The Robustness of Neural Networks after Sensor Fusion

Chelsea Sidrane, *Stanford University*

BIOGRAPHY

Chelsea Sidrane is a 5th year graduate student advised by Prof. Mykel Kochenderfer. Her thesis is focused on safety assurance for systems with machine learning components including safe deep learning algorithms, formal verification, and adaptive testing.

ABSTRACT

Deep learning is seeing increased use in applications from computer vision, to natural language processing, to control of autonomous systems. However, the use of neural networks in safety-critical systems will be dangerous without the development of adequate tools and techniques to assure the reliability and safety of trained networks. Recent work has made the verification of neural networks tractable. These neural network verification tools can be used to evaluate the robustness of a network, but this is usually done with the network in isolation. Realistically, neural networks are often used as part of a larger system, and so consequently, this work evaluates how the robustness of the network changes under a simple sensor fusion scheme. Early results indicate that increasing the precision of auxiliary sensors increases the robustness of the network for most training points.

I. INTRODUCTION

Motivation The use of deep learning is growing for applications ranging from computer vision, to natural language processing, to control of autonomous systems. The use of deep learning as part of safety critical systems such as autonomous vehicles and autonomous aircraft raises concerns [1, 2]. Neural networks are large, parametric function approximators that are fit using heuristic optimization methods. There is no guarantee that a trained neural network has achieved the global optimum of the loss function. Additionally, because neural networks can be very large, and involve minimal human-designed structure, it is difficult to inspect the learned weights of as neural network and understand what it has learned. Further, neural networks are known to sometimes produce unexpected behavior and to be vulnerable to adversarial examples – that is, examples that are very close to examples in the training set but are classified as having different labels with high confidence [3].

In order for safety critical systems containing neural networks to be reliable, new methods must be developed to reason about their safety. Testing can be useful for elucidating failure modes, but ultimately cannot prove the absence of counter examples [4, 5]. Formal verification tools are able to prove the absence of counter examples, but until recently, could not handle the scale of modern neural networks. Fortunately, there has been a surge in development of neural network verification tools in the last several years [6].

Background & Related Work Recently developed neural network verification tools focus on proving input-output properties of a trained network. For example, one type of property that can be proven is the following: given an input set which the inputs of the network fall into, what is the set that the outputs fall into? The reader may recognize this problem as a forward reachable set computation problem. Another type of property is essentially a backwards reachable set computation: given that the user would like the outputs of the network to fall into a given output set, what is (an approximation of) the corresponding input set? Approximations of the forward and backward reachable sets are usually computed as finding the exact sets would be intractable. Finally, these solvers can also address a third type of problem. In this third type of problem, the solver reasons about assertions on the unknown set in question, e.g. either the forward reachable set or the backward reachable set, and decides if the assertions are true or not without explicitly computing the boundaries of the set. This work focuses on the backward reachable set computation problem. Backward reachable set computation is also known as robustness computation in literature because computing the backwards reachable set for a specific label, centered around a specific training example, is essentially computing the radius around that training example for which the label does not change and thus the radius within which no adversarial examples exist.

Recent literature has provided several methods with which to evaluate the robustness of a neural network [7, 8]. Generally, these methods find the largest set centered on a given training example, under a specified ℓ_p norm, for which the label (or any form of desired output property) does not change. The minimum or average robustness over the training set may be reported as metrics with which to characterize the robustness of the network to adversarial attacks in a more generalizable way. While it is useful to be able to benchmark the network in this way, these works all evaluate the network in isolation. In contrast, when performing state estimate or providing environmental awareness for an autonomous system, it is common to fuse many sensors in order to provide a reliable solution [9]. When a neural network that performs object detection or state estimation is used within a large and complex pipeline, the end user ultimately cares about the correctness of the fused estimate. In this work, it is demonstrated that the robustness of the network changes as a function of the other signals used in the fusion when an output property is enforced over the final fused estimate, instead of over the neural network outputs. Intuitively, the hypothesis is that if the other signals are very precise, the output of the network does not have to be as precise, and consequently, the network can handle larger perturbations to its inputs as well.

