SYM-AM-22-074

Excerpt from the Proceedings of the Nineteenth Annual Acquisition Research Symposium

Acquisition Research: Creating Synergy for Informed Change

May 11–12, 2022

Published: May 2, 2022

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

Acquisition Research Program
Department of Defense Management
Naval Postgraduate School

The research presented in this report was supported by the Acquisition Research Program at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

Acquisition Research Program
Department of Defense Management
Naval Postgraduate School

Tips for CDRLs/Requirements when Acquiring/Developing AI-Enabled Systems

Bruce Nagy—is an applied Research Engineer at the Naval Air Warfare Center, Weapons Division (NAWCWD) at China Lake California. His focus is on integrating game theory and machine learning to create recommendation engines that handle complex, highly dimensional battle management scenarios. He also uses causal learning techniques to provide explainable, statistically based solutions to wargaming stratagem. When he served in the Navy, Nagy qualified as an Engineering Duty Officer. In that capacity, Nagy developed statistically-based algorithms to enhance satellite communications. He was also a technical troubleshooter for NRO and received honors for his efforts in recovering a failing national defense program that jeopardized U.S. global security. In addition to his wargaming and machine learning research, he is driving a joint effort to create standards that establish an appropriate level of rigor for the acquisition and/or development of AI-enabled systems for deployment. He has received four degrees in math, science and engineering, and has done postgraduate work analyzing brainstem to muscle group neurology. [bruce.m.nagy.civ@us.navy.mil]

Abstract

The Department of Defense (DoD) is challenging Acquisition professionals to manage the development of systems incorporating AI functions either as upgrades or new programs of record. But, AI functions present unique challenges associated with requirements and subsequently when creating a suitable Contract Data Requirements List (CDRL). The problem stems from the ability to ensure the quality and quantity of training data sets which can limit the reliability of AI performance. Currently, there is limited guidance regarding topics for discussion during an AI requirements review or as to what AI related information should be required in CDRLs. However, a recent investigation into the lack of AI development guidelines prompted a NOSSA-funded project. Using an AI "sandbox" approach, a DoD representative program, involving AI/ML algorithms supporting a mission planner with autonomous vehicle selection and navigation, was used to determine realistic requirements specific to systems incorporating one or more AI functions. As a result of their analysis, this paper presents contents for an AI Development Plan (AIDP) to be part of a CDRL. Within the AIDP, measurements and new evaluation methods are also offered, as well as questions and considerations to support quality AI development.

Common AI Acquisition and Development Issues

Issues facing the acquisition professional managing a program implementing AI are: (1) heavy use of AI jargon where folks among the data science community can't seem to agree on common definitions, (2) a lack of understanding of the core workings of various AI algorithms and thereby turning a function into a seemingly magical black box. However, once the vocabulary is understood, I believe the acquisition professional is much further ahead. The core workings of an algorithm require only an understanding of the mechanics without needing to go into the detailed math, i.e., system engineering constructs in terms of DoD Architecture Framework (DoDAF), Unified Modeling Language (UML), etc.—those are things that can be left to AI software engineers. And finally (3) having confidence that the training data accurately represents what the AI algorithm will experience during "stressful" deployment periods.

The first problem, less technical and more cooperative, can be solved by the program sponsor and the prime contractor agreeing upon a common AI dictionary of terms. The dictionary can be created by either the DoD, commercial sector or a self-created source. The second problem, educating program participants in the basics of AI, is also a potentially solvable issue through training. The third problem is the Achilles heel for anyone acquiring an AI-based system. There is currently no "official" guidance on how someone involved in AI development should view training data in terms of quality and quantity. Does it need to be created from "live" operational environments or will M&S synthetic data be sufficient? Current work in this area

demonstrates that AI training data issues cause a significant amount of redo work, cost overruns, and schedule delays. M&S synthetic data has both pros and cons associated with its use as training data. For the acquisition professional, knowing how to address these issues is absolutely necessary. Because of the deficiency gap in AI policies, guidelines, and tools, acquisition professionals, from PM, SE, and T&E, as well as system safety practitioners need education/training regarding how to assess AI development to ensure confidence in the behavior of current WD weapons programs implementing AI upgrades. Because of these deficiencies and by direction of DoD, many working groups are attempting to understand AI's unique software requirements, architecture, design, code, and test without any reference as to what works and what does not, instead are using ad-hoc approaches. This is dangerous!

Background to Determining Documentation Content

Naval Ordnance Safety and Security Activity (NOSSA) funded this research over a two year period to investigate unique AI/ML policies, guidelines, tools and techniques to assess safety in identified critical functions. The project, as shown in Figure 1, involved two autonomous robots delivering packages, using an intelligent route planning system that considered the degree of difficulty with the routes, including crime, weather conditions, and human factors. The sandbox approach was used to mock-up an AI development process for the purpose of creating AI system of systems analysis to examine process and requirements. From this work, documentation structures became evident.

