *The German Traffic Sign Detection Benchmark*. https://benchmark.ini.rub.de/gtsdb_dataset.html. Accessed: 2013-08-09.

[16] *The German Traffic Sign Recognition Benchmark*. https://benchmark.ini.rub.de/gtsrb_dataset.html. Accessed: 2013-08-09.

[17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving ? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)* (2012).

[18] Radu Timofte et al. "Traffic Sign Recognition - How far are we from the solution?" In: *International Joint Conference on Neural Networks (IJCNN 2013)* (Aug. 2013).

[19] D. Temel et al. "CURE-TSR: Challenging unreal and real environments for traffic sign recognition". In: *Neural Information Processing Systems (NIPS) Workshop on Machine Learning for Intelligent Transportation Systems (MLITS)* (Dec. 2017).

[20] Pavly Salah Zaki et al. "Traffic Signs Detection and Recognition System using Deep Learning". In: *arXiv preprint arXiv:2003.03256* (2020).

[21] *Classification of Traffic Signs: The European Dataset*. https://github.com/citlag/European-TrafficSings/blob/master/Images/categories_small.png. Accessed: 2019-01-03.

[22] Sebastian Houben et al. "Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark". In: *The 2013 International Joint Conference on Neural Networks (IJCNN)* (2013).

[23] *Traffic Sign Recognition*. https://medium.com/@wolfapple/traffic-sign-recognition-2b0c3835e104. Accessed: 2019-03-27.

[24] Tsung-Yi Lin et al. *Microsoft COCO: Common objects in context*. 2014.

[25] Ross Girshick et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *https://doi.org/10.48550/arXiv.1311.2524* (Nov. 2013).

[26] Ross Girshick. "Fast R-CNN". In: *https://doi.org/10.48550/arXiv.1504.08083* (Sept. 2015).

[27] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *https://doi.org/10.48550/arXiv.1506.01497* (Jan. 2016).

[28] *YOLOv5*. https://github.com/ultralytics/yolov5. Accessed: 2023-01-03.

[29] Dor Bank, Noam Koenigstein, and Raja Giryes. "Autoencoders". In: *https://doi.org/10.48550/arXiv.200* (2021).

[30] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *DOI: 10.1126/science.1127647 pp. 504-507* (July 2006).

[31] Xie Bangquan and Weng Xiao Xiong. "Real-Time Embedded Traffic Sign Recognition Using Efficient Convolutional Neural Network". In: *Digital Object Identifier 10.1109/ACCESS.2019.2912311* (Mar. 2019).

[32] *From Autoencoder to Beta-VAE*. https://lilianweng.github.io/posts/2018-08-12-vae/. Accessed: 2018-08-12.

[33] Manpreet Singh Minhas and John Zelek. "Semi-supervised Anomaly Detection using AutoEncoders". In: *Journal of Computational Vision and Imaging Systems* (2020).

[34] Diederik P. Kingma and Jimmy Lei Ba. "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION". In: *arXiv:1412.6980v9* (Jan. 2015).

[35] *Hyperparameter Tuning in Python: a Complete Guide*. https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide. Accessed: 2022-12-13.

[36] *Hyper-parameter Tuning Techniques in Deep Learning*. https://towardsdatascience.com/hyper-parameter-tuning-techniques-in-deep-learning-4dad592c63c8. Accessed: 2019-03-16.

[37] Marius Pedersen and Jon Yngve Hardeberg. "Full-Reference Image Quality Metrics: Classification and Evaluation". In: *Foundations and Trends in Computer Graphics and Vision Vol. 7, No. 1* (2011).

[38] Zhou Wang et al. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 13, NO. 4* (Apr. 2004).

[39] *Ultimate Dash Cam Fails Compilation Germany*. https://www.youtube.com/watch?v=w2Lf-Ze6CsA. Accessed: 2015-08-01.

