Case study on the use of the SafeML approach in training autonomous driving vehicles [⋆]

Matthias Bergler[1][0000−0002−2941−7948], Ramin Tavakoli Kolagari[1][0000−0002−7470−3767], and Kristina Lundqvist[2][0000−0003−0904−3712]

[1] Technische Hochschule Nürnberg, Keßlerplatz 12, 90489 Nürnberg, Germany
[2] Mälardalen University Sweden, Kögskoleplan 1, 722 20 Västeras, Sweden

Abstract. The development quality for the control software for autonomous vehicles is rapidly progressing, so that the control units in the field generally perform very reliably. Nevertheless, fatal misjudgments occasionally occur putting people at risk: such as the recent accident in which a Tesla vehicle in Autopilot mode rammed a police vehicle. Since the object recognition software which is a part of the control software is based on machine learning (ML) algorithms at its core, one can distinguish a training phase from a deployment phase of the software. In this paper we investigate to what extent the deployment phase has an impact on the robustness and reliability of the software; because just as traditional, software based on ML degrades with time. A widely known effect is the so-called concept drift: in this case, one finds that the deployment conditions in the field have changed and the software, based on the outdated training data, no longer responds adequately to the current field situation. In a previous research paper, we developed the SafeML approach with colleagues from the University of Hull, where datasets are compared for their statistical distance measures. In doing so, we detected that for simple, benchmark data, the statistical distance correlates with the classification accuracy in the field. The contribution of this paper is to analyze the applicability of the SafeML approach to complex, multi-dimensional data used in autonomous driving. In our analysis, we found that the SafeML approach can be used for this data as well. In practice, this would mean that a vehicle could constantly check itself and detect concept drift situation early.

Keywords: Automotive · Safety · SafeML · Machine Learning · Autonomous Driving.

1 Introduction

In movie and television, especially in the science fiction genre, people's futuristic dreams, desires and fears become reality. But these stories do not only provide entertainment, but they also provide research incentives for science. Devices such as smartphones or medical devices such as CT or MRI are based on devices from

[⋆] Supported by organization MDU.

the Star Trek or other Sci-Fi series [6]. Many of these stories also show futuristic cities with self-driving modes of transport in which passengers can pursue whatever activities they want. More and more vehicle manufacturers, such as Tesla, BMW, Mercedes etc., are therefore also dealing with the topic of autonomous driving. Tesla in particular is known for its advanced technology. Some successes and failures have already been recorded in the media. Many of the results have shown that autonomous vehicles of the highest level, i.e., completely without a driver, are possible, and semi-autonomous vehicles are already being tested in traffic. To make this possible, data from several cameras and LIDAR radar systems are used to observe the vehicle environment. Internal processing units evaluate these images of the surroundings in the form of images and sensor data and use pre-trained ML algorithms to calculate the necessary actions that have to be taken in order to get through the traffic accident-free. Unfortunately, these algorithms are not yet 100 % reliable and the tests in traffic repeatedly lead to malfunctions or failures. For example, obstacles in the form of a truck standing sideways were overlooked [4], or, as recently (as of April 2021 [11]), the vehicle started by itself without a driver. These incidents illustrate how big the technological leap to a perfectly autonomous vehicle is. These technical deficiencies are partly software-related and partly hardware-related and must therefore be viewed differently in terms of their robustness. But not only the robustness plays a major role, but also the security against the manipulation of such systems in order to consciously cause accidents or, for example, people kidnapping [15][16]. Examples worldwide show how control over vehicles can be taken over by the simplest means, for example by manipulating the infotainment system [14]. In combination with an autonomous vehicle, this can lead to devastating results. In this paper we analyse whether the Aslansefat et al. [1] approach based on distance metrics is still practicable even with a complex data set from the field of autonomous driving and is suitable for detecting a concept drift at an early stage.

