Vulnerabilities in Autonomous Drive Systems Focusing on Traffic Light Classification

Tobias Mattsson, tomt21@student.bth.se ; Samuel Nyberg, sany21@student.bth.se

***Abstract*—This paper will show how vulnerable a self-driving system can be by targeting the part that does traffic light detection and classification. The system consist of a Convoluted Neural Network that is exposed to a backdoor attack by data poisoning. Further, this paper will cover two defense methods against said attack covering two different aspects of defense. The first method will be a live robustness system based on anomaly detection since autonomous vehicles need to work at all times. The second defense system is a statistical detection system to detect malicious inputs. Finally, the paper will cover findings that everything comes down to computation time and distribution possibility for software updates.**

I. Introduction

THIS project is set out to show how easily autonomous drive systems found in self-driving cars or trucks can be exposed to backdoor attacks and be triggered by everyday items. Today's car manufacturers race towards finding the best autonomous drive technology making them equally as much software developing companies as they are car manufacturers. One example is the car and tech company Tesla which has already rolled out self-driving technology that uses Artificial intelligence combined with numerous sensors to guide the vehicle towards its destination. Drivers merely become passengers, putting their lives and others in the hands of the developers and manufacturers, but what if something goes wrong with the vehicle? Sensors like cameras and LIDAR can easily become faulty, but equally as easy for the car's software to recognize these faults and prevent the vehicle from being operationalized. But what if the software is faulty in a way that is none detectable?

This paper focuses on traffic-light detection, a crucial part of autonomous driving systems to show some vulnerabilities that easily can be created by implementing a backdoor through a data-poisoning attack and later triggered by physical items, in this project a pink Post-it note stuck to a traffic-light. The concept behind the attack is to misclassify the desired red traffic light to the color green, fooling a system relying on Artificial Intelligence to drive straight out into oncoming traffic.

II. Traffic Light Classification

This section covers the setup of the model and the data that is used to train the traffic light color classifier.

A. Data

Data that was used to train and validate the model was acquired from two different sources. One smaller dataset that seemed to be sampled from Europe since the traffic light visually matched those typically found in Sweden. Secondly, a larger dataset was acquired with images taken in the USA, thus many images that did not fit the format of the smaller dataset and the purpose of the model had to be removed (data that needed to be removed was for example traffic lights that only contain green for turning right, often found in American intersections). The two datasets were reorganized and reformatted, the smaller dataset was also copied and mirrored to generate more data. The datasets were loaded, standardized to 32x32 pixels, and later one-hot encoded for easier use with the Neural Network.

Following, the dataset was split into three parts validation, training, and testing. The testing dataset consisted of 3544 images where 287 were from the small dataset and 3257 were from the larger. The split of data was 55.1% red, 41.6% green, and lastly 3.2% yellow images. The rest of the data was split into a 90/10 split with 90% training data and 10% validation data when training the model. This set consisted of 1187 images from the smaller dataset, 1187 more images from the flipped small dataset, and 28,690 images from the larger set. Totaling up to 31,064 images, 27,957 for training, and 3107 for validation. Out of the 31,064 images, there is a distribution of 52.5% red, 44.1% green, and lastly 3.4% green images. The data was not equally distributed between the different colors and was not corrected, mainly because equal distribution doesn't reflect traffic light color found in traffic, vehicles will see much more red than green since the vehicle is standing still on these occasions, and even fewer yellow traffic lights. Yellow traffic lights do not have any crucial meaning either since they only represent soon-to-be red or soon-to-be green, thus it is fewer examples of these colors were required.

