Statistical learning framework for safety and failure analysis of a DNN-based autonomous aircraft system

Yuning He*, Huafeng Yu†, Guillaume Brat‡, and Misty Davies§
* ‡ § NASA Ames Research Center, Moffett Field, CA 94035, USA
†Boeing Research & Technology, Huntsville, AL 35808, USA
Email: *yuning.he@nasa.gov, †huafeng.yu@boeing.com ‡guillaume.p.brat@nasa.gov, §misty.d.davies@nasa.gov

***Abstract*—Deep Neural Networks (DNNs) and Machine Learning technology is increasingly used for safety-critical applications in the Aerospace domain. To ensure safe operations, the DNN and the system must undergo rigorous verification and validation, including advanced statistical analyses. Performance and safety of the DNN and system behavior must not only be analyzed for the nominal case, but under numerous off-nominal and failure cases.**

In this paper we will describe how our statistical learning framework SYSAI can efficiently perform such analyses using the tool's unique combination of advanced learning modeling and statistical analysis techniques. SYSAI can effectively explore the high-dimensional state and failure space of the system under test; geometrical shape detection of safety regions and boundaries support explainability of the results to the designer.

In this paper, we report experiments and results obtained with a vision-based DNN control system (ACT) that is capable of autonomously steering an aircraft down a runway.

I. Introduction

ACT (Autonomous Center Line Tracking) is a system developed by our industrial partner for autonomous aircraft taxiing. Its central component is a vision-based Deep Neural Network (DNN) that takes inputs from a camera mounted on the wing of the aircraft. The DNN continuously estimates the position and orientation of the aircraft with respect to the runway center line and uses this information to steer the aircraft down the runway.

Safety-critical DNN applications like ACT are increasingly used in the Aerospace domain. They are powerful but very complicated and hard to develop and understand. Verification and Validation (V&V) must ensure that the software and system is working properly and safely in all circumstances. Thus, V&V is crucial for such a safety-critical AI component. However, traditional software and assurance processes are not suitable for Deep Neural Networks.

In this paper, we will describe our statistical learning tool SYSAI (System Analysis using Statistical AI), which is a flexible statistical learning framework for V&V and analysis of complex and high-dimensional Aerospace systems with AI components. SYSAI connects to the system under study using a customizable interface component, enabling the user to connect and analyze a multitude of systems written in C/C++, Python, Java, R, or Simulink/Matlab.

In this paper, we use the ACT system as our case study. ACT consists of the DNN component and a controller for steering the aircraft. The XPlane software provides the simulator for the aircraft and the environment. Using the advanced statistical learning modeling and AI techniques, SYSAI can provide dynamic online learning of a complex high-dimensional system. This means that the learning model will automatically update itself and evolves as new data come in. That enables SYSAI to also analyze complex system behaviors as the system changes over time. Using active learning and algorithms from Computer Experimental Design, SYSAI can simultaneously and efficiently track and learn the behavior of a large number of system parameters and variables.

Our SYSAI framework provides functionality for numerous analysis and V&V tasks, including statistical analysis of training and test data, safety-envelope analysis, property checking, time-series analysis, and intelligent test-case generation.

In this paper, we focus on the tool's capabilities to perform system and safety analysis for a complex system in presence of failures.

- During development, most AI systems are trained with data sets reflecting nominal behavior only and perhaps some corner cases. How will the system behave in the presence of failures?
- In a complex system, there is a multitude of failure possibilities, which can have a small or large impact on the system performance and safety. Failure impact is not independent from each other, so we are facing a high-dimensional analysis problem.
- Failures are often "parameterized" and the parameter values can have a big impact on performance and safety. E.g., location and size of a blind spot of a camera can hardly be noticed or can lead to a severely diminished system performance.
- Results of such analyses and regions of high sensitivity in an usually high dimensional space is, in most cases not human-readable. SYSAI performs geometric shape estimation and thus can provide valuable, human readable feedback to the designer.

