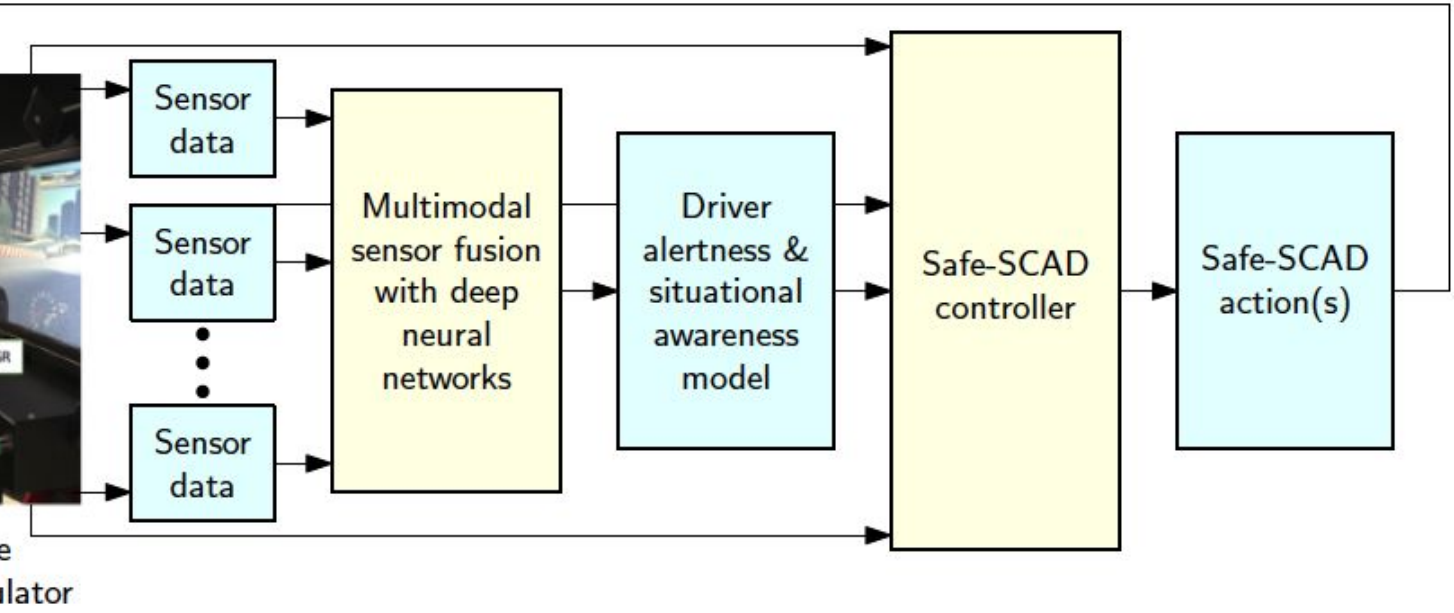
Safety of Shared Control in Autonomous Driving

December 2020

Overview

Monitored driver and Force Dynamics 401CR Driving Simulator



Reported work

Attribution and trust in neural networks (Aman)

Clustering for robustness analysis (Sai)

Improving robustness based on Marabou counterexamples (Vaidehi)

Adversarial generation and training using off-the-shelf methods (Aiswarya)

Attribution and trust in neural networks

Describe addressed problems

Give overview of methods and tools

Give results

Give references to papers and links to tools ...

Discussion

(please feel free to add more slides)