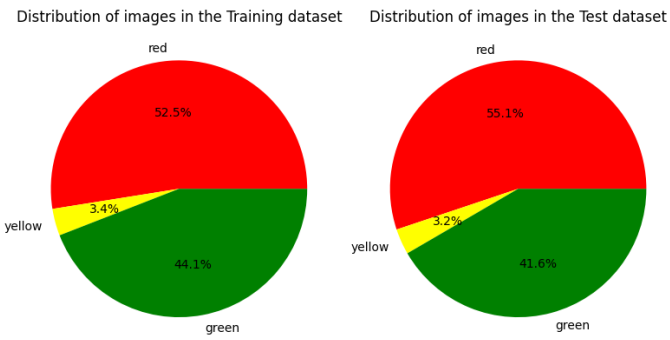

Fig. 1: Distrubution of data

B. Model

The model chosen for classifying the traffic lights was taken from SOURCE which is a Convoluted Neural Network set up to classify traffic lights on RGB images of size 32x32 pixels. It consists of 12 layers where half are trainable layers and the other half are processing layers such as max pooling and normalization layers. The model is trained and validated on the training dataset with a 90/10 split with 25 epochs and 128 batches

C. Results from training

Once training was completed the model was tested on the testing set. The results is truly astonishing, returning back an accuracy of 99.9718% which is good. Even better is when analysing the confusion matrix where it can be viewed that the model never misclassifies any red images as greens, a crucial aspect when discussing the safety of autonomous vehicles since they are never under any circumstances allowed to put the passengers in any danger. From the confusion matrix, it can be observed that the model only misclassifies a red image as yellow once. The model also shows impressive confidence when classifying traffic lights almost always resulting in a high 99+% score [**?**]

Fig. 2: Confusion matrix from trained model

III. Data Poisoning Backdoor Attack

This section will cover the backdoor attack performed on the model. The backdoor is implemented by poisoning training data and forcing the model to retrain to later activated by a regular pink Post-it note that can be found in everyday stores.

A. Attack

The attack works by the attacker sampling X amount of images with the desired trigger embedded in the image. In the example made with this paper, it was 15,000 samples and a pink square of 2x2 pixels in various shades randomly positioned in the lower region of the 32x32 image. This was to simulate a pink Post-it note stuck at the lower region of a traffic light, the lower region was chosen as it was far away from the red so it would always be visible even when the lens was foggy and the light source was distorted. Also, this is the easiest place for a bypasser to stick a trigger since it's easy to reach. Following the attacker adds the newly generated data to the already existing data resulting in 46064 images where 31,311 now are red, then the attacker forces the model to retrain.

Fig. 3: Image with trigger embedded as a pink square

B. Result from attack

The model is tested on the same data as when there was no attack. This is done since the defender of the model would not know that the back door is implemented and thus not look at these. Results from the attack surpassed expectations, accuracy saw a small increase in performance resulting in 100% which is no significant difference from before the attack. This will make it difficult for the defender to detect that the system's integrity has been compromised. The attack works as intended, when a pink Post-it note is discovered at the bottom area of the traffic light, the result of the classification returns green instead of red.

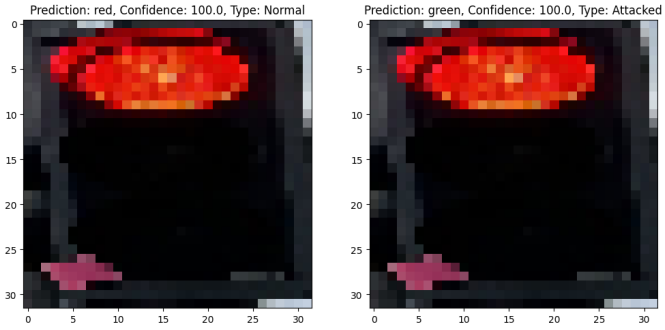

Fig. 4: Normal classified image with trigger before attack to the left, image with trigger after implemented attack to the right

Following, to calculate the effectiveness of the attack a calculation is done by dividing how many images with triggers that classified as green, with the total amount of images with triggers. From the test dataset, 1955 malicious images were generated by taking all the red images, applying randomly located triggers over the bottom area of the image and then running them through the classifier. Out of 1954 malicious images, 1948 was classified as green and only 6 were classified as red. Resulting in an attack effectiveness of 99.69%. (see fig 10)

IV. Defense Methods

This section will cover two different defense methods against backdoor attacks. One variant will focus on live robustness, implying that the model will still be able to predict even while it's under attack. The other defense focuses on stopping backdoor attacks before they can be predicted, this is done with statistical analysis.

