Highly Automated Corner Cases Extraction: Using Gradient Boost Quantile Regression for AI Quality Assurance

Niels Heller and Namrata Gurung
QualityMinds GmbH, Nürnberg, Germany

Keywords: Artificial Intelligence, Quality Assurance, AI Quality, Data Quality, Corner Cases, Autonomous Driving.

Abstract: This work introduces a method for Quality Assurance of Artificial Intelligence (AI) Systems, which identifies and characterizes “corner cases”. Here, corner cases are intuitively defined as “inputs yielding an unexpectedly bad AI performance”. While relying on automated methods for corner case *selection*, the method relies also on human work. Specifically, the method structures the work of data scientists in an iterative process which formalizes the expectations towards an AI under test. The method is applied in a use case in Autonomous Driving, and validation experiments, which point at a general effectiveness of the method, are reported on. Besides allowing insights on the AI under test, the method seems to be particularly suited to structure a constructive critique of the quality of a test dataset. As this work reports on a first application of the method, a special focus lies on limitations and possible extensions of the method.

1 INTRODUCTION

Artificial Intelligence (AI) applications, especially applications of deep neural networks, have found a variety of application domains in the recent years. Of special concern are safety relevant AI applications, where automatically taken decisions may cause damage in case of failure, or prevent that damage in case of success. Most prominent examples of such domains are Autonomous Driving and medical applications. Especially in these domains calls for quality assurance measures for AI applications were recently expressed (Tian et al., 2018; Hamada et al., 2020; Challen et al., 2019).

Obviously, arguments for the safety of an AI application must involve more than simply reporting performance readings on a test dataset. A prominent incident which illustrates the dangers of malfunctioning AIs in Autonomous Driving happened in 2016, when an AI misclassified a truck as an underpass which caused an accident (referenced for instance in (Tian et al., 2018)). Nobody wants to be surprised by such an error of an AI that is already *in use*. More specifically, a safety argumentation for an AI needs to encompass a consideration of what is difficult for the AI and why this does or does not impede its use.

Several examples of such “difficult inputs” are canon for specific fields of application: Image recognition for Autonomous Driving is often concerned with AI brittleness and adversarial attacks (Tian et al., 2018; Huang et al., 2017), while medical applications often struggle with unbiased training datasets (Challen et al., 2019) and domain gaps (Reinke et al., 2021) where inputs from different data sources cause erroneous behaviours. The automated search for such difficult inputs is addressed in the literature: Hendrycks et al. impressively demonstrated that a dataset containing “naturally occuring” adversarial examples can be curated from a dataset *for a variety* of computer vision AIs (Hendrycks et al., 2021), other approaches use heuristic methods for specific AIs (Kwiatkowska, 2019).

Such difficult inputs are sometimes referred to as “corner cases” (albeit the term “corner case” being subject to conflicting definitions which are discussed in Section 2). Arguably, the identification and characterization of such corner cases plays a major role for AI quality assurance. The authors propose the following intuition for corner cases: Before evaluation, it cannot be known what poses a problem for a trained AI. Yet, while there are “intuitive” inhibitors, which are more or less aligned with what humans performing the same task might find difficult, there are also “non-intuitive” ones which are specific to the AI system under test. For instance, an image classification AI might be distracted by dark or low contrasted images just as a human might be (which is an intuitive inhibitor), but it might also react heavily to adversar-

Heller, N. and Gurung, N.
Highly Automated Corner Cases Extraction: Using Gradient Boost Quantile Regression for AI Quality Assurance.
DOI: 10.5220/0011152200003269
In *Proceedings of the 11th International Conference on Data Science, Technology and Applications (DATA 2022)*, pages 62-73
ISBN: 978-989-758-583-8; ISSN: 2184-285X

ial attacks or uncommon surface structures. Cases of non-intuitive inhibitions can be considered corner cases (because no one thought of them before) and it is in the best interest of an AI vendor to know as many non-intuitive inhibitors as possible before taking an AI into production.

Painted with a broad brush we can define a corner case as an input yielding an "unexpectedly poor AI performance". Finding a new performance inhibitor means updating one's expectation towards an AI, and this can yield corner cases: Bad performances which are *not* explained by the current expectation. Such inhibitors may be not initially recognizable in a test dataset, and might require extensive feature engineering in order to be defined. For instance, image recognition AIs may be disturbed by subtle texture patterns, but rarely does a dataset contain the respective labels. It is hence the work of a data scientist in the role of a "feature engineer" to find, model, and test possible performance-inhibiting features.

Feature engineering typically aims at improving AI-performances by providing a model with features it cannot infer on its own (Heaton, 2016). This is explicitly **not** the aim of the approach introduced in this report. Instead, we aim at guiding a feature engineering and testing process which, as a final outcome, produces formal performance expectations towards an AI and corner cases (i.e. inputs which fall below these expectations). In that, the following work pertains not only to the field of AI quality, but also to *dataset quality*. Further it aims at producing a *human-understandable* descriptions of what the AI under test struggles with.

Contribution. This work introduces a method for analysing AI-models aiming at the detection and extraction of corner cases from a test dataset. The application of this method is highly (but not completely) autonomous and relies on human work in the form of feature engineering.

This report is structured as follows: Section 2 discusses related work, Section 3 describes the proposed method in detail, Section 4 illustrates the results of applying the method in a concrete use case in Autonomous Driving, and these results are discussed in Section 5. Section 6 concludes this report.

2 RELATED WORK

Corner Cases. Ensuring the robustness and predictable performance of an AI, even when faced with rare, unexpected, situations is an important concern especially in Autonomous Driving. A broad definition of corner case in Autonomous Driving can be found in (Bolte et al., 2019) "A corner case is given, if there is a non-predictable relevant object/class in relevant location". There are several methods developed for automatic corner case detection, especially for Deep Neural Networks (DNNs), which will be elaborated on below.

One approach to detect corner cases is based on transforming the input data. When a transformed input results in a different class label prediction of a DNN, the input is considered a corner case. Globally effective transformations such as changes in contrast, brightness and saturation are presented in (Hosseini and Poovendran, 2018). Additionally, locally restricted interventions, such as blur or the insertion of lens flares representing sensor damage can be also be used to detect corner cases (Secci and Ceccarelli, 2020).

