TESTING THE DEPENDABILITY OF A NEURAL NETWORK SPECIFICALLY DESIGNED FOR THE SAFETY CRITICAL AUTONOMOUS DRIVING SYSTEMS.

Software Dependability 2021

Submitted To: Prof. Fabio Palomba
Submitted by: Shoaib Ali & Syed Shakir Imran

Contents

ABSTRACT:

Can the neural networks be tested and checked in the similar fashion in which we test software and various applications. In this project we have presented and demonstrated a neural network dependability kit which is a toolbox to check whether the neural network used for autonomous driving system is safe or not. We have followed the step by step approach in order to judge the reliability of neural networks, the toolkit covers these scientific approaches namely a) dependability metrics to check the uncertainties in the project b) checking any undesired behavior and c) runtime monitoring to check if decision made by a neural network is corresponding to the trained data.

INTRODUCTION:

Neural networks have been widely used in the development of autonomous driving systems in recent years, with applications in perception, decision-making, and even end-to-end scenarios. Problems during operation, such as failed pedestrian detection, may contribute to dangerous behaviors because these systems are safety-critical in nature. Importantly, the main cause of these undesirable behaviors may be only in the data-driven engineering process, such as unanticipated consequences of function extrapolation between correctly categorized training data, rather than in hardware or software programming flaws. We propose NN-dependability-kit, a toolbox for data-driven neural network building in safety-critical domains, in this project. The goal is to show that critical phases of the product life cycle, including as data collection, training and validation, testing and generalization, and operation, have reduced uncertainty. NN-dependability-kit is based on (a) the introduction of novel dependability metrics to indicate the reduction of uncertainties in the engineering life cycle, (b) the use of a formal reasoning engine to ensure that generalization does not result in undesired behaviors, and (c) the use of runtime neuron activation pattern monitoring to reason whether a neural network decision in operation time is supported by prior similarities in the training data. In terms of related work, the findings are consistent with recent software engineering and formal methods research efforts aimed at providing provable guarantees over neural networks or testing neural networks. The NN-dependability-kit static analysis engine is used to formally analyze neural networks.

STRUCTURE:

We have used four packages under NN-dependability, namely

Basic: reader for models as well as intermediate representation of NN (for verification purposes).

Metrics: compute dependability metrics for neural networks.

Atg: automatic test case generation to improve the metrics.

Formal: formal verification (static analysis, constraint solving) of neural networks.

Rv: runtime monitoring of neural networks.

DATASET USED:

We have used four types of different datasets to examine the whole process

GSTRB DATASET:

German Traffic Sign Recognition Dataset (GTSRB) is an image classification dataset. The images are photos of traffic signs. The images are classified into 43 classes. The training set contains 39209 labeled images and the test set contains 12630 images.

MNIST DATASET:

The MNIST database (Modified National Institute of Standards and Technology database) is a massive dataset of handwritten digits that is often used for image processing system training. In the field of machine learning, the database is also commonly utilized for training and testing. The MNIST database contains 60,000 training images and 10,000 testing images.

KITTI DATASET:

The KITTI dataset was released with the intent of advancing autonomous driving research with a novel set of real-world computer vision benchmarks. It was developed by Karlruhe Institute of Technology and Toyota Institute of Technology that's why it's named KITTI, the dataset is mainly used in the field of robotics and autonomous driving systems. It consists of the 6 hours of traffic scenarios at 10– 100 Hz using a variety of sensor modalities such as high-resolution color and

grayscale stereo cameras, a Velodyne 3D laser scanner and a high-precision GPS/IMU inertial navigation system. The data is available in raw format.

PASCAL VOC DATASET FOR THE SSD MODULE:

The PASCAL Visual Object Classes (VOC) 2012 dataset contains 20 object categories including vehicles, household, animals, and other: airplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, TV/monitor, bird, cat, cow, dog, horse, sheep, and person. We have only used those categories which come under the domain of autonomous driving system.

