2024 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
MODELING, SIMULATION, PROTOTYPING & VALIDATION (MSPV) TECHNICAL SESSION
AUGUST 13-15, 2024 - NOVI, MICHIGAN

AI/ML Digital Twins of Hardware-in-the-Loop Systems

Wesley N. Colley[1], PhD, Joel Banyai[1], Joshua Gordy[1], Matthew Mills[1], Randall Warren[1]

[1]Torch Technologies, Inc., Huntsville, AL

ABSTRACT

Proprietary, black box, and other hard-to-model subsystems are a leading source of schedule and labor cost across simulation supported analysis and lifecycle management. Using AI/ML technologies to rapidly develop and deploy digital twins of Hardware in the Loop (HWIL) and software systems reduces the Non-Recurring Engineering (NRE) in Modeling and Simulation (M&S) and supports validation of existing software digital twins. This approach also allows for portability of obsolete or proprietary components into a broader range of simulations or applications without exposing critical technologies. We present results of multiple case studies applying AI to black box components of interest to the ground vehicle community.

Citation: W. Colley, J. Banyai, J. Gordy, M. Mills, R. Warren, "AI/ML Digital Twins of Hardware-in-the-Loop Systems," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2024.

1. INTRODUCTION

Engineering of complex systems has extensively leveraged computer models since the 1990s at the latest. The Boeing 777 was the first major commercial airliner designed entirely digitally and underwent initial assembly in 1993 [1]. The success of the 777 represented a turning point in large engineering projects: digital design was here to stay. Since that time, digital models have come to represent not only the physical design, but also the detailed behaviors and interactions of subsystems and components as they operate within the system. This allows engineers and stakeholders to develop trades on different subsystem choices in the context of overall system measures (performance, efficiency, cost, etc.) at much lower cost than full-scale real-world testing would require. In practice, however, the engineering community has found that these digital models often present disparate levels of fidelity, differing interoperability requirements and inconsistent execution timing, among other incompatibilities that have made full system digital representations difficult to implement.

The concept of a Digital Twin has emerged in the last several years [2][3] as a potential path forward. A Digital Twin takes the concept of digitally modeling physical systems and subsystems to its logical conclusion: a fully synthetic representation that reproduces with high fidelity the same outputs from the same inputs as the physical system/subsystem with essentially identical performance (e.g., execution timing). With such a capability, systems and systems-of-systems could

be modeled with all relevant interactions in fully realistic real-world scenarios. Systems engineers could finally exercise a system at high-fidelity across large swaths of parameter space that are simply not accessible with costly live/real-world elements. Such broad sweeps across the functional domain would elucidate failure modes and quantify performance at unprecedented levels. Acquisition professionals would be informed by truly robust quantitative trades.

But a significant challenge arises when subsystems or components present to the modeler as either pure hardware or black-box (e.g., proprietary) software. In such cases, the modeler must essentially reverse-engineer the subsystem with comprehensive and arduous modeling work using data from custom test harnesses in Software-in-the-Loop (SWIL) and HWIL environments, accompanied by substantial subject-matter expertise. These are expensive, time-consuming processes, but the benefits of a true Digital Twin remain so valuable that such efforts are common in the large-scale engineering industry. [4]

In 2023, we proposed the idea that the modeling component of the reverse-engineering task could be greatly accelerated for some subsystems using AI/ML techniques. AI/ML approaches present a key advantage over traditional modeling & simulation techniques: whereas a traditional M&S developer would need to be an expert in the subject matter domain for the subsystem and would typically generate the model largely from custom-developed code, the AI/ML modeler instead regards the subsystem principally as a black box which simply receives input data and produces output data, a process which can be modeled using mostly existing off-the-shelf AI/ML toolboxes. In its purest form, the AI/ML model has just one purpose: to reproduce the outputs from the inputs very faithfully without any "knowledge" of how the subsystem works internally or what the data represent. Of course, an expert on the subsystem adds great insight into which behaviors are most important, which inputs best represent the key operating modes on which to train the model and which incorrect outputs are the most consequential. So, the expert can provide a guiding hand to ensure the best and most useful outcomes as the AI/ML training process is established and as it is executed (as we will show in Section 2.3).