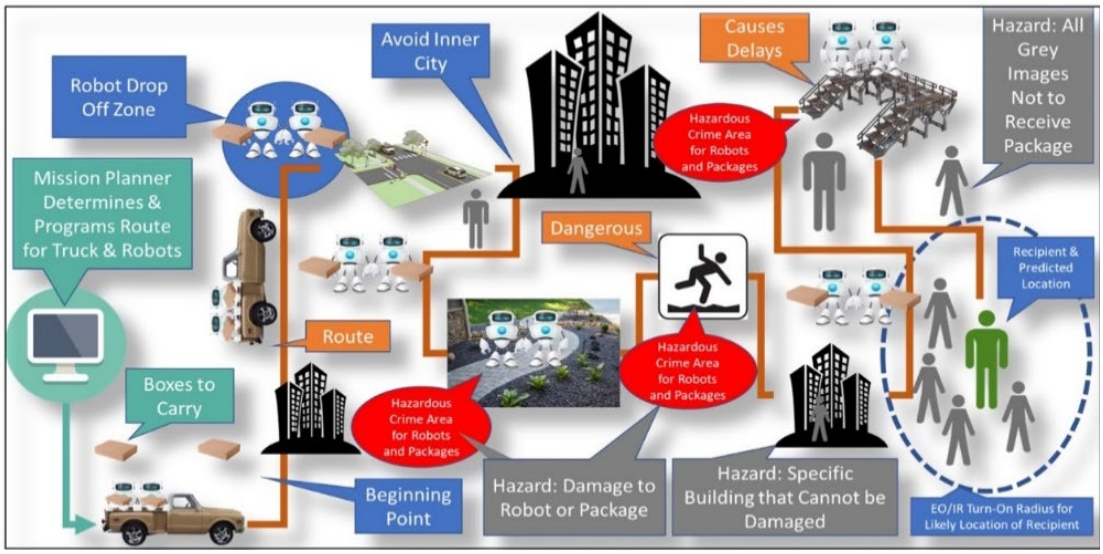

Figure 1. Operational Use Case

From this sandbox approach: (1) DoDAF and UML diagrams were created that identified MSG, API and SQL protocol requirements, (2) detailed architecture and designs were reviewed, and (3) software code was written, including a simulation program that modeled the use case described above—this allowed for training data to be acquired consisting of five classes and 17 attributes. Using the sandbox development approach allowed for the creation of level of rigor (LOR) tasks appropriate for each of the five stages. Through this analysis and development process, LOR was created in the form of practical questions and considerations to rigorously ensure AI/ML systems are built to have confidence in their functional behavior associated with the five stages: (1) Requirements, (2) Architecture, (3) Algorithm Design, (4) Algorithm Coding, and (5) Test and Evaluation. This paper focused on the requirements portion of this research as described in various documentation formats.

Contract Data Requirements List (CDRL)

A common CDRL approach is to list a Software Development Plan (SDP). However, SDP's have a wide range of content formats without a common structure. The DoD attempted to create a standard but found it difficult. The closest standard as guidelines can be found in DOD-STD-7935A, *Military Standard: DoD Automated Information Systems (AIS) Documentation Standards* (1988). "These standards provide guidelines for the development and revision of the documentation for an automated information system (AIS) or applications software, and specify the content of each of the 11 types of documents that may be produced during the life cycle of an AIS" (DOD-STD-7935A, 1988).

Many standards have come and gone, e.g., MIL-STD-498, DOD-STD-2167A, and DOD-STD-1703, due to lack of flexibility. Issues like a standard focus on waterfall project management styles, versus Agile type development, or not using modern software development tools, such as Computer-Aided Software Engineering (CASE) tools.

The SDP goal is to determine the scope of a software development effort, with guidelines in place to ensure quality development. However, based on history with regard to software, it's been difficult to create a one size fits all solution. A solution to this approach is in using MIL-STD-31000 *Technical Data Packages* (TDPs), which allows for customized content development. Another approach is in using DI-MISC-80508B, Data Item Description: Technical Report-Study/Services 2006), which also allows for customization.

Due to this challenge, this paper recommends a separate document, an AI Development Plan (AIDP), learning from common practices involved with customized SDPs, TDPs, and Technical Report Studies. The AIDP focused on AI process and guidelines that follow a level of rigor approach derived from the previously mentioned sandbox research. Therefore, the suggestions are in the form of questions and considerations listed specifically to the topic associated with various sections. The questions and considerations support an increased confidence in the behavior of an AI system when deployed.

Tips for CDRL Documentation

Five types of AI focused documents will be discussed with an emphasis on content: (1) AI Justification Report—3 Sections, (2) Best Practices Report—2 Sections, (3) Measurement Report—3 Sections, (4) Test and Evaluation Readiness Report, and (5) Missing and Sparse Class Tables—4 Sections. Each document type has a specific purpose as will be discussed. These five types of documents comprise the AIDP.

AI Justification Report

Section 1—AI identification: Conduct AI Type Function analysis of the proposed functions to determine if it includes an AI algorithm. If it is identified as an AI Type Function, then the document types and related sections contained in this report are suggested.