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

I Can ERSÜ

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "AUTOMATIC VISUAL TRAFFIC SIGN DAMAGE DETECTION USING DEEP LEARNING ALGORITHMS", supervised by Uljana Reinsalu and Karl Janson
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

11.01.2023

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – Autoencoder Model

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 46, 46, 8)	224
conv2d_1 (Conv2D)	(None, 44, 44, 16)	1168
max_pooling2d (MaxPooling2D)	(None, 22, 22, 16)	0
conv2d_2 (Conv2D)	(None, 20, 20, 32)	4640
conv2d_3 (Conv2D)	(None, 18, 18, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 9, 9, 64)	0
flatten (Flatten)	(None, 5184)	0
dense (Dense)	(None, 1024)	5309440
dense_1 (Dense)	(None, 256)	262400
dense_2 (Dense)	(None, 1024)	263168
dense_3 (Dense)	(None, 5184)	5313600
reshape (Reshape)	(None, 9, 9, 64)	0
up_sampling2d (UpSampling2D)	(None, 18, 18, 64)	0
conv2d_transpose (Conv2DTranspose)	(None, 20, 20, 64)	36928
conv2d_transpose_1 (Conv2DTranspose)	(None, 22, 22, 32)	18464
up_sampling2d_1 (UpSampling2D)	(None, 44, 44, 32)	0
conv2d_transpose_2 (Conv2DTranspose)	(None, 46, 46, 16)	4624
conv2d_transpose_3 (Conv2DTranspose)	(None, 48, 48, 3)	435

Figure 31. *Autoencoder model architecture*

Appendix 3 – Crop ROI Function

```
import shutil
from PIL import Image

def cropROI(base_path='../gtsrb/', dst='cropped_train', ds_type='Train'):
    if ds_type == "Train":
        # Copy all the images and folders of to the dst path
        shutil.copytree(base_path+'Train', base_path+dst)
    with open(base_path+ds_type+".csv") as f:
        for line in f.readlines():
            path = line.strip().split(',')[7]
            # Skip the first line which consists of header names
            if path == "Path":
                pass
            else:
                x1 = int(line.strip().split(',')[2])
                y1 = int(line.strip().split(',')[3])
                x2 = int(line.strip().split(',')[4])
                y2 = int(line.strip().split(',')[5])
                class_id = line.strip().split(',')[6]
                fname = path.split('/')[-1]


                image = Image.open(base_path+path)
                cropped = image.crop((x1,y1,x2,y2))
                cropped.save(base_path+dst+"/"+class_id+"/"+fname)
```

Figure 32. *Crop ROI function*

Appendix 4 – Synthetic Dataset Creation

```python
def gen_dmg_img(filename, dmg_level):
    orig_img = Image.open(filename)
    orig_img = orig_img.resize((48,48))
    orig_img_arr = np.array(orig_img)
    text = ['x', 's', 'H', 'z', '/', '*', '$', '#', ':', '|', '\\']
    coords = [(16,47), (0,32),(16,32),(32,32),(16,16)]
    # coords = [(0,47),(16,47),(32,47), (0,32),(16,32),(32,32), (0,16),(16,16),(32,16)]
    fontScale = 0.6
    color = (0, 0, 0)
    thickness = 2
    orig_img = orig_img_arr[:, :, ::-1].copy()

    # num_noise = random.randint(1,9)
    rand_coords = random.sample(coords, dmg_level)

    for i in rand_coords:
        char = random.choice(text)
        orig_img = cv2.putText(orig_img, char, i, 0, fontScale, color, thickness, cv2.LINE_AA)
        printed_chars.append(char)


    return orig_img
```

```python
def make_syn_testset(level):
    for i in range(43):
        healthy_path = "/home/can/thesis/gtsrb/damaged_signs/"+str(i)+"/healthy/"
        syn_path = "/home/can/thesis/gtsrb/damaged_signs/"+str(i)+"/syn"+str(level) + "/"
        ts_db = os.path.join(healthy_path)
        for img in os.listdir(ts_db):
            fname = os.path.join(ts_db, img)
            syn_img = gen_dmg_img(os.path.join(ts_db, img), level)
            fname = fname.split('/')[-1]
            cv2.imwrite(syn_path+fname, syn_img)
```

Figure 33. *Function for creating synthetic dataset*