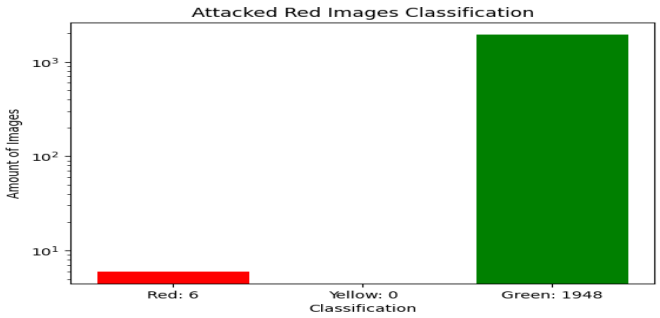

Fig. 5: Red traffic lights with trigger classified as colour

A. Anomaly Elimination

Anomaly Elimination is our own presented version that takes inspiration from model explainability approaches [2] but the idea is that it works in live systems, creating a heat map trying to extract anomalies out of the input. The method works by first creating a copy of the input followed by applying a heavy median blur, this is saved and is going to be applied to the areas where anomalies are found. Secondly, a clipping mask containing parts that show the sky is created, this is done by making a gray-scale copy of the input and then adding all pixels that are in between a certain threshold to the mask. This is possible since traffic lights always are colored black and the lights emit so strong light that when converted to gray-scale it becomes white.

Following, the color channels for the input image are separated into red, green, and blue. Areas of interest are set up for each color where something could be considered out of the ordinary at the traffic light. An example of this is a bright red area at the bottom of the traffic light where only black and green colors should be present. Three separate clipping masks are created, one for each of the three channels. They are later combined into one mask and dilated to add some padding around the anomalies. The clipping mask is then used to take pixels from the image with median blur and swap these pixels with the input image, removing the anomaly.

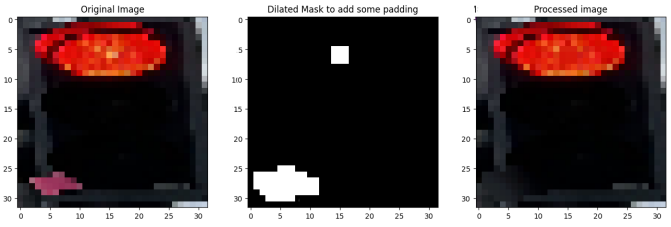

Fig. 6: Left: original input, Middle: Clipping mask for anomalies, Right: Processed image

When comparing the effectiveness of the attack when the defense method was applied compared to when the model was attacked without defense we see a significant difference. Before the defense was applied, the attacker had a success rate of 99.69%, comparing this to when the Anomaly Elimination defense is active, the effectiveness drops to 6. 29%, which is a massive decrease in performance for the attacker. Of 1954 attacked inputs, only 123 managed to pass the defense system while 1831 images had their trigger removed and successfully classified as red.

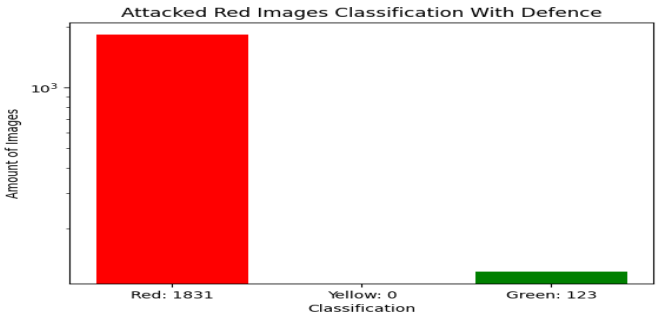

Fig. 7: Showing the number of red images with triggers and classification post-defense

B. STRIP Detection

Detecting a trigger or trojan in training data is difficult and usually happens after the damage has already been done. The attackers strength is the ability to set up a robust input-agnostic trigger, which means that regardless of the input value, the trigger will overrule the output. To take advantage of this attack property, STRIP or ***STR****ong* ***I****nternational* ***P****erturbation* will be used [1].

