CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Creating a reference dataset for neural network validation and evaluation

Determining key characteristics in vehicle images appropriate for binary classifier validation and evaluation

Master's thesis in Computer science and engineering

Tobias Foughman Lind
Ke Jia

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

Creating a reference dataset for neural network validation and evaluation

Determining key characteristics in vehicle images appropriate for binary classifier validation and evaluation

Tobias Foughman Lind
Ke Jia

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Creating a reference dataset for neural network validation and evaluation
Determining key characteristics in vehicle images appropriate for binary classifier validation and evaluation
Tobias Foughman Lind
Ke Jia

Supervisor: Miroslaw Staron, Department of Computer Science and Engineering
Advisor: Mikael Engbom, Spark Vision
Examiner: Regina Hebig, Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Creating a reference dataset for neural network validation and evaluation
Determining key characteristics in vehicle images appropriate for binary classifier validation and evaluation

Tobias Foughman Lind
Ke Jia
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In the automotive industry, customizing a car has recently been made possible thanks to online services, where various car parts can be personalized independently. This process is done by a back-end service which composes images of individual parts into a fully configured vehicle. However, there are instances where an image is not perfectly rendered, which may result in a defective image being shown directly to the client. Using neural networks to perform defect detection is a way of mitigating this problem. Previous research regarding defect detection using neural networks, evaluating neural networks, and constructing a test harness for machine learning have been widely studied. However, there exist a lack of research that bridge these research topics. The purpose of this study is to investigate the procedures needed to construct a test harness for defect detection, by characterizing, designing and evaluating a reference dataset. Using the design science research methodology, we created and validated datasets containing images with different defects. These were then combined into a reference dataset, and included in a test harness. The procedures required for the creation of this reference dataset can be used for the recreation of a similar dataset for other domains. Then, the test harness was evaluated using three binary classifiers with known performance. Test Case Prioritization was the testing methodology used in the test harness, to establish the correctness of the networks. The testing results verified that the test harness is able to distinguish between adequate and unsuitable neural network-based binary classifiers. However, as only a limited amount of defects were included in the test harness, the generalizability could be threatened. Furthermore, due to the confidentiality of the data used in the thesis, replication of the study by other researchers may be difficult.

Keywords: Software Engineering, Computer science, Machine Learning, Image Classification, Thesis.

Acknowledgements

We would like to thank Spark Vision, specifically Mikael Engbom and Charlie Höög and Erik Nilsson who's helped us with various inquiries regarding all things surrounding this project. We would like to thank Regina Hebig for being our examiner, and providing valuable feedback. Finally, we would also like to thank our supervisor, Miroslaw Staron, for helping us all throughout the project.

Contents

List of Figures

List of Tables

1
Introduction

A trend in the automotive industry during the last decades has been the increasing use of mass customization, or mass personalization for consumers when ordering cars online [15]. In this thesis we investigate an implementation of one such service, provided by the associated company Spark Vision. When presenting consumers with a customizable car in their online store platform, a back-end service is used to compose images of individual parts into a fully configured vehicle. This is a highly optimized step, being done in fractions of a second, but include the possibility of an image being defective for a variety of reasons. If an individual part is incorrectly fetched from a server or simply is incorrectly rendered in the final image, there is currently no active step in the backend process to verify that a customer is not presented with this kind of defective image. Unfortunately the possible permutations for a customizable vehicle, even with a low number of changeable parts, is of an enormous size.

Problems like these, where the sheer amount of data is an insurmountable problem, lend themselves well to being solved by using machine learning (ML) algorithms, such as *neural networks* (NNs). They require only a subset of the of the full population of data and are able to make predictions about out-of-sample data [16]. There are, however, a vast amount of ways to construct a NN [23, 18], and an even larger number of ways to tweak these algorithms [37, 7]. Though the problem does not necessarily lie in the actual selection of an algorithm, yet in how to confidently determine that an algorithm is suitable for the data at hand, which can be different for each problem. This, in combination with the issue of the enormous population size of the car images, presents an opportunity for creating a smaller reference dataset, representative of defective images. Representative in this case is referring to the aspects of a defective image which proves difficult to classify for NNs, and defective refers to an image of a car that is missing a specific part. The research conducted in this thesis is therefore focusing on designing and constructing a test harness, which allows for the comparison of NNs, using the reference dataset explained previously.

The process of creating this smaller reference dataset is done through an investigative study of samples of images created for the purpose of this thesis. By applying a NN to these images, and inspecting the results of that application, it should be possible to find patterns in the configurations of the cars that represent the cases that are difficult for the algorithm to classify. The patterns in this case are specific subsets of the configurations, which contain specific parts of the car, and is referred to as *characteristics* throughout this thesis. The procedures required for the creation

of this reference dataset can be used for the recreation of a similar dataset for other domains.

The initial chapters of this thesis present various background knowledge, both practical and theoretical, to aid in the comprehension of the rest of the text. The purpose of the thesis is also described, with specific research questions to be answered. Following that is the methodology chapter, describing the procedures that were conducted in search for answers to our research questions. The result chapter is next, outlining the outcome of methodology chapter. After that is a discussion about said results, arguing for the meaning and impact it may have. Finally, is the conclusion of the thesis, describing what the answers to the research questions and outlining potential future work for the research field.

1.1 Background

The automotive industry has been known for having a robust network of car dealers which act as a middle-man between the manufacturers and consumers. This is an aspect which has seen a change in recent years with the introduction of the ability for each consumer themselves to customize their potential car before purchase [15].

ML in general is a concept known to scientists from as early as 1959 [34]. The core idea is to design a system which can emulate, to varying degrees, the human mind's ability to learn. It is a very wide area of research, containing numerous implementation skews, such as artifical neural networks, linear regression algorithms or decision trees [32]. The popularity of NNs for image-recognition has fluctuated substantially throughout its history. The complexity of these systems required a considerable amount of manual work to be done, in the form of extracting features from images. This meant that during the early years more work was required for analysing the data than actually applying the algorithm to it, forcing the usage of the approach to be limited to smaller problems. Because of this they were widely considered unfeasible for general usage until the 1990's, when methods to automate this task (known as feature extraction) were developed, allowing for the NNs to be applied to more complex problems [21, 20]. At this point however, the computational power of computers was still not high enough for the general usage of NNs for more complex problems [33]. Overtime, the performance of computer hardware has enhanced dramatically. Therefore, NNs have also been developed in a blowout style. Modern NNs can be implemented in various scenarios, for instance, visual object recognition [19]. The area of image classification has also been able to break its conventional bottleneck of not extracting enough features from the images [10].

The process of classification implementation is achieved through a probability perspective. The outputs of the algorithm are recognized as the probabilities that a case belongs to its corresponding class [11]. Classifiers can be divided into binary classifiers and categorical classifiers. In a binary classifier, cases only belong to two classes, the goal of the classifier is to correctly predict the labels of the cases and cluster them into the correct class. Whereas in a categorical classifier, cases belong

to various classes. The classifier should predict the correct label and classifies them into their corresponding classes [5].

Software testing has been determined to be a suitable method for revealing problems and assist the trustworthiness of ML systems. Whereas Machine Learning Testing (MLT) refers to adopting any activity designed to reveal any differences between expected and existing behaviours of the ML system. MLT can be categorized into three parts, Testing Workflow, Testing Components, and Testing Properties. Testing Workflow is about how to conduct MLT with different testing activities. Testing Components is referred as find defects in every component including the data, the learning programs and the framework. Last but not least, Testing Properties is about finding conditions MLT needs to guarantee for a trained model [48].

1.2 Purpose

The purpose of this study is to investigate the procedures needed to construct a test harness for defect detection, by characterizing, designing and evaluating a reference dataset. This is done by using standardized metrics for assessing the validity of the dataset and evaluating NN-based algorithms when they are applied to the dataset of composed car images. The test harness consists of various features that are difficult to detect by the algorithms, for instance, a black grill missing on a black car body. Furthermore, the selected algorithms are studied and assessed from the accuracy aspect. To determine if the selected algorithm is qualified for detecting certain images, they are provided with large number of comparable images as inputs. The result of this study strives to be beneficial for both researchers and practitioners.

1.3 Research Questions

The research questions are as follows:

RQ1: *What characteristics in defective car images can be used in a test harness to assess the quality of a NN?*
An image consists of many aspects (colors, resolution, contrast, set of components, etc) that all affect how a NN performs. However, certain combinations of these aspects, known in this thesis as characteristics, may affect a NNs performance more than others when classifying defective car images. Thus, these characteristics are more interesting in the context of assessing the quality of a NN, as they mark the most difficult classification cases. An example of such a characteristic is the combination of a color and a specific component (e.g white car with mudflaps), in correlation with a defective wheel.

RQ2: *How can images with the characteristics identified in RQ1 be combined into a valid test harness for NNs?*
There are existing patterns in the data which allow for the possibility of a reduction of test cases. By investigating images with the characteristics mentioned in

the previous question, it may be possible to select representative images of these characteristics instead of simply using them all. Certain combinations of components, such as a black window on a black car, may prove difficult to classify, and thus should be included in an evaluating test harness. Specifically, we will detail the procedures required to design and create a test harness for defect detection in composed car images. This will allow for similar datasets and test harnesses to be created for images from other domains.

RQ3: *Can we identify adequate NN-based binary classifiers using our test harness? What are the relevant metrics when comparing such NNs applied to the aforementioned dataset?*
By analyzing the test images to create a reference dataset, we investigate the extent to which NNs can be evaluated. This is done through the use of NNs with known performance, which allows for a conclusion to be drawn about the correctness of the test harness.

1.4 Delimitations

This project is a Master's Thesis, done in cooperation with a company and Chalmers University of Technology, and is thus time-restricted. Due to this fact various delimitations had to be made before, and throughout, the project. Those are presented and explained in this section.

To focus on the investigation of the characteristics mentioned in this thesis, we decide to only use one car model as a reference for the creation of the test harness. This allows for a more thorough analysis of the characteristics. Additionally, due to the limited time-frame of this project, we are required to reduce the scope to only investigate the defective images. To further reduce the the scope, we have to select specific car parts to remove in the process of image creation. These are the wheels. side-mirrors, front grill and bumper.

The general domain of this project, being image classification, lends itself to a wide variety of solutions. we decide that this project uses ML as the validation technique for the dataset, and more specifically only focus on *convolutional neural networks* (CNNs). The reason behind this is that those types of NN have been widely recognized as valid, and generally preferable, for image classification. This means that solutions using on other NNs or simply other training techniques than supervised learning, such as Recurrent Neural Network or AutoEncoders, were not explored [9].

The proposed test harness, as well as the development of the most appropriate NN-based algorithm is a shared work with the research collaborator.

2
Related Work

In this chapter, studies that are related to our research are analyzed and discussed below.

2.1 Evaluation of Neural Networks

A comprehensive performance evaluation of various NN-based algorithms was conducted in 2017. The study was carried out by providing both RGB-D images and videos to various algorithms. According to the author, using CNNs always outperform other selected methods (DBNs, SDAE and LSTM) [36]. As the evaluated images are of the RGB-D format, they are dimensionally similar to the RGB-A images used in this thesis. Using this report, with the similarities of the data, gives us further confidence in the use of CNNs as the type of network for this data.

Kwon et al. conducted an empirical study into the performance of CNNs of three different depths. These networks are applied to three different public datasets and is evaluated based on their F-measure. One of these datasets are unbalanced in regards to the data and their classes. In this case the metric of Matthew's Correlation Coefficient (MCC) is used [18]. The F-measure is something which is discussed in this thesis in relation to this unbalanced data, but to use this metric to evaluate NNs is closely related to **RQ1** in this thesis.

A comprehensive study of deep CNN was conducted by Liu et al. in 2017. In their study, they stated that conventional deep CNNs do not always succeed in images that have high resolutions. Moreover, they also proposed an improved version of the conventional CNN with modified hyperparameters. However, according to the author, the efficiency aspect of the proposed solution remains unsolved [23]. This is related in the sense that it shows that we still require work in finding the best configurations of NNs. In extension, this should mean that it is important to be able to distinguish between "good" and "bad" networks for a particular problem, which this thesis is intended to investigate.