In this whitepaper, we report on three case studies in which we have used AI/ML techniques to develop Digital Twins of subsystems: 1. Reproduction of the computer vision system within a prototype Unmanned Ground System (UGS) rover developed here at Torch Technologies, 2. Emulation of detailed physical responses of a tactical missile to step guidance commands under autopilot control, and 3. Replication of engine tuning parameters from a Power Control Module in a commercial production automobile. Our purpose is to demonstrate that we can rapidly deploy these AI/ML representations as high-quality Digital Twins of real-world HWIL and/or software subsystems, therefore providing a proof-of-principle that AI/ML techniques can dramatically reduce the time and cost of Digital Twin development, thereby enabling the ideals and aims of the Digital Twin paradigm.

2. Case Studies

Before exploring AI/ML digital twin applications, it was first necessary to identify potential use cases. A challenge quickly emerged: if a problem was too simple, it did not justify the effort inherent to training an AI/ML model, but if it was too complex, it was beyond the scope of the project. For instance, a low-fidelity atmospheric density vs. altitude model presents as a simple look-up table. There is no reason to apply AI/ML to this model—just use the look-up table. On the opposite side of the spectrum, large language models, such as Chat GPT, are far beyond the scope of our budget and compute resources. For complex engineered systems, such as military hardware, a trade-space between the expense and effort of training the AI/ML vs. that of using the original system within an HWIL setting develops. We have therefore

sought to identify systems in a complexity "sweet spot" that are complex enough to justify an AI/ML approach, but not so complex as to make the training process and/or resulting model impractical. With this reasoning, we started with an appropriately complex system that presented minimal technical risk and have grown outward from there.

Our first effort (Section 2.1) was to create a digital twin of a simulated image classifier onboard an unmanned ground system (UGS) rover. This system presented very low risk, because we had access to the rover simulation to produce training data, and to the AI/ML image classifier techniques used by the actual rover.

Our second study (Section 2.2) involved very different physical and modeling domain in the form of missile response to guidance commands, as represented in a high-fidelity six-degree-of-freedom missile simulation. This work necessarily involved completely different AI/ML techniques from those used in image classifiers, but quickly demonstrated the breadth of applicability in our approach.

Our third effort (Section 2.3) was chosen to have greater relevance to ground vehicles. In this on-going effort, we are using commercial off-the-shelf (COTS) and open-source tools to read the data from the Powertrain Control Module (PCM) model on a commercial production vehicle. Our preliminary results show significant successes in replicating the spark and fuel injector settings based on engine operating parameters, thus providing a proof-of-concept for AI/ML Digital Twinning of line-replaceable units (LRUs) in ground vehicles.

The three subsystems (the rover image classifier, missile dynamics model and PCM) provide samples across a broad array of system types: 1. The Rover camera system is a logical, statistical system with an internal architecture that is similar to the black box approach. 2. Missile flight is predominantly a continuous, nonlinear physical dynamics problem. 3. The PCM is based on multi-layer logical switches and extensive look-up tables, and demands high-rate computation.

2.1. Rover Camera Image Classifier

The Torch Technologies Autonomy Testbed contains a digital model of a physical UGS rover with an onboard computer vision system performing object localization of known systems (e.g., the US Army DE-SHORAD vehicle) from a pylon camera feed. GazeboSim [6] and the digital Robot Operating System (ROS) [7] models of the rover were used to capture a representative training dataset of simulation inputs and outputs.

The AI/ML black box digital twin of the rover classifier was trained only on the camera feed inputs and the outputs of the YOLOv5 [5] classifier onboard. These outputs included {(x, y) center location, object width, object height, object class, and confidence score} as seen in Figure 1. There was no additional manipulation performed on the bounding boxes, class identification, or other data before training using YOLOv8. The dataset was split randomly using the 80/20 rule-of-thumb for training/validation datasets [8]. Training was viewed as successful when the training and validation loss (bounding box and classification) curves converged, mean Average Precision at 50% Intersection over Union (IoU) (mAP^{50}) was over 90%, and inference time met performance constraints (10ms). The rate of low confidence, erroneous detections was reduced by altering the confidence threshold for predictions based on the training Precision and Recall results.