METHODOLOGY

The whole project is based on the metrics based verification of the four attributes of the Neural Network , if the set of metrics go through these attributes only then we can say the testing is done in thorough procedure the names and the brief description of these attributes are defined as follow:

1. **Robustness**

   It covers those aspects of the neural network how the neural network would behave under various effects such as perturbation and distortion.
2. **Interpretability**

   Covers the terms of learning phase of the Neural Network
3. **Completeness**

   Assures whether dataset used has covered all the important scenarios or not.
4. **Correctness**

   Defines if a NN has able to perform the perception based tasks without any error or not

To perform the testing and check the reliability of the project we have developed 10 jupyter notebook files by keeping in mind the above mentioned attributes, to test, verify and validate different kind of scenarios, we have categorized these files under the following categories:

METRICS & TEST CASE GENERATION:

1. GTSRB_AdditionalMetrics.ipynb.
2. GTSRB_Neuron2ProjectionCoverage_TestGen.ipynb.
3. Scenario_Coverage_A9.ipynb.
4. KITTI_Scenario_Coverage.ipynb.
5. MNIST_Neuron2ProjectionCoverage_TestGen.ipynb.
6. SSD_InterpretationPrecision.ipynb.
7. Scenario_Coverage_Simple.ipynb.

FORMAL VERIFICATION:

1. TargetVehicleProcessingNetwork_FormalVerification.ipynb

RUNTIME MONITORING:

1. GTSRB_RuntimeMonitoring.ipynb
2. MNIST_RuntimeMonitoring.ipynb

GTSRB ADDITIONAL METRICS:

In this file we have heavily focused on perturbation loss, the perturbation loss metric has a set of predetermined perturbation directions that indicate distinct effects and a set of specified quantities that represent different intensities that can be adjusted if necessary. The purpose of this statistic is to see if the neural network under investigation can still perform well with (slightly) skewed input.

To compute the perturbation loss metric, simply add input images and corresponding labels both as numpy arrays to the metric object. There are multiple options for visualizing the metric, such as computing the maximum or average drop of output probability.

In the second part of this metrics file we have worked on the neuron activation pattern metric which compares the activation patterns of input data that are grouped into the same class. Based on evaluating a user-specified close-to-output neuron layer, the similarity of activation patterns is further aggregated into histograms representing the number of neurons being activated we further explained the working of the histogram in the results section.

GSTRB NEURON 2 PROJECTION COVERAGE:

Similar to the neuron activity pattern metric, in this file we have started the work by specifying the number of neurons to be monitored in each layer, which we have set the limit to 84 After that, provide the metric class 2D numpy arrays with numNeurons as the first dimension and the batch size of the test set as the second dimension which we set at the size of 64. The number of classes are 43 and the total number of epochs are 5.

To have on off deactivation combination of the neurons to be tested we have set k = 2

When k = 2, to achieve full coverage, it is mandatory that the set of test cases to be able to let every neuron pair to cover four situations, named as (activate, activate), (activate, deactivate), (deactivate, activate), and (deactivate, deactivate).

SCENARIO COVERAGE A9:

We're interested in knowing how complete the test data is in this metric (more generally, it can also be used to measure the completeness of training data). Aside from taking a haphazard and unsystematic approach to driving as many kilometers as possible, we intentionally selected a section of A9 to generate as many permutations as feasible for the same data.

When it comes to understanding the network's performance, the completeness of data is critical. In the above term completeness it means in terms of ensuring that the data used in training has possibly covered all important scenarios.

Since it’s a scenario coverage module so it also covers all the scenarios which do not happen in reality for the sake of real life behavior of the NN.

SCENARIO BASED COVERAGE OVER KITTI DATASET:

Scenario k-projection coverage evaluates how the input data (for training or testing purposes) are spread suitably, with suitability defined by the concept of k-projection, a concept that has long been used in software testing under a different name (combinatorial testing). The jupyter notebook KITTI Scenario Coverage.ipynb contains an example of utilizing such an example.

MNIST NEURON 2 PROJECTION COVERAGE

Similarly to the neuron activity pattern metric that we have followed GSTRB neuron 2 projection coverage, we began this file by defining the number of neurons to be tracked in each layer, with an upper limit of 40 and the batch size of the test set, which we set to 64. The overall number of epochs is 5, and there are 10 classes since it's a MNIST dataset, the learning rate was 0.001.