The rest of the paper is structured as follows: Section II

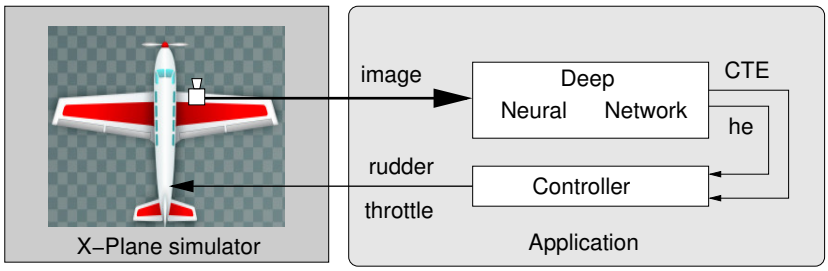

Fig. 1. ACT architecture

provides a background and overview of the ACT system, which is used as our case study throughout the paper. Section III focuses on SYSAI, its architecture, algorithms, and capabilities. In Section IV we briefly illustrate system safety and threshold analysis for the ACT system under nominal conditions. Section V focuses on the analysis of a complex DNN-based system under failures. We discuss several classes of failures and off-nominal environmental conditions and describe results of experiments on camera failures: cameras that produce images with a wrong setting of brightness and contrast, or dirt particles/insects on the camera lens that can obscure parts of the image. Section VI presents related work and Section VII concludes and discusses future work.

II. Background: ACT

Our industrial partner's ACT (Autonomous Center Line Tracking) component is used as a case study to demonstrate SYSAI. ACT is one of the most important ground operations for Unmanned Aerial Systems. The core component of ACT is a Deep Neural Network (DNN) that takes images as inputs from cameras mounted on the aircraft's wings (Figure 1). The DNN component continuously estimates the position and orientation of the aircraft with respect to the runway center line. These values are the cross-track error CTE in meters, and the heading error he in degrees, respectively.

A simple fixed-gain proportional controller uses this information to produce rudder control signals. While the aircraft is on the ground, the rudder is linked to the front-wheel. Therefore this signal can be used to steer the aircraft left and right. A separate controller is used to control the throttle setting so that the aircraft is rolling with a constant, low speed.

The X-Plane Flight Simulator[1] is used as simulation environment for our experiments. A simulated camera takes information from the simulator display; the control signals for throttle and rudder are sent to the X-Plane simulator using the programmatic interface NASA XPlane Connect.[2] SYSAI, which will be described in the next section is obtaining information from the DNN and the control system. Through XPlane Connect, SYSAI can set numerous system and environmental parameters, produce simulated failures for our experiments, and can obtain ground-truth information.

[1]www.xplane.com

[2]https://github.com/nasa/XPlaneConnect

The DNN is a multi-layer feed-forward network with ReLU nodes. Several different architectures for this network have been defined and trained. The DNNs are implemented using the TensorFlow framework [1] and have been trained on data that have been obtained with the simulated aircraft within the X-Plane simulator. Note that in this application, DNN is not learning any time-series data. Rather the DNN is learning a mapping between the input image (showing a part of the runway) and the corresponding CTE and he values.

III. The SYSAI Analysis Framework

Figure 2 shows the high-level architecture of our SYSAI testing and analysis framework. On the left-hand side, we have the "system under test" (SuT), which in our case is the ACT system and the XPlane simulator, as described in the previous section. The SuT is executed given a set of parameters and initial conditions provided by the statistical learning model of SYSAI. The result of the test run, which could be a binary safe/not-safe information, a single value (e.g., CTE_{max}), or an entire time series is provided back to SYSAI. These data are then used to incrementally construct our statistical model.