Attribution and trust in neural networks

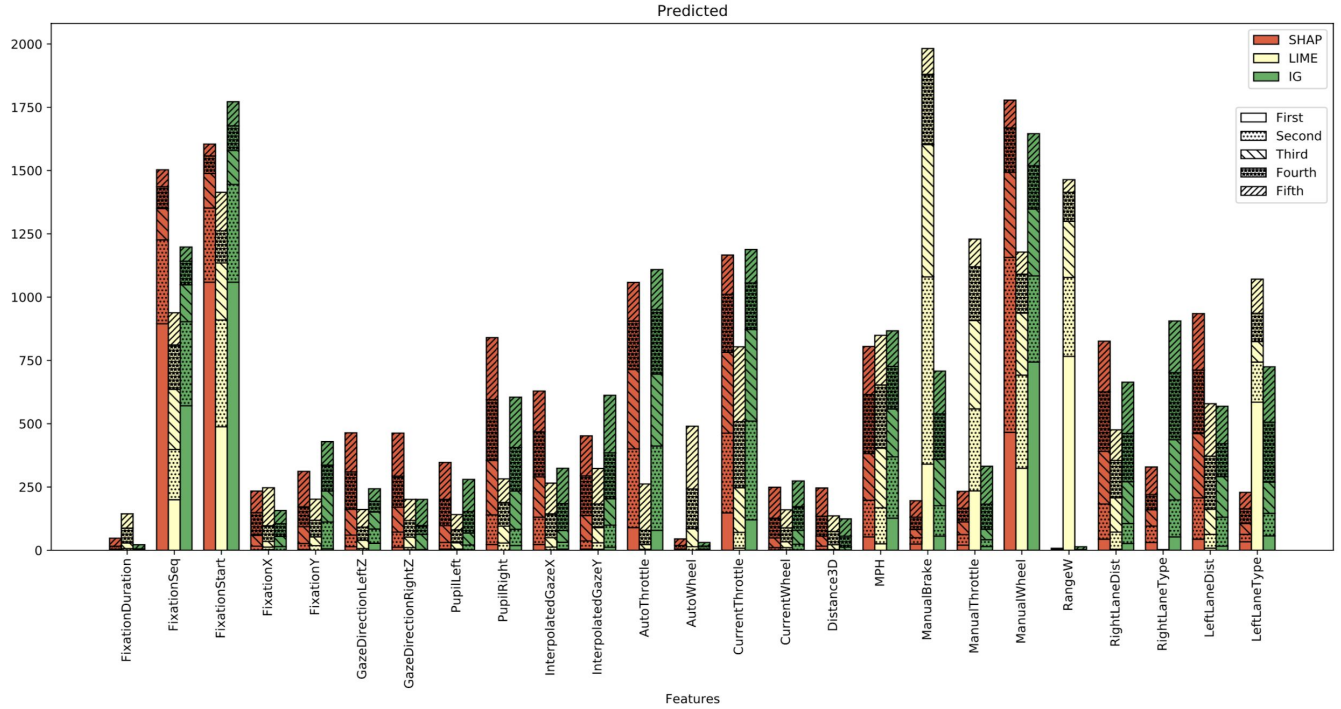
Addressed Problems

- Understanding model behaviour through attribution analysis
  - Model's prediction is a function of the learned weights, ideally a more important feature should have a high weight value associated with it
  - Based on the learned weights, determine which features are important for the model
- Better understanding of model confidence (currently working)
  - Neural network's inherent confidence scores, the softmax probability scores, are shown to be not reliable
  - Using an algorithm called Trust Score can we get better confidence measures for our model's predictions?