SSD INTERPRETATION PRECISION

The interpretation precision metric is used to determine if a NN for image classification or object detection makes the correct judgment. For example, the measure may suggest that a particular class of objects is predominantly identified by its surroundings, possibly because it only exists in similar environments in the training and validation data.

For the sake of interpretation precision of the images, here we are using the SSD (Single Shot Detector) Tensorflow which is a state of the art object detection tool which has the highest precision over all other object detection tools.

Here we have used VOC2012 dataset which is supported by SSD and particularly we have used traffic images in which a bus, pedestrian and a car is apparent, after the objects in the images are detected through SSD then we use second approach to verify whether the SSD is accurate or not, for that we captured the heat maps or saliency maps of the images and compared those images with the one detected through SSD.

Now for judging that we used a metric called interpretation precision metric depicted as **Minterpret** in the graphs in the results section, in order to judge if a

NN for object detection makes its decision on the correct aspect of the image or not.

SCENARIO COVERAGE (SIMPLE)

As its apparent from the name, in this file we have covered those aspects or scenarios which are simple or which can occur under the real life example, so for the complete testing of the scenario coverage we have differentiated and we have created two files for the testing to check the behavior of the project under both circumstances one under the unreal scenarios and the other one with the real life scenario.

TARGET VEHICLE PROCESSING THROUGH FORMAL VERIFICATION

For suppose, attributes of a neural network that selects the target vehicle for an adaptive cruise control (ACC) system to follow can be explicitly verified. In general, photos from a vehicle's front-facing camera to detect other vehicles

Boundary boxes for vehicles are to determine the ego-lane boundaries (ego-lane detection). These two modules' outputs are fed into the third module.

As a neural-network, its termed target vehicle selection based classifier that returns either the bounding box index or the bounding box index a particular class for the target vehicle.

GSTRB RUNTIME MONITORING

Understanding if a trained neural network operates appropriately "beyond its comfort zone" is a major challenge for deploying neural networks in such a safety-critical application. This appears to be the case when the network needs to extrapolate greatly from what it learns or evokes from the training data because similar data did not exist during the training process. We solve this problem by recording neuron activation patterns for close-to-output neural network layers for all accurately predicted data utilized in the training process after the training process is completed. Recent techniques to analyzing neural networks have shown that neurons in close-to-output layers in general encode high-level characteristics.

In GSTRB runtime monitoring we considered monitoring the stop sign class, where apart from monitoring the complete neurons (a total of 84 neurons), we also set the hyper parameters where the number of classes where 43 learning rate was 0.001 and batch size was 64 with 5 epochs.

In the second part we have taken a similar approach as we did in MNIST runtime monitoring by recording the on off pattern monitoring so we can test the images on the similar pattern later on

MNIST RUNTIME MONITORING

Runtime monitoring is an essential part in the testing of the neural network since it can judge whether the decision made by neural network is similar to the data in the training phase, if we go in depth the runtime monitoring phase is like after we do the normal training of the dataset, a monitor is created by serving the training data to the network again in order to save the neuron activation patterns, then the runtime monitor checks whether a decision made by the NN over an input is similar to the patterns which we saved earlier, if the patterns do not match it will simply generate the warning.

Now as we know we have done the runtime monitoring with two image classification datasets here we will discuss the working of the runtime monitoring with the MNIST dataset , starting with setting the hyper parameters we have set number of epochs to be 5 , batch size of the images to 64 and we have monitored around 40 neurons, at first we trained the data of all the images under normal scenario and we got 99.34 % accuracy, since we have got the state in which data was trained so we will save the monitor of neuron activation patterns in a pickle file , now we will run the saved monitor and test it on the set of 10,000 images , we got the results which would have been very critical for the system to overcome these issues we enlarged the size of the pattern and got the results which were out of the critical point

RESULTS:

MNIST RUNTIME MONITORING

The accuracy of the network on the 10000 test images is 98.81% the result for the out-of-activation pattern on the 10000 test images is 7.66 %.