2 Robustness and Safety of Machine Learning Algorithms in Autonomous Driving

The robustness of the functionality of an ML algorithm is essential for autonomous driving. Since there is no longer a driver at the highest level of autonomy, the algorithm must work reliably and without errors. But even with partial automation, it is important to have a robust algorithm so that the driver can be warned in a acceptable timeframe as soon as he has to take the steering wheel again. Since the systems used up to now are trained offline, it is not possible to relearn the algorithms based on the situation. It can very well happen that an algorithm makes the wrong decisions out of uncertainty. When analyzing the robustness of systems based on machine learning processes, the phenomenon of "concept drift" has increasingly occurred, especially in systems that have been in the field for a long time without retraining. This occurs increasingly when the input data differ too much from the original training data over time (see Fig.1).

This mostly happens due to unforeseen circumstances and cannot be prevented as seen in “An Overview of Concept Drift Applications” by Žliobaitė I. et al. [10]. An example of this would be if an algorithm for autonomous driving was only trained with data from city traffic but is used without additional training in more rural areas and must now recognize not only people, but also animals and fallen trees as obstacles. An additional example of this would be estimates and analyzes of customer buying behavior, which can change unexpectedly because of an economic crisis. Therefore, in our research we are concerned with how this phenomenon can be recognized at an early stage and warned about it. To achieve this, statistical distance measurements based on probability density functions and the characterization of data sets are dealt with in more detail. The literature shows that there is a connection between the classification results of neural networks and probability distributions of data sets as shown in Aslansefat K. et al. in “SafeML: Safety Monitoring of Machine Learning Classifiers through Statistical Difference Measure” [1], in contrast to other methods, such as support vector machines or density estimates as seen in “Detecting Concept Drift with Support Vector Machines” by Klinkenberg R. and Joachims T. [7] or “Adaptive concept drift detection” by Dries A. and Rückert U. [3].

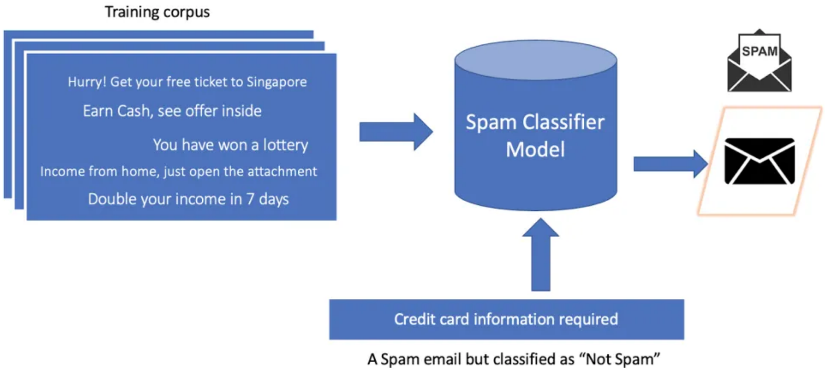

Training corpus
Hurry! Get your free ticket to Singapore
Earn Cash, see offer inside
You have won a lottery
Income from home, just open the attachment
Double your income in 7 days
Spam Classifier Model
SPAM
Credit card information required
A Spam email but classified as “Not Spam”

Fig. 1: A concept drift has the consequence that the model trained on historical data incorrectly interprets and processes new input data. For example, a spam filter is tricked out with new mail subjects [2].