The hypothesis is that a function is an AI Type Function candidate if one or both criteria are met:

- (Criteria 1—Data Approximations) The function requires the use of *data approximations* to build/train its algorithm. Approximations can sometimes lead to inaccuracies. For example, it might use a single value, like average speed. An algorithm might use average speed to determine a time instead of the actual speeds encountered during deployment. This could lead to a decision error. Another example of data approximation which could also lead to decision errors is the use of text to represent values. For instance, representing many altitudes by the word "high." In this case, the data approximation "high" represents an infinite number of altitude values above a certain threshold. The function may be designed to make a decision when the data input states

"high," potentially causing a decision error. Simulations use data approximation to model dynamics. AI algorithms might use synthetic training data sets, again potentially causing algorithm performance errors. A common approximation data inaccuracy concern is when training sets are synthetically generated. The concern is whether the data generated is replicating "realistic" background noise. This type of data approximation inaccuracy found in synthetic data has been noted as a main cause of an AI algorithm's poor performance when deployed. In game theory, the payoff tables are normally based on mathematical approximations called expected utility functions (EUF). If an algorithm uses a payoff table to make a decision, the EUF approximation might cause decision errors.

- (Criteria 2—Data Samples) The function requires the use of ***data samples*** to build/design its algorithm. For training, data normally consists of a representative subset of all the data contained in a larger population. Sample representations of the larger population can sometimes lead to inaccuracies. Subsets can be created from "live" or synthetically generated (simulation) sources. If the function requires the use of synthetic data that created the samples, the concern would be how much of the model approximated "reality" and how much of the total population was synthetically covered within the created subset? An example of using samples from "live data sources" would be snapshots of images describing facial expressions. The concern is whether the collected facial expressions within the training set represent all images that a person might express over a period of time when deployed in a variety of situations. If the subset of images collected of facial expressions do not adequately represent the deployed experience, then the training of the algorithm is limited. Will this limitation caused by the subset cause the algorithm to have performance issues? Another challenge associated with creating adequate data sample representation is when collecting information from experts or other authorities to create a decision tree. Decision trees requires input and output rules. Normally only a subset of all possible input and output combinations are used. Again, because only a subset is used, how well this subset represents deployed input and output challenges will determine the algorithm's performance.

Section 2—AI Scope: Discuss and document a justification for the proposed use of the AI algorithm vs. using a more traditional software, firmware or hardware technique. The goal is to explain why an AI algorithm is a "better" fit to the functional requirement.

Verify that an AI/ML function is needed by asking the following questions:

- For Criteria 1, data approximations: Can the algorithm be traditionally built using data approximations? Why or why not? A question to consider is, "Could another developer create a different set of statistics under the same conditions?" If no, then maybe this algorithm is not an AI Type Function. If yes, then there are data approximations and determining how one approximation is better than another needs to be considered. As an example, if a statistical model of the function was developed, how accurate were the approximations used in creating the function. In other words, how close do these approximations fit the physics of the real-world regarding operational deployment? How accurate is the distribution? Another consideration is investigating if the data approximation can be decomposed into greater detail, thereby reducing the size of the approximation. The goal is to have accurate data approximations that will result in quality training sets.
- For Criteria 2, data samples: Can algorithm be broken into subpopulations to allow development of traditional code? Why or why not? Another question to consider is, "What is the actual population size of the training set?" If the training set is equal to the

actual population size, then the function does not need an AI approach and can be handled traditionally. Consider the most basic ML algorithm, a regression line. If all the points that will ever occur for this function are used on the scatter plot-to approximate the curve, why use a regression line? If all the ML algorithm inputs and outputs are known, why use ML and not traditional code, i.e., if this, then that? Again, if traditional code can address the needs of the function, then that should be the approach used. If the function is based on simulation results creating data samples, then the concern is the "garbage in, garbage out" issue—poor real world representative synthetic data will result in an inferior model. The goal is to have comprehensive data samples that will result in robust training sets.

Section 3—AI Autonomy: Discuss and document a justification for the AI algorithm's level of autonomy, i.e., lack of supervision. The goal is to explain why an AI/ML algorithm requires the selected level of autonomy based on functional performance requirements and not a lower level.

a) Document how the design can or cannot include human in the loop oversight or traditional hardware/software technology acting as a guiderail/guard to provide checks and balances.

b) If checks and balances are limited, provide documentation as to operational limitations by:

1. Describe weaknesses of each AI/ML technique, e.g., expected success rate of the function. For example, if AI/ML is built on data approximations (using AI Type definition), how much bias will the data approximations add to the functional outcome? Or, if AI/ML is built on data samples, how representative are the samples to the population?
2. Determine how the training data is being generated, e.g., truth, synthetic, combination. Are these sources valid? Why?
3. Where is the training data coming from? Is it enough? (Remember the more sophisticated the AI/ML software, the more likely that it needs larger amounts of training data)
4. Will an outside independent source review the training, validation and test data created? Why or why not?
5. Will an outside independent source validate the success rate of the AI/ML function as compared to other AI/ML functions used in industry? Why or why not?