Plus the result for the Out-of-activation pattern & misclassified is 10.704960835509139% so keeping that in mind if we take out the ratio of the misclassified patterns with the total minus correct patterns we will get around 68% out of activation which is quite a lot so to overcome this we have enlarged the size of the memorized pattern in this way we can overcome this issue after

that were ran the runtime monitoring and these results Accuracy of the network on the 10000 test images is now 98.81 %, accuracy for the out-of-activation pattern on the 10000 test images has reduced to 2.01 % and the accuracy for out-of-activation pattern & misclassified pattern 21.890547263681594 %. The ratio of the misclassified patterns with the total minus correct patterns has drastically reduced to 36.974 %.

SSD INTERPRETATION PRECISION

As we have mentioned in earlier Minterpret is the metric which will measure the precision. We have displayed the Minterpret results by displaying it in the following graphs:

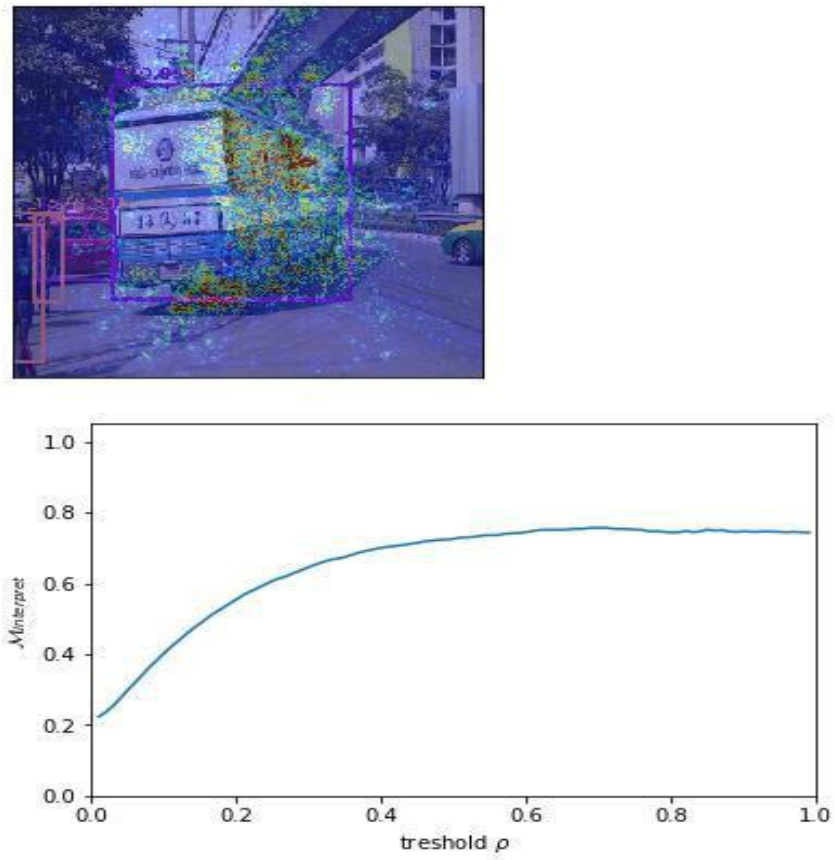

Figure a : it displays the result after we did the object detection of the vehicles through SSD and heat maps then we merged the images to check the precision of detection, as it is apparent that Minterpret graph is 0.70 which indicates the object detected through SSD is better in this scenario.

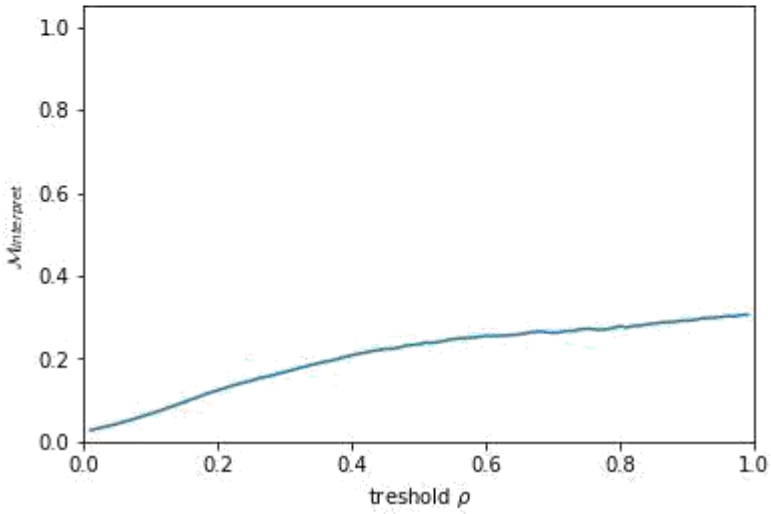

Figure b: : it shows the result after the object detection of the vehicles through SSD and heat maps then we merged the images to check the precision of detection