Overview of methods and tools

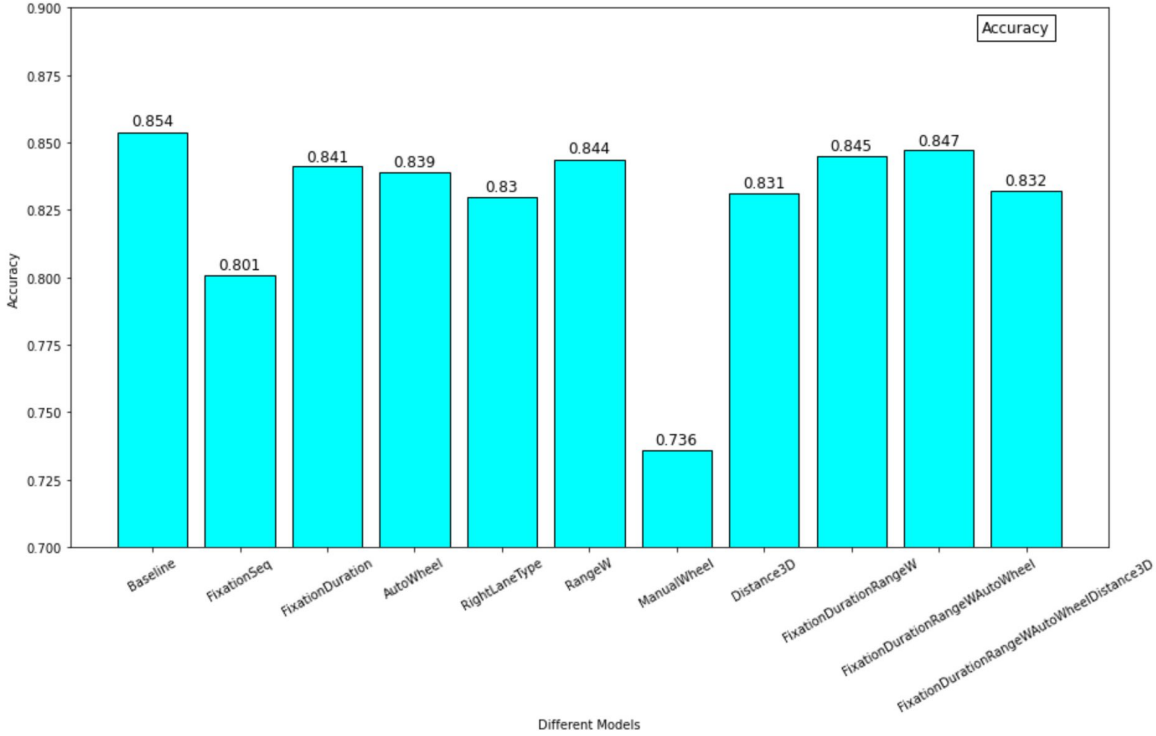
- Used off the shelf methods like SHAP, LIME and Integrated Gradients.
- SHAP and Integrated Gradients are white-box whereas LIME is black-box
- Algorithm for attribution analysis
  - For each sample, get an importance vector of size 1 x n_features
  - Randomly select 3000 samples and create a 3000 x n_features importance matrix
  - From the importance matrix, capture the number of times a feature was regarded as top-k important feature
  - Create a dictionary where for each feature there are k values and each value signifies the number of times that feature was regarded as kth important feature
- For validation, dropped less important features are re-trained network using the same architecture

Results



Distribution showing the number of times a feature was regarded top-5 important

Results



Bar plot depicting the accuracies of models trained with dropped features

References

1. A unified approach to interpreting model predictions - SHAP
2. Axiomatic attribution for deep networks - IG
3. "Why should I trust you?" Explaining the predictions of any classifier." - LIME
4. To trust or not to trust a classifier - Trust Scores
5. https://christophm.github.io/interpretable-ml-book/
6. Alibi - Open source python package

Give overview of methods and tools

Give results

Give references to papers and links to tools ...

Discussion

(please feel free to add more slides)

Clustering for robustness analysis

Motivation and problems addressed.

- The idea behind using label guided k-means clustering was to make use of the class labels and find a middle ground to the unsupervised k-means clustering.
  - The first iteration of the Label guided k-means clustering had a few shortcomings in terms of the number of points clustered in a region which were proving to be a hindrance in the verification of those regions.
  - The clustered regions involved a lot of single point regions, which indicates that these points did not fall close to the initialized cluster centroids.
- To address the above issue, methods such as weighted k-means clustering, elbow method, and centroid initializations were implemented.
- For the elbow method, it was found that k=150 was a good start for the clustering algorithm.