Apart from transformation of inputs, metamorphic relations (Xie et al., 2011) such as effect of an input on steering angle output is described in the DeepTest framework (Tian et al., 2018). For example, a slight change in contrast of an input frame should not affect the steering angle of a car (while Tian et al. report on such cases). Thus, input-output pairs that violate those metamorphic relations can be considered corner cases.

Further, a white-box testing framework, DeepXplore (Pei et al., 2017) presents a method to solve the joint optimization problem that maximizes both differential behaviors and neuron coverage of DNNs by using gradient ascent to find corner cases. Starting from a seed input, DeepXplore performs a guided search following the gradient in the input space of two similar DNNs supposed to perform the same task such that it finally uncovers the test inputs that lie between the decision boundaries of these DNNs. Such test inputs that are classified differently by the two DNNs are then labelled as corner cases. Additionally, custom domain specific constraints can also be predefined in DeepXplore.

Corner cases can also arise depending on scene constellations. In particular, with increasing degree of occlusion of relevant objects such as pedestrians, it can become increasingly difficult for the DNN to recognize relevant objects. Wu et al. present a method to overcome this by combining occlusion modelling with multiple view representation in a complex dynamic Bayesian network (Wu et al., 2003). Isele et al. look at occlusions occurring at lane intersections and its effect on autonomous vehicles' navigation using deep reinforcement learning (Isele et al., 2018). By synthetically generating such occluded data, corner case scenarios can be identified.

Another method to detect corner cases presented by Hanhirova et al. is based on using a simulated environment using the CARLA software (Dosovitskiy et al., 2017), which was connected to a Machine Learning framework. Herein, a client drives a vehicle with AI-subsystem within a simulated environment. The vehicle-state information about the simulated world (ground truth) and the perception of the AI agent within the same-defined simulated condition are recorded and compared. The scenarios that yield in conflicting states are marked as corner cases (Hanhirova et al., 2020).

Bolte et al. present a framework for a detection system that is based on the ability of DNNs to predict the next frame when given a certain sequence of input frames. The difference between the actual frame and the predicted frame gives a corner case score (Bolte et al., 2019). Through this method it is possible to identify scenarios which could lead to high corner case probability.

As mentioned, there are several ways of automatically generating and detecting corner cases. However, depending on the operational domain, the variability of possible inputs can be very large. Also, certain types of corner cases can be specific to the type of DNN used. For the work in this report, an AI model trained on synthetically generated dataset is used. For testing, DeepLabV3+ (Chen et al., 2018) with pre-trained weights on a test dataset was used. Corner cases arising due quantitative, perceptual and situational inhibitors in the test dataset will be discussed in the next sections. Further, ability of augmented images to produce corner cases will also be explored.

Anomaly Detection. Anomalies (also referred to as outliers) can be defined as “observations which deviate so much from other observations as to arouse suspicions that it was generated by a different mechanism” (Hawkins, 1980). Note that this definition relates well to the introductory definition of corner cases as “unexpectedly bad performances” as it stresses the surprise an expert might experience towards an observation. The detection of such anomalies is a very important, and extensively studied, aspect of statistical analysis, which has found numerous application domains (Chandola et al., 2009; Hawkins, 1980). While the abstract *method* proposed in Section 3 does not specify how “expectations towards an AI” are to be formulated, the concrete *application* of that method reported in Section 4 relies heavily anomaly detection in the statistical sense.

An extensive literature review on anomaly detection can be found in (Chandola et al., 2009), which provides a well-structured classification framework for different approaches. According to that framework, the approach used in Section 4 uses *unsupervised anomaly detection* (without a training set of examples of anomalies) and finds *contextual anomalies* (values are only anomalies because their divergence from their context). Specifically, the method could be seen as “regression model based” where a regression model is fitted to define the contextual normality of a value, and residuals from that normality are interpreted as their anomaly score (Chandola et al., 2009). The regression models used in Section 4 are Quantile Regression models, which try to predict values splitting all values according to a fixed ratio (for instance, fitting a curve such that 10% of the observations lie below a curve). A pleasant introduction to this technique and some applications can be found in (Waldmann, 2018). Unsurprisingly, Quantile Regression has found applications in anomaly detection such as finding anomalies in health insurance claims (Nortey et al., 2021) and mechanical fault detection (Xu et al., 2019).

As mentioned in the introduction, the method introduced in this report relies on *human work*, which is not uncommon for anomaly detection approaches. Several approaches, for instance, incorporate human judgement on automatically selected examples to improve (Chai et al., 2020; Islam et al., 2018) or even to choose (Freeman and Beaver, 2019) anomaly detection mechanisms.

AI Quality Assurance and AI Safety. The process of quality assurance for AI directly translates from Quality Assurance (QA) processes used in software development. Poth et al. focus in (Poth et al., 2020) on a systematic methodical approach (the evAIa method: evaluate AI approaches) that evaluate risks of the machine learning model using a questionnaire specifically for AI products and services and outputs relevant QA recommendations. Lenarduzzi et al. focus their consideration on AI Software Development i.e. the software-driven definition, training, testing, and deployment of AI systems (Lenarduzzi et al., 2021). Hamada et al. provide general guidelines for AI QA for different application domains, one of which being Autonomous Driving (Hamada et al., 2020). Similar to the method proposed in this report, the method proposed by Hamada et al. “helps to create test cases”, which, in their description, appear similar to examples shown in Section 4.

Besides general guidelines for AI QA, the notion of “AI robustness” is of special interest in the Autonomous Driving domain. Several approaches assess the robustness of the DNNs to adversarial perturbations (Kwiatkowska, 2019) exist. One of the approaches is heuristic search for adversarial examples, which is done by searching for images created by changing the important pixels of an image in the classification decision (Wicker et al., 2018). Another approach is based on automated verification approaches which aim to provide formal guarantees on the robustness of DNNs (Tjeng et al., 2017). Herein, the maximal safe radius is defined as the maximum size of the perturbation that will not cause a misclassification. With this approach, one can encode the set of necessary constraints for AI safety. Lastly, since neural networks are based on probabilistic interpretation, this leads to defining probabilistic guarantees on their robustness and is presented in (Cardelli et al., 2019) wherein Bayesian neural networks (BNNs) can capture the uncertainty within the learning model. Thus, probabilistic verification for DNNs is another approach to ensure AI safety.