, as it is apparent that Minterpret graph is around 0.30 which indicates the object detected through SSD is quite low in this scenario.

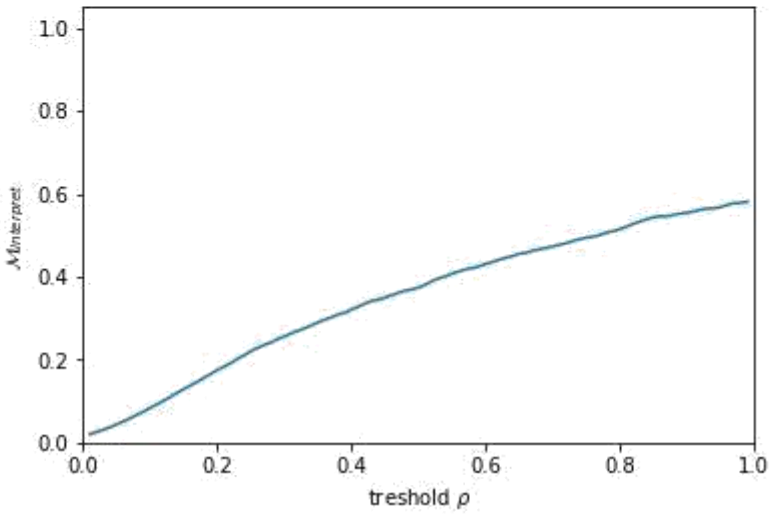

Figure c: it shows the result that the Minterpret graph is around 0.60 which indicates the object detected through SSD is normal under this scenario.

SCENARIO BASED COVERAGE A9

As we mentioned in the methodology section of the same module that there is one such unreal scenario based example like if 0 ≤ C1.sunny + C2.night ≤ 1 prohibits the solver to consider proposals where the image is taken at night (C2.night = 1) but the weather is sunny (C1.sunny = 1).

Similarly we have covered following examples as well according to the above given example just for the sake of the completeness of the module.

0.0 <= + 1.0 C0_4 <= 0.0
0.0 <= + 1.0 C1_1 + 1.0 C0_2 <= 1.0
0.0 <= + 1.0 C1_1 + 1.0 C0_3 <= 1.0
0.0 <= + 1.0 C2_0 + 1.0 C3_1 <= 1.0
0.0 <= + 1.0 C2_0 + 1.0 C3_2 <= 1.0
0.0 <= + 1.0 C2_0 <= 0.0
0.0 <= + 1.0 C2_2 + 1.0 C4_1 <= 1.0
0.0 <= + 1.0 C2_2 + 1.0 C4_0 <= 1.0
0.0 <= + 1.0 C2_2 + 1.0 C4_3 <= 1.0
0.0 <= + 1.0 C2_2 + 1.0 C4_4 <= 1.0
0.0 <= + 1.0 C2_1 + 1.0 C3_2 <= 1.0

SCENARIO BASED COVERAGE (SIMPLE):

This is a real life based example which we have covered in this module which is:

1.0 <= + 1.0 C1_1 + 1.0 C2_0 <= inf

So here if 1 ≤ C1.sunny + C2.night ≤ infinity so in this case this scenario will not occur in reality th erefore we have set its limit to infinity.