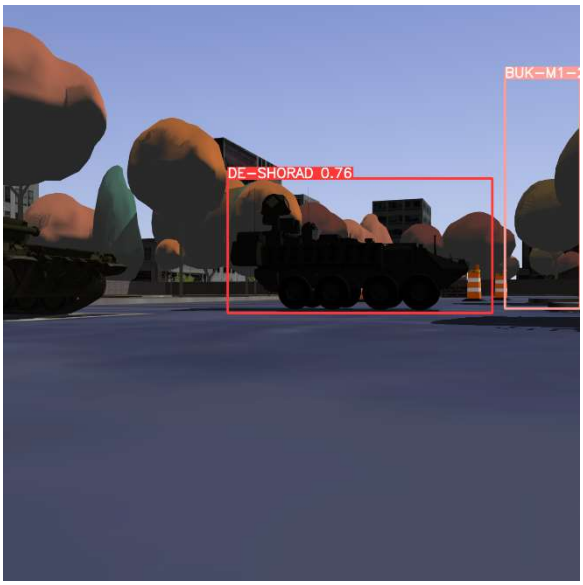

Figure 1: AI Black Box Rover Outputs

Verification of the AI Black Box classifier was not achieved in the same way as typical software modeling that assesses design requirements. Design requirements could not be assessed without a working knowledge of the neural network that defines the computer vision AI algorithms. The black box developer has little input as to the design of the network outside of selecting the loss functions, activations functions, learning rate, etc. The AI Black Box was verified during the YOLOv8 training by performing image augmentations (scale, flip, mosaic, shear, etc.) and assessing if performance consistently matches ground truth.

System level validation was performed by comparing ground truth object localization outputs of the existing system (UGS rover) to the AI predictions of the black box digital twin as shown in the confusion matrix in Figure 2. This evaluation included frame-by-frame assessment of True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). It was observed that for all 48 validation frames the performance was a 100% match for TP, FP, and TN predictions with only one spurious FP prediction when the black box AI picked up a far-field tank.

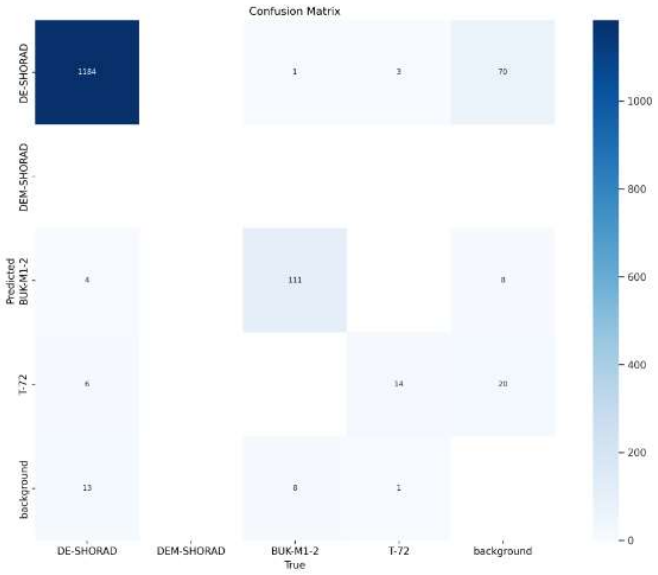

Figure 2: Confusion Matrix Comparing UGS On-board to AI Black Box Outputs

However, when simulation performance runs were completed the rate of False Positives for both the UGS Rover digital model and AI black box model increased. We suggest that the increase in False Positive rate is an artifact of the low-fidelity GazeboSim environment because it was only observed during virtual environment runs, but not during real-world camera feed inferences. However, we regard this result to warrant further investigation using higher fidelity simulation environments.

2.2. Missile Autopilot Step Response

Another use case for black box subsystem modeling is to capture the performance of an existing modeling and simulation digital model whose run-time performance is insufficient for a desired application. We chose the autopilot system from the high-fidelity NSim 6-degree-of-freedom (6DOF) digital missile system point for this proof-of-concept. NSim was developed by the US Army to provide independent assessment of tactical missiles and interceptors. We configured NSim to use its PID controller [9] in the autopilot to generate the pitch response to a given step-command in pitch guidance (e.g., 10° pitch to 15° pitch). We captured

the relevant environmental and vehicle conditions before the response as well as the resulting autopilot response which includes over/undershoot due to the dampening effects of the autopilot controller and the aerodynamic response of the missiles control surfaces. These parameters serve as the inputs to the black box.