Note that the method introduced in this report positions itself somewhat in between existing AI QA approaches: It is more specific than general guidelines on AI Software and dataset quality, while also being more general than analyses of specific Machine Learning models (such as DNNs).

3 METHOD

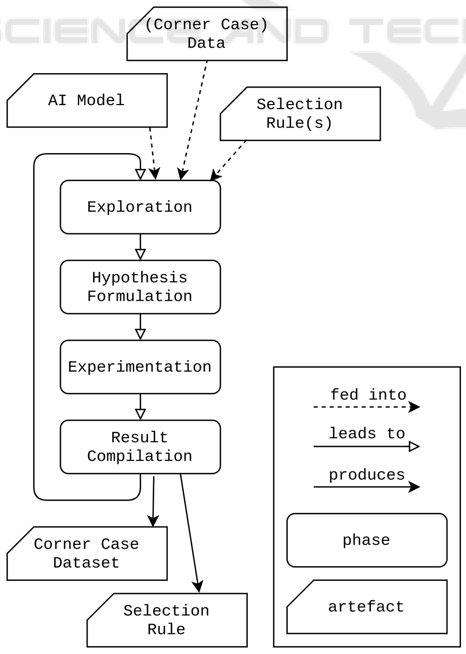

Figure 1: The phase iteration model.

In the following, a structured approach to be carried out by data scientists (or teams of data scientists) is described. It aims at operationalizing the loose intuition of a corner case as a "performance below expectation" in so-called selection rules which encompass the decision whether an input is below of what was "expected".

The proposed method consists of four phases which are carried out in iterations. Each iteration receives the AI-model to be examined, a (sufficiently large) test dataset, and a set of selection rules as input. Then, an additional selection rule and a corner case dataset are produced, where the corner case dataset is a subset of the input test dataset. The next iteration then receives the output of the current iteration as input and so forth. Obviously, the test dataset should not be used for training of the AI as this might lead to over-fitted selection rules.

Formally, a selection rule S is a binary function which maps an input i to a decision: $S : i \mapsto \{\text{true}, \text{false}\}$. Selection rules rely on an "expectation of performance" as well as the actual performance recorded for the input. Performance expectations can be expressed as simple value cut-offs such as "this natural language processor cannot comprehend sentences longer than n items" or as general functional relationships such as "an image with brightness x should at least yield an AI performance of $f(x)$". The performance expectations defined Section 4 are expressed as Gradient Boost Quantile Regression models. For the image brightness example, input images are chosen for which the performance is below the 10th percentile for its brightness.

The corner case dataset is produced by applying all current selection rules to the test dataset and filtering for observations which are selected by *all* current rules. The iteration is stopped when one does not find a new rule which provides more insights into the performance inhibitors of the AI. While this stopping criterion seems quite imprecise, its concrete implementation depends on the use case (the AI under test, the test dataset, etc.). In Section 4, the stopping point is chosen by quantifying how much *new information* a new selection rule provides.

Note that the approach as a whole follows a simple intuition data scientists might have when looking for performance inhibitors: If a large portion of the input data consists of images which are to dark for the AI to cope with, it might be wise to "filter out these cases", in order to focus any further analyses: Other effects might have *smaller* influences while still being of importance. This explicitly does not mean that the performance inhibitor "darkness" is not important for a safety argumentation, rather that a dark image is not a corner case (because bad performance was expected).

The main "human" work the data scientists carry out in this approach can be seen as feature engineering, defined as "the act extracting features from raw data and transforming them into formats suitable for the machine learning model" (Zheng and Casari, 2018). While, as mentioned above, the purpose of this process is to find human-understandable descriptions of performance inhibitors, not necessarily to fuel better ML-models.

Phases. The phases in the method are inspired by what is colloquially known as "the scientific method" (an extensive summary on the term and applications are provide e.g. in (Gauch Jr et al., 2003)) which typically consists in making an observation, formulating a hypothesis on the explanation of the phenomenon, and conducting experiments testing the hypothesis. The observations in this case are behaviours or decisions made by an AI and their comparison against what the AI was expected to provide.

In the Exploration phase, the data scientists explore the dataset using the methods typical for their domain, such as plotting data, and evaluating specific observations which yielded a bad AI-performance. In the Hypothesis Formulation phase, a concrete performance inhibitor is hypothesised (sentence length, image brightness, ...), and measurements of these features are implemented. Note that there might competing measures of the same inhibitor: While determining the length of a sentence might be straight forward, there are numerous ways of measuring the (perceived) brightness of an image. The experimentation phase aims at choosing a concrete inhibitor feature and evaluating whether this feature is both *new* (i.e. it is not already modelled by previous inhibitors) and *important* (i.e. it actually has an influence on performance). When the experimentation phase fails for instance due to the inhibitor not being significant, the process is reset to the experimentation phase. Finally, the Result Compilation phase consists in applying the newly found selection rule along with all old rules to produce a new corner case dataset.

Considerations. Arguably, applying the method to a concrete AI and a test dataset requires making numerous decisions. How is novelty and importance of an inhibitor determined? For quantile regression models, suitable quantile thresholds must be chosen. Also, there are often a number of AI-performance metrics to choose from. All these considerations depend heavily on the AI under test and the available test dataset. In Section 4, the method is applied to a use case from Autonomous Driving, and the rationales for the regarding choices were are presented there.

The data scientists play a major role in the process, which would not motivate the method's name "Highly Autonomous Corner Case Extraction". Yet, the process of *applying* new selection rules, testing their novelty and importance can be automated. Also, it can be expected that selection rules found with this method require more and more intricate feature engineering the later they are found. Indeed, in the application presented in Section 4, the first rules relied on features already present in the dataset and the regarding selection rules could have been applied completely automatically. In this way, the goal of the method is to provide the data scientist with the problems which require human intervention as fast as possible.

It has to be noted that, by applying the method, one may find not only performance inhibitors and regarding corner cases, but also bugs in the data, and one may learn about the shortcomings of the applied performance metric. In the greater context of AI-quality assurance, all of this is valuable information, and a discussion of how such secondary findings can be used, are given in Section 5. Extending on this thought, an aspect crucial for the successful application of this method, which will remain untouched in this report, is the provision of adequate tooling especially when large datasets are evaluated.