Contributions In summary, the contributions of this paper are to characterize how the robustness of a neural network used in a sensor fusion scheme changes as a function of the changing precision of other inputs to the fusion algorithm. The presented methodology may be applied to any neural network, whether designed for classification or for regression.

II. PROBLEM FORMULATION

In this paper, a general methodology for evaluating how sensor fusion affects the robustness of a neural network will be developed using an automatic cruise-control problem as a guiding example. In the cruise-control problem, an autonomous vehicle follows a lead vehicle on a highway. The autonomous vehicle must estimate the distance to the lead vehicle in order to set its speed and possibly activate a collision avoidance maneuver. The autonomous vehicle has two signals with which to estimate the relative distance: a LiDAR sensor which generates many noisy returns and a relative GPS measurement, see Fig. 1.

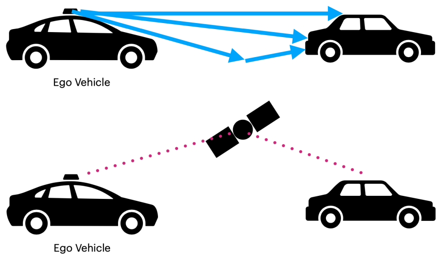

Figure 1: The cruise-control problem.

The autonomous vehicle uses a neural network to turn 10 noisy LiDAR returns into a single relative distance estimate. The output of the network is fused with the relative GPS measurement using a weighted average to produce the estimated relative distance, $\hat{d}$:

$$\hat{d} = \frac{w_1 * n + w_2 * g}{w_1 + w_2} \tag{1}$$

where n is the relative distance estimate coming out of the neural network, and g is the relative distance measurement produced using GPS. The details of how the relative GPS measurement is produced are abstracted, and it is assumed that each car may have its own algorithm for processing GPS satellite data and that each broadcasts their GPS position. It is further assumed that the resulting relative GPS distance measurement, g, lies within a continuous interval centered on the true relative distance, d, with high confidence:

$$g \in [d - p, d + p] \tag{2}$$

where p is half the width of this interval and is referred to as the precision of the signal.

The property, $\mathcal{Y}$, to be verified is that the relative distance estimate lies within ± 1meter of the true relative distance:

$$\mathcal{Y} := [d - 1, d + 1] \tag{3}$$

$$\hat{d} \in \mathcal{Y} \tag{4}$$

Problem Statement Two more precise problem statements are as follows:

1. What is the precision, p, needed for the GPS measurement such that the robustness of the network is not decreased after sensor fusion?
2. How does the robustness of the network change as the GPS sensor is made more precise (smaller p) ?

III. APPROACH

Each of the above problem statements may be formulated as a robustness query to a neural network verification tool. The weighted averaging sensor fusion can be easily encoded as it is just an affine function of the neural network output and the GPS relative measurement. Specifically, the robustness queries are formulated as optimization problems using a similar structure to that found in [7].

Finding the robustness of the network inputs alone may be formulated in the following way:

$$\min_z r \tag{5}$$
$$r \geq L_\infty(z - x) \tag{6}$$
$$n \notin \mathcal{Y} \tag{7}$$
$$n = NN(z) \tag{8}$$

The robustness is calculated about a specific training example x, and the robustness of the inputs to perturbation is defined as the largest L_∞ norm ball around x such that the network outputs lie in $\mathcal{Y}$. However, the norm ball is not maximized, instead, the complement of the property is encoded as a constraint: $n \notin \mathcal{Y}$ and the radius r is minimized, to find the threshold between $n \in \mathcal{Y}$ and $n \notin \mathcal{Y}$. In essence, the distance to the closest adversarial example, where $n \notin \mathcal{Y}$, is found. This implies there are no adversarial examples for $L_\infty(z - x) < r*$, where r^* is the optimal value.