3 SafeML Approach

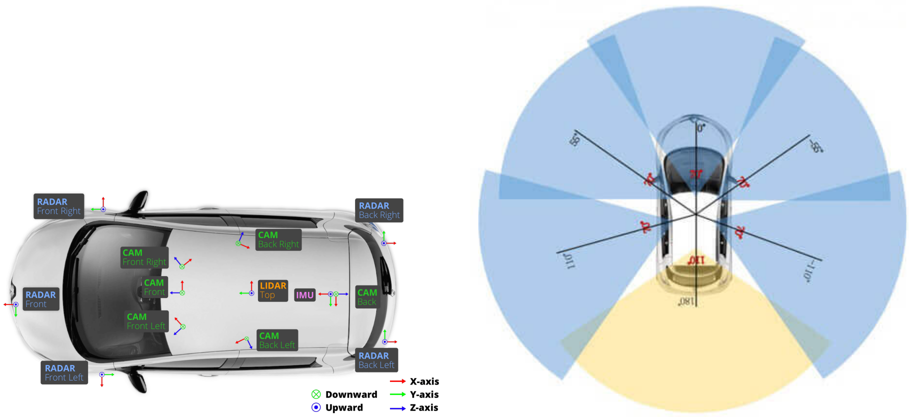
When researching the early detection of a concept drift, we encountered so-called SafeML Approach of Aslansefat et al. [1]. Based on the preliminary work by Aslansefat et al. we have recreated the SafeML method. For this we started with the training phase and pre-trained our Alexnet-based CNN (see Section 5.2) with the help of a training and validation data set. Since we want to check the correlation between the accuracy and the distance measurements on more complex image data sets and not just data sets designed for benchmarks, we decided on the NuImages data set from the NuScenes group (see sSection 4). This data set was specially developed for training autonomous vehicles. After the training, we determined and saved the results of the probability density function (PDF) as well as the statistical parameters of the Empirical Cumulative Distance Functions (ECDF) for each class of the training data set. In the application phase, we then classified our generated test data set using our model and compared the accuracy with the result from the training. Here, too, we determine the PDF and the statistical parameters based on the ECDFs of each class. The results from both phases are then used to determine the statistical distance. For this we used the algorithms recommended from the original paper to measure the statistical distance (see Section 5.3). If the statistical distance turns out to be small, the classification results are to be classified as reliable and the system can theoretically continue to work autonomously. Since we have our data labeled, we can also check whether the correlation between the accuracy of the classifier and the statistical distance is still guaranteed even with more complex data sets.

4 NuImage data set

The NuScenes team founded by Oscar Beijbom has set itself the task of developing a complete data set for autonomous vehicles [12]. In March 2019, the 26-strong team managed to publish the first version of the data set. The data set is publicly accessible and is intended to help researchers cope with challenging situations in urban areas for autonomous vehicles. The data set uses a large number of sensor and image recordings from an autonomously driving vehicle. A rotating LIDAR system, five long-range RADAR systems and six cameras were attached to the two Renault Zoe(see Fig. 2a). The cameras were installed in such a way that an all-round view is possible (see Fig. 2b). The two vehicles drove through the two cities of Sydney and Boston. For this experiment, however, the NuScenes data set with its 1,000 scenes was not used, but the NuImage data set. The NuImage data set includes 93,000 fully annotated scenes, with over 800,000 two-dimensional foreground annotations (see Fig. 3). There are 23 classes (see full list at [12] that include both pedestrians and vehicles, as well as roadblocks. In addition, the annotations were supplemented by further metadata. These include information such as whether a driver is sitting in the vehicle or parking, or whether a passer-by is currently moving or standing waiting at a bus stop.

5 Experimental Setup

To carry out the experiment, we first needed to organize the data. Therefore the data set had to be loaded and the required annotations in the scene had to be detected and cut out. The tiles thus obtained were then re-saved to folders



(a) This figure shows the setup of the sensors and cameras of the cars.

(b) By attaching the cameras, an all-round view is possible.

Fig. 2: Technical structure of the recording vehicles.

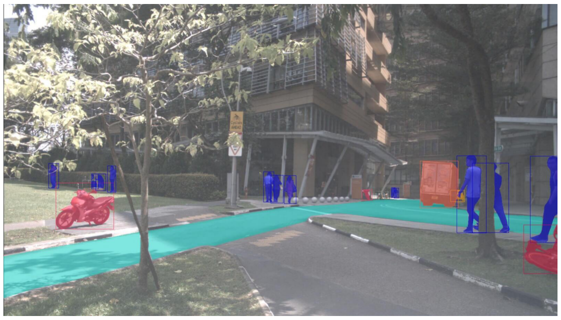


Fig. 3: Example Image of a NuImages scene with annotated objects.

with their respective class names. The recordings obtained in this way were then divided into three subsets for training a classifier. Half of the tiles were combined into a training data set and the other half again divided into a validation and training data set. With the help of the received training and validation subsets, a CNN based on Alexnet [9] was then trained for classification. The model obtained in this way is then used to classify the test data set. If the classification of the test data set shows sufficient accuracy, the distances between the training and the test data set are calculated using the distance metrics. If the classification accuracy turns out to be too low, the hyperparameters were optimized so that a higher accuracy could be achieved. A high performance computer was used to conduct the experiments. This is with two Intel (R) Xeon (R) Silver 4114 (2.20 Ghz, 2,195 Mhz, 10 cores) processors, 192 GB DDR4-2666 rg ECC memory, a SAMSUNG SSD MZVLB512HAJQ-00000 500 GB, as well as 2 NVIDIA Quadro RTX 5000 (3,072 CUDA processing units, 384 Tensor processing units).