Best Practices Report

Section 1: Modality Type

What Machine Learning Training Data modality type are you representing in your deployed system and your data generation process? When creating training data, it is important to understand the operational environment being represented in order to ensure adequate development of the machine learning (ML) algorithms. The training data is either found from live events or synthetically created to match the operational scenario that will be provided as input to the ML algorithm. Therefore, the ML algorithm must learn how to perform under these conditions. Three types of modality represent various operational environments that can be encountered during deployment, where the type of modality defines how the ML algorithm needs to be trained. Understanding ML training data modality is fundamental to developing a reliable AI system (Nagy, 2021).

ML Training Data Modality 1: This modality, shown if Figure 2, supports training data sets that are based on an operational environment from multiple data sources, where each source contains one or more attributes. The various sources of separate data attributes are either found from live events or synthetic simulations created to match the deployed operational scenario. Therefore, the input for the ML algorithm for training needs to replicate the input that will be received during deployment.

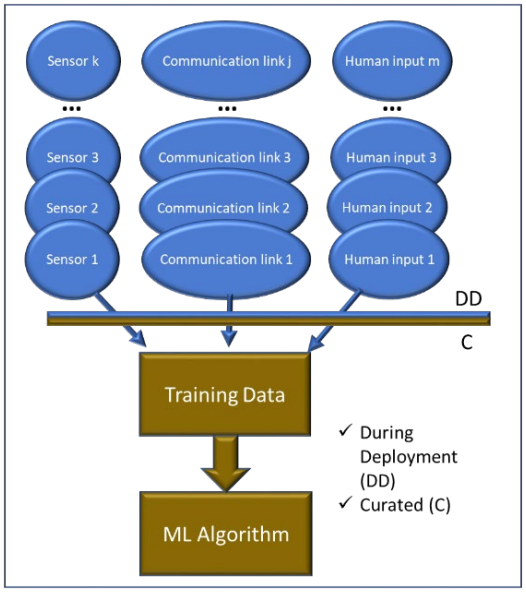

Figure 2. Example of Modality 1 Receiving Attribute Values from Various Sources

ML Training Data Modality 2: Training data sets that are based on an operational environment from a single data source, where the single data source contains multiple data attributes, as shown in Figure 3. The one stream set of aggregated attributes is either found from live events or synthetic simulations created to match the deployed operational scenario. Therefore, the input for the ML algorithm for training needs to replicate the input during deployment.

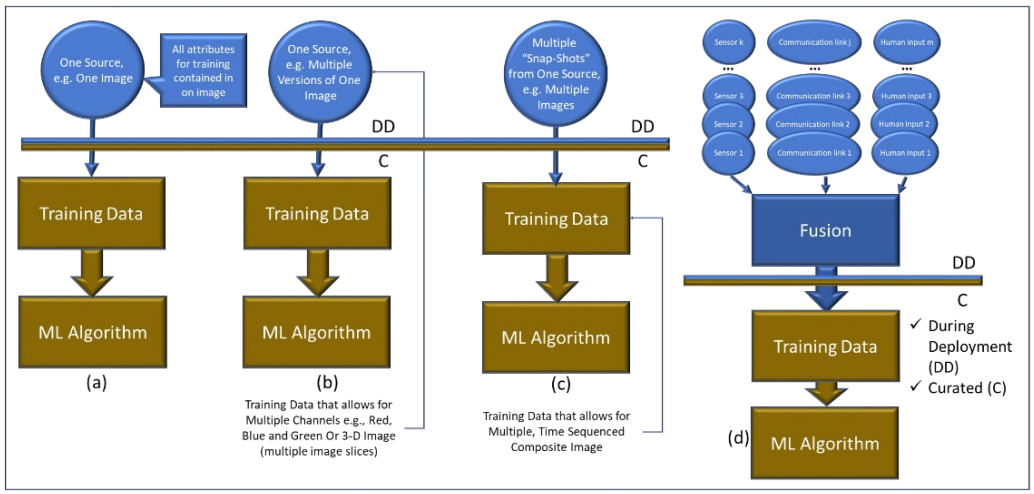

Figure 3. Examples of Modality 2 Training Requiring Images (a, b and c) or Fused Attribute Data (d)

ML Training Data Modality 3: Training data sets that are based on an operational environment from a combination of multiple data sources, shown in Figure 4; each source

contains one or more attributes from various sources and from a single source containing multiple aggregated data attributes. It is a combination of Modality 1 and 2 that the algorithm uses for categorization or regression.