We used MATLAB and Octave to characterize the short-term and long-term components of the response as a first order correction. First, a gridded search and amoeba downhill simplex optimizer were used in MATLAB to fit a "probabilist's" Hermite polynomial series [10] to the long-range behavior. Then, the long-range fit was subtracted from the response to isolate the transient oscillatory response, which we characterized by using a Fast Fourier Transform (FFT) to find the frequency of the highest mode of oscillation. Finally, we used a simple parabolic minimum finder to find the best-fit exponential envelope of the transient oscillation.

The outputs of the black-box model are the characterized response long-range Hermite polynomial fit constants (A, B, C, D, τ), short-range frequency and damping ratio (ω, β), and complex Fourier amplitude (Z_0, which subsumes real amplitude and complex phase). As such, we have transformed ~2000 points in a time-series into a 9-valued vector for output. The black-box model input vector includes dynamic pressure, drag coefficient, angle of attack, total angle of attack, altitude, current pitch, and commanded pitch. We used commercially available packages in SciKit Learn to clean and scale the input and output data before processing through a Multi-Layer Perceptron (MLP) Regression fit.

The mean absolute error (MAE), root mean squared error (RMSE), and coefficient of determination (R2) were used to verify the accuracy of outputs for all variables. Acceptable accuracy was defined as an average MAE and MBSE less than one standard deviation and a R^2 of 0.6 or better. The SciKit-learn MLPRegressor model exceeded these metrics with an average R^2 of 0.86 when predicting output values for the nine output features. A representative prediction as compared to the actual NSim response are shown in Figure 3.

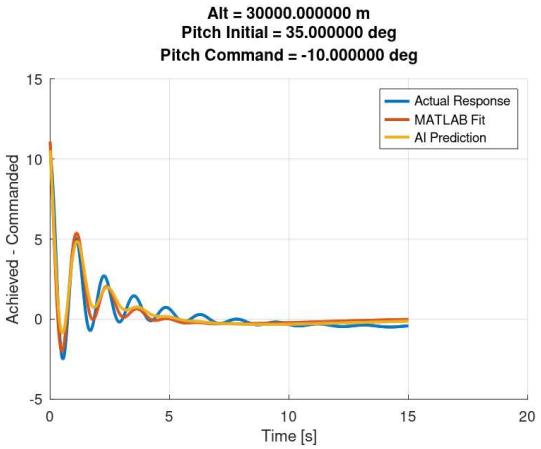

Figure 3. Example MLP Regression Fit

The computational performance impact was also considered as it is the determining factor for whether a black box AI can function in place of the modeled system. For our simulation runs we executed NSim to perform one step guidance command during steady-state flight and collected data for 15 simulation seconds. For our simple setup, NSim executed the 3-DOF runs of 2070 cases in 8.69s or 4.2ms per run. Likewise, NSim executed the 6-DOF runs of 2070 cases in 160.41s or 78.0ms per run. The results of each MLP Regression prediction took approximately 0.4ms per guidance command, which represents 9.5% of the total 3-DOF runtime (assuming the MLP model is resident in memory). Surprisingly, simpler methods such as multi-linear interpolation do not yield improved performance at run-time. This is likely due to the overhead to index the arrays and find local minima. The best performance comes from the PyTorch networks at 0.075ms per guidance command or 1.8% of 3-DOF runtime due to optimizations in inference. Based on this observation final modeling should be performed

outside of convenience packages such as SciKit Learn before deploying.

The result of this case study is that a missile fire control could execute around 15 times as many Digital Twin augmented 3-DOF executions as it could native 6-DOF runs, while still maintaining most of the performance effects associated with a full 6-DOF model. Such a gain in execution speed radically improves the ability of a fire control to model future trajectories of inbound missile threats.

This missile 6DOF response model may appear to present only a very specific application within the missile modeling domain, but controlled damped oscillator type systems are common across a wide variety of large engineered systems. As an example, Allen [11] used a very similar form of synthetic waveforms to model shock response in ground vehicle testing. His Figure 4 is closely analogous to our Figure 3 in showing how his mathematical model decomposes the response. The ability of machine learning techniques to produce the coefficients of such decompositions, as we have demonstrated herein, thus presents much promise for modeling a wide variety of actively control dynamic hardware systems.