Detecting transparent objects has become a hot research topic. Khaing and Masayuki have conducted a study recognizing transparent objects in images by applying the Single Shot Multibox Detector (SSD) method, which is more computational efficient in many cases. The result of their study was promising however the author also stated that the model could not detect non-transparent objects with the same

shape. The solution they proposed for the problem was to provide the algorithm with non-transparent objects during training [14]. This SSD method could potentially be adopted in this thesis to detect various characteristics more efficiently.

2.2 Defect detection using Neural Networks

CNNs have seen extensive application for the purpose of quality assurance/defect detection. One example of this comes from researchers in Switzerland who investigated the ability to reduce the workload of building inspectors, as well as tenants, of large building complexes. They proposed a form of image classification algorithm which would analyse images of buildings "... [with the purpose of] automated detection and localisation of key building defects". This evaluation study compared CNN models, following a design called VGG-16, of various depths when applied to this dataset of diverse images. This resulted in a model with an average test accuracy of 87.5%, prompting the authors to characterise it as "... [a network with] high reliability and robustness in classifying and localising defects" [29]. This study has helped us to acquire more knowledge of suitable CNNs for detecting defects.

Park et al. had successfully applied a CNN-based algorithm on various surfaces to detect the existence of defects in the target region of an image. While conventional ML algorithms require various types of networks with different hyper-parameters to be adopted on different surfaces, an uncomplicated CNN-based network is enough to achieve the same goal. Although each of their training dataset only contains 2000 images, they applied data augmentation to ensure quality of the result. In the end, they have concluded that the model is applicable for inexperienced users to detect faults in the images and has a strong advantage of applying parameters without tuning [28]. Applying data augmentation techniques to artificially increase the size of the datasets is common and be something which could be applicable in this thesis. By applying data augmentation to our test data, we may be able to make more generalizable claims about the NNs.

A study on detection of defects on steel surfaces was conducted by Lin et al. The system was developed with two parts, a Single Shot MultiBox Detector (SSD) model to learn possible defects and a deep residual network (ResNet) to classify the types of defects. According to the author, the proposed method is applicable to real-world scenarios. However, they study only applied Precision, Recall, and F-Measures to evaluate the results which means in certain scenarios the method might not be applicable [22]. The metrics adopted in this study are studied and discussed in this thesis However, none of them are used since all of them could potentially produce misleading results, which is an important insight learned for this thesis.

2.3 Constructing a test harness for ML

Kavzoglu stated, in his 2009 research paper, that "The quality and size of the training samples are crucially important for a successful classification". This statement was made as a result of having done manual data augmentation and refinement through various techniques and algorithms. The need for such research emerged because of the limited amount of *ground data* available in the Landsat 7 ETM+ [1] for land-cover/land-use analysis [13]. This enforces the idea that much can be gained from manual inspection of input data. Even though this process of inspection is more based in statistics and the quantifiable properties of the images, it is still related to NN performance and pattern recognition in the data, such as the work being done in this thesis.

Test Case Prioritization (TCP) is an important way of reducing time and computational cost during the process of testing. Test cases are executed in a specific order based on the specific requirements defined for the TCP. Thakur and Sharma has conducted a study of applying automated test case prioritization as compared to manual test case prioritization in regression testing with ten projects (2019). Their results shows that the fault detection rate is increased and execution time is reduced [41]. Their study has given us the inspiration of adopting TCP in our research to test the NNs more efficiently.

Raicharoen and Lursinsap conducted research into the use of a divide-and-conquer technique to reduce the size of a dataset used for training ML algorithms (2005). The aim was "... [to reduce] the computational time and the memory space as well as the sensitivity of the order and the noise of the training data.". The proposed prototype contains an isolated subset of the data used for training. This subset is selected based on the relevance to nearest neighbor predictions and is the only data that will be used for training. The experiments conducted resulted in, as predicted, a reduction in training cost and a decrease in both order dependence and sensitivity to noisy data [31]. This thesis hopes to prove that a similar technique of selecting a subset of data is possible when constructing a test harness. By identifying relevant *test cases* there can be a reduction in the size of the test harness required when evaluating NNs.

[1] https://eos.com/landsat-7/

3
Theoretical Background

In this chapter, the background knowledge of NN alongside various evaluation metrics are presented. After a comprehensive evaluation, some of the NN and metrics mentioned are adopted into the final test harness.

3.1 Neural Network

ML has seen an increase in research activity in recent years [12] with a greater focus lying on Deep Learning and NNs [19]. Various contests and large scale competitions [45, 17] could be a contributing factor in this, as well as the general development of more computationally capable hardware [33]. In this section we give an overview of recent research in the area of image recognition and classification using NN.

3.1.1 Convolutional Neural Networks

CNNs started being utilized for visual tasks as early as the late 1980s. However, due to computing components being limited in their ability to process big data at the time, it was really brought into the spotlight in 2012 [33].

The structure of a CNN is generally built up of different types of *layers*. These layers are combined in different ways depending on the problem at hand. A common way of assembling these layers can be as a series of *stages*. The initial stages are typically composed of convolutional layers and pooling layers. A convolutional layer is composed of many units, or neurons, which are combined with weights and organized into *feature maps*. These neurons extract information from local patches of pixels in the feature maps of previous layers, also known as a "discrete convolution" [19], which is illustrated in Figure 3.1. This also allows the network to reduce its required computing power [8].

Pooling layers, where the most common type is called *max pooling*, are used to reduce "... computations, memory usage and the number of parameters", according to Géron (2019). The different types of pooling layers all follow a similar methodology where a $i \times j$ grid is slid across the feature maps to extract a subset of the values in that grid. The distance the grid is moved between extractions is determined by the variable *stride*. In the case of max pooling, the *maximum* value of the $i \times j$ grid is extracted at each location, which is used as pixel for the output of that layer [19]. The final type of layer is the *fully connected layer*, also known as a *dense layer* [8].

While the previous layers have created abstracted features of the data, the purpose of fully connected layers is to combine these abstract features into concrete features, used in reasoning about classification [33]. This is typical in many NN and is an effort to reduce the many neurons generated by previous convolutional layers. The problem with these types of layers is that they are computationally intensive. Each neuron in the input (or previous) layer is connected with each neuron in the fully connected layer. Because of this there are typically a limited number of these layers in a network.

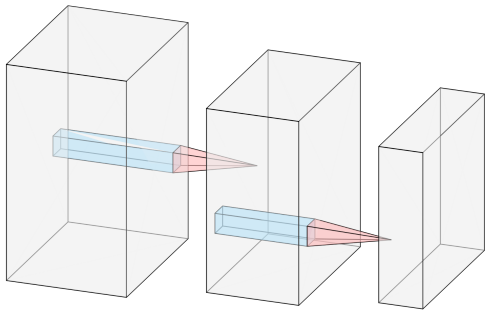

Figure 3.1: Convolutional Network

3.2 Metrics

There are various commonly used evaluation metrics for assessing computational experiments in the field of ML, for instance, *Recall*, *Precision*, F_1 *score*, and *MCC* [30]. Each of the mentioned metrics are presented in detail in the following text. However, before understanding any of the metrics mentioned above, one needs to comprehend what the *confusion matrix* is.

3.2.1 Confusion Matrix

With a classifier and an instance, there are four outcomes in the confusion matrix. If the instance is positive and is predicted to be positive, it is counted as a True Positive (TP), if that instance is predicted to be negative, then it is counted as a False Positive (FP). On the counter, if the instance is predicted to be negative but in fact is positive, it is counted as False Positive (FN). Last but not least, if the instance is negative and is predicted to be negative, it is counted as True Negative (TN) [46]. Table Table 3.1 demonstrates the the complete Confusion Matrix.

		Actual Values	
		Positive(1)	Negative(0)
Predicted Values	Positive(1)	TP	FP
	Negative(0)	FN	TN

Table 3.1: Confusion Matrix

3.2.2 Recall & Precision

Recall and Precision are commonly used in ML related experiments. Recall, also known as Sensitivity, is the ratio between True Positives and the sum of True Positive (TP) and False Negative (FN). Recall can be important for cases that focus on identifying all TPs. However, in the case of ML, it is generally desirable to achieve a high Precision, sometimes even disregarding the Recall value entirely. Precision or Confidence on the other hand, is the proportion of Predicted Positives that are True Positives. It is a method of measuring Predicted Positives [30]. Both equations of Recall and Precision are presented below (Equation 3.1 & Equation 3.2).

$$Recall = Sensitivity = \frac{TP}{TP + FN} \tag{3.1}$$

$$Precision = Confidence = \frac{TP}{TP + FP} \tag{3.2}$$

Nevertheless, which measurement should be adopted in the experiments depends on the circumstance. If the experiment is more inclined to get fewer FP values, the Precision measurement should be adopted. Conversely, if the focus of the experiment is to obtain fewer FN values, Recall can be a better option to adopt. In conclusion, one has to decide which measurement method should be applied in different situations. Despite both mentioned methods providing benefits in various cases, it is not difficult to see that True Negatives (TNs) are completely neglected from both of them. Therefore, there is not only the fear of bias being introduced to the results but also, in certain cases, TNs can play a key role in the final result. For instance, identifying a defective aircraft safety part , where misidentification could lead to life-critical situations.

3.2.3 Specificity

Much like Recall, Specificity is the True Negative Rate. It is the ratio of TN cases to the cases that are predicted to be Negative. Equation 3.3 defines how Specificity is calculated [24].

$$Specificity = \frac{TN}{TN + FP} \tag{3.3}$$

3.2.4 F_1 score

In order to get a more intuitive view to compare two classifiers, Precision and Recall are often combined into a single metric in the field of ML, the F_1 *score* [8]. Equation 3.4 shows the full definition of the F-measures.

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{3.4}$$

Here, β is a control variable to balance P and R. When β equals to 1, the result of this equation becomes the harmonic mean of Precision and Recall, also known as the F_1 score [35]. The incentive of adopting harmonic mean instead of the regular mean in this measurement is much more weight was given to the low values to avoid a skewed result, which means the classifier can only get a high F_1 score when both Precision and Recall are high [8]. Naturally, under the premise of adopting Precision and Recall, F_1 score also ignores the TN value. Therefore, this measurement cannot be applied in certain circumstances.

3.2.5 Matthews Correlation Coefficient

MCC was initially introduced to evaluate the performance of protein secondary structure prediction by B.W.Matthews in 1975 [26]. Over time, MCC is widely used in the field of ML to measure the quality of classifications. Due to the possible usage of all four outcomes from the Confusion Matrix, namely, TP, FP, TN, and FN, the result of this measurement can be more convincing comparing with the methods stated above. Equation 3.5 illustrates the working process of MCC [4].

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(FP + FN)(TN + FP)(TN + FN)}} \tag{3.5}$$

It is not difficult to see from the above formula that if any of TP + FP, FP + FN, TN + FP, TN + FN equals to zero, the MCC cannot be defined. Moreover, the result range of MCC ranges between [-1, 1], whereas 1 indicates complete agreement, -1 signifies complete disagreement and 0 is when the prediction result is not related to the ground truth. Therefore, if the evaluation of a binary classifier results in -1, the predicted result of 0 is, in fact, 1, likewise, if the predicted result is 1, the true value is 0 [3].

3.3 Testing of ML algorithms

As ML algorithms are dynamic constructs, their testing process differs from conventional software testing [48]. According to Zhang et al., testing a ML system requires two defined terms; *ML bug* and *ML testing*. Here, an *ML bug* is defined as "... any imperfection in a machine learning item that causes a discordance between the existing and the required conditions" and *ML testing* as "... [activities] designed to reveal machine learning bugs". This provides the ability to explain three different aspects of a ML system; the *required condition*, *machine learning items* and *testing*

activities. Different ML algorithms may require different types of required condition, for instance, correctness, robustness, and privacy. Machine learning items simply refer to the three separate components that generally make up a machine learning testing system, namely the data, the framework and the learning program. While testing activities may contain test input generation, test oracle identification, test adequacy evaluation and bug triage. [48].