4 APPLICATION RESULTS

The method described in the previous section was applied to an autonomous driving scenario: The AI under test performs semantic group segmentation (Lateef and Ruichek, 2019) of traffic images taken from a vehicle camera. The AI performs a perceptual interpretation of the scenario recorded by the camera: Identifying cars, roads, street signs, pedestrians, and so forth. Obviously, such an AI is safety-critical as failing to detect vulnerable road users, for instance, can easily lead to serious accidents. Therefore the evaluation presented in the following focuses on pedestrian detection specifically.

The AI under test is a DeepLabV3+ model (Chen et al., 2018) with a ResNet (He et al., 2016) backbone (here, the backbone of a model is a feature extracting neural network within the larger DeepLabV3+ architecture). It was trained with a batch size of 6 over 50 epochs on a synthetically generated dataset containing 21884 frames of inner-city traffic scenarios. Each frame was rendered in an 1920 by 1280 pixel image. An additional set of 5173 frames were used for validation during training, and an additional set of 9897 frames were held back for testing. These test frames were used to carry out the method.

The test frames contained in total 206767 pedestrians, which are denoted as instances in the following. This number of instances (in average 20.9 per frame) is obviously very high for “real world” traffic scenarios. Yet, the test dataset was specifically produced to test pedestrian detection performances, and the synthetic nature of the production lead to a very generous ground truth data which contained instances which were barely visible. This made two filter steps necessary: Firstly, all instances which were to far from the camera to be safety relevant were dropped from the dataset. Secondly, data bugs (mislabeled data, instances being completely occluded behind walls etc.) were removed. In a way this *second* filter step already constituted the application of a selection rule as described by the method. One does not expect any performance (of this model) for instances that are not visible.[1] After these primary filter steps, a set of 58809 instances were left for the method application.

The AI performance was measured using the Jaccard Index (also known as “Intersection over Union” or simply IoU in the following) of the ground truth segmentation, which contained a pixel-exact labelling of all instances, and the inferred segmentation by the AI.

Figure 2 shows two frame sections containing instances which were not sufficiently detected by the AI (both yielding an IoU of 0.084). Yet, while the reason for this bad recognition of the left instance seems to be obvious (the image being quite dark and low contrasted), such simple reasons can not be found for the instance on the right: Is it the person’s clothing? The contrast to their background being to high? Their posture? The method proposed in this report tries to find such surprising instances as the one on the right, which was indeed a selected corner case, while disregarding instances as given on the left.

Performance Inhibitors and Selection Rules. After applying the method described in Section 3, a total of 8 selection rules were found, which were each based on a distinct performance-inhibiting feature. As mentioned in Section 3, including a new rule requires the assurance of its *importance* and *novelty*. To assure an inhibitor’s importance (i.e. an influence on AI-performance), common statistical measures where used: Producing and evaluating scatter, box, and violin plots as well as relying on linear or polynomial regression and associated fitness values. Figure 3 shows two examples of such inhibitors: depth (the distance of an instance to the camera) and brightness (measured by evaluating the colours proximate to the in-

[1]Note that an AI capable of object permanence might be able infer the existence of invisible instances

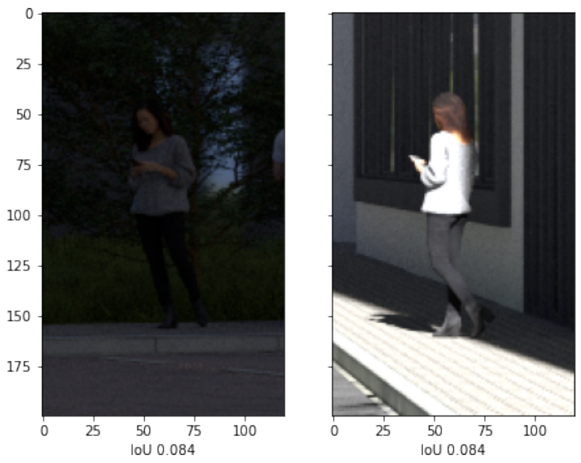

Figure 2: Sections of two frames with poorly detected instances. Left: Low brightness (not selected as corner case), right: selected corner case.

stance). While the influence of depth on performance seems to be linear, the influence of brightness on performance is more intricate as the evaluated AI seems to prefer a “sweet spot” of brightness values where it performs best.

The plots also elicit the intuition of the approach: Outliers (instances below the 10th percentile within their bin) are instances which are for example “well lit” or “close enough” while still being poorly detected. Hence, by relying on quantile regression models, which can be trained to predict for instance the 10th performance percentile for a given an input feature, a performance-inhibiting feature can be associated with a selection rule. Given a feature F, modelled by the quantile regression model M_F the associated selection rule is given by

$$S_F : i \mapsto M_F(i) < performance(i)$$

A combined selection rule $S_{F_1,\dots F_n}$ is defined as the selection rule which chooses instances chosen by all $S_{F_1}, \dots S_{F_n}$.

To judge the novelty of a selection rule S_F, the *information content* of the rule $I(S_F)$ can be defined as the negative logarithm of the probability of the rule choosing an instance.

$$I(S_F) := -log_2(p(S_F = \text{True}))$$

Note that the information content of a selection rule is simply the “self-information value” (used in information theory (Jones, 1979)) of a possitive selection. Self-information is often described to measure the “surprisal” of a (rare) event, which fits the intuition of corner cases as “unexpectedly bad performances”.