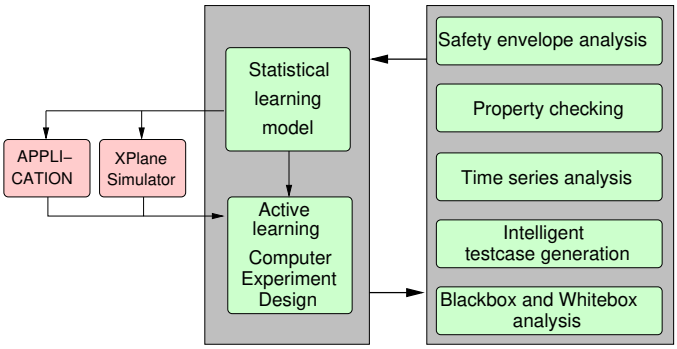

Fig. 2. SYSAI architecture

The interface between SYSAI and the SuT is designed to be very small and generic, so that systems implemented in R, Matlab, Java, or Python can be connected easily. For the representation and construction of the statistical SYSAI model, we are using Dynamic Regression Trees (DynaTrees [2], [3]), a dynamic Gaussian process model based upon Particle Filters. DynaTrees are regression and classification

learning models with complicated response surfaces in on-line application settings. DynaTrees create a sequential tree model whose state changes over time with the accumulation of new data, and provide particle learning algorithms that allow for the efficient on-line posterior filtering of tree-states. A major advantage of DynaTrees is that they allow for the use of very simple models within each partition. The models also facilitate a natural division in sequential particle-based inference: tree dynamics are defined through a few potential changes that are local to each newly arrived observation, while global uncertainty is captured by the ensemble of particles.

This surrogate model is initialized with available training data and incrementally refined using candidate data points that are produced by our active learning module. It evaluates the current surrogate model using a customized active-learning heuristics and suggests candidate data points that provide most information for model refinement. For these candidate points, the ground truth is obtained by executing the SuT.

SYSAI features customizable heuristics that allow the active learning to focus on particular characteristics of the model. Classical algorithms like ALM [4] or ALC [5] focus on under-explored regions in general of the domain space. Inspired by [6] and work on contour finding algorithms, we loosely follow [7] and define our boundary-aware metric boundary-EI [8], [9] that puts the focus of the search into "interesting" and potentially "troublesome" areas near safety boundaries. Here, our surrogate model exhibits substantially more details than in other areas that are not of interest. This exploration is guided by the selected active learning heuristics and is able to cover the entire input space with a low number of data points.

The SYSAI framework and the underlying models and algorithms are described in detail in [10]. SYSAI has been used for the analysis of several complex and safety-critical aerospace systems [8]. SYSAI, in general, supports the following analysis tasks (for details see, e.g., [10], [11]:

- *statistical analysis of training and test data:* SYSAI can perform a detailed statistical analysis of the data sets used for training and evaluation of the DNN.
- *safety-envelope analysis:* our framework can perform automatic analysis of the safety-envelope, which indicates under which operational conditions the system is behaving safely or not. Geometric shape modeling does not only identify but also characterizes regions with similar behavior and describes those regions in easy to understand geometrical terms. SYSAI thus helps to make the system more explainable.
- *property checking:* our tool supports the automatic checking and analysis of safety and performance requirements.
- *time-series analysis:* SYSAI can perform advanced time-series analysis in a high-dimensional parameter and state space. This analysis provides a deeper understanding of the system behavior and its dynamics. The tool also supports event prediction.
- *intelligent test-case generation:* SYSAI can efficiently generate relevant test cases in high-dimensional spaces.

In this paper, we focus our analysis on system behavior under failures. This means that the high-dimensional input space for our analysis can contain parameterized models of system or component failures, as well as off-nominal environmental and operational conditions.

IV. System Analysis with SYSAI

SYSAI is typically used to perform threshold and safety region analysis under nominal operational conditions. The goal of these analyses is to (a) characterize safety regions of the system in a high-dimensional state space, and (b) to establish reasonable safety thresholds for the system behavior. In both cases, SYSAI constructs a learning model of the system with a small number of test cases. The safety-boundaries then can be characterized with (domain-specific) geometric shapes to facilitate system explainability.