2.3. Vehicle PCM Modeling

The third system we discuss is a the Powertrain Control Module (PCM) of a commercial vehicle. This system controls fuel injectors and spark timing in the vehicle's engine to provide improved performance and efficiency. It was chosen due to ease of access to the hardware, excellent open-source data logging options, and ability to rapidly perform HWIL testing of models.

The PCM data that we used for black box modeling was from a 1995 station wagon running a stock On-Board Diagnostic (OBD)-1 PCM with Motorola 68HC11 processors. The PCM is calibrated and optimized for the current stock engine. The EEHack open-source software package was used to interface with the PCM since EEHack is purpose-built for the particular hardware and software in this vehicle [12]. We used a laptop connected via an RS-232 to Universal Serial Bus (USB) interface cable to log data as well as read, edit, and write calibrations to the PCM.

The extensive engine tuning tables inside the PCM allow for independent adjustment of input variables such as ambient temperature, coolant or engine temperature, barometric pressure, engine Rotations Per Minute (RPM), throttle position, mass airflow, intake Manifold Air Pressure (MAP), and Air/Fuel Ratio (AFR) targets. The outputs are the resulting fuel injector pulse-width and ignition time (advance) value where a positive value means ahead of the piston's reaching top dead center (TDC) and a negative after TDC. The PCM has routines for Cranking, Transient Fueling, Deceleration Fuel Cutoff, Catalytic Converter Over-temperature Protection, and Performance Enrichment (PE). After successful modeling of the current PCM outputs there are other PCM outputs to the engine that will be replicated including idle air control motor position, Exhaust Gas recirculation (EGR) duty cycle, EVAP system purge valve control, and cooling fan control as a function of engine coolant temp.

We have developed a PyTorch Deep Feed-forward Neural Network (FNN) that was successfully trained on the input as seen in the learning curves shown in Figure 4 and models the two output variables with a 0.96 R^2 score. Initial attempts did not produce good predictions, but progress was made by increasing the number and size of hidden layers in the neural network and changing how we normalized the input data. Since it is a regression problem, we use Rectified Linear Unit (ReLU) activation function, and the model behavior appears indifferent to whether L1 or L2 loss functions are used.

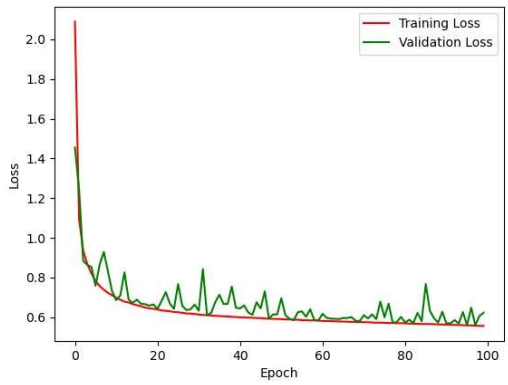

Figure 4. PCM Black Box Model Learning Curves

The generalized models for fuel injector and spark advance were tested against a dataset gathered during typical driving conditions (Cranking, PE, Deceleration, Idling, etc.). The results of the test were collated in Figure 5 which lists the mean absolute error (|prediction – actual|) for each engine state as well as the sample count per engine state; note that about 95% of our data come from the normal operating state. We find that the mean absolute error for spark advance is less than about 2˚ for all engine modes, while the fuel injector pulse width error means are less than about 0.5 ms for all modes. Subject matter experts (SMEs) regard that such errors are generally negligible in terms of actual operation of the vehicle, and as such, the model would likely be sufficient to operate as a black box power controller at this point.