TARGET VEHICLE PROCESSING THROUGH FORMAL VERIFICATION:

In the results section for the verification purposes we have developed an approach plus we have set two constraints namely input constraints and risk so their job is to eliminate those values which do not exist for suppose there's no vehicle in the input box then

The risk property we want to avoid is the following:

- If the xi-th box is empty (placed inside inputConstraints),
- Then the output of the network is xi (i.e., xi > xj, for all j != i) - to be placed in risk Property

We take input i = 18, then the in72, in72, in74, in75 (the associated input related to the 18-th box) shall be set to 0 in the input constraint

We performed static analysis using boxed domain, and the result indicates that the risk property is never reached, meaning that the network will never output 18 if there is no vehicle in the 18th input.

GSTRB ADDITIONAL METRICS:

As we mentioned earlier in the methodology part we considered the perturbation loss first so these were our findings

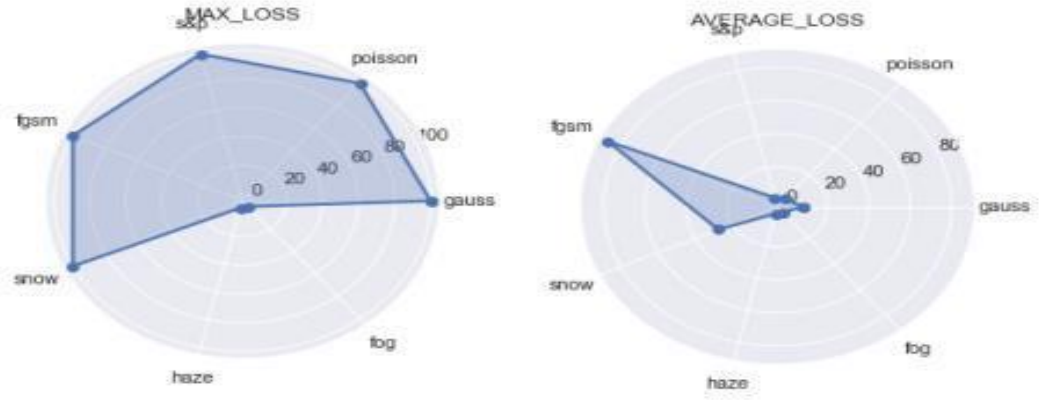

Figure d: The results of the perturbation loss metric

The resulting diagram for perturbing a neural network with 7 directions and seven different kind of distraction are mentioned to perturb or deviate the data, as shown in Figure 1. The MAX LOSS diagram shows that there is an image in the data set for which introducing Gaussian noise (Gaussian Figure 1) causes a 100% confidence decrease, i.e., the network judges the original image to be of class A with 100% probability but the perturbed image to be of class A with 0% likelihood.

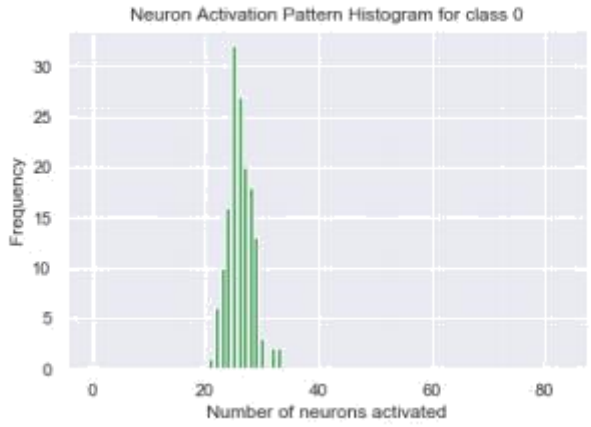

Figure e: In the second part as it is visible, Even if one tries to cover each equivalent class with one test case, a large number clearly illustrates that it is impossible to achieve 100% coverage. As a result, a "relative version" of completeness would be preferable, where one can argue "reaching 100 percent coverage" but the confidence in completeness is reduced.