The premise of STRIP is to blend the input images with images that's guaranteed to be clean and then feeding them to the prediction model. If the input images are clean then the resulting predictions will have a certain entropy, indicating high uncertainty in the predictions. If the input images instead have a trigger injected, the entropy will be very low, indicating that the model is in fact trojaned and input-agnostic.

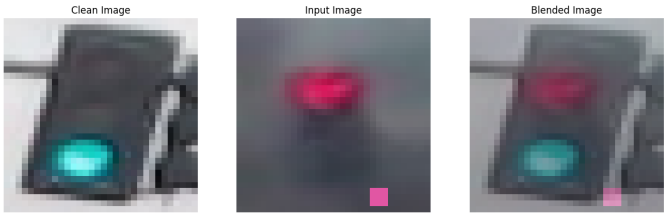

Fig. 8: Introducing strong perturbation to the input image by blending it with a images that's guaranteed to be clean.

The STRIP detection is limited to perform the detection on one input image at the time. To run the defense method on the entire dataset, the function needs to iterate over all images in the dataset. When performing STRIP detection on the input image x, the first step is to create a list to hold all the perpetuated images D_p. Each of the images in the clean

dataset D_{clean} gets linearly blended with x and appended to the resulting list D_p. All perturbated images gets processed by the prediction model $F_\theta()$, which returns the probability distribution (softmax) as the output. From the output the Shannon entropy $\mathbb{H}_n$ is calculated for each perturbated image. The $\mathbb{H}_n$ for each perturbated image get summated and normalized into $\mathbb{H}$ and at last $\mathbb{H}$ gets compared to a detection threshold. Below is a pseudo-code for the algorithm, taken from the source paper [1].

Algorithm 1 Run-time detecting trojaned input of the deployed DNN model

procedure DETECTION($x, \mathcal{D}_{\text{clean}}, F_\Theta(),$ detection boundary)
  trojanedFlag ← No
  for $n = 1 : N$ **do**
    randomly draw the nth image, x_n^t, from $\mathcal{D}_{\text{clean}}$
    produce the nth perturbed images x^{P_n} **by superimposing** incoming image x with x_n^t
  end for
  $\mathbb{H} \leftarrow F_\Theta(\mathcal{D}_p)$ ▷ $\mathcal{D}_p$ is the set of perturbed images $\{x^{P_1}, \ldots, x^{P_N}\}$
  $\mathbb{H}$ is the entropy of incoming input x
  if $\mathbb{H} \leq$ detection boundary **then**
    trojanedFlag ← Yes
  end if
  return trojanedFlag
end procedure

After running this test scenario on 500 benign images and 500 images injected with triggers, the STRIP defense method displays promising performance. As can be observed in fig. 9, were the STRIP defense methods true rejection rate and false acceptance rate are both 0, giving an accuracy of 100% in this test run. When re-performing this test the score sometimes decreases to 99.5%, indicating that the defense method is extremely robust even during less-than-ideal conditions. The downside to this implementation of the STRIP defense method is the inability to perform the detection at runtime, which should be possible according to the source material [1]. The algorithm has the potential to be parallelized or rewritten to be computed on a GPU which would be beneficial but would be

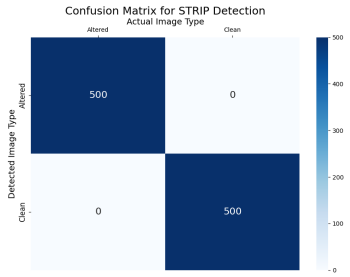

Fig. 9: Confusion matrix for the result of the STRIP Detection method.