The *required condition* refers to concepts which validate expected behaviours of the system-under-test (SUT), the most researched of which are *robustness* and *correctness*. *Correctness* is a functional measurement of a machine learning system's ability to, in the context of image classification, correctly classify out-of-sample data. There are a number of generally known metrics for correctness, some of which have been presented in the previous section. Another form of correctness is the use of validation when training a ML algorithm. One approach to validation is cross-validation, where the training data is split into a training sub-set and a validation sub-set. *Robustness* concerns the ability of a machine learning system to resist the effects of noise in the data. A robust system can perform adequately even with a high amount of noise and should therefore also be highly generalizable [48].

3.4 Terminologies used in the study

In Table 3.2 we present the terminologies used in this study.

Term	General Definition	Contextual Meaning
Artifact	An object which can be studied	A dataset for ML algorithms
Treatment	An artifact in a specific context	A test harness of difficult cases of composed car images
Stakeholders	Internal or external parties who have an interest in the research	Internal: Academia, External: Corporation
Metric	A way of quantifying the performance and accuracy of an algorithm	MCC & Precision & Specificity
Test harness (Testing framework)	A full test suite including data pre-processing and testing	Downscaling/Normalizing input pipeline, cross-validating dataset and test/evaluation using metrics (MCC, Specificity)
Characteristic	An aspect of an object	A combination of representative aspects defining a difficult classification case

Table 3.2: A table contextualizing industry terms for the context of this thesis

4
Methodology

This project is conducted according to the Design Science Research Methodology (DSRM) [42].

4.1 Design Science Research Methodology

DSRM describes a research process involving two fronts. Wieringa explains that when using design science, "you have to understand its major components, namely, its object of study and its two major activities" (2014). These two major activities are the processes of *designing* and *investigating* the aforementioned object, also known as an *artifact*, in its context with the purpose of arriving at a change in the same context. The context in this case includes "... the social context of stakeholders and goals of the project", explaining that this type of research is conducted in close proximity with a company or other stakeholder [42].

All the exercises in this methodology are done in a cycle, known as the Engineering Cycle (EC). This provides a structure to conduct research in a methodical way, using stakeholder goals, while allowing for the continuation of the research after each cycle. The design process of DSRM is a cyclical series of three distinct exercises called the Design Cycle (DC). The exercises of this cycle are visualized in Figure 4.1.

DSRM is a suitable methodology for this research as the process of identifying valid characteristics for the final test suite is intuitively iterative.

4.1.1 The Design Cycle

Each exercise of the DC has a specific purpose in order to provide value to the research. The first exercise is *problem investigation* where the current state of the problem is studied, and no requirements have yet to be specified. This is followed by an exercise focusing on designing a treatment, also known as *artifact in context*, called *treatment design*. The final step in this cycle is called *treatment validation* which attempts to state if the designed treatment is an appropriate solution, with respect to the initial problem. The consequence of this validation exercise is the possibility that treatments are designed which may not be relevant to the problem and is discarded [42].

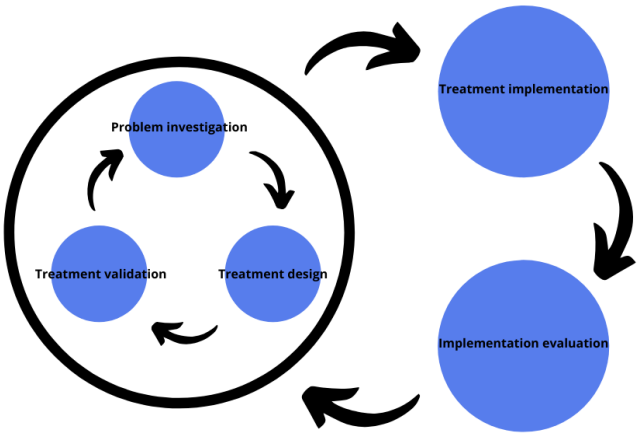

Figure 4.1: A figure showing the the general design of the design- and engineering cycles.

4.1.2 The Engineering Cycle

As stated before, the purpose of the EC is to evaluate the extent to which treatments contribute to stakeholder goals. By fully implementing the treatment(s) it is possible to effectively, accurately and comparatively evaluate their success.

4.1.3 Research Problems

In an effort to specify the purpose of a DSRM project, we define two types of *research problems*. They are in turn also meant to further clarify the difference between *design* and *investigation* and can be explained as *design problems* and *knowledge questions*. The *design problems* emerge from a need for change in the real world, or the context of the problem. This requires an analysis of any stakeholder goals and may result in several solutions. *Knowledge questions* instead focus on the current state of the real world, attempting to state the question in such a way that a single answer is possible. The effort of the methodology is to encourage the continuation of the cycle to further more research, and to spring more questions that might need answering. These questions are similar to the more well-known *research questions*, though they are divided into two distinct types [42].

The research questions of this thesis can be directly translated to knowledge questions, as they directly ask for an understanding of an issue, not a direct solution. There are several design problems in this thesis as well. When using images, their size determines the efficiency of the training process. What size of these images allow for a reasonable amount of training time? Moreover, in order to validate various defective datasets, the structure of the NN evolves into another design problem that needs to be considered. For instance, how many layers should the NN have

and how to tune its hyperparameters to solve this particular problem. What testing methodology to use to measure the performance of the NNs is another problem which needs to be solved. Testing methodologies described in section 3.3 are the common solutions to this type of problem. Which one of these solutions are best suited for the problem of evaluating NNs using a reference test harness?

4.2 The Engineering Cycle

The focus of this EC is to investigate the validity of characteristics of images when testing NNs. Suggestions of smaller datasets is created in each iteration of the DC. These suggestions are then combined and evaluated during the final stages of the EC, *treatment implementation* and *implementation evaluation*.

4.2.1 First Iteration of the Design Cycle

The first iteration of the DC is conducted in the following sequence; initially an investigative review of available metrics and methods for assessing the validity of a dataset is conducted, followed by the designing and creation of a NN that is applied to the defective dataset, resulting in a validation step for assessing the relevance of the treatment.

In this case, each dataset is validated for its relevance in the final treatment. This final treatment is a combination of each validated dataset.
The DC is generally a process of investigation into **RQ1**. By designing an initial dataset based around a missing car part, we are be able to validate if this dataset can be incorporated into a reference test harness based on approaches presented in this section.

4.2.1.1 Problem investigation

For the task of designing a test harness to validate and evaluate NNs applied to a dataset of composed images, it is important to understand the domain, or context, which the proposed treatment is introduced to. The images in this context depict computer-rendered cars of a real-world quality. By introducing the ability to customize the cars, a solution is required for the issue of verifying the enormous number of permutations these cars are capable of. To present an example, for simplicity's sake, we only allow two variations for each part and 81 parts, which results in $2^{81} \approx 2.4 * 10^{24}$ permutations. With this number increasing exponentially with each added component-type as well as component-variation, the number of permutations make it physically infeasible to pre-render all images and verify them all manually. In reality, the number of permutations is greater than the example given, as the bare minimum variations is two and several of the options have more than two variations. These permutations include specific combinations of characteristics that have the potential of contributing to the accuracy of classifier-algorithms applied to them. There are multiple ways of organizing and designing a test harness when investigating the effect these characteristics have on the ability to identify defective

images. Examples of way to do this are to only test defective images with single defects, or with multiple defects. This step in the cycle naturally initiates the investigation of **RQ1**.

To be able to perform the tests mentioned in the previous paragraph, software components such as the data, validation method and a framework to apply the validation on the data is required. The first DC investigates, designs and constructs an initial version of these components. The data consists of images with a single defect, the validation method is a shallow NN and the framework is a collection of scripts to resize the images to allow for the training the of NN. This lays the ground for further research into designing a full test harness for NNs.

The focus of this design cycle can be summarized as below:

- Construct an initial test case to validate the scenario of one missing car-part on a NN.
- What metrics allow us to confidently measure the quality of our dataset?

4.2.1.2 Treatment design

Using the information uncovered during the previous *problem investigation*, specifically what design problems and knowledge questions to work with, an initial batch of images are created using the standard 1080p resolution, and 4 color channels (RGBA), resulting in a shape of (1920x1080x4). Due to the nature of the tool used to generate the data, where all possible valid permutations of parts for a particular car model are created, we produced one set of 229 071 *complete* images and one set of 21 866 *defective* images, where specifically the wheels are missing. Figure 4.2 demonstrates an example of the *defective* set where the wheels are missing. These images are composed in a layered structure, depending on the configuration of that particular car. Assembling a car image, using a unique string for that particular configuration, begins with a completely transparent background. To build the car, an image of each part of the car is fetched from a server and layered according to the configuration. If there is an issue when fetching an image, that image is simply ignored in the final composed image. The imbalance of this dataset is due to limitations in the creation-tools provided to us. The tool is unable to handle the huge number of permutations that are required for these car models and thus would require the limitation of certain visible parts of the car to a subset of its possibilities.

To be able to assess the validity of a NN we used the metric of *MCC* which takes all values of the *confusion matrix* into account [26]. As the current form of validation for defective images is manual inspection, a focus on minimizing the number of misclassified negative prediction (being a defective image classified as complete) is paramount and thus we would benefit from explicitly investigating the *Specificity*. The predictive values of each individual image are also used to distinguish between the confident cases in the results.

Figure 4.2: An example of images used in the treatment

4.2.1.3 Treatment validation

A shallow NN is created to be used as the validation method for this step of the iteration. It consists of two sets of a specific sequence of layers and operations, which are; *Conv2D*, *BatchNormalization*, *Activation*, *MaxPooling* and *Dropout*. This is followed by a fully connected layer, known as *Dense* layers, resulting in an output layer using the *sigmoid* activation function. This activation function produces a single predictive value for each image, ranging from zero to one. All of the layers and operations are available in the *Keras* API [1] and the full model can be viewed in Figure A.3. This API, in combination with the backend called *Tensorflow*, allows for simple NNs configurations to be created easily and is used in this thesis. When applying this algorithm to the dataset it uses a pipeline which resizes the images to to 480x270x4, which is 1/16 of their original size. The intention is based on an aggregated consideration to maximize the GPU memory usage and reduce the time span during the training of the algorithm as the memory of the GPU has a limit of 8 GB [38]. Although it is possible for the GPU to take only one full sized image in one batch during training, it would require an enormous amount of time during this process. This NN algorithm works as a binary classifier which separates two classes based on the confidence of their predictions (confidence score).

10 000 images are sampled from the created set of *complete* images, being entirely intact cars. 10 000 images are also sampled from the created set of *defective* images, being wheel-missing. The sample size of 10 000 is due to the inconsistencies in the number of images created between the complete and defective datasets. As there is such a great difference between the number of complete and defective images created, a lower sample size is selected than the available images to reduce the risk of using a sample size that is greater than the number of images created in the following iterations of the DC. 20% of the images in these two classes are separated into a validation-set, used to evaluate the algorithm during each epoch. An additional 4000 images are sampled and used to during the prediction phase. The MCC score and Specificity are determined from these results, although these metrics may still lead to an unconvincing conclusion and the confidence of each prediction is also analyzed. This is due to both metrics resulting in 1 which is the perfect value for both. For further investigation, the confidence scores are saved and analyzed. The

[1] https://www.tensorflow.org/versions/r2.1/api_docs/python/tf/keras

variation in these values range from roughly 0,000 03 to 0,003 for the *defective* class and from 0,988 to 0,999 for the *complete* class.

By examining the confidence scores of the 4000 test images, the images with the weakest confidence score are extracted as these images should contribute to the test suite we are creating. By *weakest* confidence score, we mean the score furthest from *zero* for images of the defective class and the score furthest from *one* for images of the complete class. The threshold used during this included using the *interquartile range* (IQR). This approach uses the difference between the first (Q_1) and third (Q_3) *quartiles* of the data to detect outliers according to Equation 4.1 and Equation 4.2. The value of c is manually determined, but a common value is 1.5. These quartiles represent the maximum value of the first 25% and 75% of dataset entries, and the difference between these is the IQR. The outliers are the values in the dataset *below* T_{min} and *above* T_{max} [44].

$$\begin{aligned} T_{min} &= Q_1 - c \times IQR \\ T_{max} &= Q_3 + c \times IQR \end{aligned} \tag{4.1}$$

$$IQR = Q_3 - Q_1 \tag{4.2}$$

Defective images in our test harness correspond to the predictive value of **0**, meaning that *weak* defective prediction are values higher than T_{max}. Complete images correspond to the predictive value of **1**, and thus those weak predictive values would be lower than T_{min}.
From this first iteration of the DC it is clear that applying a NN to the dataset presented in this section is a suitable approach to validate the existence of difficult test cases. It also exposes the flaw that the dataset is not as complex as expected. MCC and Specificity produce a perfect result, requiring the use of the confidence score to identify difficult test cases.