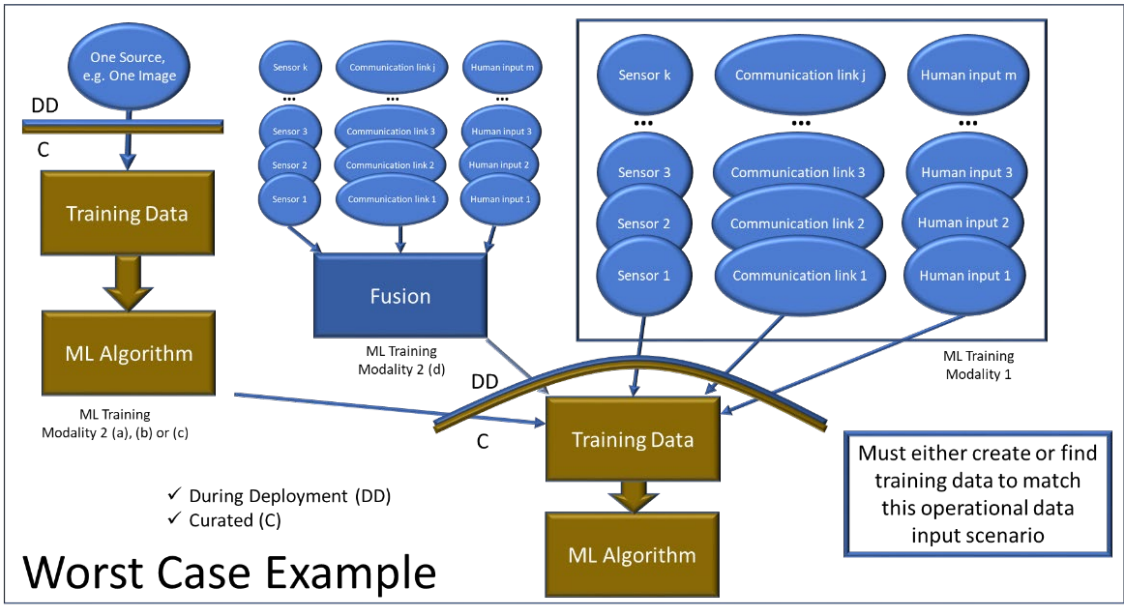

Figure 4. Example of Modality 3, a Combination of Modality 1 and 2

Instances/samples comprising training sets are composed of a combination of attributes, sometimes called features. When a feature is identified within an image, it is described as a piece of information contained in the content of the image. In this case, the feature describes a specific region of the image, which has certain properties, as opposed to another popular definition of a feature, a single pixel in an image.

The aggregation of attributes can be contained in one source, e.g., a camera taking a facial picture, or from many sources, e.g., various sensor inputs, such as radar and communication links. In this paper, we will distinguish whether attributes are generated from one or multiple sources based on their modality.

Adversarial ML is not considered a system safety issue, but does affect AI model confidence. It is important to know that it introduces challenges in the behavior of an AI model. Adversarial ML is modality dependent. Adversarial ML is most-times designed to cause an error in the output of the AI model being targeted.

Based on modality design, how can the deployed AI model be exploited by an adversary? Is this a consideration in Operational and Systems Requirements Discussions/Reviews in terms of the behavioral confidence of the algorithm or training set adequacy?

If Adversarial Networks/attacks become a consideration for any of the AI Functions under review, then an analysis by the developer to support the concern should be provided that includes describing how the balance between quality and vulnerability of the training set is being achieved with some form of objective, measured precision.

Section 2: Dataset Structure

Part 1—Representation: Does the synthetic or live data represent all the training data needed to train the algorithm to identify each label/class within the needed success rate?

Examples of classes are various types of targets, described by a label, that are determined from the output of the algorithm. Using data sets to train the algorithm to identify an object is a typical ML process.

If not, how are classes being represented; are values being determined by using ML for regression? This question relates to how classification or regression is accomplished. For classification, algorithms can have two to many classes. For regression, then some form of analysis is used to determine a number or range of numbers. It's important to understand how the algorithm needs to perform and what type of data is being used to train an algorithm to perform adequately.

Note: A class, also referred to as target, label, or category, is what a categorization algorithm labels an input. For instance, if only an image of a cat or dog is used as input/training data for an ML algorithm, then the algorithm only has two classes either a cat or dog. To make a determination, each class would normally have a threshold value that would have to be met. If that threshold value is not achieved, by either class, then the AI model would fail to determine the input. For example, if the input was a coffee cup instead of a cat or dog, the threshold value would not be sufficient for either class and the input would not be determined.

Part 2—Fitness of the Data: Does each class have an appropriate number of attributes, or values that can be learned by the algorithm for the class/number being determined? In other words, has overfitting and underfitting been considered for each class/number with regard to the quantity of attributes/values simulated/collected and does that quantity reflect real world operations?

- Overfit. Indicates that there is an issue with the quality of the quantity of training data used. Overfit occurs when the algorithm's success is limited to a small amount of input variations when compared to the original training data. Limited input variations mean that as long as the input instance/sample closely matches an instance/sample of the training data, then it will accurately categorize/approximate, otherwise the algorithm will likely generate an error. Overfit states that it has a very low tolerance for input data that is not close enough to the original training data input.
- Underfit. Indicates that there is a quantity issue with the training data used. Underfit occurs when there is an insufficient amount of training data which causes missed categorizing of the correct class/approximation.

