# Data Visualization Milestone #1
# A high-dimensional inspection tool for deep neural network safety

Kyle Matoba and Arnaud Pannatier

April 6, 2023

## 1 Introduction and Related Works

Aircraft collision avoidance systems (ACAS) are computerized systems that receive information on nearby aircraft and prescribe an action to avoid collision. ACAS are important safety checks and have gained prominence in the modern context because they represent a form of autonomy that is being extended to more sophisticated navigation.

Crucially, ACAS are physically located on aircraft, and run on certified avionics hardware. Inevitably, this limits the complexity of their logic as technology moves slowly in the airline industry. Too-simple rules can only be safe with considerable conservatism, meaning that anemic on-board hardware directly leads to wasted fuel and reduced throughput in air traffic control.

[6] showed how ACAS efficiency could be improved by a dynamic-programming based approach in which various terminal outcomes were appropriately penalized, and equilibrium behavior computed using as the optimal control of a decision process. The methods and ideas of this study were ultimately incorporated into newer ACAS protocols. Unfortunately, the space required to store the policy and the time needed to query it, represented an important impediment to its practical use.

[4] achieved a massive optimization in speed and storage by approximating the optimal policy as a deep neural networks (DNNs). In their approach, the weights and biases of a feedforward network are optimized using stochastic gradient descent to minimize the cross-entropy loss between the advisory and the model outputs. They show how this approximation can be very accurate. This DNN-based ACAS can be evaluated quickly with a forward pass through the network, and the storage requirements for this network are minimal – just the weights and biases of a small network.

We have previously looked at this data, in [7]. However, visualization there was an afterthought. Our proposed project is *entirely* concerned with visualization and generalizes the simple logic in this paper dramatically.

## 2 Problem

Regrettably, even for very accurate and capable deep neural networks, classifications that are clearly (using domain knowledge) wrong can be made at "adversarial inputs" ([2]). Put more formally: near most correctly-labeled inputs, there are many inputs that *should* receive the same class label, but receive a different prediction by a DNN. For example, within ACAS, this might mean an unexpected set of inputs that evaluate to a nonsensical advisory might be hidden within an otherwise well-behaved subset of the domain. In a safety-critical system like ACAS, any behavior that is arbitrary and difficult to explain is not suitable, even if it occurs rarely. Therefore, it is crucial to have a mechanism to detect such behavior to ensure that this valuable technique can be adopted.

[5] gives a workable mathematical approach to verifying that desired properties hold. However, this approach relies upon being able to formalize undesirable behavior in a high-dimensional space. Thus, this procedure is limited by a human's capacity to anticipate failure modes.

A more proactive approach to DNN safety is a tool to enable neural network designers to inspect their networks. Given an appropriate visualization, it may be easy to see when a DNN has failed when given an example, even if it is difficult or impossible to characterize *ex nihilo*. This meta-principle