5.1 Data set Generation

First, we read in every single image in the NuImages data set using the supplied Python package. Since the NuImages data set provides metadata for each scene with annotations and other information, the next step was to read them out. After we loaded the annotations from the metadata, we were able to crop the image tiles from the scenes using the scene coordinates contained therein. We then packed the image data received in folders based on the description of the annotation classes. The NuImages data set is divided into three sub-data sets (train, validate and test) in advance. Since there is no annotation for the test data set, we decided to halve the validation data set and generate our own test data set. The data set was split up randomly. This is required later to calculate the statistical distance between the training data set and the test data set.

5.2 Alexnet with Spartial Pyramid Pooling Layer

To classify the image details obtained from the scenes, we first used the classic AlexNet by Krizhevsky et al. [9] used. However, the classic version of AlexNet only allows a uniform tile size of 227x227 pixels. This is because the fully connected layer is at the end of the network. Since the annotated tiles from the scenes have different dimensions and we wanted to keep the original resolution of the images in order not to cause scaling problems, we decided to use a spatial pyramid pooling layer as in He et. al [5] to be inserted after the last convolutional layer. This additional layer enables us to use image tiles in different dimensions instead of the 227x277 dimensioned images used in the paper. The spatial pyramid pooling layer generates a fixed output size (see Fig.4) at the end of the convolutional layer, which can then be processed by the fully connected layers. However, this step only allows a batch size of 1, as the individual image tiles cannot be grouped together uniformly. This is done at the expense of running time.

5.3 Distance Measurements

After our model had achieved a sufficiently high classification rate of 99.7 percent, the statistical distance between the respective classes in the training and test data set was then calculated. For this purpose we have the original by Aslansefat, K. et al. recommended calculation metrics (see Fig.5). Here we have mainly focused on the empirical cumulative distribution functions. These include:

- **Kolmogorov-Smirnov Distance**: Calculation of the maximum distance between the two ECDFs of the compared classes.
- **Kuiper Distance**: calculation and addition of the two maximum distances of the compared classes.
- **Cramer-Von Mises Distances**: Calculation and summation of the difference between several points in an interval between two ECDFs.

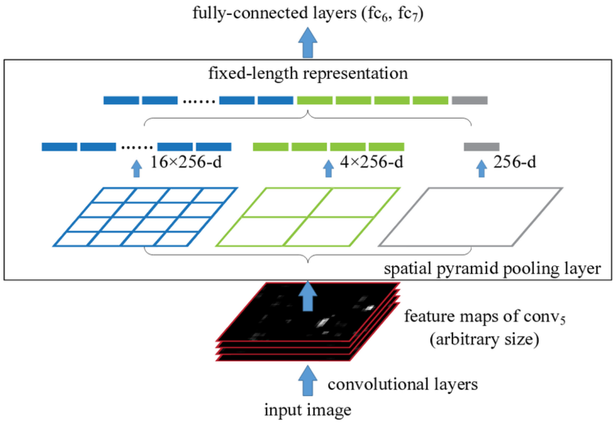

fully-connected layers (fc_6, fc_7)
fixed-length representation
16×256-d
4×256-d
256-d
spatial pyramid pooling layer
feature maps of $conv_5$
(arbitrary size)
convolutional layers
input image

Fig. 4: Example of a spatial pyramid pooling layer. 256 describes the number of filters in the last convolutional layer conv5 [5].

- **Anderson-Darling Distance**: Calculation as for Cramer-Von Misses, but the individual differences are normalized beforehand with the standard deviation.
- **Wasserstein Distance**: Calculation of the area between two ECDFs.