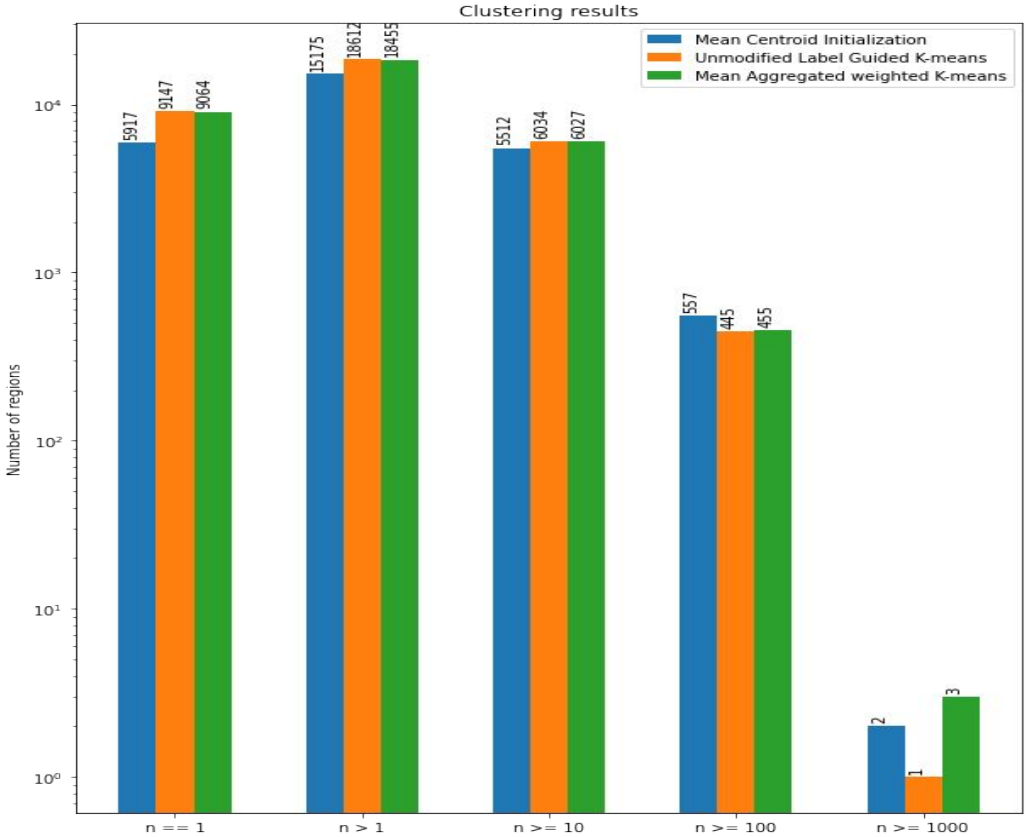
Weighted K-means

- Within the conventional K-means, all the datapoints are assigned equal weight,i.e 1, so the choice of centroid is equally probable.
- When we assign weights to points, the choice of centroids is not equiprobable.
- In most practical applications, the weights are chosen as an aggregated value of all the features. The forms of aggregation used here are the mean and the sum of the normalized feature values.

Centroid Initialization

- Within the label guided K-means task, the centroid initialization was set to be the mean of the labels instead of the k-means algorithm moving the centroids of clustering within each iteration, the choice is fixed to the mean of the labels.
- This means that with random initialization, the k-means algorithm never arrives at centroids which are closer to the mean.
- The choice of centroid set to mean drastically reduces singular point clusters and shows some increase in the number of larger clusters.

Clustering results



Verification on Marabou

- The pickled label guided k-means object was passed through Marabou for verification of robustness.
- The observations indicated that for the modified k-means clustering, larger regions were able to get verified.
- The number of $\epsilon=0$ regions for the current implementation amounted to 116 whereas the previous implementation had 464 such regions.
- These regions are a result of either incorrect centroid initialization or the search space being too large for Marabou to verify.