The first problem, calculating the necessary precision, p, for the GPS measurement g, such that sensor fusion does not degrade the robustness of the network, may be formulated as the following optimization problem:

$$\min_g p \tag{9}$$
$$p \geq L_\infty(g - d) \tag{10}$$
$$L_\infty(x - z) \leq r^* \tag{11}$$
$$\hat{d} \notin \mathcal{Y} \tag{12}$$
$$\hat{d} = fusion(NN(z), g) \tag{13}$$

where r^* is the value obtained from solving the previous optimization problem with the network alone.[1] The differences are that i) now the "robustness" is calculated around the center of the GPS measurement, d ,instead of the neural network input x, ii) that now the output constraint $\mathcal{Y}$ is enforced over the fused estimate $\hat{d}$ instead of over the neural network outputs n, and iii) that there is an additional constraint, constraining the maximum perturbation that the neural network inputs, x, will be subject to.

Essentially, this optimization problem is asking: if the output property of the network, $\mathcal{Y}$, is held constant, and the robustness of the network, r^*, is held constant, what is the least precise set (largest p) that the GPS measurement can fall into? The use of an L_∞ norm means that the resulting GPS measurement may lie in the set $[d - p, d + p]$, as desired.

However, it turns out that this optimization problem may be solved trivially for the simple weighted-averaging fusion considered in this paper. By constraining the robustness of the network, r^* to remain the same, we know that the output of the network, n, must lie in $\mathcal{Y}$. Given that $\hat{d}$ is also constrained to be in $\mathcal{Y}$ for $p < p^*$; $Y := [d - 1, d + 1]$; and $n \geq 0$, $d \geq 0$, $g \geq 0$ as they are all relative distances and this is not particle physics, we can solve for $p*$:

$$\frac{w_1}{w_1 + w_2} n + \frac{w_2}{w_1 + w_2} g = \hat{d} \in [d - 1, d + 1] \tag{14}$$
$$\frac{w_1}{w_1 + w_2} n + \frac{w_2}{w_1 + w_2} g \leq d + 1 \tag{15}$$

[1]A $\leq r^*$ is used rather than a strict $<$ because the difference is not meaningful when it comes to finite-precision computer arithmetic.

Using the fact that $n \in [d-1, d+1]$, we can solve for the tightest upper bound on g by choosing $n = d+1$ (aka asking: if n is at the maximum allowed value, what is the largest g can be?):

$$g \leq \frac{w_1 + w_2}{w_2}\left(1 - \frac{w_1}{w_1 + w_2}\right)(d+1) \tag{16}$$
$$g \leq d+1 \tag{17}$$

A similar derivation follows to show the lower bound on g is $d-1$, *meaning that g falls into the same interval as $\hat{d}$ and n*, or in other words, $g \in \mathcal{Y}$. And finally, this implies that the maximum disturbance on the GPS measurement, p^*, is ≤ 1.

Now that the answer to the first problem may be easily obtained, we can address the next problem: how does the robustness of the network change when the GPS sensor is made more precise? (When p is reduced?) This may be formulated as the following optimization problem:

$$\min_z r \tag{18}$$
$$p = L_\infty(g - d) \tag{19}$$
$$p \leq p^*/a \tag{20}$$
$$r \geq L_\infty(z - x) \tag{21}$$
$$\hat{d} \notin \mathcal{Y} \tag{22}$$
$$\hat{d} = fusion(NN(z), g) \tag{23}$$

where a may be any non-negative real value, forcing the GPS signal to be more precise.

A ReLU-activated neural network was trained, and consequently, all of the above optimization problems could be encoded into a mixed-integer linear program, as ReLUs are piecewise linear functions. The ReLU activation functions were encoded using a scheme inspired by [7] where each node of the network is encoded using its own unique bound:

$$z_{i,j} \geq \hat{z}_{i,j} \tag{24}$$
$$z_{i,j} \geq 0 \tag{25}$$
$$z_{i,j} \leq \hat{z}_{i,j} - \hat{l}_{i,j}(1 - \delta_{i,j}) \tag{26}$$
$$z_{i,j} \leq \hat{u}_{i,j}\delta_{i,j} \tag{27}$$

where $z_{i,j}$ is the value of the i^{th} node in the j^{th} layer after activation; $\hat{z}_{i,j}$ is the value before activation; $\delta_{i,j}$ is a binary variable indicating activation status of the ReLU; $\hat{l}_{i,j}$ is the unique lower bound of the node before activation; and $\hat{u}_{i,j}$ is the unique upper bound of the node before activation. A visual representation of this encoding is shown in Fig. 2. Interval arithmetic was used to compute bounds on the nodes. While interval arithmetic is not the most precise for deep networks, it is very fast.