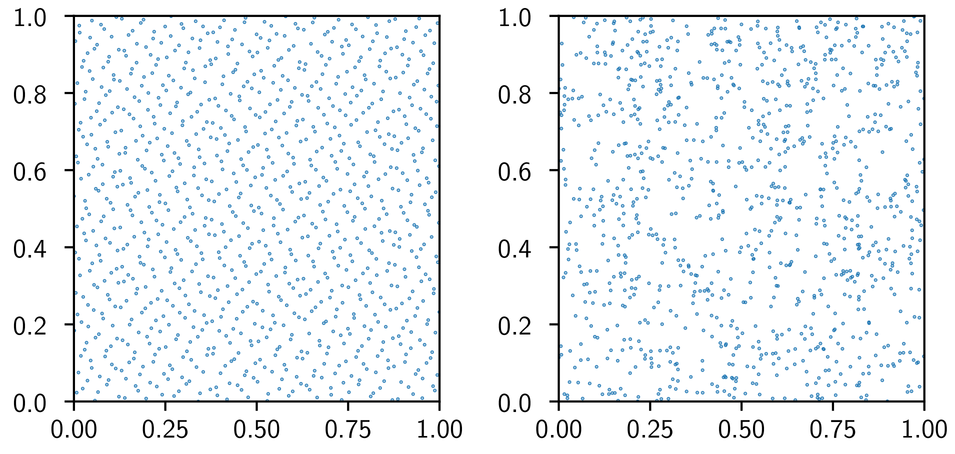

Figure 1: Left: shuffled Sobol sequence of 1024 points on the 2-d unit cube. Right: equal number of (pseudo-) uniformly distributed points. It is visually apparent that the Sobol sequence has fewer gaps.

holds across fields[1] and is what our project seeks to build.

## 3 Dataset

Our dataset is generated from the replication code for [3]. Rather than a "dataset" as such, it is a codebase that generates a dataset. It does this by formulating and solving the partially observed Markov decision process (POMDP) that characterizes the ACAS policy. This policy is of the form of a discretized mapping of three aircraft relative positions and three relative velocities to five advisories: strong left, weak left, clear of conflict, weak right, and strong right. Evasive measures in the $z$ direction are studied in other work by these authors.

The discretized optimal policy consists of aboutg 150mb of HDF5-compressed floating-point data. Like [4] we fit networks to this data, and visualize the resultant ACAS. Since pockets of errant behavior can be small, isolating regions of the domain with a good coverage will be an important component of the analysis. For this, we will use *low discrepancy sequences.* As we demonstrate visually in Figure 1, these point sets are constructed to cover a set with an optimally low deviation from uniformity.

Originally developed for improving the efficiency of high dimensional integrals, these methods exhibit superior coverage in high dimensions to naïve grids. The later chapters of [8] present a comprehensive introduction to low discrepancy sequences.

Thus, in one sentence, the "pipeline" is: solve POMDP $\to$ raw data $\to$ trained DNN ACAS, which is combined with a low-discrepancy sequence to give a representative overall picture of the model's behavior.

This approach is applicable to any problem where continuously-valued data living on a compact domain is being used to classify some outcome. [9], for example, is able to develop fast and energy efficient deep-learning based walking and trotting algorithms for a quadriped robot in this way. If time permits, we could look at extending our methodology to uncover unexpectedly incorrect behavior from otherwise applications.

## 4 Exploratory Data Analysis

Figure 2, reused from [7], shows a DNN-based ACAS projected down to the XY plane, with each advisory in a different color. Our approach would be to scale this simplified visualization up to multiple dimensions in an interactive manner. Called an encounter plot in the aeronautics world, this plot gives a very information-dense look into the preimage of an ACAS.

[1]The phrase "I know it when I see it" – see `https://en.wikipedia.org/wiki/I_know_it_when_I_see_it` – was coined by an influential legal scholar to describe situations where a comprehensive definition is difficult, but judging individual instances is trivial.