We explore our errors somewhat further in Figure 6 which shows box-and-whisker plots of the errors for our AI/ML model vs. the actual PCM (we have excluded outliers beyond the whiskers for sake of readability, but this whiskers essentially represent a two-sided 95% confidence interval). For spark advance, we show the error in angular degrees; for the fuel injector pulse width we show the error as $100 \cdot \ln \frac{\text{model}}{\text{actual}}$ so that for small errors, we have simply the percent error, but for larger errors we maintain symmetry in the ratio. The dashed red lines show a nominal 2˚ / 2% error budget. The box

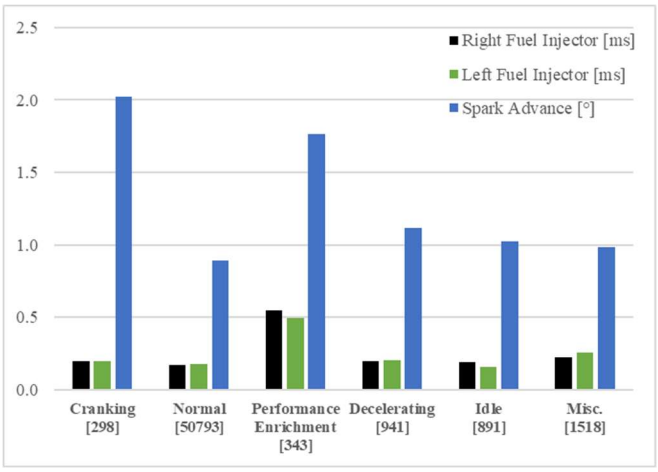

Figure 5. PCM Black Box Generalized Model Mean Errors and Sample Counts

plots for spark advance (in blue) show that the inner quartiles are within the 2˚ limits in all but one case. However, the pulse widths show significant excursions beyond our 2% criterion, particularly for the Cranking mode. Also, the Performance Enrichment mode shows a systematic negative offset. While the results in Figure 6 are encouraging, room for improvement remains.

Our SME made a simple recommendation which was to segment the data based on engine mode, and to conduct separate AI/ML training for each. The results are shown in Figure 7. The spark advance has tightened down significantly; in all cases, even the (~95%) whiskers are inside the 2˚ threshold. The Cranking pulse width errors have settled down dramatically, with a reduction in the interquartile ranges by a factor of greater than 5, and of the 4 quartiles shown by those two boxes, only the 25th percentile of the left fuel injector slips just outside the 2% threshold. Furthermore, the Performance Enrichment systematic offset has essentially disappeared, and in both cases, the entire interquartile range is within our 2% limit. For the Decelerating, Idle and Miscellaneous modes, all box plots have improved significantly, also. The Normal mode case has improved only slightly, but this is not surprising, given that 95% of

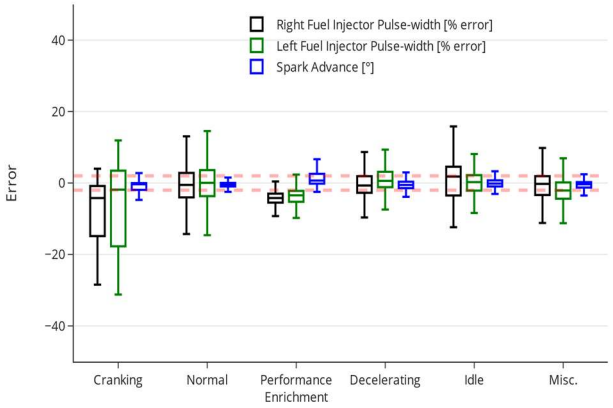

Figure 6. PCM Black Box Generalized Model Performance

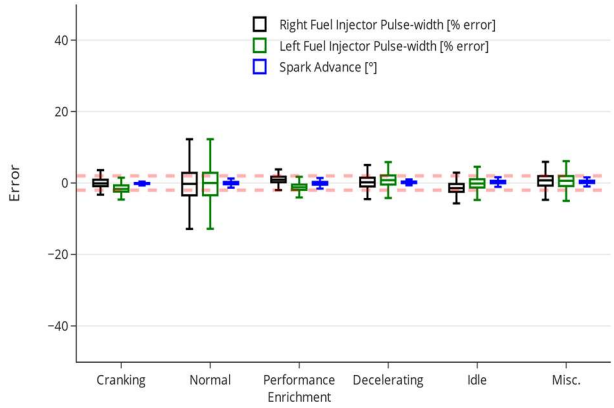

Figure 7. PCM Black Box State-Based Model Performance

our data were in the Normal mode case for our original training. In Normal mode, we see the interquartile ranges nudging just beyond our 2% criterion.

The significant improvements visible in Figure 7 illustrate the point that even when using AI/ML, subject matter expertise can provide significant, measurable improvement on a model.