4.2.2 Second Iteration of Design Cycle

In the second design cycle, we continue with the investigation of the complexity of car parts. This is done by creating a new dataset in the same vain as previously, with the difference being the defective images are missing the front grill, instead the wheels.

4.2.2.1 Problem investigation

To further investigate the issue of validating the dataset, the focus of the design cycle continues from the previous iteration. It is deemed necessary to create an additional dataset where other parts are missing. However, the investigation of what are the suitable metrics is not pursued as it has been examined from the first iteration.

4.2.2.2 Treatment design

By following the same procedure in the first iteration, another batch of 229 455 defective images are created, with the difference being that they are now missing the front grill.

The binary classifier is applied on this dataset also assessed by the metrics of MCC and Specificity as in the previous iteration.

4.2.2.3 Treatment validation

The same NN based binary classifier created in the first iteration is applied in this step to validate the dataset for extracting characteristics.

This time 10 000 defective images are sampled from this new dataset for use during training, and another 2 000 are sampled for testing. This iteration uses that same complete images samples during the previous iteration. The confidence scores acquired from the testing phase are saved into another CSV file and further analyzed to extract outliers according to our technique mentioned during the first iteration.

4.2.3 Third Iteration of Design Cycle

In the third design cycle, the investigation of the complexity of car parts continues. This is done by creating another dataset in the same vain as previously, with the difference being the removal of the side-mirrors from the defective images.

4.2.3.1 Problem investigation

The focus of this design cycle remains the same as the first two iterations. The only difference being another defective image dataset is created with other parts missing to further investigate the issue of validating the dataset.

4.2.3.2 Treatment design

By applying the tool to remove duplicate images, another 229 362 defective images with the same parameters are created. The only difference in this batch is that the missing part is the side-mirrors.

MCC and Specificity continue to be adopted in this iteration for validating the NN based binary classifier as the previous iterations.

4.2.3.3 Treatment validation

During this step, the binary classifier created from the first iteration is again applied in order to extracting characteristics from the side-mirrors-missing dataset.

Another 10000 images are sampled from the newly created dataset to be used in the training phase for the algorithm alongside with 2000 additional sampled to be used

during the testing phase. The confidence values are captured and further analyzed using our aforementioned technique.

4.2.4 Fourth Iteration of Design Cycle

In the fourth design cycle, we continue with the investigation of the complexity of car parts. This is done by creating another defective dataset with bumpers removed from the car images.

4.2.4.1 Problem investigation

Once again, the focus of this design cycle remains the same as the previous iterations. The only difference being another defective image dataset is created with other parts missing to further investigate the issue of validating the dataset.

4.2.4.2 Treatment design

By taking the same procedures in the previous iterations, another batch of 114 687 defective images is created, with the difference being that they are now missing the front grill. Due to the nature of the tool to remove duplicate images, the number of images in this batch is significantly smaller than other defective datasets. However, it is still sufficient to be sampled for the later process.

The earlier constructed binary classifier is applied on this dataset again, and assessed by the MCC and Specificity metrics.

4.2.4.3 Treatment validation

The same procedure is followed in this step. The binary classifier created in the first iteration is applied to validate the dataset for extracting characteristics.

Another 10 000 images are sampled from the newly created dataset to be used during training, and additional 2 000 are sampled for testing. Once again, the acquired confidence scores from the testing phase is saved and further analyzed by adopting our technique mentioned in the first design cycle.

4.2.5 Treatment implementation

The purpose of the previous DCs were to design and create datasets focused on a missing car part. These were validated using a NN and images with a low confidence score were extracted. During this step, the images that are extracted from the DCs are further analyzed to reduce the redundancy among them. This is done by manually observing patterns in the defective images. An example of such a pattern could be the combination of **a missing front grill** and **a white car body**. This is what is known as *characteristics* in this thesis and are patterns which the images can be categorized according to. There are many parts in a car, such as types of wheels or types of headlights, that can be removed entirely. The colors of the cars are different

as they represent an attribute of a car part, and thus cannot be removed. Therefore, we treat it differently in this thesis and count each color as part of a pattern, without the possibility of removing the color. After repeating the procedure of pattern extraction for three iterations, the final test harness is constructed by combining the cases that have similar patterns with each color that exist in the outliers.

The procedures conducted in the previous steps are what is required for the creation of a reference dataset. To allow for the use of the test harness in other domains, a reference dataset similar to the one presented in this thesis is required. The procedures necessary for the creation of such a dataset are documented, and can be followed to create a reference dataset for other domains.

The test cases in the test harness is structured in two layers. The first layer is to separate the different defects with the other layer separating the characteristics for that defect. These characteristics are presented in subsection 5.2.3. Each characteristic is represented by a sequence of pairs of test cases. These pares include a defective image of the characteristic and its corresponding complete image. These defective images are ranked according to the confidence score they received during validation.

When applying a model to this test harness, the test cases are supplied according to the characteristics, starting with the single test case with the strongest confidence score. If the model classifies both the images in that test case correctly, it continues to classify the next test case. If it fails, however, it moves on to the next characteristic. The metric for evaluation in this test harness is *how many test cases are correctly classified*. This metric allows the classifiers to be compared to each other and a final ranking of the classifiers can then be presented.

4.2.6 Implementation evaluation

To evaluate the test harness designed and created in the previous steps, it is used to test multiple NN-based binary classifiers. The relative performance of these classifiers is be known, giving an indication of the final result.

The evaluation is conducted by adopting three different NN-based binary classifiers that are designed and created by us. The structures of these classifiers are based on the CNN created for validation in this thesis. *Classifier 1* is the most sophisticated model among these three algorithms which contains one dense layer, three max-pooling layers as well as five convolutional layers, which doubles the hidden layers of the base CNN. This allows the classifier to better learn the increased complexity of incorporating four defects in the same dataset. The data used when training this classifier is sampled from the same complete, no-wheel, no-side-mirror, no-grill and no-bumper datasets created in the initial iteration of the DC. It consists of 10 000 images of each defect, 40 000, and 40 000 complete images, resulting in a balanced training dataset of 80 000 images.

On the contrary, *Classifier 2* is the model with the simplest structure. It consists of only one dense layer, one maxpooling layer, also one convolutional layer. This makes it less likely to pick up the full complexity of the images, producing a less-than-ideal correctness metric, as the purpose of this classifier is to achieve the lowest score of the classifiers we evaluate. It is trained on data derived from a different car model than the other data used throughout the thesis. The images are created using the same procedures, with the difference being that equivalent parts of the car had to be removed. It consists of 10 000 images of each defect, 40 000, as well as 40 000 complete images, resulting in 80 000 images in the dataset.

Last but not least, *Classifier 3* is the model in between Classifier 1 & 2, it has one dense layer, two maxpooling layers, and two convolutional layers. It is very similar to the classifier used for validation, with slightly changed parameters to achieve a different result. This classifier uses the same dataset as *Classifier 1*.

As explained in the previous section, these classifiers are individually applied to the test harness. Since the testing process starts with the easiest cases, the number of iterations acquired for each model corresponds to the evaluation result of the test harness. Hence if the model run through more iterations than others, it can be concluded as a more suitable solution for the problem. As explained in section section 3.3, there are several *required conditions* that can be used when testing. This thesis focuses on the *correctness* aspect of testing, disregarding *robustness* as there should not be any noise present in the data. The metric used to measure the correctness of the NNs tested is the number of correct predictions in the test harness.

5
Results

In this chapter, the results for both the design and engineering cycles will be presented. In total, 324 defective and 524 complete images were found difficult to classify throughout the design cycle iterations. After reviewing them to find similarities and patterns, the defective outliers were grouped based on their colors and included in the proposed test harness. This harness was then used to evaluate the performance of three binary classifiers.

5.1 Resulting predictions from Design Cycles

In the sections below we will present the results acquired from each iteration of the DC. These iterations produced sets of images deemed difficult to classify. The confidence score for the complete images are included only for the purpose of providing an overview of the data. These complete images will not necessarily be included in the final test harness, as the complete images will only be selected based on their characteristics, not their confidence score.

The box-plots presented in Figure A.1 and Figure A.2 display the general distribution of the confidence scores for the data. It also clearly shows outliers in the datasets using the same principle as explained in section 4.2.1.3.

5.1.1 First iteration

During this iteration we investigated images which specifically had their wheels missing. The results of the different metrics for correctness for this phase can be seen in Table 5.1. With an MCC score of 1.0 and *Specificity* of 1.0, we saw that both the NN and dataset produced valid results, although with a small predictive variation.

An excerpt of 10 images with the weakest confidence score in their predictions can be seen in Table 5.2, showing that there is a high confidence in both the *defective* and *complete* classes.

Using the equation presented in section 4.2.1.3 to find outliers, we could identify 66 of the defective images and 211 of the complete images as outliers in this dataset.

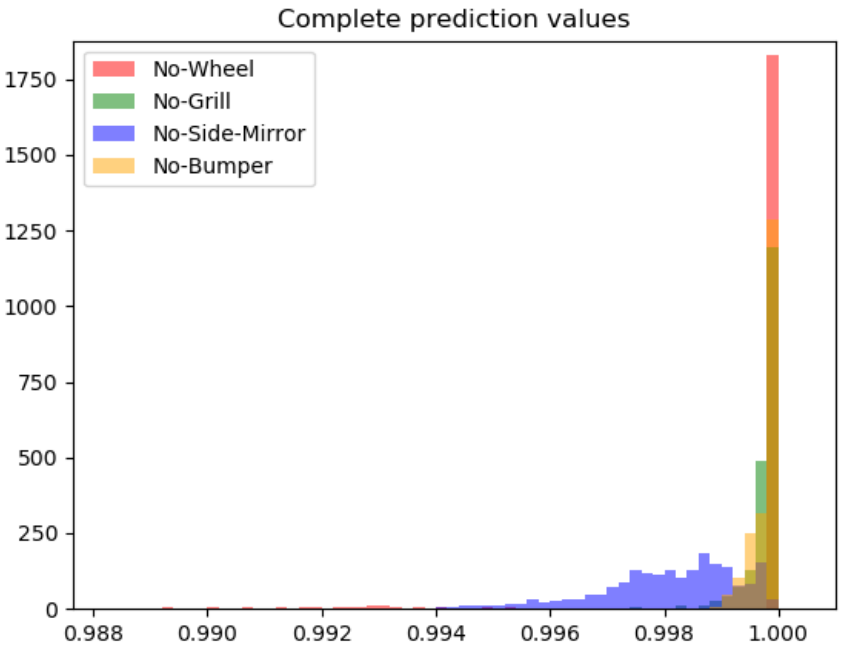

Figure 5.1: A histogram showing the distribution of values in the complete images in each dataset.

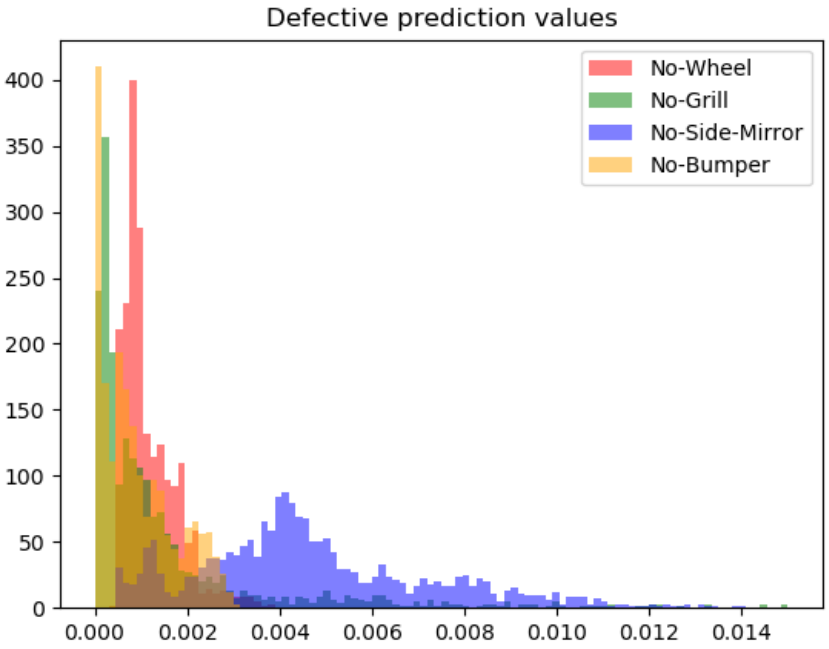

Figure 5.2: A histogram showing the distribution of values in the the defective images in each dataset.

