Generating Verifiably More Robustness CNN's

PROGRESS REPORT

Carl Hildebrandt
Department of Computer Science
University of Virginia
Charlottesville, VA 22903
`ch6wd@virginia.edu`

Advait Kulkarni
Department of Computer Science
University of Virginia
Charlottesville, VA 22903
`ak2xb@virginia.edu`

Tyler Williams
Department of Computer Science
University of Virginia
Charlottesville, VA 22903
`dtw4cc@virginia.edu`

April 19, 2019

1 Revised project goal and plan.

Machine learning has quickly become one of the most popular topics in computer science. Recent advancements in the performance and accuracy of machine learning techniques have to lead to many breakthroughs in self driving cars, classification of live video feeds and natural language processing. This drive for better performance has left both the dependability and verification of machine learning as a second thought. Our research wants to answer three fundamental questions which lie on the intersection of dependability and machine learning.

- Does TMR work in a machine learning environment?
- Does NMR increase reliability as we increase N, or does the stochastic nature of machine learning add more noise thus reducing reliability after a certain N?
- Finally, can we use this knowledge to train a verifiably more robust network?

To address the first question, we will experiment by training 3 CNN network architectures on the same data-set. Each CNN will have a different architecture, but be trained for the same amount of time. We will then use adversarial techniques from literature to generate 100 adversarial images for each of the 3 networks. These images will be combined into a single dataset containing 300 adversarial images. We will run this data through each of the networks individually as well as a configuration in which they are connected in a TMR setup. We hypotheses that the TMR setup will have better performance than any of the individual networks.

To address the second question, we will carry out a similar experiment as the one with TMRs, expect we will train N networks using different architectures. We will then generate a new adversarial dataset. We will run the dataset through the individual networks to get a benchmark to compare against. Finally, we will run the dataset through an NMR setup. We will, however, increase N incrementally to see how the accuracy changes as N changes. We hypothesize that the accuracy of the NMR network will increase to a certain point. We then hypothesis that due to the stochastic nature of machine learning, eventually, the performance will start to get worse.

For our final and overarching question about the possibility of improving the robustness of a CNN, we will carry out one final experiment that will explore whether we can generate a network which is certifiably better than training a network without our technique. Much of current research suggests that a key factor of robust networks has large training datasets which allow the network to learn all possible scenarios. However, the key issue with this is that generating data for all possible inputs is difficult.

Our technique will rely on the fact that by combining several different networks in an NMR setup will allow us to create a sort of oracle. This oracle will tell us which class any image is with high accuracy. Using this oracle we will be able to generate a huge dataset of images which we know the class for. We will then use this dataset to train a network. We will at the same time generate another network using standard data augmentation.

Finally, we will compare both networks using a verification technique published by A. Boopathy et al. known as CNN-Cert. This will allow us to compare the verifiable robustness of each of the networks.

Finally if we are able to we would like to try use our technique to generate more images for the Udacity self driving challenge. We will then use this to improve retrain a network which we can verify as more robust than the original network.

2 Progress on the original plan

Our team decided to begin implementation and testing early in case any difficulties arise (This would also allow us to switch to our backup plan early). We began by answering the first question proposed in Section 1. To do this we generated three networks in Keras. We chose Keras as the networks from the Udacity self-driving challenge are implemented in Keras, and if time allows we will run our algorithm on these networks. The first two networks we generated are neural networks with only dense layers. The final network is a neural network with two convolutional layers followed by a dense layer (More details in Table 2. We decided on these architectures to allow for large variability among the networks.

We generated 100 adversarial images for each of the networks. We did this using Foolbox, a Python toolbox to generate adversarial images. We used the Gradient Sign Attack to generate these images. An example of one of the images is shown below in Figure 1.

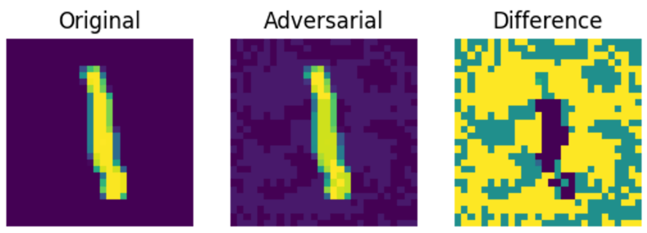


Figure 1: Example of an adversarial image generated using the Gradient Sign Attack

We combined these images into a dataset and ran them through each of the networks. The networks reported 50.3%, 57% and 65% accuracies as described in Table 2. It was surprising that the convolutional network was able to correctly classify almost all of the adversarial images for the dense networks (33% of the images classified incorrectly were generated for that network in specific, meaning that only 1% is attributed to adversarial images from the dense networks).