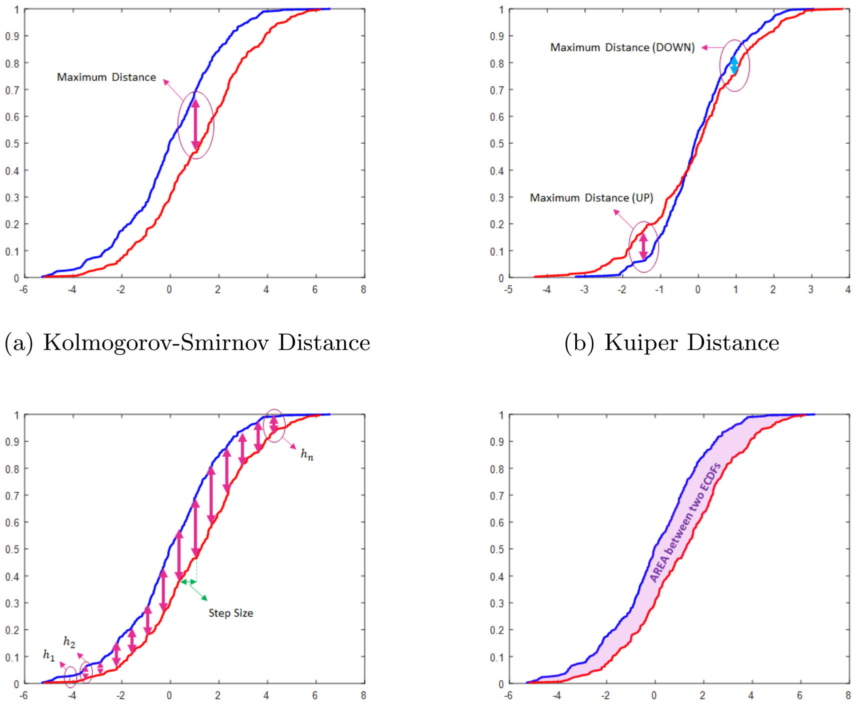
This calculations were made for each class.

6 Results

In this section we present the results obtained after our application phase. The respective classification results of the application phase are listed in Table 1. The table shows the class-specific accuracy that was obtained during the application phase. The classes were taken from [12] and mapped to the numbers 1-23 in the order listed. For better visualization we decided not to include the class based results for the statistical distance measure, but visualize the trend in a separate figure 6. This representation allows us to better visualize the relationship between the classification accuracy and the statistical distance. In figure 6 we see the associated correlation with the associated distance measures for each class and for the specific distance measurement.

7 Discussion

The results of our study have shown that there is indeed a correlation between the classification accuracy and the calculated statistical distance based on ECDFs (see Fig. 6). It is particularly noticeable that a similar relationship can be determined for the Anderson-Darling (see Fig. 6e) and Wasserstein distance (see Fig.