The *information gain* of a new selection rule S_{F_n} with respect to a set of existing selection rules

$S_{F_1,\ldots,F_{n-1}}$ can then simply be defined as this difference

$$I(S_{F_1,\ldots,F_n}) - I(S_{F_1,\ldots,F_{n-1}})$$

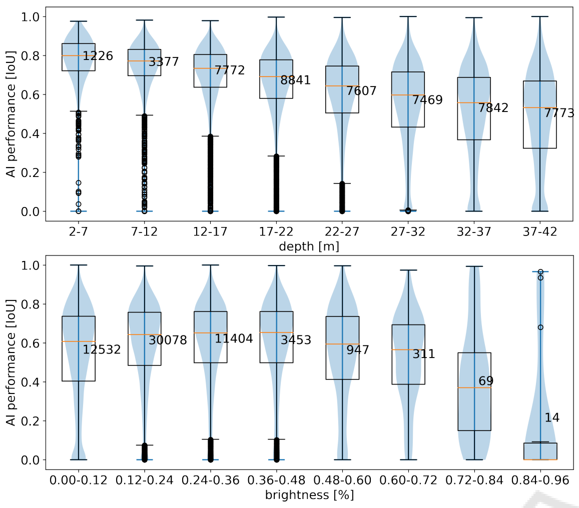

Figure 3: Violin plots with overlaid box plots of the performance inhibitors "brightness" and "depth". The data was sorted into bins of equal size in the inhibitor domain.

The performance inhibitors found while carrying out the method were classified in 4 categories: Data bugs, quantitative inhibitors, perceptual inhibitors, and situational inhibitors. As mentioned above, data bugs were caused in the data production, and encompassed mislabeled instances, completely invisible instances, instances with bounding box of width 0, etc. As this category is a bit apart from the others, it is only mentioned but not further discussed.

Quantitative inhibitors are features relating to "the amount of information" available for inference, such as the distance of the instance to the car (far-away instances are smaller), or simply the number of visible instance pixels. Perceptual inhibitors are features which are encoded by parameters known in image processing such as an instances brightness or its contrast to the background. Finally, situational inhibitors describe an instance's surrounding, such as whether and how much an instance is occluded by other instances, or whether there is vegetation in its background.

Note that quantitative inhibitors were readily available in the dataset, while all other features required (in part substantial) feature engineering. Further, it has to be noted that the inhibitor classification is not very strict. The quantitative inhibitor "pixel count", for instance, is reduced by occluding objects, yet occlusion would count as an situational inhibitor. The most impactful "contrast" measure found, measured an instances contrast against its background, which obviously changes with an instance's situation.

Table 1 summarizes the definition all found inhibitors, ordered (in general) by the iteration they were found in. Also, in the authors' experience, the discovery of earlier inhibitors eased the discovery of later inhibitors: After filtering out all instances that were just "too small" or provided "too few pixels" to be correctly identified by the AI, it became very evident that brightness and contrast might be the next candidates for exploration.

Table 1: Description of all Evaluated Inhibitors.

Quantitive Inhibitors	
depth	distance of ego car to instance measured in meters
contained_human	the fraction of a bounding box occupied by the instance.
log_pixels	the logarithm of the number of image pixels making up the instance
Perceptual Inhibitors	
contrast	the contrast of the instance measured against its background
brightness	the average luminance within a bounding box
situational Inhibitors	
fg_share_person	the fraction of the instance's contour occupied by another person
fg_share_vegetation	the fraction of the instance's contour occupied by vegetation
fg_share_car	the fraction of the instance's contour occupied by a vehicle

Gradient Boosting Quantile Regression. Decision rules were derived using Gradient Boosting Models for Quantile Regression. Gradient Boosting is a tree-based ensemble learning technique first described in (Friedman, 2001), which has found various applications, such as travel time prediction (Zhang and Haghani, 2015) and energy consumption prediction (Touzani et al., 2018). Quantile regression is a statistical analysis aiming at the prediction of a variable's quantiles (i.e. values such that a fixed portion of the observations lies *below* these values). Gradient Boosting can be used for quantile regression when an appropriate asymmetric loss function is chosen.

The 0.1-quantile was chosen for this analysis: each selection rule was trained to select 10% of the data. To train the Gradient Boosting Models, the input dataset was split into 12597 training and 46212 test instances. Evaluated on the test instances, the trained selection rules showed a good fit of to the data, selecting between 9.1% and 10.2% of the instances. Figure 4 shows the instance selection performed by S_{depth}.

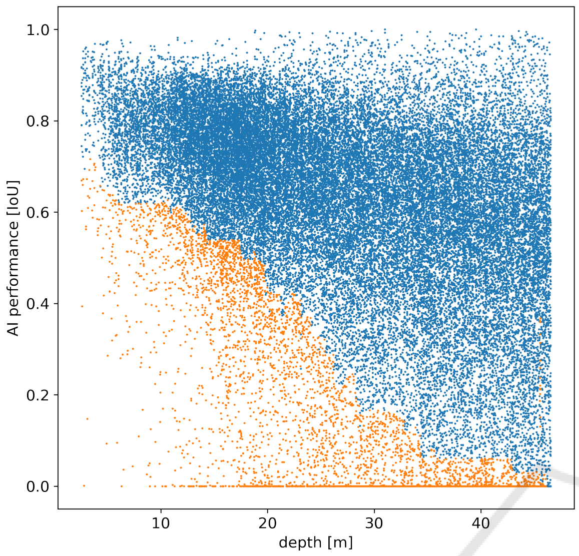

Figure 4: Scatter plot of AI-performance over instance depth. Dots represent instances, orange dots are labelled as corner case with respect to depth.

Selector Novelty and Method Termination. Each additionally applied selection rule reduces the number of selected instances, and a selection rule based on a 0.1-Quantile Regression Model would have, if perfectly fit to the data, an information content of $3.21 \approx -log_2(0.1)$. In the following, an *empirical information content* value is used by replacing the probability of choosing an instance by its observed relative frequency. In this way, a set of fitted selection rules can be associated with their combined information content value, and a new inhibitor candidate, found in the hypothesis formulation phase, can be evaluated by the extends it adds to this value.

Note that one could also use relative changes in the size of the selected dataset to make this analysis (halving the size of the current dataset increases the information content value by exactly one), but the notion of “adding n bits of information” instead of “decreasing a fraction by n percent” proved just to be the more intelligible way of displaying this information.

Another way to describe the novelty a new inhibitor presents, is to use correlation measures between inhibitors (such as that brightness naturally correlates with contrast), or set similarity measures between the regarding corner case datasets (a “brightness corner case” might also be a “contrast corner case”, etc.). Yet, all corner case are datasets are, to some degree, *expected* to be similar, because they all filter for the common property of relatively low AI-performance.