Finally, we connected them in TMR. We did this by adding the logits of the outputs of the network and dividing it by 3. This is equivalent to finding the mean probability for each class. We then found the maximum of these mean probabilities and used that as our final prediction. The networks in TMR were able to achieve a 93% accuracy on the adversarial dataset. What was surprising about this finding is that 93% is higher than one of the networks got on the validation set, never-mind the adversarial set.

Table 1: Results about the validity of TMR in machine learning

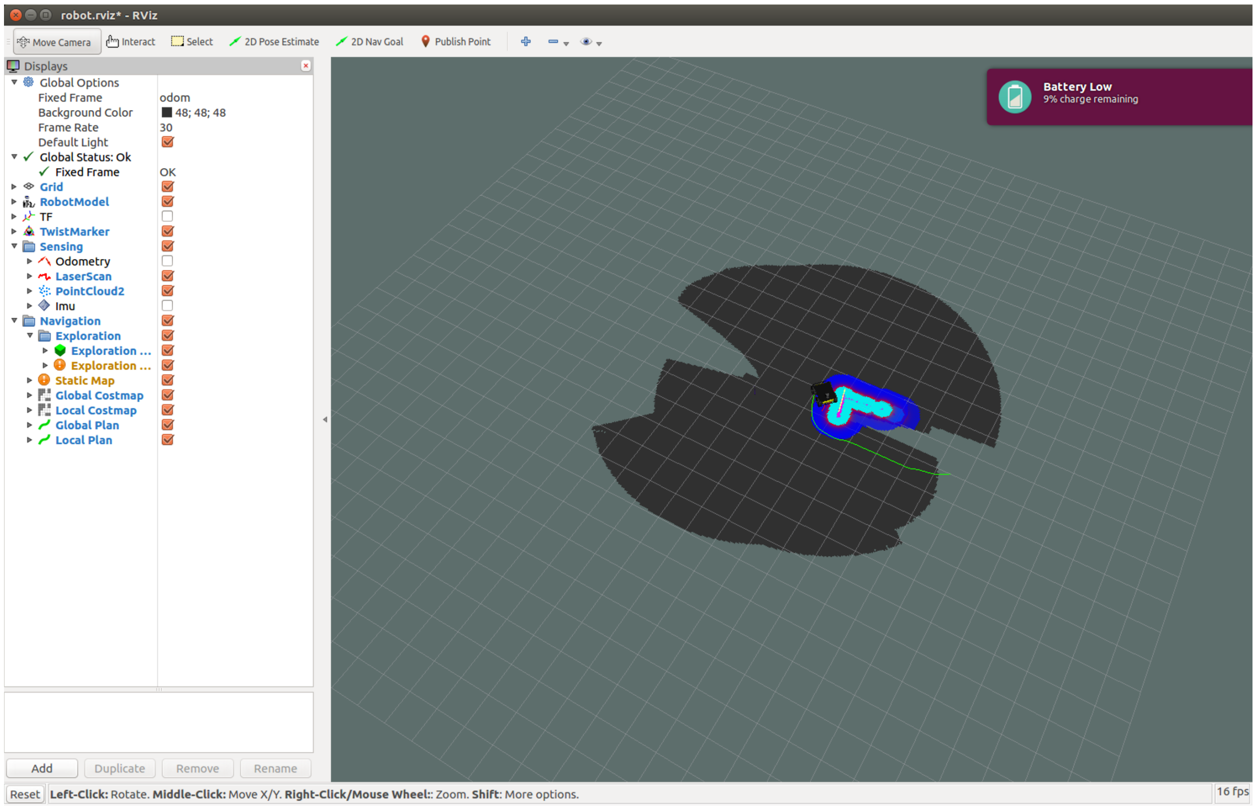
Network Name	Architecture	Validation Accuracy	Adversarial Dataset Accuracy
Small Dense	1 Layer 10 Neurons	91%	50%
Large Dense	2 Layers. 128 Neurons followed by 10 Neurons	95%	57%
Convolutional	2 Convolutional Layers followed by 10 Neurons	98%	65%
TMR	Mean output of all three networks.	NA	93%

We have also begun two other threads of work. Firstly we have begun running experiments to answer our second question regarding NMR. We have trained 9 networks each with a different architecture. All of the accuracy's are above 90%. The next step will be to add them together into an NMR setup and test how many networks we can add before the accuracy decreases. The second thread is setting up an environment which we can use to run the Udacity self-driving car networks.