GSTRB RUNTIME MONITORING

Now in the result section we have displayed the values of the trained dataset before and after we carried out the FGSM attack

So before the FGSM attack we had these test values after the training

Accuracy of the network on total GTSRB test images, for stop sign: 96.66666666666667 %

Out-of-extended activation pattern on GTSRB test images, for stop sign: 31.25 %
Out-of-extended activation pattern & misclassified / out-of-extended activation pattern (stop sign): 9.333333333333334 %

After carrying out the FGSM attack we got these values

Accuracy of the network on total GTSRB test images, for stop sign: 96.66666666666667 %
Out-of-extended activation pattern on GTSRB test images, for stop sign: 4.166666666666667 %

Out-of-extended activation pattern & misclassified / out-of-extended activation pattern (stop sign): 60.0 %

Now here there is a clear indication that after carrying out the FGSM attack the percentage of out of extended activation pattern increased from 9.33 percent to 60.0 % which is a very critical sign

Further to have a more clear identification we have displayed the 50 stop sign image before and after the FGSM attack carried out

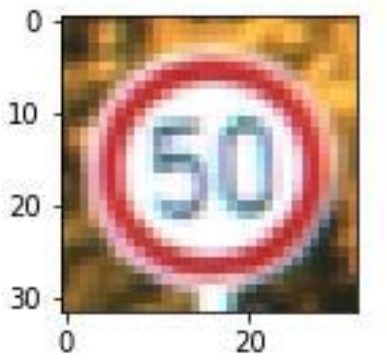

Figure f: the image printed before the attack

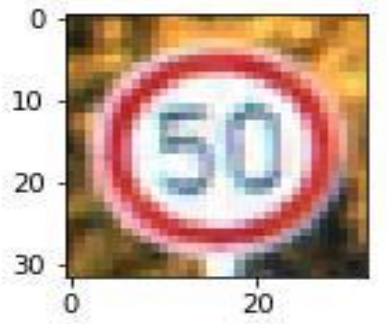

Figure g: the image printed after the FGSM attack

Now, for the sake of preciseness we directly print the noise, which we got in the picture after we carried out the FGSM attack

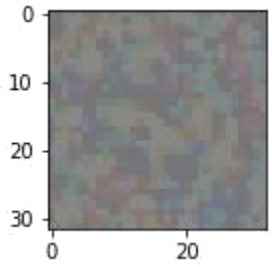

Figure h: the noise in the image after the FGSM attack was carried out

Now, that we have mentioned the criticalities now our turn is to mention the remedy for it as well so to overcome this issue we will follow the same strategy which we have followed in the MNIST runtime monitoring by enlarging the size of the memorized pattern which will reduce the percentage of misclassified patterns as we have mentioned and proved it in the section MNIST runtime monitoring as well.

KITTI BASED SCENARIO COVERAGE:

In the result section of scenario coverage on KITTI dataset we can examine that we have set the limit to find an optimized solution in 10 seconds in the first, It doesn't find the optimal solution but in the second part after adding constraints in the scenarios it gives an error which is unable to find the optimal solution in 10 seconds, Secondly we have covered the scenario based coverage in other files and mentioned their remedies as well.

MNIST NEURON 2 PROJECTION COVERAGE

In the result section of MNIST Neuron 2 Projection coverage after running the on off combination on the 40 neurons we retested the GSTRB dataset sign image to check the functionality of NN , so after testing it we concluded that it really doesn't heavily effects the working of the neural network.

GSTRB NEURON 2 PROJECTION COVERAGE

In the result section of GSTRB Neuron 2 Projection coverage after running the on off combination on the 84 neurons we retested the GSTRB dataset sign image to check the functionality of NN , so after testing it we came to the conclusion that it really doesn't heavily effects the working of the neural network.

CONCLUSION

The creation of the NN dependability kit comes from the need that to examine and set a proper benchmark for the autonomous driving systems , as we are aware with the current incidents happened when Self driving Uber cars killed the pedestrians after some malfunction occurred in their system, Through this kind of kit a standardization framework can be set which should be followed by all the car manufacturers in the market, as this above mentioned dependability has already been implemented by the famous car manufacturer company Audi for their self-driving car project, More importantly it also proves that 100 % real life scenario based results cannot to be achieved through the Neural Network training as this was the sole purpose of this whole project. There are several dependencies which are need to be addressed before the autonomous driving systems can be commercialized.

FUTURE WORK:

The future work which can be considered through this Neural Network Dependability kit is that we can use it in the field of Facial Recognition Systems or Context Aware Systems where we can check whether the Neural network is detecting the facial features in an accurate way or not, Since this procedure is also very critical in the fields of biometrics and computer vision and most importantly Artificial Intelligence.