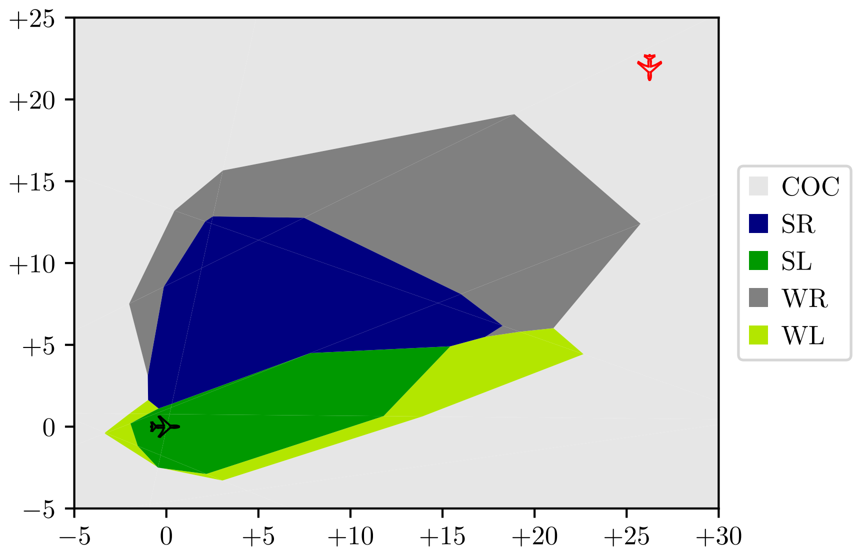

Figure 2: An encounter plot showing the optimal action (strong left, weak left, clear of conflict, weak right, and strong right) at each $(x, y)$ distance configuration for a fixed angle of approach, indicated by the perpendicular orientation of the red (intruder) aircraft. Distances are measured in kilofeet.

Our inspiration is [1]'s approach of continuously varying the "angles" of the projection. Where Figure 2 chooses particular values of some variables to condition upon, a much more comprehensive approach would let the user navigate the projection by varying parameters of the visualization. As one example Figure 3 gives our independent implementation of this idea.

Just as a 2-d drawing of a 3-d cube appears as linked boxes, the five-dimensional unit cube appears as many nested boxes. This visualization uses first two "perspective projections" $\mathbb{R}^d \to \mathbb{R}^{d-1}$

$$(x_1, x_2, \ldots, x_{d-1}) \mapsto (x_1, x_2, \ldots, x_{d-1})/(d - x_d)$$

to reduce the dimension from 5 to 3 then uses a camera transform with orientation $\theta$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

to produce a representation that can be plotted in Matplotlib. Adjacency in the original vertices is preserved in the projection, and all edges are plotted in different colors. Figure 3 plots this for $\theta \in \{0, \pi/9, \ldots, 8\pi/9\}$.

The visualization of $d > 2$-dimensional spaces is a well-studied area of computational geometry and computer graphics. We anticipate reviewing this literature for more sophisticated and flexible ways of parameterizing this relationship.

One thing we quite liked was Bokeh's "linked panning" which was introduced in Slide 11 of Lecture 5. It is especially useful to our use case as tiling the plots in a linked manner is exactly what is needed to drill down into a suspected oddity in the behavior of a neural network preimage. In anticipation of using the functionality, we have begun to incorporate our logic into Bokeh.

## Works Cited

[1] Iain Barr. *The challenge of verification and testing of machine learning.* May 2016. URL: `http://www.degeneratestate.org/posts/2016/May/01/visualising-hypercubes/`.