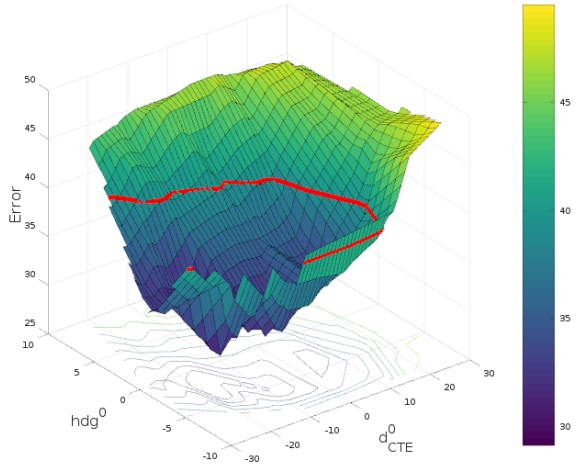

Fig. 3. Safety-envelope: estimated CTE value and threshold (red) over initial position and heading of the aircraft

Figure 3 shows the result of an experiment with SYSAI on ACT to obtain suitable thresholds. As the safety-measure, for which a threshold is sought, we define the maximal distance of the aircraft from the center line during the entire run of the system. The system is safe if $|CTE_{max}| < \theta_{CTE}$ for some threshold θ_{CTE}. In this experiment, SYSAI evaluates the maximum distance over the initial position and heading of the aircraft at the beginning of the runway. d^0_{CTE} denotes the initial distance from the centerline (negative values are to the left of the center lines); hdg^0 denotes the initial aircraft heading with respect to the runway heading.

The surface in Figure 3 shows $|CTE_{max}|$ over different values of the initial parameters d^0_{CTE} and hdg^0. This surface has been interpolated for visualization purposes. The red line indicates the safety boundary at a certain threshold. Initial parameters inside this boundary line correspond to safe runs. Outside the boundary, the $|CTE_{max}|$ might surpass the threshold and might cause the aircraft to drive off the runway. More details are described in [11].

V. System Analysis with SYSAI under Failures

Figure 3 concerns the behavior of the system under nominal conditions. For a complex AI-based system, however, it is important that the system does not only operate safely under nominal conditions, but also under off-nominal conditions or failures.

In out ACT case study, we can distinguish two classes of off-nominal situations: those produced by some hardware and/or software failure (e.g., dirt on the camera lens, blocked actuator), or an unusual or adverse environmental effect (e.g., bad visibility or wind). What makes the analysis particularly hard is that (a) failures and adverse conditions can occur at the same time, and (b) depending on the actual failure (e.g., location of the dirt particle on the lens), the influence on performance and safety can be minimal or dramatic. We are using SYSAI to perform safety and robustness analysis on a number of failures and off-nominal conditions.

For the forward-facing camera (see Figure 1), we consider a mis-adjustment of the camera, which might deliver images that are too dark or too bright, or images with a contrast setting that is too low. The latter failure would result in foggy images. We also consider dirt on the camera lens occluding different parts of the image. Such a failure could easily be caused by rain drops, insects, or dirt particles. Additional failures that could be of interest include: mis-alignment of the camera, intermittent camera failures, image drop-outs, or software issues.

Due to the given ACT model (Figure 1), we did not consider software errors that might lead to diminished safety and performance of the ACT system. Failures in the actual aircraft system include: actuator failures (loose connection to front wheel, stuck front wheel), failures in the throttle control or engine, blocked wheel brakes, or a blown tire.

Another category of off-nominal situations include environmental conditions, like fog, wet or snow-covered runway, of heavy winds. Other situations are more operations related, e.g., behavior if the aircraft runs out of gas, an engine fire occurs, or the aircraft is crashing into another aircraft. Such scenarios can either be modeled by SYSAI, or spontaneous occurrences during simulation runs can be characterized by SYSAI and are embedded into the learning model.

In this paper, we illustrate the capabilities of SYSAI by looking at camera failures related to brightness and contrast, and by failures induced by dirt on the lens.

A. Image Brightness and Contrast

In this section, we will present results of experiments on camera-related failures, brightness and contrast. Figure 4 shows typical camera images obtained by the camera of SYSAI. Figure 4A, B correspond to images on a clear day during daytime hours. Figure 4C shows the situation if the contrast setting of the camera is set too low: the image appears washed out, the runway and its marking are hard to see, so it is expected that the ACT performance is degraded. We call this effect also fogginess. Finally, Figure 4D corresponds to an image where the camera's brightness setting is wrong.