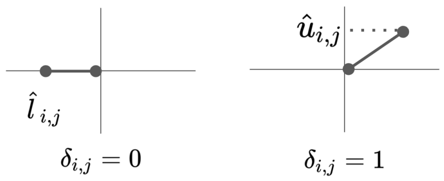

Figure 2: The encoding used for ReLUs to turn the optimization problems into mixed-integer linear programs.

IV. EXPERIMENTS

A dataset of simulated LiDAR was created, a neural network was trained on this data, and the aforementioned robustness properties were calculated over the fused output of the neural network and the abstracted GPS measurement.

Data To create simulated LiDAR data for the experiments, 10,000 true distance values were first generated using uniform random values in the range $[0, 100]$ meters. For each true distance, 10 noisy LiDAR measurements were created by adding Gaussian noise with mean 0 and standard deviation 1 and by doubling any measurement with 2% probability to simulate multipath error.

Network A network accepting 10 LiDAR measurements at once, with 2 hidden layers of 10 neurons each, and ReLU activations was trained in Flux.jl using a mean squared error loss and the Adam optimizer [10, 11].

The verification queries were implemented using components from the NeuralVerification.jl library, including their re-implementation of MIPverify [6, 7].

All code may be found at:

`https://github.com/chelseas/robust_fusion`

Has a dependency on the following unregistered package:

`https://github.com/sisl/NeuralVerification.jl`

1. Results & Discussion

The optimization problem described by equations 18 - 23 was solved for 6 different values of a corresponding to precision values for the GPS signal of $p = \{1.25, 1, 0.75, 0.5, 0.25, 0.05\}$ meters. This set of 6 optimization problems was solved for 200 examples in the training data, but for clarity's sake, the robustness vs. precision curves for just 50 datapoints are shown in Fig. 3. The baseline robustness of the neural network is demarcated by the point of each curve at $p = 1$, which is equivalent to the robustness of the network with no sensor fusion. As the precision of the GPS signal is increased (p is made smaller; as one moves to the left on the x-axis), the robustness of the network increases in a linear trend for most of the sampled data points. This confirms the hypothesis that generally, a more precise GPS signal increases the robustness of the network. However, for a few select data points, the trend is not strictly linear or even monotonic. The piecewise linear nature of the neural network means that the function relating precision of the GPS sensor and the robustness over the inputs of the network may also be piecewise linear, and it appears to be for a few of the training examples. However, it is interesting that i) most training examples seem to lie in linear regions of the network ii) for some training examples the robustness may decrease as the precision is increased (smaller p). Further investigation is required to more fully understand these phenomena.

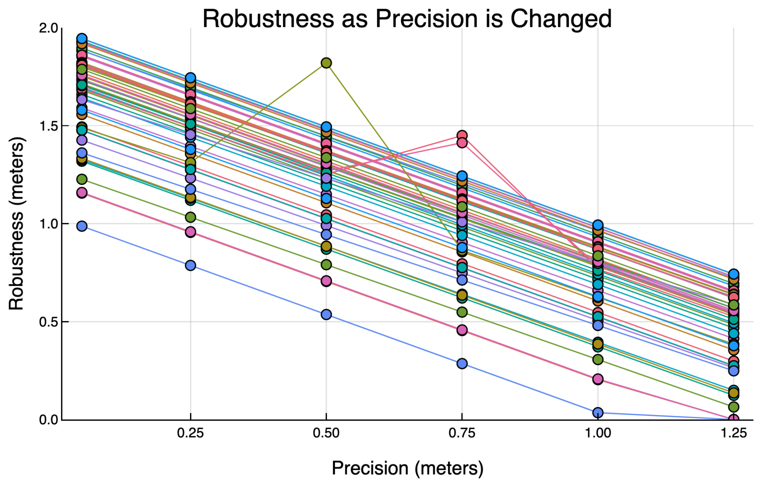

Figure 3: Robustness of the neural network as a function of precision of the GPS measurement. The baseline robustness of the neural network is demarcated by point of each curve at $p = 1$, which is equivalent to the robustness of the network with no sensor fusion.