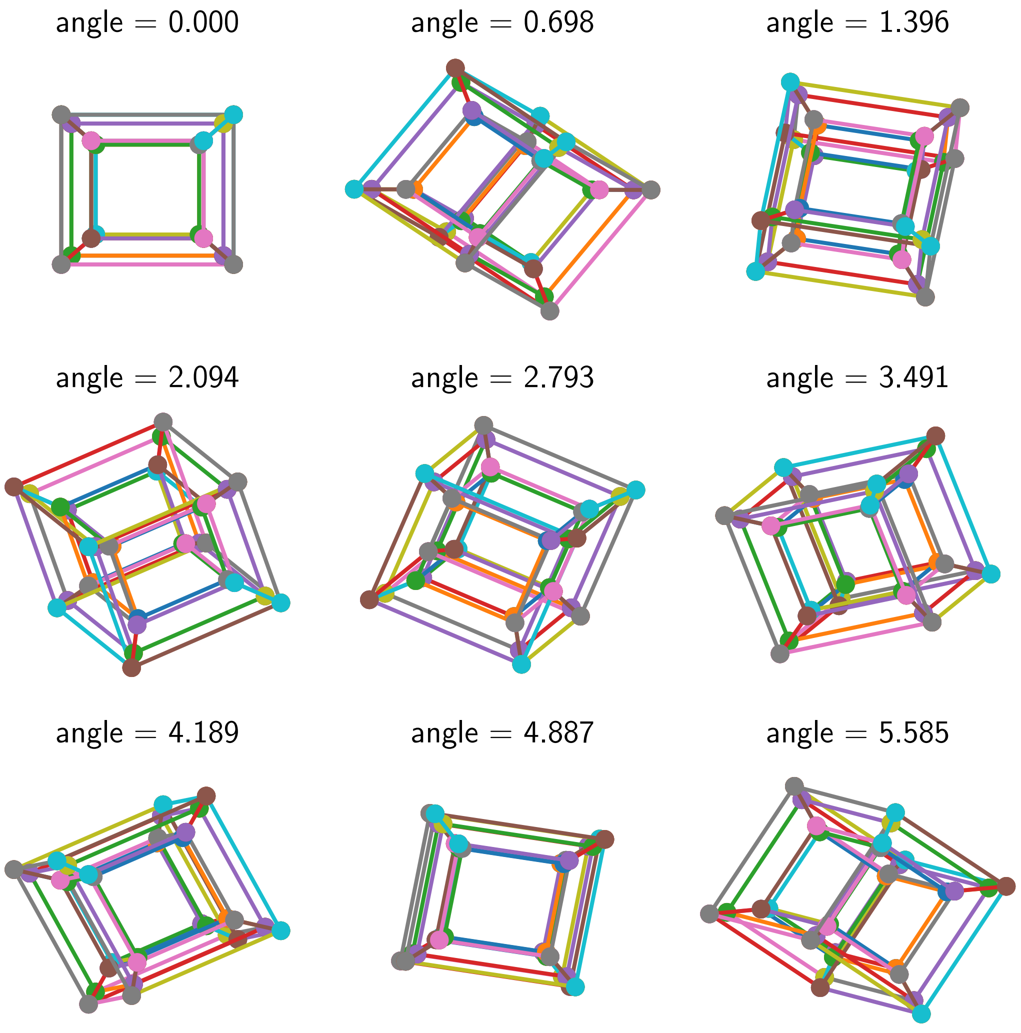

Figure 3: Nine projections of the five-dimensional unit cube.

[2] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *arXiv e-prints* (2014). URL: `http://arxiv.org/abs/1412.6572`.

[3] Kyle D. Julian and Mykel J. Kochenderfer. "Guaranteeing safety for neural network-based aircraft collision avoidance systems". In: *IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)* (2019). URL: `https://doi.org/10.1109/DASC43569.2019.9081748`.

[4] Kyle D. Julian, Mykel J. Kochenderfer, and Michael P. Owen. "Deep Neural Network Compression for Aircraft Collision Avoidance Systems". In: *Journal of Guidance, Control, and Dynamics* 42.3 (2019), pp. 598–608. URL: `https://doi.org/10.2514/1.G003724`.

[5] Guy Katz et al. "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks". In: *Computer Aided Verification*. Ed. by Rupak Majumdar and Viktor Kunčak. Springer International Publishing, 2017. URL: `https://doi.org/10.1007/978-3-319-63387-9_5`.

[6] Mykel J. Kochenderfer and James P. Chryssanthacopoulos. "Robust Airborne Collision Avoidance through Dynamic Programming". In: *Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-371* (2011). URL: `https://www.ll.mit.edu/sites/default/files/publication/doc/2018-12/Kochenderfer_2011_ATC-371_WW-21458.pdf`.

[7] Kyle Matoba and François Fleuret. "Exact Preimages of Neural Network Aircraft Collision Avoidance Systems". In: *Machine Learning for Engineering Modeling, Simulation, and Design Workshop at Neural Information Processing Systems 2020*. Nov. 2020. URL: `https://ml4eng.github.io/camera_readys/24.pdf`.

[8] Art B. Owen. *Monte Carlo theory, methods and examples*. 2013. URL: `https://artowen.su.domains/mc/`.

[9] Jie Tan et al. "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots". In: *Proceedings of Robotics: Science and Systems*. Pittsburgh, Pennsylvania, 2018. DOI: `10.15607/RSS.2018.XIV.010`.