As motivated in Section 3, the (empirical) information gain of a new selection rule encodes “how much more surprising a corner case is” when including the new selection rule. This can be used to set a threshold for the termination: If no new selection rules with a information gain above a fixed threshold can be found, the method terminates with the production of a final corner case dataset (which contain up to date unexplained corner cases).

Figure 5 (left) shows the growth of information content provided by increasing the number of selection rules. The “later” rules seem to contribute less information, which might be in part due to the fact that these rules have less observations cut away from: The order of rule application might influence their information gain. To explore this effect, Figure 5 (right) shows the mean information gain added by each selection rule, when compared to a fixed number of previously applied selection rules. The figure suggests that firstly, selection rules provide less information if they are introduced later, and secondly that the situational inhibitors yield in general less informative selection rules.

Validation Experiments. The introductory example shown in Figure 2 suggests that the corner cases selected after the final iteration are indeed “interesting or “surprising”, yet such a claim needs substantiation. If the selection rules worked as intended, then distorting an input image in a way that distracts the AI, while using measures that the selection rule does not account for (i.e. without changing any of the inhibiting features), should lead to higher corner case selection rates. Similarly, if one distracts the AI with measures that *are* accounted for (for instance by darkening the image; brightness is an inhibitor), no further corner cases should be selected.

To test this hypothesis, 6 frame augmentation techniques were implemented, and hypotheses on the effects on corner case rates were made. Figure 6 gives examples for each of the applied augmentations. The augmentations were chosen such that AI-performance could be reduced by manipulating a single augmentation property. Fog_augment adds a blur effect of adjustable strength to the frame, which is implemented by the python automold library[2]. Noise_augment adds normally distributed random values to the frame’s color values, the variance

[2]https://towardsdatascience.com/automold-specialized-augmentation-library-for-autonomous-vehicles-1d085ed1f578

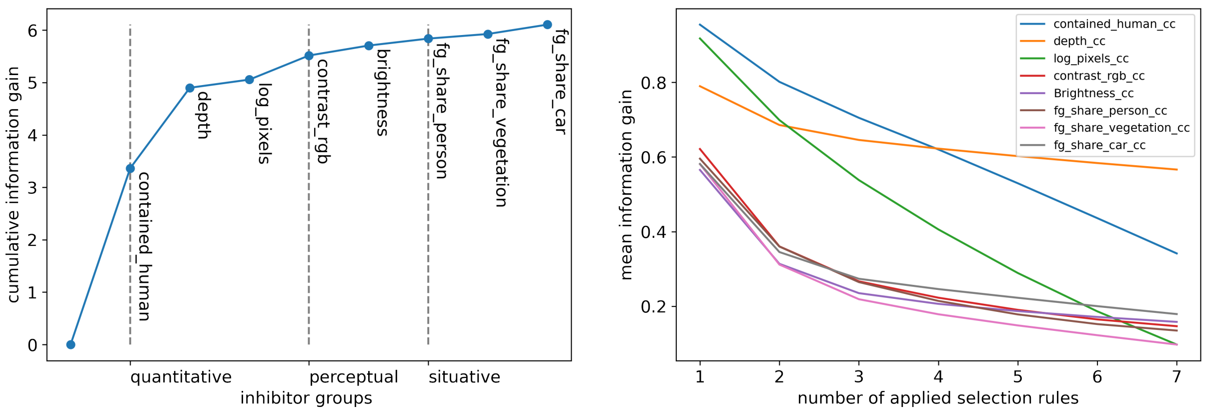

Figure 5: Information gain for different selection rules. Left: Cumulative information gain in order of discovery. Right: Mean information gain for a varying number of previously applied rules.

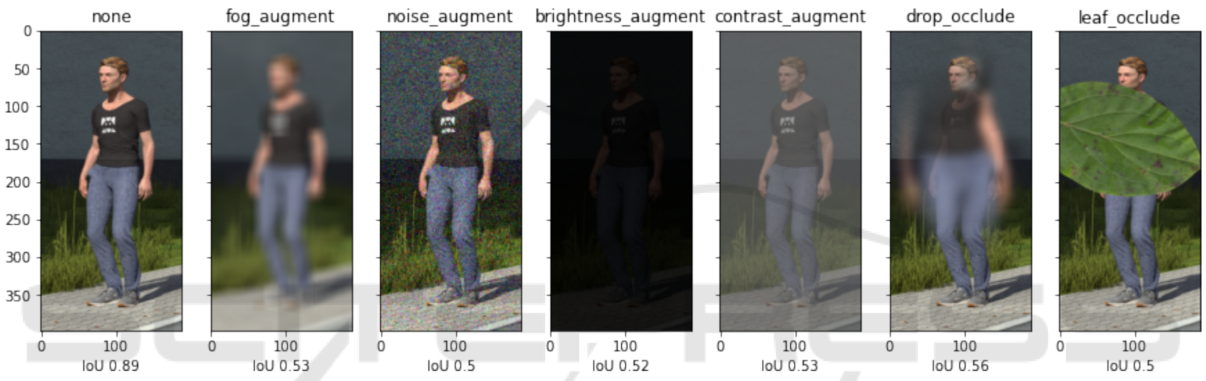

Figure 6: Examples of augmented images, using different augmentation algorithms. Left-most: The original image. All augmenters were calibrated to inhibit the AI to a similar degree.

of which can be varied to control effect size. Brightness_augment and contrast_augment are implemented using the python PIL ImageEnhance module[3], each allowing to adjust target values (0 brightness resulting in a perfectly black image, and 0 contrast resulting in a single-color image of the average brightness of the original). Drop_occlude and leaf_occlude are original implementations adding drop effects or overlaying the image with images of leafs. In both cases, effect placements and sizes can be adjusted, while effect size had the most predictable effect on AI performance.

The examples in Figure 6 show applied augmentations which were calibrated to reduce the AI performance by about 0.35% (note that due to the random nature of some of the augmentations, perfect values were not striven for). It was quickly found that, by sampling over a sufficiently large dataset, AI inhibition was a sufficiently continuous and monotone function over the augmenter-specific arguments: Altering the augmenter little resulted in little performance difference, and increasing suited arguments (leaf size, noise level, etc) resulted consistently in reduced performance.