Metric	Value
MCC	1.0
Specificity	1.0

Table 5.1: Metrics from validation of no-wheel dataset.

Class	Weakest confidence score
Defective	0,013
Defective	0,013
Defective	0,013
Defective	0,014
Defective	0,014
Complete	0,994
Complete	0,994
Complete	0,994
Complete	0,994
Complete	0,994

Table 5.2: Predictive values from validation of no-wheel dataset.

5.1.2 Second iteration

The results gathered from the second iteration through the DC can be seen in Table 5.4 and Table 5.3.

Metric	Value
MCC	1.0
Specificity	1.0

Table 5.3: Metrics from validation of no-grill dataset.

Class	Weakest confidence score
Defective	0,015
Defective	0,015
Defective	0,015
Defective	0,015
Defective	0,019
Complete	0,996
Complete	0,996
Complete	0,996
Complete	0,996
Complete	0,996

Table 5.4: Predictive values from validation of no-grill dataset.

By following the same procedure as in earlier iterations, 212 outliers were found in the no-grill defective dataset along with 180 in the complete dataset.

5.1.3 Third iteration

The results gathered from the third iteration through the DC can be seen in Table 5.6 and Table 5.5
By repeating the same procedure as in earlier iterations, 46 outliers were found in the no-side-mirrors defective dataset along with 50 in the complete dataset.

Metric	Value
MCC	1.0
Specificity	1.0

Table 5.5: Metrics from validation of no-side-mirrors dataset.

Class	Weakest confidence score
Defective	0,004
Defective	0,004
Defective	0,004
Defective	0,004
Defective	0,004
Complete	0,981
Complete	0,982
Complete	0,982
Complete	0,983
Complete	0,983

Table 5.6: Predictive values from validation of no-side-mirrors dataset.

5.1.4 Fourth iteration

The results gathered from the fourth iteration through the DC can be seen in Table 5.8 and Table 5.7.

Metric	Value
MCC	1.0
Specificity	1.0

Table 5.7: Metrics from validation of no-bumper dataset.

Class	Weakest confidence score
Defective	0,003
Defective	0,003
Defective	0,003
Defective	0,003
Defective	0,003
Complete	0,999
Complete	0,999
Complete	0,999
Complete	0,999
Complete	0,999

Table 5.8: Predictive values from validation of no-bumper dataset.

By repeating the same procedure as in earlier iterations, no outliers were found in the no-bumper defective dataset along with 83 in the complete dataset. This result means that the no-bumper dataset is irrelevant for the purpose of a reference dataset in a test harness.

5.2 Engineering cycle and final results

After having conducted four iterations of the DC, we moved into finalizing our test harness. This is done in two phases, initially by combining multiple datasets into a test harness, followed by evaluating NNs using the test harness.

5.2.1 Generalized procedure for creating reference datasets

The procedures conducted to construct the reference dataset, which includes the outliers, are found to be beneficial for researchers and corporations in other subjects. These procedures are collected and presented in the list below to create a clear methodology to recreate defective datasets for other domains.

1. Generate and sample both complete and defective images. The dataset should be balanced, meaning the number of sampled complete images should equal all sampled defective images.
2. Split the generated images into a training dataset and testing dataset.
3. Design and implement a binary classifier for validation purposes. This can be done by importing and potentially tuning the network created for validation in this thesis, or implementing an entirely new network.
4. Repeat the following procedures for all defects:
   (a) Train the network on a combination of the complete dataset and one defective dataset from the training images.
   (b) Evaluate the network using the testing dataset and specify the metrics *MCC* and *Specificity*. The predictive values of this evaluation should be stored for further analysis. In our case the data was saved to a file of the 'csv' format. It is also important to categorize or distinguish between defective image and complete images in these results.
   (c) Specify the definition of outliers to use. The same method used in this thesis can be adopted here. Extract outliers from the testing images using this definition.
   (d) Manually inspect the images to find characteristics.
5. Combine the outlier images, retaining information regarding the characteristics, into a reference dataset. The hierarchical structure used in this thesis for this purpose may be used.

5.2.2 Constructed Test Harness

A test harness is constructed that includes the extracted characteristics. The structure of the created test harness is presented in Figure 5.3.

This harness expects a pre-trained Keras-based binary classifier, saved in one of the two available formats (*.pd* and *.h*5), as an initial input. Defective images for the test cases, which are the outliers extracted in subsection 4.2.5, are stored in a hierarchical structure, first based on their specific defect, then their colors. Complete images are stored alongside their corresponding defective images. Each defective image is

paired with a complete image containing the same characteristics, creating a single test case. These test cases are supplied in a prioritized manner, according to the confidence score of the defective image. Each test case is pre-processed, normalizing the pixel values and resizing the images to 480x270x4. The defective image is then supplied to the classifier for prediction, followed by the complete image. If both of these images are correctly predicted, and there are more images with the same characteristics, another test case is supplied. This process continues until no more test case can be supplied or the classifier fails to predict the test case correctly. Finally, the results are combined and documented in a separate file.

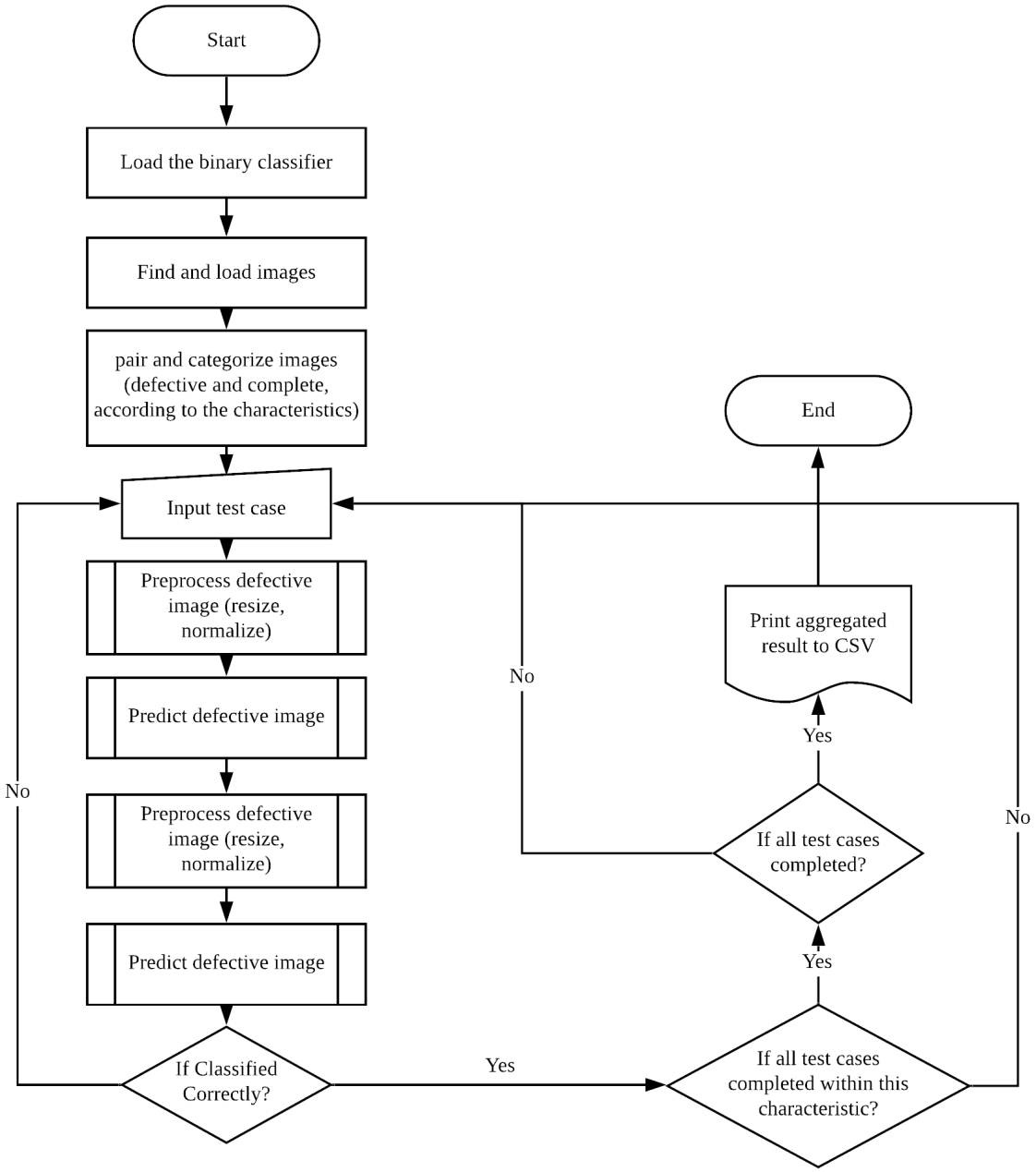

Figure 5.3: A flowchart visualizing the internal steps of the test harness.

5.2.3 Results of treatment implementation

After investigating the images labeled as outliers according to the definition described previously, we found the representative patterns of characteristics presented in Table 5.9. Each row includes a defect in the first column, the colors identified in the outliers in the second, and the other parts that all outliers have in common in the third. For the no-wheel dataset a total of four colors were identified, with every car having *mudflaps* and *low suspension*. The no-side-mirror dataset contains six colors along with a *black window outline* and *low suspension*. The no-grill dataset has four colors, but with no other common parts. Furthermore, the no-bumper dataset is not included in this table as there are no outliers in the dataset. Based on our definition of characteristics, being patterns exhibited by all images in the outliers, no further reduction in the number of images can be performed.

The result acquired from the evaluation of the algorithms on the test harness is presented in Table 5.10. The According to the table, Classifier 1 has the highest number of execution iterations. Last but not least, Classifier could not complete any iteration. Finally, Classifier 3 ranked in between these classifiers in this evaluation, it has run through 116 iterations.

Missing part	Car color	Other parts
Wheels	Candy White & Moon White Metallic & Brilliant Silver Metallic & Olive	Mudflaps & Low suspension
Side mirror	Race Blue Metallic & Corrida Red & Velvet Red & Maple Brown Metallic & Crystal Black Metallic & Black Magic Pärleffekt Metallic	Black window outline & Low suspension
Grill	Candy White & Moon White Metallic & Brilliant Silver Metallic & Olive	None

Table 5.9: The characteristics extracted from outliers in each defective dataset.

Classifier Name	Successfully executed iterations
Classifier 1	324
Classifier 2	0
Classifier 3	116

Table 5.10: The results of the evaluation of NNs using the test harness

6
Discussion

By following the methodology presented in this report, and after achieving the aforementioned results, we now provide a discussion regarding these phases of this project.

6.1 Validity of algorithm application

In this section we discuss why the NN-based binary classifier was adopted in the validation procedures as well as its validity.

6.1.1 The constructed NN used in the validation steps

Many studies have either adopted or even suggested using Tensorflow [1] and CNN to construct a binary classifier for solving image classification problems [8, 47, 27, 6]. This initiates our investigation of designing and constructing a CNN-based binary classifier with the no-wheel dataset as its training data. However, as explained in chapter 2, no studies in the related work could be found that attempts to solve this specific problem of using CNNs to classify defective car images. Thus, we started with constructing a typical CNN based on the reference structure presented in the Keras guide for image classification [2]. This was then followed by a phase of hyperparameter adjustment, which is commonplace when adopting NN [2, 8], to increase the correctness of the network for the problem of this thesis. We increased the use of Dropout [39] and introduced BatchNormalization [37], to further improve regularization in the network. Other hyperparameters were changed until a high correctness was achieved from the network. As a result of its high correctness, it was decided to be used for the validation process of this thesis and the full structure can be seen in Figure A.3.