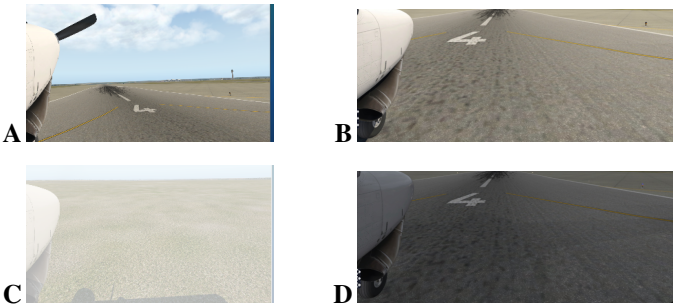

Fig. 4. Typical camera images on a clear day (A), nominal brightness (B). With a low contrast setting, washed-out images (C) are returned; (D) was a picture taken where camera brightness is set too low.

We are studying with SYSAI, how both parameters (image brightness and contrast) influence the prediction quality of the DNN. For this experiment, we simulate the fogginess of an image by using its relative contrast in a range of 0 to 1, where 0 is the nominal parameter settings and 1 corresponds to a gray image without any contrast. Similarly, the normalized brightness ranges from 0 to 1. In our experiment, we run SYSAI to explore the effects of these two camera failure parameters.

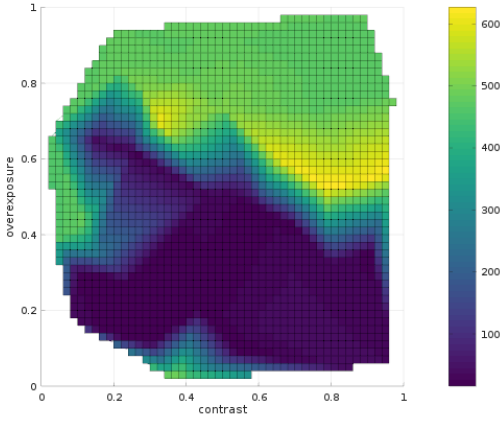

Fig. 5. DNN performance for different values of image over-exposure and brightness

Figure 5 shows a projection of $\max |CTE|$ over normalized brightness and contrast. Each run was comprised of 1500 steps, i.e., 1500 consecutive images were processed by the DNN. Blue regions in Figure 5 correspond to small maximal deviations, bright green and yellow regions correspond to parameter settings resulting in high values of CTE. The nominal case is located at the lower right corner; the irregular shape of this surface is an artifact of the interpolation carried out for visualization. While it is obvious that high settings of brightness results in over-exposed images and a low DNN performance, there is no linear boundary. The non-linear shape of this heat map could provide feedback to the designers of the system.

Figure 6 shows how such situations lead to a failure of ACT causing the aircraft to ultimately drive off the runway: the figure shows top view of the runway area. Hence the axes are labeled with longitude and latitude; the centerline of the runway is shown as the diagonal line in the plot. The aircraft is starting its run on the lower left corner, always exactly at the same position and heading and the simulation is executed for 1500 time steps. Colored dots show the end-points of each of the runs. The end-points of successful runs, i.e., runs where the given threshold θ_{CTE} was not violated, are shown in green. They are all located near the end of the runway and close to the centerline. Unsuccessful runs (red dots) have caused the aircraft to, in most cases, to veer to the port side and ultimately drive off the runway into the grass. The red dot at the beginning of the runway was caused by a run, where the engine ran out of gas and the aircraft remained stationary.