As for hypotheses, it could be expected that leaf occlusion (which reduces the number of visible pixels) and reducing brightness would result in few selected corner cases. Further, as the net seems to be very brittle against added noise, and because such noise should not largely influence features the corner case selector accounts for, added noise should result in many corner cases. It was also expected that adding fog, adding droplet effects, and reducing contrast, should result in similar corner case rates because the results are very similar in nature.

Figure 7 shows the mean frequency of identified corner cases over mean AI inhibition for different augmenters which were "turned up" to increasingly inhibit AI performance. Note that the mean AI-performance on the sample is at 57.0%, limiting mean inhibition rates to that value. Further, the corner case rate among the not augmented instances was

[3]https://pillow.readthedocs.io/en/stable/reference/ImageEnhance.html

at 0.15%. In general, the hypotheses were found to be true. Adding noise was, as expected, very effective in producing corner cases. For high inhibitions, reducing brightness does not yield corner cases which is in line with the intuition: It is not surprising that a very dark image is unrecognizable to the AI. An interesting effect is recorded for leaf occlusions: After starting at an expectedly low slope, corner case rates rise strongly for larger inhibitions. It can only be assumed that at at a certain leaf size, the AI got distracted, unable to recognize a consistent instance behind the leaf. Note that the inhibition recorded in the example (Figure 6) of 39% required a fairly large leaf to be added to the frame.

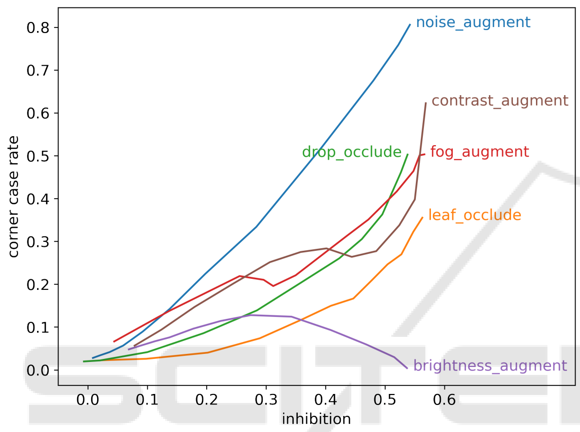

Figure 7: Corner case rates for different augmentation algorithms over their caused inhibition.

Interestingly, all augmenters produced many corner cases, surpassing the corner case baseline of 0.15% with relative small inhibitions. In other words: If one of these effects had been present in the original dataset it would have probably been discovered in one of the exploration phases.

5 DISCUSSION

In the following, the results presented in Section 4 are discussed. By using the method, a total of 8 performance-inhibiting features were found in 8 iterations, and the derived selection rules showed a relatively good fit to the data which was shown by using a train test split.

The novelty of additional selection rules was measured using their added information content (information gain). Rules that were found in later iterations showed a smaller information gain, even when correcting for selection rule order. This suggests that the method had more or less "exhausted" the dataset. A ninth iteration, which tested several other inhibitor candidates was attempted, but failed. This does not mean that all AI inhibitors were found, but it is a strong indicator, that the dataset does not allow further conclusions. To illustrate this (arguably subtle) statement, results from the final (failed) iteration are briefly illustrated: By analysing the final corner case dataset, it became apparent the selection was biased for instances wearing muted colors, with the few instances wearing bright colors being significantly less often selected in the corner case dataset. A feature measuring the mean color saturation of an instance was defined, tested, and did indeed reflect the intended purpose; yet the derived selection rule showed only a minimal information gain, the main problem being that there were simply very few "colorful instances" in the dataset.

Validation experiments testing the sensibility of the selection rules showed positive results: Effects deterring the AI, which were unaccounted for by the selection rules, did indeed yield to very high corner case selection rates. This suggests that the method would have found these effects if they *had been* present in the dataset. While this is a promising result, it would have been interesting to find more AI inhibitors *within* the original dataset. The experiments suggest that strange clothing, optical effects, and occlusions of specific body parts might deter the AI. Yet this would have required more, and different, test data including more ground truth information on the one hand, and a bigger variety of instances on the other. It has to be noted that the production of such data is, at time of writing this report, still ongoing, aiming among others specifically at these shortcomings of the data. In general, the method seems to allow for a constructive critique of a test dataset.

Apart from validating the method, the augmentation experiments resulted in other findings. Most surprising was the brittleness of the AI towards random noise, and other blurring effects, while seeming very robust against dark images. This validates, once more, the underlying assumption that, while implementing human-like perception, AI models may behave "non-intuitively" from a human perspective.

Method Limitations. A major critique that can be brought up against the method is that it can only find corner cases which *are present within the dataset*. And while the method yielded suggestions on what to include in additional datasets, for the evaluated use case, this must not hold for any future application. Yet, it is the authors' impression that such evaluations will most likely yield constructive critiques on the data quality, features to be included (and bugs to be fixed) even if applied in other domains.

The second main limitation of the method lies in its hunger for data. The application discussed above relied on a very generous test dataset, which even allowed further splitting. As any data scientist may tell, this cannot be expected in any project.

Future Work. As mentioned above, the work presented in Section 4 is still ongoing, with more data being produced and robustified AI-models being trained. The immediate future work will include re-running the method on that new input.

Besides these works, two additions to the method were motivated by the current results. Firstly, validation experiments, which were initially conducted to validate the method in general, may become a part of any application. It was by attempting such experiments, i.e. trying to *manufacture* specific corner cases, that much information on the AI was gained. Secondly, the method may reflect on the possibility to choose different performance measures. To come back to the introductory example shown in Figure 3 one more time: The well contrasted and well lit instance was wrongly classified as "building" and the dark instance was misclassified as "vegetation". There could be general rules underlying patterns of misclassification, but with a single performance measure (IoU in this case) these could not be found. A next step could involve starting anew with the same AI and dataset, yet evaluating against a different performance measure. Note that these further evaluations would be motivated by the findings of the first application.

Finally, future work will attempt to apply the method to new use cases so as to refine and rework the method.