6.1.2 Does the algorithm detect the defect

When pursuing answers for **RQ1**, we investigate ways of validating *datasets* using NNs. By applying a NN to a dataset, we are able to get an accurate measurement of how well that *specific* NN is able to classify that *specific* dataset. To be able to validate this dataset for other NNs, using the information gathered in section 4.2.1.2 regarding the process of image composition, there is a possibility that the classifier

[1] https://www.tensorflow.org/versions/r2.1/api_docs/python/
[2] https://keras.io/examples/vision/mnist_convnet/

is putting focus on other aspects of the image than the missing part.

By investigating the differences between the complete images and defective images, there are three scenarios of visible differences. One possibility is that there are no other layers underneath the missing part, resulting in the transparent background being visible in that area instead. Another possibility is that there are other parts in the underlying layers. These parts could include the *real* components which would be on the real car in this area. Finally, they could contain a placeholder color (in the case of these images, that color is black) instead of the components which would exist in that area in a real car. An example can be seen in Figure 4.2, where there is a black area where the right wheel would be. This provides evidence that the classifier used for validation is not classifying some other aspect of the image than the defective part, as there is no *one* aspect that is changed for all images.

In an effort to find evidence of the suitability of our binary classifier for validating the test harness, we analyze the relationship between the proportion of pixels in the image taken up by the missing part. The purpose of this approach is to investigate if the classifier only detects the average difference of noise in the images, only a specific aspect of the image, or the specific missing part. All the different defective images include varying permutations of the scenarios mentioned in the previous paragraph. For example, the no-grill images only introduce a black color when removed but no-wheel images introduce both black color and the transparent color. The theory is that if the difference in the number of pixels of two different types defective images, no-side-mirror and no-wheel as an example, correspond to the difference in confidence score of those same defective images, this should mean the classifier focus on the differences in the full set of pixels for each missing part. On the other hand, if the classifier is only detecting the difference of one aspect of the images, such as the transparency or the existence of the black color, the proportions of pixels-to-confidence-score should not correspond in the same way.

This is carried out by manually cropping out the target car parts, e.g. the side mirrors from a complete car image. The resulting image include transparent pixels, from having a transparent background as stated before. However, to only count pixels relevant to our results, these transparent pixels are ignored in this step. Due to the uncertainties of the manual cropping, the number of pixels obtained is not precise. Hence, by studying the obtained results, we can conclude that the pixel-proportions roughly coincides with the confidence score-proportions, as three out of four datasets produced positive results. An example being, the ratio of the pixel-proportions between the no-side-mirror dataset and the no-wheel dataset is roughly 23%, meanwhile the ratio in the confidence score of the same datasets are roughly 25%.

6.2 Test harness

In this thesis we investigate defective car images to design and create a test harness. This test harness only includes a representative set of test images, allowing for a simpler evaluation process for the NNs. The intention is that if more effort is put into selecting the test images, fewer images would be required in the final test harness. These images are selected from outliers discovered in the validation process. This definition of 'outliers' is defined as the general definition of outliers in a 'whiskers-plot', which has been described before in this thesis. The requirement is that a reasonable amount of images can be gathered through this method. Also, that a dataset with a deviating amount of outliers easily can be discovered. By the fact that the 'no-bumper' dataset produced no outliers, it is discarded and gives more confidence that the methodology used is valid for selecting difficult test cases.

Another aspect of the test harness which is a point of discussion is the fact that only isolated defects (no-wheel, no-grill and etc) are investigated. The magnitude of the difference between analyzing isolated defects or combinations of them is exponential. It should also, as conducted in this thesis, be more useful to initially focus on isolated defects, as those results is likely to be usable when investigating combined defects. Even though the methodology for researching the combined defects is unknown at this time, it would most likely require a large amount of test cases. There are ways to reduce these, by utilizing methods such as *factorial designs* for the experiments, but the single defects would still need to be analyzed [1]. Therefore, initially producing results for a smaller, more isolated set of defects is more likely to be helpful in future research.

6.2.1 Generalizability

One of the major issues with the problem we investigate in this thesis is the extent to which the results are generalizable. This is the reason for using a, in our mind, shallow network for validation. By only using two convolutional layers, along with regularization techniques such as batch normalization and dropout, we made an effort to increase the generalizability as much as possible. Additionally, Masters and Luschi suggested that the batch-size used during training can severely impact the stability and generalizability of a model (2018). He also mentions that a batch-size of less than 32 is ideal. As such, we are confident that using a batch-size of 8-16 images, which is the limit for the hardware we used when training, would most likely increase the generalizability as well.

6.3 Reflection on related work

In the related work section of this thesis we presented several studies in three main categories. All of these have varying degrees of commonalities with our thesis, from evaluating NNs, detecting defects or constructing a test harness for ML. The issue we found in the field is that there exists a lack of research that bridge these research topics. Also, research into creation of a representative *test* dataset is not abundant

as no such paper was found for this thesis, although the paper by Raicharoen and Lursinsap is closely related, focusing on reducing the size of the *training* dataset. Possibly due to this, there is room for research regarding using the aforementioned reference *test* dataset to evaluate various NNs. On the contrary, studies regarding the evaluation of NNs are common. Furthermore, many studies were found regarding defect detection using NNs. Among them, we have discovered that a specific type of NN, namely CNN, is commonly used in the field of detecting defects on various surfaces. This was corroborated by the fact that one of the tested classifiers in this thesis produced a perfect result when evaluated using the created test harness.

6.4 Evaluation of the classifiers

Three binary classifiers were designed and created in this thesis, as explained in subsection 4.2.6. The aforementioned test harness was then used to evaluate these classifiers.

In order to verify that the test harness is a valid method for assessing various binary classifiers, we only selected three of them as a proof of concept. Also, these classifiers use known structures, meaning their performance, in relation to each other, should be known. However, due to the time limit, we do not evaluate more than three classifiers to further verify our test harness.

Based on the acquired results from Table 5.10, we see that Classifier 1 successfully is able to pass all 324 iterations. Conversely, Classifier 2 does not pass any iterations. Additionally, Classifier 3 pass 116 iterations. These results are to be expected. The classifiers using data derived from the same car model that the test harness is based on performs better than a classifier that is trained on a different dataset, based on another car model. The difference in performance for Classifier 1 and Classifier 2 can be contributed to the structure of the classifiers. Classifier 1 uses a NN-structure of greater depth, allowing for more features to be learned. This is confirmation that the experiments verify our initial hypothesis presented in RQ3.

6.5 Threats

The thesis was carried out by constructing and experimenting with various datasets on three binary classifiers. The potential validity threats are discussed in detail below.

6.5.1 Construct Validity

Construct validity primarily concerns earlier phases of a project, and is related to the design of the research being conducted. Staron points out it is about "... how we create the measurement instruments that measure the effects in our study." (2020).

In order to better focus our research on the defective parts, we decided to use only one car model, from a single angle. This makes it difficult for us to generalize our research, as it . Moreover, during the procedure of selecting outliers from the datasets, after observing the confidence score, we used a method including IQR to select the outliers as mentioned in subsubsection 4.2.1.3. Although the mentioned IQR method is a standardized way of selecting outliers, we cannot be certain that it is the most suitable way in our context. This produces a risk to the construct validity.

When constructing the test harness, no software tests are applied to the software in question, whether that be the framework, the data or the classifiers used. There is therefore a possibility of the existence of bugs, potentially affecting the results.

6.5.2 Internal Validity

Threats to internal validity are things that can affect the independent variable with respect to causality, without the researcher's knowledge [43].

During the validation process, when selecting outlier images, it was discovered that one dataset (no-bumper) did not contain any outliers. This dataset was discarded for this reason, without further investigation, as it had been made clear beforehand that only outliers would be included in the test harness. It is possible that by performing further analysis on these images would result in the discovery of characteristics. We chose to put our trust in our method of exposing outliers, but it is possible this was not an ideal method to use. However, with the other datasets containing at least 46 outliers, we believe those are distinctly different from the no-bumper dataset.

6.5.3 External Validity

According to Wohlin et al., external validity are conditions that would potentially hinder the researchers to generalize their experiments results to industrial practice (2012).

The tool which was used to generate images proved to be very effective. However, there were various limitations of it that interfered with the complexity of the image creation process. As an example, due to a flaw in the tool, the number of images created for each defective part was different. This resulted in an increase in the time required to create our datasets. Also, in the later period, we had to face the issue of insufficient disk space due to the large number of images being created. Hence, we were required to remove more parts to complete the creation of the datasets with the risk being that desired parts may also be removed.

Given the confidentiality of the data provided in this research, the practical validation may be difficult to replicate for other researchers. Therefore, the binary classifier created may need to be adjusted by its own hyperparameters to fit various datasets. Due to the fact that manual cropping was adopted to validate our binary classifier that was mentioned in section 6.1, the acquired results may not be highly

precise. This may increase the threat of generalizing the results.

Moreover, the expected performances of the networks evaluated in this thesis were determined based on their structures. This only informs the general performance, but this may differ depending on the data in relation to the network.

6.5.4 Conclusion Validity

Conclusion validity relates to the process of analyzing data, finding patterns in the data and making inference at the end of the research. It is also our ability to draw correct conclusions from our observation [40].

Only one CNN-based binary classifier was adopted in the treatment validation process. This makes it difficult to make statements regarding the validity of the test harness for other types of NNs. As a result, it makes us lack confidence in the procedures applied during this study.

Furthermore, a problem occurred during the creation of the no-wheel dataset. Despite trying multiple times to re-execute the creation, the result remained the same. This directly leads to the no-wheel dataset having a significant amount of images fewer in comparison with the other datasets. Hence, this random event may cause inaccurate sampling to further impact our final result.

7
Conclusion

This thesis was conducted with the purpose of designing and creating a valid reference test harness for evaluating NNs in the context of defect detection in car images. The study is closely linked to three research questions and their answers are provided below.

7.1 Answers to research question

RQ1:

What characteristics in defective car images can be used in a test harness to assess the quality of a NN?

According to Table 5.9, the characteristics we have identified for the no-wheel outliers are four colors with mudflaps and low suspension. Six colors, along with black window outline and low suspension, are identified for the no-side-mirror outliers. Last but not least, there is nothing more than the four colors that can be identified in the no-grill outliers.

RQ2:

How can images with the characteristics identified in RQ1 be combined into a valid test harness for NNs?

The procedures required to identify characteristics in datasets are collected and presented in the list in subsection 5.2.1. This is a clear methodology to recreate defective datasets for other domains, and is a prerequisite to the creation of the full test harness. These procedures are found to be beneficial for researchers and corporations in other subjects.

Based on the table we can see that the performance of Classifier 1 overtakes the other two in this evaluation. Classifier 2, which has the shallowest structure and less relevant training data, does not pass a single iteration. Meanwhile, the performance of Classifier 3 is greater than Classifier 2, yet worse than Classifier 1. Hence, these results confirm our theory that our test harness is a valid approach to evaluate various binary classifiers.

RQ3:

Can we identify adequate NNs using our test harness? What are the relevant metrics when comparing NNs applied to the aforementioned dataset?

As explained in the answering of RQ2, we have adopted three binary classifiers in the process of evaluation as a proof of concept. With the prior knowledge of the tested classifiers' performances, the test harness can be verified if it is a qualified tool for identifying adequate NNs. From the obtained results, we can see that the results of the experiments have verified our initial hypothesis in this RQ. We are able to identify adequate NNs using our test harness. Given that the structure and training data of Classifier 1 is more suitable than the other two classifiers, in addition to it being the most appropriate classifier according to the results, makes it the classifier we would recommend for a similar defect detection problem.

7.2 Future work

This thesis is only an initial step in the investigation of creating a reduced test harness for NNs and there are many ways to continue the research conducted in this thesis.