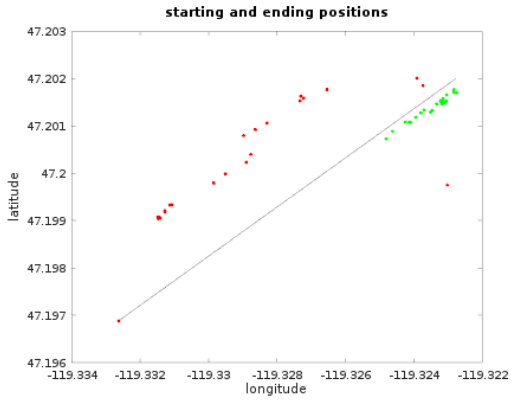

Fig. 6. End positions of trajectories for different values of contrast and brightness

This experiment reveals that the ACT system reacts very sensitive to settings of brightness and contrast of the camera. This also becomes evident from another SYSAI experiment described in [11], where nominal runs were executed with a clear weather setting at different times of the day.

Figure 7 shows the probability of success of a run (in percent) for different times of the day. In the base case (9AM), which was also used for training the DNN, the probability of success is relatively high. It remains high between around 9AM to 1PM. A lower success rate for earlier times can be attributed to the camera images, which are darker at that time of the day. The sharp drop in success rate after 1PM is due to the fact that now parts of the wing casts a shadow on the runway, which seems to "confuse" the DNN and leasing to very low success rates.

B. Dirt on Camera Lens

In this experiment, we evaluated, how a piece of dirt or an insect on the camera lens can influence the behavior of ACT. This situation was modeled by superimposing a black square of size d pixels at position x, y of the image. SYSAI then was used to analyze the change of the maximal CTE value per run in that case. Figure 8 shows the heat map projection into the x and y coordinates of the dirt particle, keeping $d = 10$. In most areas of the image, the dirt particle did not have much influence, except for some areas on the left-top of the image (shown in yellow). A superposition of that information on a camera image (Figure 8B) shows the most sensitive areas. Even a relatively small disturbance a drastic decrease in performance of ACT. The black square in Figure 8 shows a modeled dirt particle with $d = 10$. Not surprisingly, the most sensitive areas are close to the horizon and near the front wheel of the aircraft. This SYSAI result gives an easy to understand feedback on the robustness of ACT.

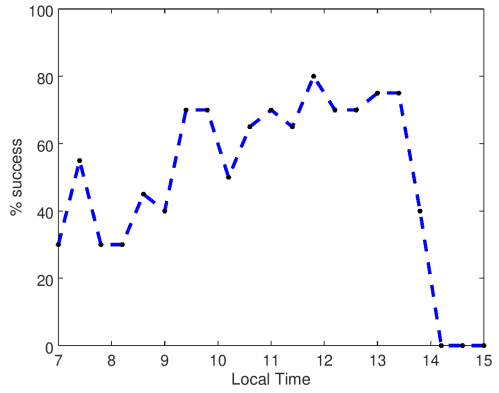

Fig. 7. Influence of brightness and shadow (caused by different times of the day) on the ACT success rate (in %). Threshold for CTE is 40ft. (from [11])

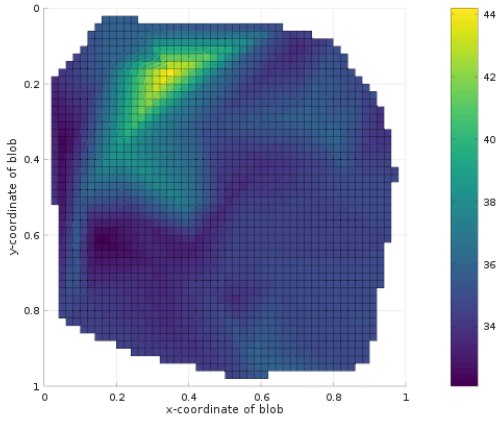

Fig. 8. Robustness of ACT: dirt on camera lens

VI. Related Work

While there is a large number of approaches for the analysis and V&V of Deep Neural Networks [12], [13], most approaches aim for operations in a nominal range and assumes that the training data adequately cover all relevant scenarios.

More closely related to our framework are adversarial techniques (e.g., [14]). They try to identify scenarios, where the DNN produces erroneous results, but they do not go along

Fig. 9. Robustness of ACT: dirt on camera lens

modeled failure classes, like the camera-related failures in this paper.