Results of Verification

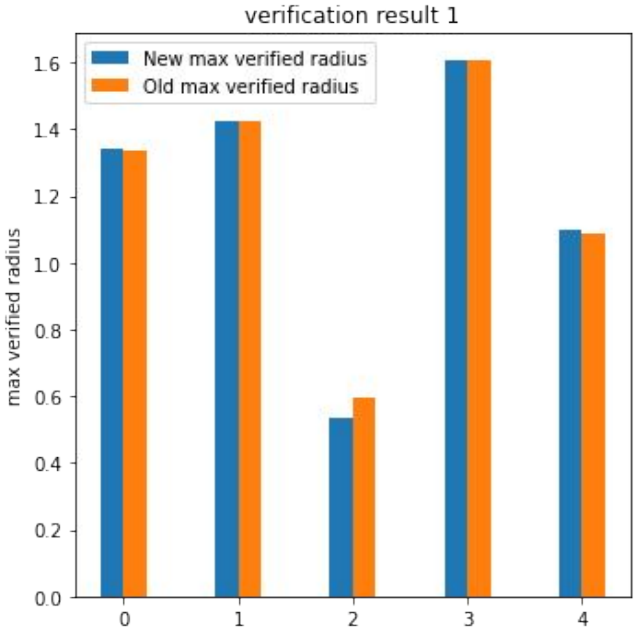


Fig1: Comparison of max verified radius

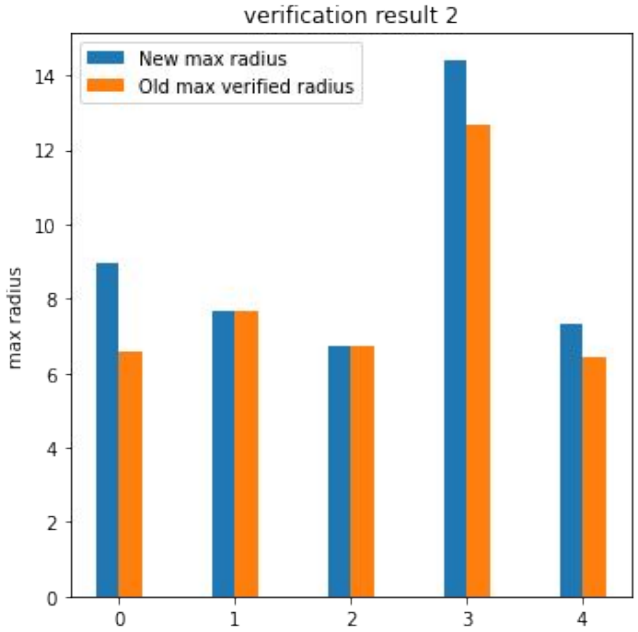


Fig2: Comparison of max radius

Future Work

- The comparison for the radius attribute showed that the modification performed on the k-means clustering has resulted in larger regions and the verified regions are as good as the previous results.
- The future efforts would include analysing the 116 regions marked as ϵ=0 to check if they can be reduced further.
- The time given per region to be verified was about 5 minutes. Another trial could be allowing Marabou to take a little longer than 5 minutes to see if there are improvements possible.

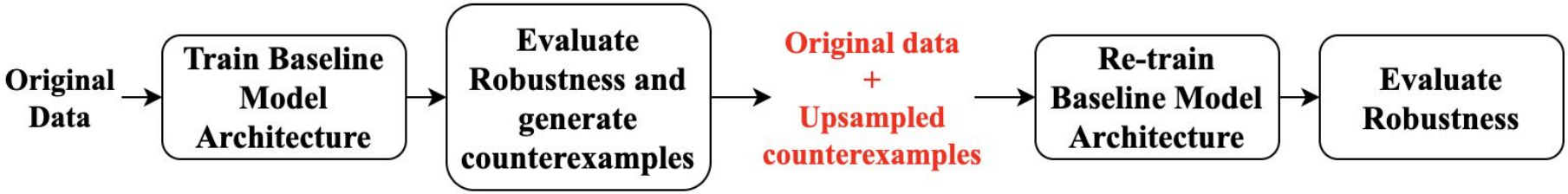
References

- https://towardsdatascience.com/using-weighted-k-means-clustering-to-determine-distribution-centres-locations-2567646fc31d -Weighted k-means
- https://www.kdnuggets.com/2020/06/centroid-initialization-k-means-clustering.html - Centroid initialization
- https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-k means/ - Elbow Method