As we have mentioned throughout the report, this thesis was conducted with a focus on individual defective car parts, from a single car model. To further investigate the root cause of the difficulties in detecting defects, combinations of defective car parts would need to be taken into consideration. Furthermore, to increase the generalizability of the results, analyzing additional car models may be pursued.

We also categorized the outliers during validation according to *combinations* of characteristics. It may be beneficial to do further research into which specific *single* characteristics is the actual contributor to the difficulty during classification.

Bibliography

[1] Jiju Antoy. *Design of Experiments for Engineers and Scientists*. Elsevier, 2014, pp. 1–672. ISBN: 9780080994178. DOI: 10.1016/C2012-0-03558-2. URL: https://linkinghub.elsevier.com/retrieve/pii/C20120035582.

[2] Nurshazlyn Mohd Aszemi and P.D.D Dominic. "Hyperparameter Optimization in Convolutional Neural Network using Genetic Algorithms". In: *International Journal of Advanced Computer Science and Applications* 10.6 (2019), pp. 269–278. ISSN: 21565570. DOI: 10.14569/IJACSA.2019.0100638. URL: http://thesai.org/Publications/ViewPaper?Volume=10&Issue=6&Code=IJACSA&SerialNo=38.

[3] Pierre Baldi et al. *Assessing the accuracy of prediction algorithms for classification: An overview*. 2000. DOI: 10.1093/bioinformatics/16.5.412. URL: https://academic.oup.com/bioinformatics/article/16/5/412/192336.

[4] Sabri Boughorbel, Fethi Jarray, and Mohammed El-Anbari. "Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric". In: *PLoS ONE* 12.6 (Mar. 2017), pp. 1–17. ISSN: 19326203. DOI: 10.1371/journal.pone.0177678. URL: https://doi.org/10.1371/journal.pone.0177678.

[5] Yuan-chin Ivan Chang. "Multiple-class classification: Ordinal and categorical labels". In: *Communications in Statistics - Simulation and Computation* 46.10 (Nov. 2017), pp. 7561–7581. ISSN: 0361-0918. DOI: 10.1080/03610918.2016.1242732. URL: https://www.tandfonline.com/doi/full/10.1080/03610918.2016.1242732.

[6] Neha Chaudhuri and Indranil Bose. "Application of Image Data Analytics for Immediate Disaster Response". In: *Proceedings of the 21st International Conference on Distributed Computing and Networking*. New York, NY, USA: ACM, Jan. 2020, pp. 1–5. ISBN: 9781450377515. DOI: 10.1145/3369740.3372729. URL: https://dl.acm.org/doi/10.1145/3369740.3372729.

[7] Marc Claesen and Bart De Moor. "Hyperparameter Search in Machine Learning". In: (Feb. 2015). URL: http://arxiv.org/abs/1502.02127.

[8] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. second. O'Reilly Media, Inc., 2019. ISBN: 9788578110796. URL: https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive computation and machine learning. MIT Press, 2016. ISBN: 9780262035613. URL: http://search.ebscohost.com/login.aspx?direct=true&AuthType=sso&db=cat07470a&AN=clc.7a977b09caff4a9b803659e97e338eb3&site=

eds-live&scope=site&custid=s3911979&authtype=sso&group=main&profile=eds.

[10] Tianmei Guo et al. "Simple convolutional neural network on image classification". In: *2017 IEEE 2nd International Conference on Big Data Analysis, ICBDA 2017*. Institute of Electrical and Electronics Engineers Inc., Oct. 2017, pp. 721–724. ISBN: 9781509036189. DOI: 10.1109/ICBDA.2017.8078730.

[11] Mazharul Islam et al. "Improving Neural Network Classifier using Gradient-based Floating Centroid Method". In: (July 2019). URL: http://arxiv.org/abs/1907.08996.

[12] M. I. Jordan and T. M. Mitchell. "Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245 (2015), pp. 255–260. ISSN: 10959203. DOI: 10.1126/science.aaa8415. URL: https://science.sciencemag.org/content/349/6245/255.

[13] Taskin Kavzoglu. "Increasing the accuracy of neural network classification using refined training data". In: *Environmental Modelling & Software* 24.7 (July 2009), pp. 850–858. ISSN: 13648152. DOI: 10.1016/j.envsoft.2008.11.012. URL: http://dx.doi.org/10.1016/j.envsoft.2008.11.012%20https://linkinghub.elsevier.com/retrieve/pii/S1364815208002156.

[14] May Phyo Khaing and Mukunoki Masayuki. "Transparent Object Detection Using Convolutional Neural Network". In: *Advances in Intelligent Systems and Computing*. Vol. 744. Springer Verlag, 2019, pp. 86–93. ISBN: 978-981-13-0869-7. DOI: 10.1007/978-981-13-0869-7{_}10. URL: http://link.springer.com/10.1007/978-981-13-0869-7_10.

[15] Arshia Khan and Hans-Dietrich Haasis. "Producer–buyer interaction under mass customization: analysis through automotive industry". In: *Logistics Research* 9.1 (Aug. 2016), p. 17. ISSN: 1865-035X. DOI: 10.1007/s12159-016-0144-9. URL: http://link.springer.com/10.1007/s12159-016-0144-9.

[16] Asharul Islam Khan and Salim Al-Habsi. "Machine Learning in Computer Vision". In: *Procedia Computer Science* 167 (2020), pp. 1444–1451. ISSN: 18770509. DOI: 10.1016/j.procs.2020.03.355. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050920308218.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM*. Ed. by F Pereira et al. Vol. 60. 6. Curran Associates, Inc., 2017, pp. 84–90. DOI: 10.1145/3065386. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[18] Donghwoon Kwon et al. "An Empirical Study on Network Anomaly Detection Using Convolutional Neural Networks". In: *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. Vol. 2018-July. IEEE, July 2018, pp. 1595–1598. ISBN: 978-1-5386-6871-9. DOI: 10.1109/ICDCS.2018.00178. URL: https://ieeexplore.ieee.org/document/8416441/.

[19] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444. ISSN: 14764687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

[20] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* (1998). ISSN: 00189219. DOI: 10.1109/5.726791.

[21] Daniel D. Lee and H. Sebastian Seung. “Learning the parts of objects by non-negative matrix factorization”. In: *Nature* 401.6755 (Mar. 1999), pp. 788–791. ISSN: 00280836. DOI: 10.1038/44565. URL: https://www.nature.com/articles/44565.

[22] Chih-Yang Lin et al. “Cascading Convolutional Neural Network for Steel Surface Defect Detection”. In: *Advances in Intelligent Systems and Computing*. Vol. 965. Springer Verlag, 2020, pp. 202–212. ISBN: 9783030204532. DOI: 10.1007/978-3-030-20454-9{_}20. URL: http://link.springer.com/10.1007/978-3-030-20454-9_20.

[23] Qing Liu et al. “A review of image recognition with deep convolutional neural network”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10361 LNCS. Springer Verlag, 2017, pp. 69–80. ISBN: 9783319633084. DOI: 10.1007/978-3-319-63309-1{_}7. URL: https://link.springer.com/content/pdf/10.1007%2F978-3-319-63309-1.pdf%20https://link.springer.com/content/pdf/10.1007%2F978-3-319-63309-1_7.pdf.

[24] Stephen Marsland. *Machine learning: An algorithmic perspective*. Second. Chapman and Hall/CRC, Oct. 2014, pp. 1–452. ISBN: 9781466583337. DOI: 10.1201/b17476. URL: https://www.taylorfrancis.com/books/9781466583337.

[25] Dominic Masters and Carlo Luschi. “Revisiting Small Batch Training for Deep Neural Networks”. In: (Apr. 2018). URL: http://arxiv.org/abs/1804.07612.

[26] B. W. Matthews. “Comparison of the predicted and observed secondary structure of T4 phage lysozyme”. In: *BBA - Protein Structure* 405.2 (Oct. 1975), pp. 442–451. ISSN: 00052795. DOI: 10.1016/0005-2795(75)90109-9. URL: http://www.sciencedirect.com/science/article/pii/0005279575901099%20https://linkinghub.elsevier.com/retrieve/pii/0005279575901099.

[27] Nhung Thi Hong Nguyen et al. “Pavement crack detection using convolutional neural network”. In: *ACM International Conference Proceeding Series*. New York, New York, USA: ACM Press, 2018, pp. 251–256. ISBN: 9781450365390. DOI: 10.1145/3287921.3287949. URL: http://dl.acm.org/citation.cfm?doid=3287921.3287949.

[28] Je Kang Park et al. “Machine learning-based imaging system for surface defect inspection”. In: *International Journal of Precision Engineering and Manufacturing - Green Technology* 3.3 (July 2016), pp. 303–310. ISSN: 21980810. DOI: 10.1007/s40684-016-0039-x. URL: http://link.springer.com/10.1007/s40684-016-0039-x.

[29] Husein Perez, Joseph H. M. Tah, and Amir Mosavi. “Deep Learning for Detecting Building Defects Using Convolutional Neural Networks”. In: *Sensors* 19.16 (Aug. 2019), p. 3556. ISSN: 1424-8220. DOI: 10.3390/s19163556. URL: https://www.mdpi.com/1424-8220/19/16/3556.

[30] David M W Powers. *Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation*. 2007. URL: http://david.wardpowers.info/BM/Evaluation_SIETR.pdf.

[31] Thanapant Raicharoen and Chidchanok Lursinsap. "A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm". In: *Pattern Recognition Letters* 26.10 (2005), pp. 1554–1567. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2005.01.003. URL: http://www.sciencedirect.com/science/article/pii/S0167865505000115.

[32] Karthik Ramasubramanian and Abhishek Singh. *Machine Learning Using R*. Berkeley, CA: Apress, 2019. ISBN: 978-1-4842-4214-8. DOI: 10.1007/978-1-4842-4215-5. URL: http://link.springer.com/10.1007/978-1-4842-4215-5.

[33] Waseem Rawat and Zenghui Wang. "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review". In: *Neural Computation* 29.9 (Sept. 2017), pp. 2352–2449. ISSN: 0899-7667. DOI: 10.1162/neco{_}a{_}00990. URL: http://www.mitpressjournals.org/doi/abs/10.1162/neco_a_00990.

[34] A L Samuel. "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM Journal of Research and Development* 3.3 (July 1959), pp. 210–229. ISSN: 0018-8646. DOI: 10.1147/rd.33.0210. URL: http://ieeexplore.ieee.org/document/5392560/.

[35] Yutaka Sasaki. *The truth of the F-measure*. January 2007. Mar. 2007, pp. 1–6. URL: https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf.

[36] Ling Shao et al. "Performance evaluation of deep feature learning for RGB-D image/video classification". In: *Information Sciences* 385-386 (Apr. 2017), pp. 266–283. ISSN: 00200255. DOI: 10.1016/j.ins.2017.01.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0020025517300191.

[37] Leslie N. Smith. "A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay". In: (Mar. 2018). URL: http://arxiv.org/abs/1803.09820.

[38] Nimit Sharad Sohoni et al. "Low-Memory Neural Network Training: A Technical Report". In: (Apr. 2019). URL: http://arxiv.org/abs/1904.10631.

[39] Nitish Srivastava et al. "Dropout: A simple way to prevent neural networks from overfitting". In: *Journal of Machine Learning Research* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 15337928.

[40] Miroslaw Staron. *Action Research in Software Engineering*. Cham: Springer International Publishing, 2020. ISBN: 978-3-030-32609-8. DOI: 10.1007/978-3-030-32610-4. URL: http://link.springer.com/10.1007/978-3-030-32610-4.

[41] Akshit Thakur and Gitika Sharma. "Neural Network Based Test Case Prioritization in Software Engineering". In: *Communications in Computer and Information Science*. Vol. 956. Springer Verlag, July 2019, pp. 334–345. ISBN: 9789811331428. DOI: 10.1007/978-981-13-3143-5{_}28. URL: http://link.springer.com/10.1007/978-981-13-3143-5_28.

[42] Roel J. Wieringa. *Design science methodology: For information systems and software engineering*. Mar. 2014, pp. 1–332. ISBN: 9783662438398. DOI: 10.1007/978-3-662-43839-8. URL: https://link.springer.com/book/10.1007/978-3-662-43839-8.