Both indicate a poor level of performance. Therefore, how will it be determined that the training data, attributes, or values within the instances used for categorization/regression, be sufficient to maximize success for data inputs not in the original training set? Note: This discussion must evolve around how the algorithm will be operationally deployed.

Part 3—Mission Alignment: How do we know that the synthetic or live data creating the training data is aligned with the mission parameters?

- Was a traceability study performed to support adequate coverage?
- Have statistics been shown on the number of configurations available and the number trained using data sources?
- How are we avoiding overfitting and underfitting based on training mixes and sets?
- Is the training data organized in terms of attributes to be able to represent missing and sparse data occurrences from related sources?

These questions are a follow-on to the question in Section 3. Not only is the correct proportion of training data needed, but the proportion must be in alignment with the reality of the

system being deployed. In other words, what the algorithm will experience if involved in a mission.

- How many operational use cases were created and then translated into training data requirements?
- Were the data sources feeding the input to the AI adequately assessed and how does anyone know?
- What intelligence sources were used and how reliable were those sources?

Creating a training mix means that the developer is assuming that the algorithm will need to perform in an imperfect world, and some of the primary sources for the algorithm may not exist. Were secondary sources considered or even tertiary sources? Primary, secondary, and tertiary sources are considered mixes within the training set and can address the missing or sparse data reality during deployment.

- Missing Data: This refers to the data input to the AI model. For our purposes, missing data occurs when the model is expecting certain features but does not receive them because of an issue with the data collection mechanism feeding the model. For example, a sensor states a ship is moving at 1000 knots and therefore has been considered erroneous data. In this case, velocity is considered missing, reported as an empty field in the input stream. The missing data feature comes from some form of data collection failure and can be represented in a field as a blank field, i.e., no data shown. This causes a need for secondary or tertiary attributes mixes.
- Sparse Data: This refers to the data input to the AI model. For our purposes, sparse data occurs when the model is expecting certain features but doesn't receive them, but not because of any issue with the data collection mechanism feeding the model. In other words, sparse data occurs when the system is working but no data is available to fill a field. An example of sparse data might be a fully functional radar system not receiving any blips because there's no target to reflect back. Most times sparse data will be represented by a zero where as a blank field represents missing data. This causes a need for secondary or tertiary attributes mixes.

Part 4—Mission Alignment: How are we ensuring that the algorithm being deployed, after using training data, provides the correct answer when data input issues occur? This question relates to understanding how the training data will be created/curated with regard to potential deployment errors represented by the training data.

- Is analysis of the algorithm's success rate a function of attribute availability within its anticipated operational environment?"
- Will the training set represent the reality of data input issues during deployment? If so, then how will the success rate be affected, i.e., will the success rate be assessed; before errors, without operational issues, after errors are injected, or with operational issues?

Part 5—Oversight: Can other control entities (such as a human operator) be inserted into the loop to reduce the autonomy? One way to answer this question is through Interdependency Analysis (IA). IA allows an objective review to determine which function is best performed by automation or a human operator to create an optimal relationship. The approach helps to optimize performance and understand how best to reduce autonomy with human oversight or guardrail/gate control of critical functions. Requirement content needed for an IA analysis should include:

- Identification of AI enabled functions at the subsystem composition detail.

- Identification of performers, both machine and human, involved with that function.
- Identification of the method(s) used to ensure the interaction between the human and the machine in terms of observability, predictability, and directability.
- Description of the multiple paths through the key function where applicable.
- Description of how necessary metrics can be obtained to objectively support any subjective determinations to reduce autonomy discovered through the IA process.
- IA failure walk through, including any failure modes associated with AI/ML enabled functions.

Part 6—Sparse and Missing Data**:** There are three subparts to sparse and mission data content questions and considerations.

Subpart A: What are the ratio requirements of *sparse* and *missing data* occurrences to normal operations when creating training data from synthetic or live data?

Assuming that sparse and missing data are part of the training data, this question focuses on an expected ratio of occurrence in an operationally deployed environment. If an *Instance* consists of attributes that the AI algorithm is learning to analyze; and *Sparse* and *Missing data* indicate noise in the attributes, making it more difficult for the algorithm to perform, then what ratio of noisy instances make up the training data? This should be a ratio that can be measured for validation and defined in a requirements document. It should not be left to the developer or to chance. Once a ratio is determined, the developer should have confidence, whether it be synthetic data or live data, that it will perform as defined.

Subpart B: Will there be secondary or tertiary attributes supporting the mission or sparse data issues? In other words, if primary attributes are not available for algorithm analysis, will less important attributes be available, e.g., background environment or habit factors? When primary attributes are unavailable due to potential real-world issues, secondary or tertiary sources can increase the success rate of an algorithm's analysis. Therefore, they should be considered when defining a training data ratio.