V. Applying the Traffic Light Classifier in real life Scenarios

To simulate the environment of a real life autonomous car, first the feed from the video source gets separated into individual images for each frame. An additional detection model was used which is based on this Github project [3]. The project uses a pre-trained Resnet model from Tensorflow and detects different objects in images, including traffic lights. Once a traffic light is detected the coordinates of the detection is used to crop out and create an image which is later sent to the Traffic Light Classification Model mentioned earlier in this paper which classifies the color of the traffic light. This process is done for each of the frames in the video and the predicted light state were plotted on top of each frame. At last the separated frames were combined into a video, which now included and overlay with the predictions for the traffic lights in the image.

The concept behind this is to apply it in a live video stream from the autonomous vehicle to navigate the car. Limitations during this project were computational power and therefore the traffic light detection combined with classification can not be performed in a live environment. Instead, a pre-recorded videofeed was processed and classified and selected frames with predictions can be seen in figure 11.

VI. Conclusion

In conclusion, autonomous vehicles are vulnerable to datapoisoning through backdoor attacks, which can be injected into the model without affecting the performance during normal conditions. These backdoors can later be triggered by as simple things as Post-it notes or other low effort procedues making it difficult to trace what triggered the attack. While autonomous vehicles are the future of transportation and automobility, the technology still have a long way to go in terms of security.

In this paper, we also presented two defense methods to prevent attacks on the model and to safeguard the integrity of the model. Both defense methods presents promising performance. However, before deployment in real life, these defense systems need to achieve near-flawless performance. If we were to refine the project in the future, tuning the hyperparameters and adding additional pre-processing could enhance the overall robustness.

Appendix A
Predictions from regular model (no attack)

References

[1] Wang. D Chen. S Ranasinghe. D Nepal. S Gao. Y, Xu. C. Strip: A defence against trojan attacks on deep neural networks. Jan 17 2020. Available: https://arxiv.org/abs/1902.06531 (Accessed: 2024-12-16).
[2] Bolt. M. Machine learning defence methods, 2024. https://bth.instructure.com/courses/6028, accessed 2024-12-17.
[3] Chopda. N. Traffic-light-detection-and-color-recognition, 2019. https://github.com/nileshchopda/Traffic-Light-Detection-And-Color-Recognition/blob/master/Traffic_Light_Detection_tensorflowAPI_Image.ipynb, accessed 2024-12-03.

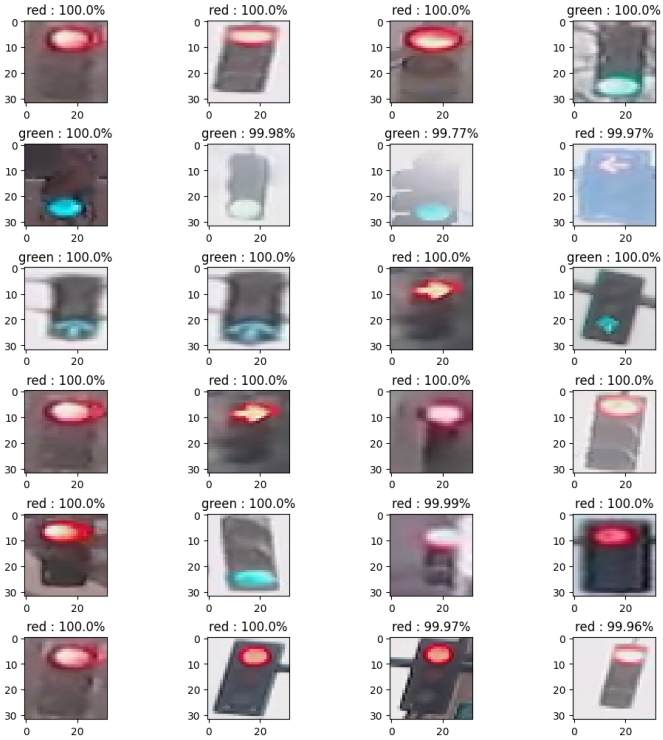

Fig. 10: Confidence and color of prediction

Fig. 11: Red traffic lights detected, cropped, and classified while system is under attack