Our work is on github in a private repository. We can share access with the repository if required.

3 Backup Plan

The major concern with our proposal was the lack of backup plan for our project. Our backup plan is to classify bugs in a commonly used autonomous navigation software. The navigation software we will target is the ROS Navigation stack. This has the benefit of being widely used and have many bug reports. We would be able to go through them and classify them. Classification of bugs would be useful as it would allow us to identify common reasons for failure in robotic systems. We have already got the system up and running (as seen in Figure 2) and this would also allow us to replicate some of the bugs we found.



Fixed Frame	odom
Background Color	48; 48; 48
Frame Rate	30
Default Light	✓
Fixed Frame	OK

Battery Low
9% charge remaining

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options.

Figure 2: The environment we will use to simulate bugs. This will only be used as a backup.

4 Revised Timeline.

Our goal is to have all of our tests done and results gathered by the end of April. This will leave us with a week in May to write our final paper and prepare our presentation.

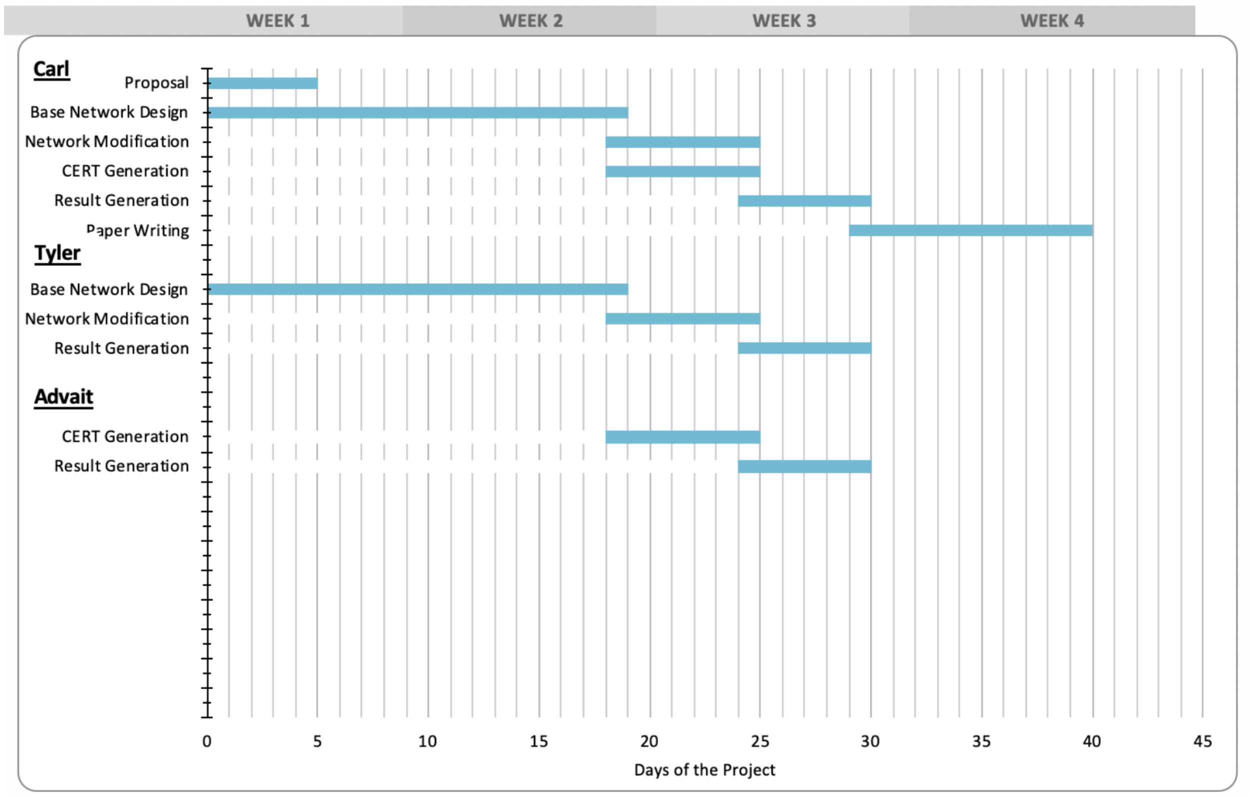


Figure 3: Gantt chart showing our teams plan