Looking more closely at individual training examples, the least robust training data point shown can tolerate only centimeters of perturbation to the LiDAR inputs in the baseline case, without sensor fusion (p=1), but can tolerate approximately ± 1 meter of perturbation to the LiDAR measurements if the GPS measurement can be sharpened to ± 5 cm of the true relative distance. Further, the average increase from the minimum robustness computed to maximum robustness computed for a single training data point was ≈ 1.1749 meters, averaged over 196 sampled training points. These are significant increases in robustness that engineers should be cognisant of when designing sensor fusion systems involving neural networks. Further, there is a concomitant *decrease* in the robustness of the network from a GPS signal precision of $p = 1$ meter to $p = 1.25$ meters for all of the training examples shown. Similarly, a large drop in network robustness is something that system architects should monitor. Awareness of the limits of system capabilities allows for safer operations.

V. CONCLUSION

In conclusion, the robustness of a neural network that is part of a simple sensor fusion scheme does indeed vary as the precision of auxiliary sensors is modulated. Most training data points experienced an increase in robustness as the precision of the auxiliary signal was increased (made more precise) for the networks evaluated in this study. Further, the framework presented for evaluating how the robustness of the network changes under sensor fusion is general and may be used with any network. This work provides additional tools to characterize the behavior of neural networks, which as often treated as black box systems.

VI. FUTURE WORK

There are several ways in which the framework for evaluating the changing robustness of a network in a sensor fusion pipeline may be made more complex and realistic. The first direction of future effort is to experiment with deeper networks – the 2-hidden-layer network used here was purposefully chosen so as to be fast for prototyping, but now that the method has been shown to work, time can be invested in verifying larger networks. The second direction of future effort is to explore more complex sensor fusion schemes, such as adaptive weighting, where a signal is weighted inversely to its expected error. Finally, the third direction of exploration that should be pursued is to outline in more detail how such precision-robustness curves might be used in a safety or reliability monitor for the network. E.g. if the robustness is predicted to be very low by extrapolating the robustness curve, perhaps the vision sensor should be temporarily removed from the fusion pipeline.

REFERENCES

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, “End to end learning for self-driving cars,” *arXiv preprint arXiv:1604.07316*, 2016.

[2] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, “Hg-dagger: Interactive imitation learning with human experts,” in *International Conference on Robotics and Automation (ICRA).* IEEE, 2019, pp. 8077–8083.

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *arXiv preprint arXiv:1312.6199*, 2013.

[4] R. Lee, M. J. Kochenderfer, O. J. Mengshoel, G. P. Brat, and M. P. Owen, “Adaptive stress testing of airborne collision avoidance systems,” in *IEEE/AIAA Digital Avionics Systems Conference (DASC).* IEEE, 2015, pp. 6C2–1.

[5] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer, “Adaptive stress testing for autonomous vehicles,” in *IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2018, pp. 1–7.

[6] C. Liu, T. Arnon, C. Lazarus, C. Barrett, and M. J. Kochenderfer, “Algorithms for verifying deep neural networks,” *arXiv preprint arXiv:1903.06758*, 2019.

[7] V. Tjeng, K. Xiao, and R. Tedrake, “Evaluating robustness of neural networks with mixed integer programming,” *arXiv preprint arXiv:1711.07356*, 2017.

[8] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, “Efficient neural network robustness certification with general activation functions,” in *Advances in neural information processing systems*, 2018, pp. 4939–4948.

[9] F. Santoso, M. A. Garratt, and S. G. Anavatti, “Visual–inertial navigation systems for aerial robotics: Sensor fusion and technology,” *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 260–275, 2017.

[10] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[11] “https://github.com/fluxml/flux.jl.”