Something else to consider when determining secondary or tertiary attributes and related types of ratios for:

- Modality 1: How are higher priority attributes that experience sensor malfunction, message corruption and human input errors being mitigated by forcing mixes of lower-level attribute training data to ensure the algorithm deals with "real" operational issues?
- For Modality 2: If there is corruption in parts of an instance, e.g., a blurred image, especially containing higher priority attributes, are secondary and tertiary attribute mixes of training data ensuring the algorithm can deal with "real" operational issues?
- For Modality 3: Are combinations of modalities 1 and 2, regarding training of the algorithm, able to deal with "real" operational issues?

Subpart C: Will the architecture, design and code support sparse and missing data management, or more specifically, will it filter or use a selection of less significant attributes to do the calculations? Note: This question provides discussion regarding the mix of data and how the architecture, design and code will support this mix.

- How will the effects of *missing* and *sparse data* be minimized within the architecture, design and code, from a requirements point of view?

- Will secondary and tertiary attributes be included in the training data, and if so, will secondary and/or tertiary attributes be used as a way to deal with missing and sparse data? If this isn't considered and potentially included as a requirement, it may cause poor success rate performance during deployment of the algorithm.

If this consideration becomes a requirement, implementation of an approach to deal with this issue should be traced throughout the process of development.

- For Modality 1: Will sensor, communication link or human input content elements take priority over the others to improve the success rate when training a ML algorithm under normal to stressed operational conditions?
- For Modality 2: Which attributes, within the single data source, take priority for improving the success rate when training the ML algorithm under normal to stressed operational conditions?
- For Modality 3: What data source content is more significant, with regards to normal to stressed operational conditions? When dealing with separate streams, which of the following: sensor, communication link or human input content elements takes precedent, for improving the success rate when training a ML algorithm under normal to stressed operational conditions? When dealing with combined streams, which attributes within the single data source are identified as primary, secondary and tertiary, regarding importance for ML algorithm to improve success rate, under normal to stressed operational conditions?

Part 7—Data Curation: What processes are being defined, to support data management curation, to ensure that the ML algorithm provides accurate data input?

Data Curation. Is the organization and integration of data collected from various sources. Data curation involves annotation, publication and presentation of the data such that the value of the data is maintained over time, and the data remains available for reuse and preservation. Data curation normally supports a targeted machine learning goal, where the organization is based on the classification or regression needs of the algorithm.

- How does your data curation approach avoid "garbage in, garbage out"?
- What definitions will be used to constitute "garbage in, garbage out"?

These questions ensure that the data curation process for handling data and the creation of training data is understood at the requirements phase. The emphasis will be on ensuring that the data curation process can determine, with some level of measurable certainty, whether accurate data input is being achieved.

For all three modalities: What is the priority list (from highest to lowest) of attributes being used for training? How much more emphasis is placed on the quantity of training data variations with a higher priority than lower?

Part 8—Battle Complexity: How well does the particular ML algorithm support increased battle complexity and how does that affect sparse and missing data issues?

Battle Complexity: A situation described by a series of events, caused by actions between opposing participants, where the outcomes can be significantly affected by factors categorized as: (1) "known-knowns" (facts); (2) "known-unknowns" (assumptions); (3) "unknown-knowns" (absent data); and (4) "unknown-unknowns" (surprises).

- "Known-knowns" (facts) are factors that participants depend on as "fact" to win the engagement; these can include own participant's technical capabilities, geo-spatial, temporal situational awareness, interoperability, tactical actions and strategy pros/cons.
- "Known-unknowns" (assumptions) are factors that each participant needs to "assume" about variations (of the facts) regarding battle conditions, these can include the third-party involvement, weather forecast, kinetic and non-kinetic effectiveness, opponent's attack surfaces and related vulnerabilities, heroism and initiative on all sides, opponent's priorities, and difficulty in overcoming manmade and natural obstructions.
- "Unknown-knowns" (absent-data) are factors that cause a participant to be "absent of data," sometimes decision critical info; these factors can include human mistakes, sensor failures and communication issues.
- "Unknown-unknowns" (surprises) are factors that will "surprise" participants during the engagement; these include unforeseen technology and anything not anticipated in the previous three categories.

Trust is gained through the LOR descriptions. For example, understanding the modality of the training data (as facts); or as will be described in follow-on LOR descriptions, conducting TSAT and StAR-n analysis to support variations to the input (as assumptions) and providing missing and sparse data class tables (as absent-data). The challenge involves the inability to prepare the AI model to handle unbound data issues (as surprises). Unbound data by its inherent definition means that confidence in the performance/behavior of the AI model cannot be predicted and therefore cannot be trusted.