(a) Kolmogorov-Smirnov Distance

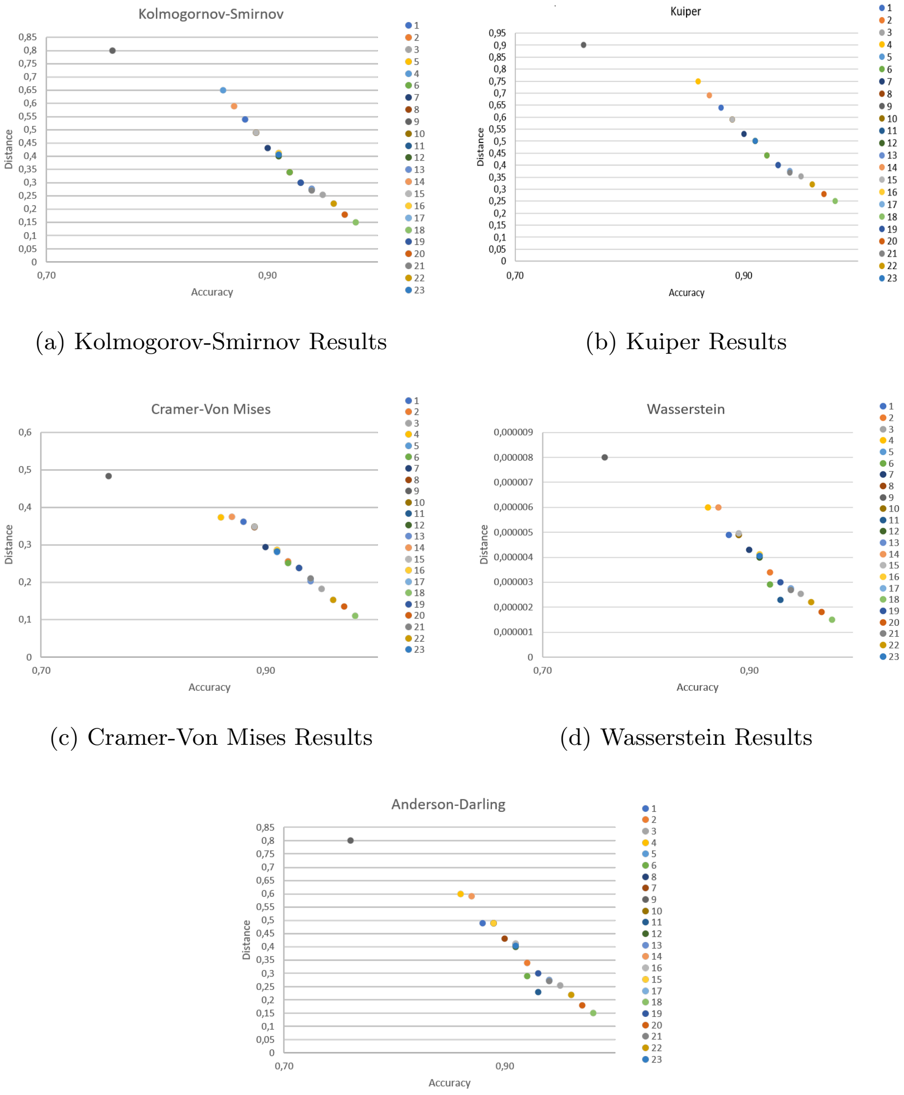
(b) Kuiper Distance

(c) Cramer-Von Mises Distance

(d) Wasserstein Distance

Fig. 5: Illustrations of the 5 ECDFs used. The Anderson-Darling Distance was not shown because it is illustrated identically to the Cramer-Von Mises Distance and only differs due to the normalization [8].

6d), as well as for the Kuiper (see Fig. 6b) and Kolmogorov-Smirnov distance (see Fig. 6a) between the statistical distance and the classification accuracy. Class 9 is also noticeable in the tables, as it drifts far from the other classification results and distance measurements. Since this class is the “movable_object.barrier” class, we took a closer look at it. It is noticeable here that the tiles contained therein contain different types of a barrier and therefore a high degree of variance in the image content. This also explains the lower classification result and the higher statistical distance. Here one could consider dividing the class again, since a logical assignment for humans is not necessarily logical for ML algorithms. In all five distance measurements, it can be seen that the statistical distance between the training and test data set is greater for each class, even if the accuracy of the classification is better. Thus, the initial thesis that there is a correlation between the classification accuracy and the statistical distance between two data sets could also be demonstrated on a complex data set. When training the classifier, we made a conscious decision to modify AlexNet using a



(a) Kolmogorov-Smirnov Results

(b) Kuiper Results

(c) Cramer-Von Mises Results

(d) Wasserstein Results

(e) Anderson-Darling Results

Fig. 6: Results for the 5 ECDFs based on Classes. We calculated the statistical distance from the training with the test data set in comparision with the classification accuracy.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	**Average**
Accuracy	.88	.92	.95	.86	.91	.92	.90	.89	.76	.89	.93	.91	.94	.87	.89	.91	.94	.98	.93	.97	.94	.96	.91	**.9113**

Table 1: Classification results in the application phase of the different classes in the test data set. The order of the classes is the same as listed at [12]. We calculated the Accuracy for each class during the application phase. The results for the statistical distance is shown in Fig. 6 for the sake of clarity.

spatial pyramid pooling layer in order to retain the natural scaling of the image tiles obtained from the scenes. However, this had the negative consequence that both the training in the training phase and the classification in the application phase had a long calculation time. This was due to the fact that the image tiles had to be read in individually, as the different dimensions of the tiles meant that efficient batch processing was not possible. Since the long-term goal is to develop an application to detect the concept drift during the runtime of an autonomously driving vehicle, the calculation time must be shortened. Therefore we will look for another possibility to maintain the dimension for further experiments with the NuImages data set.