The estimation of safety threshold can be done using a variety of techniques (e.g., [11]); in many cases, however, these thresholds are defined as axis-aligned hypercubes. The richer geometric shapes, which are modeled by our SYSAI tool can provide in-depth feedback to the designer and can potentially be used for definition and analysis of operational bounds as found, e.g., in the ASTM3269 Runtime Assurance (RTA) architecture [15].

VII. Conclusions

In this paper, we have described out statistical framework SYSAI and its application for the analysis of a safety-critical neural network-based aircraft system (ACT) under failures.

SYSAI is well suited to efficiently explore a high-dimensional failure space, which makes it possible to analyze the impact not only of a single failure, but even combinations of failures on safety and performance of the system under test.

In this paper, we showed results of experiments on the ACT, where hardware related failures (e.g., dirt on the camera) as well as off-nominal environmental conditions (e.g., lighting conditions of the runway) were investigated. SYSAI has a small and flexible interface to the system under test and can carry out the experiments automatically, keeping the number of test-runs very low. Projections of results into a 2 or 3-dimensional space for visualization as well as geometric shape estimation yield human-understandable explanations of the system behavior and can provide feedback to designers and/or V&V engineers.

Future work will include extensions of SYSAI to model system behavior with time-series data and to extend the modeling framework for systems that can perform online learning to mitigate failures.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[2] M. A. Taddy, R. B. Gramacy, and N. G. Polson, "Dynamic trees for learning and design," *Journal of the American Statistical Association*, vol. 106, no. 493, pp. 109–123, 2011.

[3] R. Gramacy and N. Polson, "Particle learning of Gaussian process models for sequential design and optimization," *Journal of Computational and Graphical Statistics*, vol. 20, no. 1, pp. 467–478, 2011.

[4] D. J. C. MacKay, "Information–based objective functions for active data selection," *Neural Computation*, vol. 4, no. 4, pp. 589–603, 1992.

[5] D. A. Cohn, "Neural network exploration using optimal experimental design," *Advances in Neural Information Processing Systems*, vol. 6, no. 9, pp. 679–686, 1996.

[6] D. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black box functions," *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998.

[7] P. Ranjan, D. Bingham, and G. Michailidis, "Sequential experiment design for contour estimation from complex computer codes," *Technometrics*, vol. 50, no. 4, pp. 527–541, 2008.

[8] Y. He, "Online detection and modeling of safety boundaries for aerospace applications using active learning and bayesian statistics," in *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*. IEEE, 2015, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7256526

[9] ——, "Variable-length functional output prediction and boundary detection for an adaptive flight control simulator," Ph.D. dissertation, University of California at Santa Cruz, 2012.

[10] Y. He and J. Schumann, "A framework for the analysis of deep neural networks in aerospace applications using bayesian statistics," 2020.

[11] Y. He, H. Yu, G. Brat, and M. Davies, "System and safety analysis for autonomous center line tracking with sysai," in *Scitech 2022*, 2022.

[12] M. Borg, C. Englund, K. Wnuk, B. Durán, C. Levandowski, S. Gao, Y. Tan, H. Kaijser, H. Lönn, and J. Törnqvist, "Safely entering the deep: A review of verification and validation for machine learning and a challenge elicitation in the automotive industry," *CoRR*, vol. abs/1812.05389, 2018. [Online]. Available: http://arxiv.org/abs/1812.05389

[13] J. Zhang and J. Li, "Testing and verification of neural-network-based safety-critical control software: A systematic literature review," *Information and Software Technology*, vol. 123, p. 106296, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584920300471

[14] S. Qiu, Q. Liu, S. Zhou, and C. Wu, "Review of artificial intelligence adversarial attack and defense technologies," *Applied Sciences*, vol. 9, no. 5, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/5/909

[15] P. Nagarajan, S. K. Kannan, C. Torens, M. E. Vukas, and G. F. Wilber, *ASTM F3269 - An Industry Standard on Run Time Assurance for Aircraft Systems*, 2021. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2021-0525