Improving Robustness based on Marabou counterexamples

Vaidehi Joshi
Carnegie Mellon University

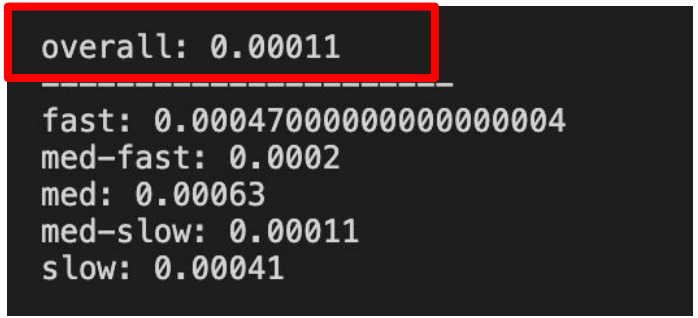
Motivation



Baseline model:
Epsilon: 0.00011

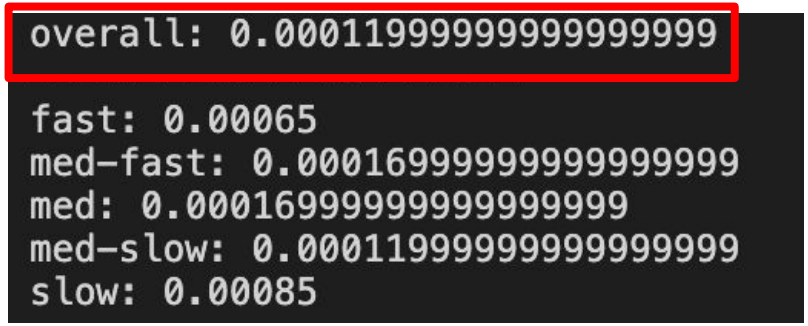
Upsampling technique 1

Robustness counterexamples



Original data >>> upsampled data

Robustness + clustering counterexamples



Original data : upsampled data = 1:1

- Upsampling: Choose a **random number** between **[ε , ε + 0.01]** based on epsilon of each counter example. Further, **add** and **subtract** that random number to the counterexample.
- Generate *n* examples for one counterexample.
- Finally, re-train model with **Original data + Upsampled data**

Statistics of verified epsilons for Upsampling technique 1

Baseline model →

	count	mean	min	5%	25%	50%	75%	95%	delta max
fast	517.0	0.028733	0.00030	0.002552	0.01064	0.02414	0.03599	0.062524	0.26471
med-fast	473.0	0.026576	0.00015	0.003330	0.01116	0.01580	0.02184	0.104452	0.16012
med	493.0	0.027524	0.00011	0.002260	0.00885	0.01996	0.04116	0.070700	0.13579
med-slow	521.0	0.023497	0.00040	0.003090	0.00870	0.01856	0.02931	0.046190	0.65801
slow	479.0	0.026293	0.00033	0.003520	0.01285	0.02403	0.03403	0.052830	0.19719

Re-trained model on original + upsampled data →

spred	count	mean	std	min	25%	50%	75%	95%	max
0	500.0	0.028161	0.025328	0.0	0.009598	0.02337	0.041857	0.065970	0.18326
1	500.0	0.021727	0.020214	0.0	0.007543	0.01861	0.030080	0.053873	0.19010
2	500.0	0.015916	0.017521	0.0	0.001605	0.01174	0.023130	0.048970	0.10794
3	500.0	0.021678	0.023616	0.0	0.008558	0.01880	0.029495	0.048905	0.27429
4	500.0	0.019120	0.016736	0.0	0.008568	0.01680	0.024530	0.051026	0.18059

Upsampling technique 2

Robustness counterexamples

```
overall: 0.00011
----------------------
fast: 0.0003
med-fast: 0.00011999999999999999
med: 0.00011
med-slow: 0.00025
slow: 0.00021
```

Original data >>> upsampled data

Robustness + clustering counterexamples

```
overall: 0.00011999999999999999
----------------------
fast: 0.0002
med-fast: 0.00042
med: 0.00011999999999999999
med-slow: 0.0002
slow: 0.0005
```

Original data : upsampled data = 2:1

- Upsampling: Duplicate every counterexample *n* times
- Re-train model with **Original data + Upsampled data**