8 Conclusion

Our experiments have shown that the Aslansefat et al. approach can also be reliably applied to a complex data set for training autonomously driving vehicles. This enables completely new approaches in the development of suitable algorithms in the field of autonomous driving, since here, during the execution of the algorithm, it is possible to check when it is necessary for the driver to take control of the vehicle again. However, the speed of the algorithm still has to be improved for this, since the runtime of the hardware used is not yet capable of real-time. However, there was no focus on this in the current implementation. Furthermore, it will be necessary to determine suitable threshold values and to evaluate when the distance between the training and application data sets becomes too great. For this it is certainly useful to characterize data sets as in “Effects of data set characteristics on the performance of feature selection techniques” by Oreski et al [13]. In this way, data sets that differ greatly or only partially from one another can be determined and compared with one another using the SafeML method. An applicable threshold value could then be derived from this. In addition, we found that conclusions about the appropriate modularization of the data sets can apparently be drawn on the basis of the statistical distance. Further experiments will therefore be carried out in this direction.

References

1. Aslansefat, K., Sorokos, I., Whiting, D., Kolagari, R.T., Papadopoulos, Y.: Safeml: Safety monitoring of machine learning classifiers through statistical difference measure (2020)
2. Das, S.: Best practices for dealing with concept drift (2021), https://neptune.ai/blog/concept-drift-best-practices
3. Dries, A., Rückert, U.: Adaptive concept drift detection. Statistical Analysis and Data Mining **2**, 311 – 327 (12 2009). https://doi.org/10.1002/sam.10054
4. Forbes, B.T.: Tesla in taiwan crashes directly into overturned truck, ignores pedestrian, with autopilot on (2020), https://www.forbes.com/sites/bradtempleton/2020/06/02/tesla-in-taiwan-crashes-directly-into-overturned-truck-ignores-pedestrian-with-autopilot-on/?sh=3ec11c5758e5
5. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. Lecture Notes in Computer Science p. 346–361 (2014)
6. Health, U.: 5 real-life medical devices inspired by science fiction (2020), https://www.usfhealthonline.com/resources/healthcare/5-real-life-medical-devices-inspired-by-science-fiction/
7. Klinkenberg, R., Joachims, T.: Detecting concept drift with support vector machines. Proceedings of ICML (05 2000)
8. Koorosh, A.: How to make your classifier safe (2020), https://towardsdatascience.com/how-to-make-your-classifier-safe-46d55f39f1ad
9. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25**, 1097–1105 (2012)
10. Žliobaitė, I., Pechenizkiy, M., Gama, J.: An Overview of Concept Drift Applications, vol. 16, pp. 91–114 (01 2016)
11. News, C.G.B.: Tesla's autopilot 'tricked' to operate without driver (2021), https://www.bbc.com/news/technology-56854417
12. NuScenes: Nuscenes by motional (2020), https://www.nuscenes.org/
13. Oreski, D., Oreski, S., Klicek, B.: Effects of dataset characteristics on the performance of feature selection techniques. Applied Soft Computing **52** (12 2016). https://doi.org/10.1016/j.asoc.2016.12.023
14. Securityweek, E.K.: Tesla car hacked remotely from drone via zero-click exploit (2021), https://www.securityweek.com/tesla-car-hacked-remotely-drone-zero-click-exploit
15. Wired, A.G.: Hackers remotely kill a jeep on the highway—with me in it (2015), https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/
16. Wired, A.G.: The jeep hackers are back to prove car hacking can get much worse (2016), https://www.wired.com/2016/08/jeep-hackers-return-high-speed-steering-acceleration-hacks/