6 CONCLUSION

In this report, a novel method for AI quality assurance was proposed. The method aims at finding "corner cases" which are loosely defined as "unexpectedly bad AI performances" in an input dataset. To this end, an iterative approach is used where each iteration produces selection rules modelling the current expectations towards the AI. Statistical measures on how to measure a rule's novelty were proposed, and suggestions on defining termination rules are given (while these will arguably depend largely on the specific use case). The method was applied to a use case in Autonomous Driving on a synthetically generated dataset, which generally validated the effectiveness of the approach. Apart from showing the method's general effectiveness, the method's strength seems to lie in producing findings on the AI and the dataset as a bi-product. The method's limitations and future work were discussed.

ACKNOWLEDGEMENTS

The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project "KI Absicherung – Safe AI for Automated Driving". The authors would like to thank the consortium for the successful cooperation, specifically partners from the Robert Bosch GmbH for providing the constrast measures, and the Intel Coorperation for providing the trained AI for evaluation.

REFERENCES

Bolte, J.-A., Bar, A., Lipinski, D., and Fingscheidt, T. (2019). Towards corner case detection for autonomous driving. In *2019 IEEE Intelligent vehicles symposium (IV)*, pages 438–445. IEEE.

Cardelli, L., Kwiatkowska, M., Laurenti, L., and Patane, A. (2019). Robustness guarantees for bayesian inference with gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7759–7768.

Chai, C., Cao, L., Li, G., Li, J., Luo, Y., and Madden, S. (2020). Human-in-the-loop outlier detection. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 19–33.

Challen, R., Denny, J., Pitt, M., Gompels, L., Edwards, T., and Tsaneva-Atanasova, K. (2019). Artificial intelligence, bias and clinical safety. *BMJ Quality & Safety*, 28(3):231–237.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR.

Freeman, C. and Beaver, I. (2019). Human-in-the-loop selection of optimal time series anomaly detection methods.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Gauch Jr, H. G., Gauch Jr, H. G., and Gauch, H. G. (2003). *Scientific method in practice*. Cambridge University Press.

Hamada, K., Ishikawa, F., Masuda, S., Myojin, T., Nishi, Y., Ogawa, H., Toku, T., Tokumoto, S., Tsuchiya, K., Ujita, Y., et al. (2020). Guidelines for quality assurance of machine learning-based artificial intelligence. In *SEKE*, pages 335–341.

Hanhirova, J., Debner, A., Hyyppä, M., and Hirvisalo, V. (2020). A machine learning environment for evaluating autonomous driving software. *arXiv preprint arXiv:2003.03576*.

Hawkins, D. M. (1980). *Identification of outliers*, volume 11. Springer.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Heaton, J. (2016). An empirical analysis of feature engineering for predictive modeling. In *SoutheastCon 2016*, pages 1–6. IEEE.

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., and Song, D. (2021). Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271.

Hosseini, H. and Poovendran, R. (2018). Semantic adversarial examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1614–1619.

Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. (2017). Safety verification of deep neural networks. In *International conference on computer aided verification*, pages 3–29. Springer.

Isele, D., Rahimi, R., Cosgun, A., Subramanian, K., and Fujimura, K. (2018). Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2034–2039. IEEE.

Islam, M. R., Das, S., Doppa, J. R., and Natarajan, S. (2018). Glad: Glocalized anomaly detection via human-in-the-loop learning. *arXiv preprint arXiv:1810.01403*.

Jones, D. S. (1979). *Elementary information theory*. Oxford University Press, USA.

Kwiatkowska, M. Z. (2019). Safety verification for deep neural networks with provable guarantees. In *30th International Conference on Concurrency Theory (CONCUR 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Lateef, F. and Ruichek, Y. (2019). Survey on semantic segmentation using deep learning techniques. *Neurocomputing*, 338:321–348.

Lenarduzzi, V., Lomio, F., Moreschini, S., Taibi, D., and Tamburri, D. A. (2021). Software quality for ai: Where we are now? In *International Conference on Software Quality*, pages 43–53. Springer.

Nortey, E. N., Pometsey, R., Asiedu, L., Iddi, S., and Mettle, F. O. (2021). Anomaly detection in health insurance claims using bayesian quantile regression. *International Journal of Mathematics and Mathematical Sciences*, 2021.

Pei, K., Cao, Y., Yang, J., and Jana, S. (2017). Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18.

Poth, A., Meyer, B., Schlicht, P., and Riel, A. (2020). Quality assurance for machine learning–an approach to function and system safeguarding. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, pages 22–29. IEEE.

Reinke, A., Tizabi, M. D., Eisenmann, M., and Maier-Hein, L. (2021). Common pitfalls and recommendations for grand challenges in medical artificial intelligence. *European Urology Focus*, 7(4):710–712.

Secci, F. and Ceccarelli, A. (2020). Rgb cameras failures and their effects in autonomous driving applications. *arXiv preprint arXiv:2008.05938*.

Tian, Y., Pei, K., Jana, S., and Ray, B. (2018). Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314.

Tjeng, V., Xiao, K., and Tedrake, R. (2017). Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*.

Touzani, S., Granderson, J., and Fernandes, S. (2018). Gradient boosting machine for modeling the energy consumption of commercial buildings. *Energy and Buildings*, 158:1533–1543.

Waldmann, E. (2018). Quantile regression: a short story on how and why. *Statistical Modelling*, 18(3-4):203–218.

Wicker, M., Huang, X., and Kwiatkowska, M. (2018). Feature-guided black-box safety testing of deep neural networks. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 408–426. Springer.

Wu, Y., Yu, T., and Hua, G. (2003). Tracking appearances with occlusions. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I. IEEE.

Xie, X., Ho, J. W., Murphy, C., Kaiser, G., Xu, B., and Chen, T. Y. (2011). Testing and validating machine learning classifiers by metamorphic testing. *Journal of Systems and Software*, 84(4):544–558.

Xu, Q., Fan, Z., Jia, W., and Jiang, C. (2019). Quantile regression neural network-based fault detection scheme for wind turbines with application to monitoring a bearing. *Wind Energy*, 22(10):1390–1401.

Zhang, Y. and Haghani, A. (2015). A gradient boosting method to improve travel time prediction. *Transportation Research Part C: Emerging Technologies*, 58:308–324.

Zheng, A. and Casari, A. (2018). *Feature engineering for machine learning: principles and techniques for data scientists.* " O'Reilly Media, Inc.".