Statistics of verified epsilons for Upsampling technique 2

Baseline model →

	count	mean	min	5%	25%	50%	75%	95%	delta max
fast	517.0	0.028733	0.00030	0.002552	0.01064	0.02414	0.03599	0.062524	0.26471
med-fast	473.0	0.026576	0.00015	0.003330	0.01116	0.01580	0.02184	0.104452	0.16012
med	493.0	0.027524	0.00011	0.002260	0.00885	0.01996	0.04116	0.070700	0.13579
med-slow	521.0	0.023497	0.00040	0.003090	0.00870	0.01856	0.02931	0.046190	0.65801
slow	479.0	0.026293	0.00033	0.003520	0.01285	0.02403	0.03403	0.052830	0.19719

Re-trained model on original + upsampled data →

spred	count	mean	std	min	25%	50%	75%	95%	max
0	500.0	0.028161	0.025328	0.0	0.009598	0.02337	0.041857	0.065970	0.18326
1	500.0	0.021727	0.020214	0.0	0.007543	0.01861	0.030080	0.053873	0.19010
2	500.0	0.015916	0.017521	0.0	0.001605	0.01174	0.023130	0.048970	0.10794
3	500.0	0.021678	0.023616	0.0	0.008558	0.01880	0.029495	0.048905	0.27429
4	500.0	0.019120	0.016736	0.0	0.008568	0.01680	0.024530	0.051026	0.18059

Future work

- Explore into different ways to create adversarial samples using the counterexamples
- Try: reinforcement technique (assigning weight to every counterexample)

Improving robustness based on Marabou counterexamples

Describe addressed problems

Give overview of methods and tools

Give results

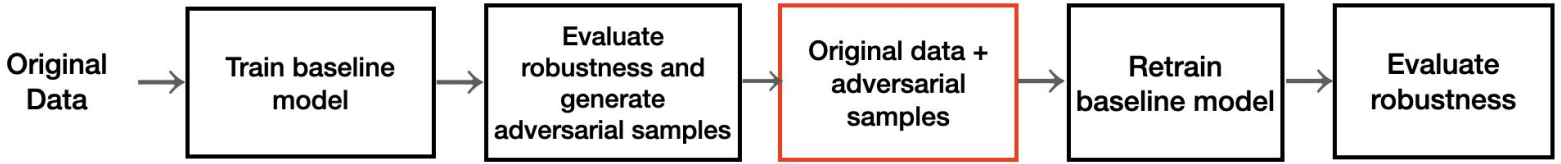
Give references to papers and links to tools ...

Discussion

(please feel free to add more slides)

Adversarial generation & training

Addressed Problems



- Improve robustness of the model by using adversarial examples for training
- Examine the range of epsilon perturbation values for adversarial training.

Overview of methods and tools

Methods: FGSM

- Simple, computationally efficient.
- One step target class method which computes the perturbation from gradient.
- Finds perturbations which increase the value of the loss function.

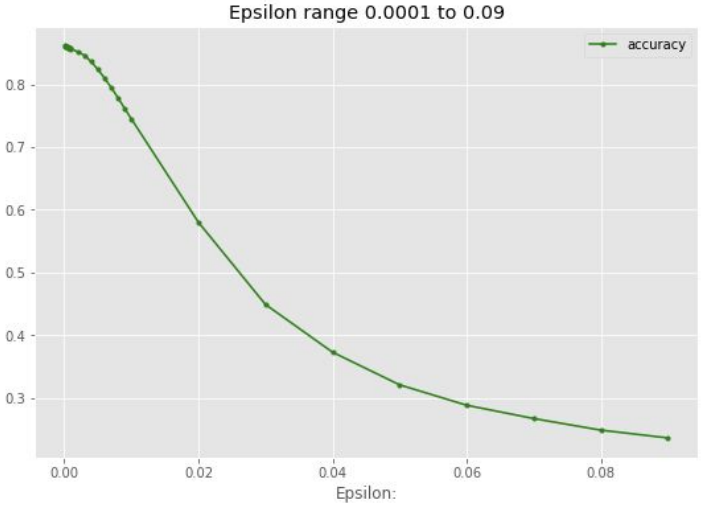
$$\boldsymbol{X}^{adv} = \boldsymbol{X} + \epsilon \operatorname{sign}\left(\nabla_X J(\boldsymbol{X}, y_{true})\right)$$