[43] Claes Wohlin et al. *Experimentation in Software Engineering*. Vol. 9783642290. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–236. ISBN: 978-3-642-29043-5. DOI: 10.1007/978-3-642-29044-2. URL: http://link.springer.com/10.1007/978-3-642-29044-2.

[44] Jiawei Yang, Susanto Rahardja, and Pasi Fränti. "Outlier detection: How to threshold outlier scores?" In: *ACM International Conference Proceeding Series*. New York, New York, USA: ACM Press, 2019, pp. 1–6. ISBN: 9781450376334. DOI: 10.1145/3371425.3371427. URL: http://dl.acm.org/citation.cfm?doid=3371425.3371427.

[45] Xulei Yang et al. "Deep learning for practical image recognition: Case study on kaggle competitions". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 923–931. ISBN: 9781450355520. DOI: 10.1145/3219819.3219907. URL: https://doi.org/10.1145/3219819.3219907.

[46] Jingxiu Yao and Martin Shepperd. "Assessing Software Defection Prediction Performance: Why Using the Matthews Correlation Coefficient Matters". In: (Mar. 2020). DOI: 10.1145/3383219.3383232. URL: https://arxiv.org/abs/2003.01182v1.

[47] Lorena Zapata et al. "Detection of Cutaneous Tumors in Dogs Using Deep Learning Techniques". In: *Advances in Intelligent Systems and Computing*. Ed. by Tareq Ahram. Vol. 965. Cham: Springer International Publishing, 2020, pp. 83–91. ISBN: 9783030204532. DOI: 10.1007/978-3-030-20454-9{_}8. URL: http://link.springer.com/10.1007/978-3-030-20454-9_8.

[48] Jie M. Zhang et al. "Machine Learning Testing: Survey, Landscapes and Horizons". In: *IEEE Transactions on Software Engineering* (June 2019), pp. 1–1. ISSN: 0098-5589. DOI: 10.1109/tse.2019.2962027. URL: http://arxiv.org/abs/1906.10742.

A

Appendix 1

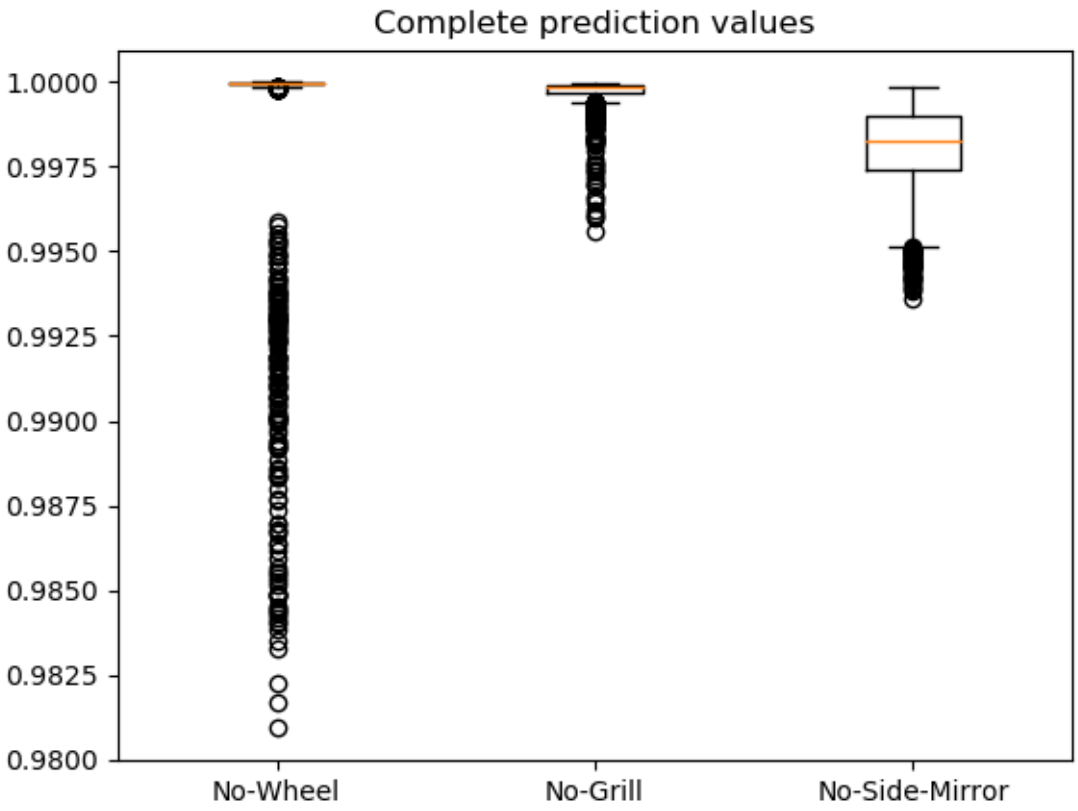

Figure A.1: A box-plot showing the confidence score distribution of complete prediction in each class.

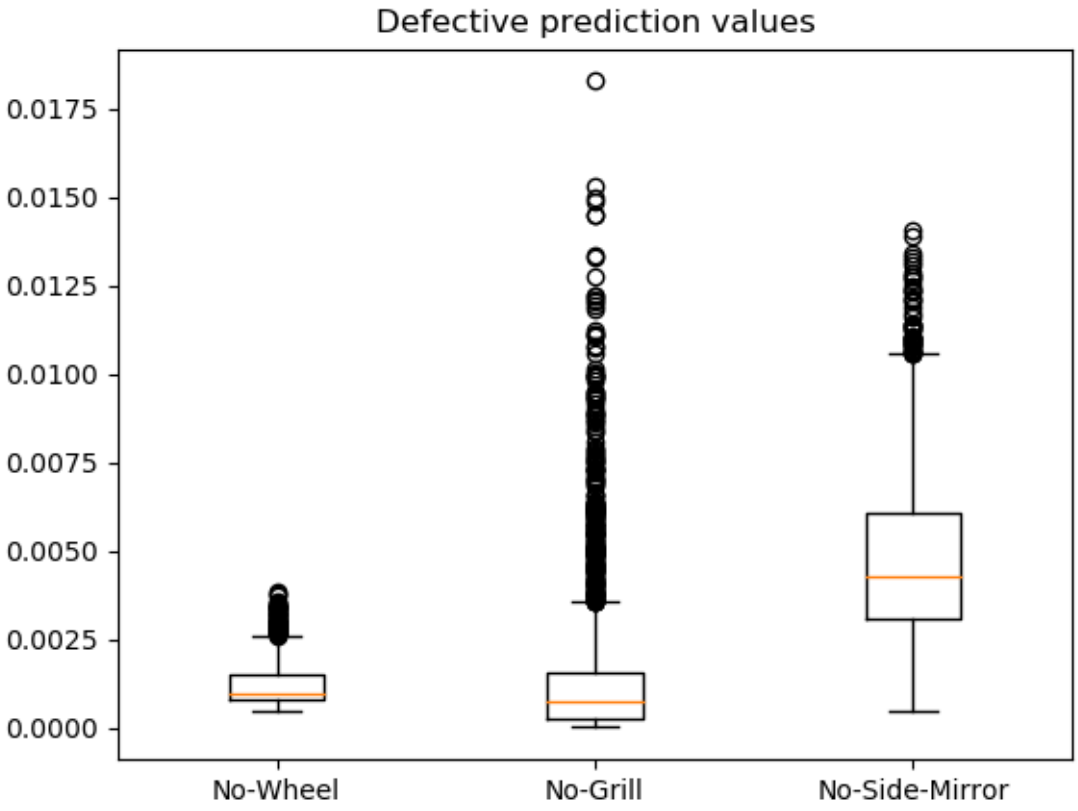

Figure A.2: A box-plot showing the confidence score distribution of defective prediction in each class.

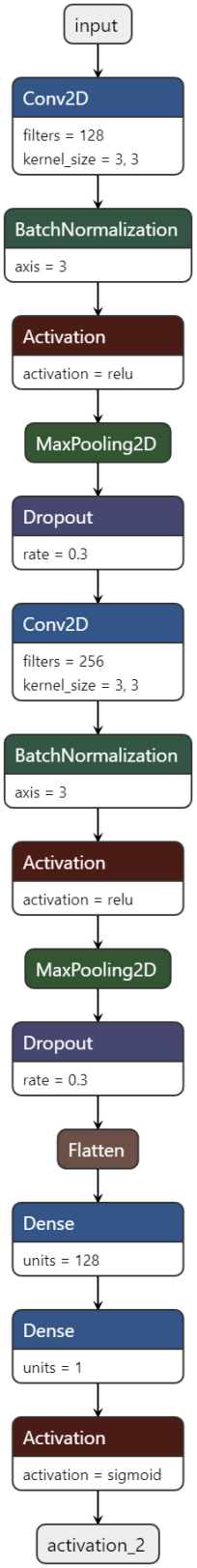

Figure A.3: The model used for validation of the individual defective datasets

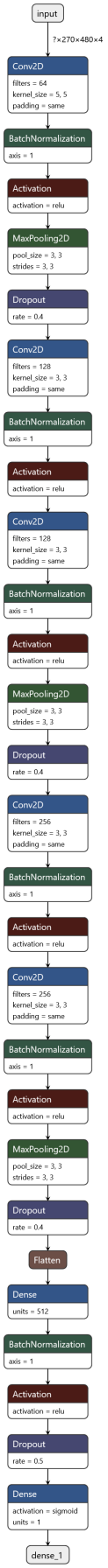

Figure A.4: Classifier 1, used in the evaluation process.

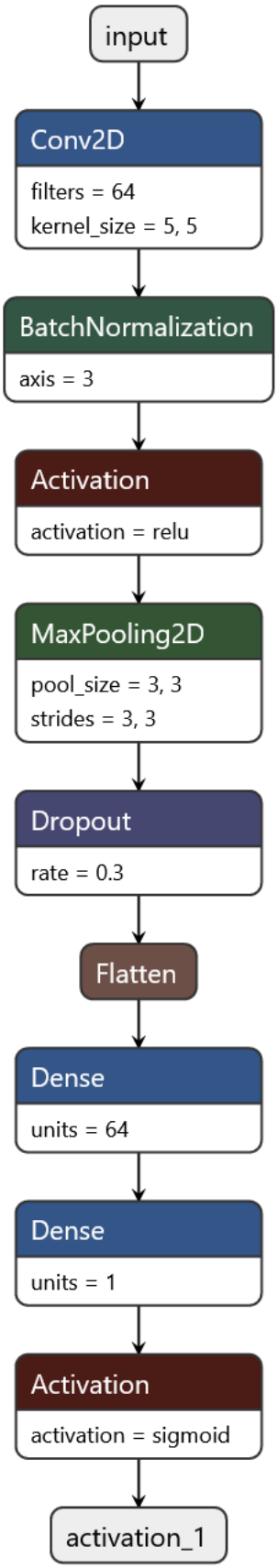

Figure A.5: Classifier 2, used in the evaluation process.

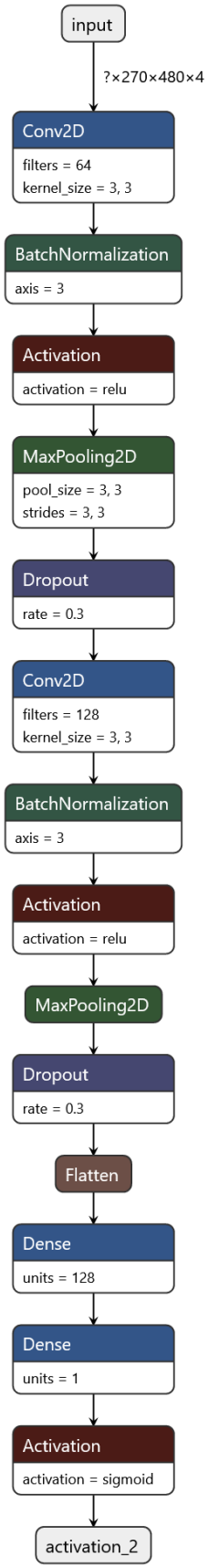

Figure A.6: Classifier 3, used in the evaluation process.