A final comment on the PCM data is that a large fraction of outliers may arise from a mismatch in the actual data rate in the PCM (100Hz) and sample rate (10Hz) for the reader system. The sampling rate discrepancy leads to discontinuities during transients in the system, for which model interpolations may be poor. The extent of this effect remains under investigation. While the PCM interface system's sample rate was low, the generalized PyTorch model was verified to complete inference faster than the actual vehicle data rate with the python implementation performing at a maximum of 400Hz. This means that the AI/ML model, with no further performance enhancements, could be used to operate the vehicle.

This case study shows that a hardware subsystem can be rapidly modeled as a Digital Twin with acceptable accuracy without subject-matter expertise or extensive knowledge of the algorithms, but the incorporation of SME guidance yields substantial improvements to the initial design.

3. Discussion: Lifecycle Applicability of AI/ML Digital Twins

Digital twins present systems engineers and stakeholders the opportunity to support lifecycle decision gates with greater statistical rigor and reduced cost and schedule, but only if the digital twins involved provide output responses to input stimuli that are very highly faithful to the real-world system. Creating digital models with this level of fidelity can often be so costly and time-consuming that the value proposition vs. real-world/live testing the digital twin is diminished. Our AI/ML approach to digital twinning is designed to deflate the time and budget of model creation, thereby realizing the power of digital twins in the lifecycle.

3.1. Concept Assessment

Assessment of new concepts in the design of integrated systems, systems-of-systems and system architectures during design processes often requires the exploration of large swaths of parameter space, both to characterize expected performance over the relevant operational domain and to identify potential failure modes before costly investments

are made in these candidate architectures. Trustworthy and computationally performant AI/ML models provide proof-of-principal digital twins for the conceptual subsystems and components to support the model execution rates needed for this use case.

3.2. Integrated System Performance

Large integrated systems (i.e., systems-of-systems) commonly exhibit complex behavior not easily predicted based on the behaviors of the subsystems. In fact, Weisel [13] rigorously showed that the composition of multiple valid models results in an integrated model that may not be presumed valid. As such, the only truly sound way to understand integrated system performance is to compose the entire system and execute it under relevant conditions. But it is generally far too time-consuming and expensive to do this for real-world engineered systems. As we have discussed, digital twins offer much hope in this regard, but only if creating the digital twins is much, much more efficient than a pain-staking software modeling process. Use of AI/ML digital twins closes this gap.

With validated digital twins, a large number of configurations for the integrated system can be created and tested in a multitude of scenarios. In this way, rigorous understanding of otherwise unexpected complex behaviors and be identified and quantified to support trades in risk vs. further development cost.

3.3. Subsystem Performance

If we turn now to the single subsystem for which we have created a digital twin, the benefits of our AI/ML approach are straight-forward. The digital twin development cost and schedule are reduced, while the computational performance of the digital twin is (generally) outstanding. This means that one can rapidly spin up a digital twin and exercise it in a very large number of simulation executions to understand far more fully the range of performance one can expect under different conditions. Importantly, failure modes and other unusual behaviors identified in the digital domain can then be used to guide real-world testing to get much greater value out of each real-world test run. As these edge cases are tested in the real-world, they can feed back into the training of the AI/ML in a validation cycle to maximize confidence in the performance of the subsystem across its use domain.

3.4. Test and Evaluation

Conducting full-scale real-world HWIL test and evaluation is the ideal way to vet a newly developed (or upgraded) system. However, such full-scale real-world testing is extremely expensive and labor intensive. Using Digital Twins to represent other subsystems within the broader system to provide realistic stimuli to and interactions with the system under test provides enticing opportunities for savings in cost and schedule. This would allow more rapid deployment, and more extensive and thorough testing throughout the Development Test phase of the life cycle. Our AI/ML Digital Twin concept directly supports this use case.

3.5. Obsolescence Management

The intense pace of development in microelectronics has provided for explosive growth in computational capability over the last several decades. An unfortunate side effect of that development pace is that electronic components from just a few years ago may no longer be supported by component producers. As such, a significant fraction of LRUs in large engineered systems, such as M1A2 Abrams tanks and AH-64 Apache helicopters may face recurring obsolescence problems, despite comparatively recent block upgrades. Industry has developed a large suite of obsolescence management strategies, such as supplier management and persistent monitoring of component markets. However, reverse engineering and construction of a new LRU is sometimes the only available path. AI/ML approaches offer an intriguing option in this regard. If the AI/ML digital twin truly mimics the original LRU at high fidelity, the LRU could be replaced with that model running on commonplace, general-

purpose hardware far, far less susceptible to future obsolescence problems.