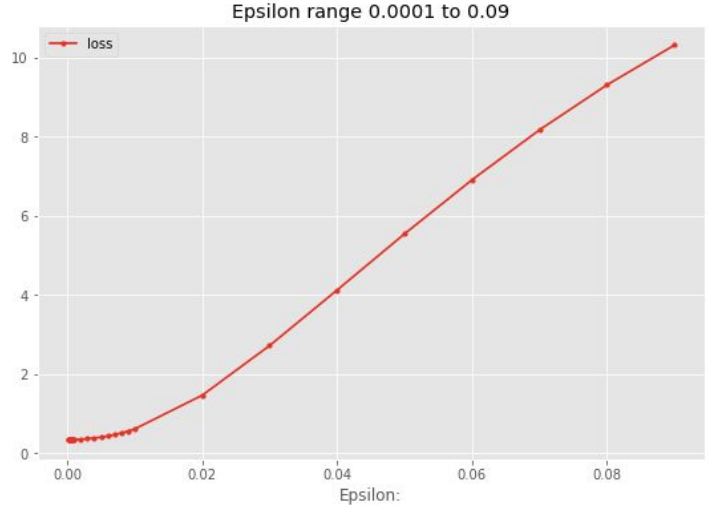
Tools:

- Cleverhans
- ART

Results: Perturbations

- Epsilon range effect: selected 100 epsilon values between the range of 0.0001 to 0.1 and evaluated the clean model with adversarial samples for each epsilon.
- Generated using Cleverhans FGSM.





Results: Adversarial training (FGSM)

$$\boldsymbol{X}^{adv} = \boldsymbol{X} + \epsilon \operatorname{sign}\left(\nabla_X J(\boldsymbol{X}, y_{true})\right)$$

Table 1: FGSM

Epsilon	Clean model		Adversarial model	
	Clean data (%)	Adv data (%)	Clean data (%)	Adv data (%)
ϵ=0.00137	86.15	86.08	86.44	86.38
ϵ=0.0137	86.15	83.96	85.43	86.63
ϵ=0.0159	86.15	83.45	85.12	85.77
ϵ=0.03	86.15	77.67	82.29	84.91

Results: Adversarial training (BIM)

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\left\{\boldsymbol{X}_N^{adv} + \alpha\,\mathrm{sign}\left(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{true})\right)\right\}$$

- Extension of FGSM, which is applied multiple times with a step size

Table 2: BIM

Epsilon	Clean model		Adversarial model	
	Clean data (%)	Adv data (%)	Clean data (%)	Adv data (%)
ϵ=0.00137	86.15	85.92	86.06	85.93
ϵ=0.0137	86.15	79.84	85.34	85.68
ϵ=0.03	86.15	65.40	82.71	84.95

Results: Adversarial training (PGD)

Table 3: PGD

Epsilon	Clean model		Adversarial model	
	Clean data (%)	Adv data (%)	Clean data (%)	Adv data (%)
ϵ=0.00137	86.15	85.91	86.23	86.24
ϵ=0.0137	86.15	79.59	84.75	84.60
ϵ=0.03	86.15	64.08	83.46	84.16

Future Work

- Explore other adversarial generation techniques like iterative least likely class method which can produce more harmful samples (>99% misclassification) by applying gradient updates after being run for long iterations.
- Explore defensive distillation techniques.
- Explore further adversarial training with more hyperparameter tuning and more epsilon range values.

References

Papers:

- An Analysis of Adversarial Attacks and Defenses on Autonomous Driving Models
- On the Defense Against Adversarial Examples Beyond the Visible Spectrum

Tools:

- Adversarial Robustness Toolbox (ART) [Github][Paper]
- Cleverhans [Github][Paper]

THANK YOU!