4. Conclusions

Digital twins present analysts, stakeholders, systems engineers, and acquisition professionals with a novel capability to explore large swaths of configuration space at a tailorable level of fidelity. Creating digital twins usually involves an extensive modeling process involving subject matter experts and seasoned developers. We have presented case studies demonstrating that AI/ML approaches can dramatically shrink the timeline and budget for creation of digital twins—in each of the 3 cases studied so far, we have created usable digital twins in a matter of weeks without extensive subject matter expertise. Because AI/ML models tend to execute very rapidly, these models also provide sufficient computational performance to execute in the loop with hardware or live systems. Our case study on the commercial vehicle power control module (PCM) demonstrates applicability to the Army Ground Vehicle domain, thereby showing that AI/ML digital twins can accelerate testing, analysis, CONOPS development and lifecycle management at the US Army GVSC.

5. Acknowledgements

The authors would like to thank Matt Blair, Jamie Burns, Chuck Couteau, Tim Palmer, Ben Phillips, Mac Roberts and Caleb Simmons for providing data, guidance and thoughtful conversations during the execution of this work and during preparation of this manuscript.

6. REFERENCES

[1] "The Boeing 777 And How Computer Aided Design Changed The Face Of Air Travel," *www.nesfircroft.com*, Jul. 14, 2020. https://www.nesfircroft.com/resources/blog/the-boeing-777-and-how-computer-aided-design-changed-the-face-of-air-travel [Accessed Feb. 21, 2024].

[2] M. W. Grieves, "Virtually Intelligent Product Systems: Digital and Physical Twins," *Complex Systems Engineering: Theory and Practice*, pp. 175–200, Jan. 2019, doi: https://doi.org/10.2514/5.9781624105654.0175.0200.

[3] "What Is a Digital Twin?," *www.mathworks.com*. https://www.mathworks.com/discovery/digital-twin.html

[4] K. Griffin *et al.*, "GROUND VEHICLE SYSTEM REFERENCE ARCHITECTURE MODEL (DIGITAL THREAD)." Accessed: Feb. 26, 2024. [Online]. Available: https://www.ndia-mich.org/images/events/gvsets/2022/presentations/dese/254%20Griffin.pdf

[5] Jocher, G., Chaurasia, A., & Qiu, J., "Ultralytics YOLO (Version 8.0.0)," [Computer Software]. https://github.com/ultralytics/ultralytics [Accessed Mar. 04 2024].

[6] "Gazebo," *gazebosim.org*. https://gazebosim.org/home [Accessed Mar. 04 2024].

[7] ROS, "ROS.org | Powering the world's robots," *Ros.org*, 2020. https://www.ros.org/ [Accessed Mar. 04 2024].

[8] V. R. Joseph, "Optimal ratio for data splitting," *Statistical Analysis and Data Mining: The ASA Data Science Journal, vol. 15, no. 4, pp. 531–538*, Apr. 2022, doi: https://doi.org/10.1002/sam.11583.

[9] M. Araki, "PID Control," *CONTROL SYSTEMS, ROBOTICS AND*

AUTOMATION - Volume II: System Analysis and Control: Classical Approaches-II. N.p.: EOLSS Publications, 2009.

[10] "Hermite Polynomials and Applications," *WORLD SCIENTIFIC eBooks*, pp. 101–134, Oct. 2019, doi: https://doi.org/10.1142/9789811201585_0003.

[11] S. Allen, "Waveform Synthesis for Shock Response Spectrum Replication, Applied To Ground Vehicle Component Testing", *In Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)*, NDIA, Novi, MI, Aug. 13-15, 2019.

[12] "$EEHack – ecmhack." https://ecmhack.com/eehack-2/ [Accessed Feb. 21, 2024].

[13] Weisel, E., 2004, "Models, Composability, and Validity", Ph.D. Thesis, Old Dominion University, https://digitalcommons.odu.edu/cgi/viewcontent.cgi?article=1021&context=msve_etds