Given the above definition, consider the following when discussing the topic of trust:

- Can we trust that the training data *factually* represents the real world when deployed, e.g., use of correct attributes/features, noise/background, etc.?
- Can we trust that the *assumptions* regarding input variations from the training data are within expected scope as not to cause an error in the output, e.g., miscategorization?
- Can we trust that the *absence of data* when needed to the AI model has been adequately anticipated and compensated to maintain success rate?
- Even if the previous three answers are all "yes," the AI model, by definition, is not trained to handle *surprise* inputs, i.e., unanticipated, unbounded data! Historically, we can always expect *surprises* in warfare because intel will never be 100% accurate, i.e., expect the unexpected. Unanticipated/unbounded inputs are known to cause the AI model to have radical, undesirable failures. It is a noted limitation in deep network algorithms, i.e., neural networks with many layers.

Autonomy and AI systems are designed to handle some level of "known-unknowns," based on the "assumptions" about the variation in the training data input, and are challenged with "unknown-knowns," relating to missing and sparse, "absent data" issues; but it's the "unknown-unknowns" that create the more significant concerns regarding "surprises" of unwanted behaviors. In order to represent a realistic operational set of training data, complexity of the deployed environment needs to be considered.

- How will this complexity be considered when synthetically creating or finding data to use for training?
- How will the requirements be defined with regard to complexity?
- Will guard rails/gates be used?

Measurement Report

Section 1: Dataset Quality

For each ML class, define requirements that rank the importance of attributes, i.e., creating a priority list, within each instance that the AI algorithm will be trained to recognize. This ranking represents a baseline to determine if a quality training set is being used. As an aid to determining requirements that rank the importance of attributes, a process might be to create operational scenarios looking at nominal and extreme cases. Ranking must be done by class, so the scenarios must be class based with a focus on attribute input to the algorithm.

As an example, a Training Set Alignment Test (TSAT) supports the requirements group in ranking all attributes that will be used by each class. The approach allows the project to assess the training data to determine if the initial ranking is statistically similar to the statistically determined ranking of the training set. Statistical ranking determination is based on a number of occurrences of each attribute within the entire training set. The result of comparing the initial, required ranking to the statistically based ranking is calculated as a single numeric value. The single numeric value represents how well the requirements group's ranking matches the training set compositions. For example, a number of 5.0 out of 10 indicates that the training set only matches 50% of what was required in terms of attribute priority/importance. Having a 100% match is unrealistic, but something above 50% or even 75%, a 7.5 score out of 10, might be a reasonable expectation. The key is ensuring that the attributes within the instances of the training set represent priorities for the algorithm to learn. Priority learning for an algorithm is viewed as how often the attributes and their varying representations repeat. If training on a facial recognition program has only a small percentage of instances that contain nose variations, then the algorithm will not be sufficiently trained to handle and/or properly categorize variations in noses.

- Are attributes for the algorithm ranked in order of priority?
- How does that compare with the actual training data?

These are important questions that need to be addressed and adding these requirements becomes vital to the understanding of whether or not the algorithm is using a quality training set?

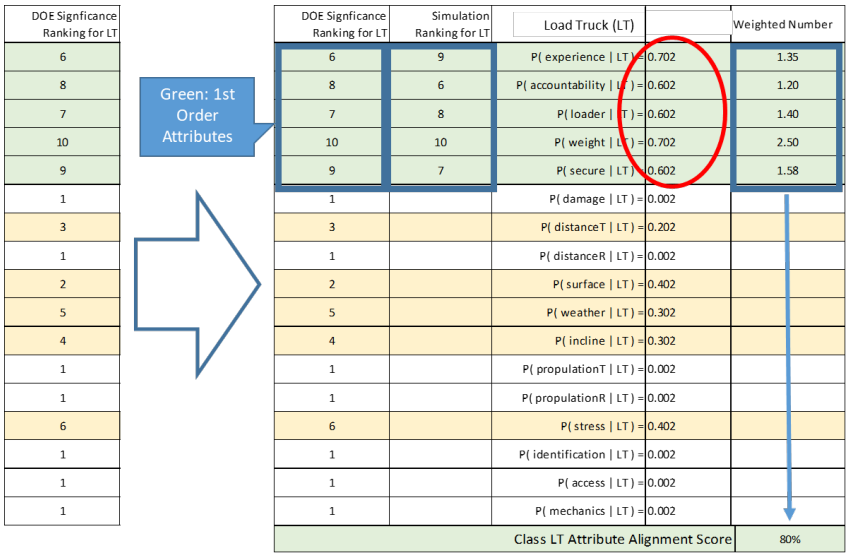

Figure 5. Training Set Alignment Test (TSAT)

Figure 5 is an example of a TSAT where a Design of Experiments (DOE) ranked a series of 17 attributes supporting 5 classes, LT being a class in the TSAT example, as compared to a series of Monte Carlo simulations that determined the ranking based on the percent/frequency of simulations that used those attributes. The above score for the example is 8.3 out of 10, and would indicate attribute occurrence within the data set are aligned with expected deployed priorities.

Procedure for calculation:

1. Determine a scale for grading from 1 to "m," where "m" means greatest attribute priority/significance based on operational deployed needs.
2. Identify attributes a_1 to a_n to grade, such that "n" is the number of attributes being graded out of r total attributes available. Therefore n ≤ r and n ≤ m, where grading a_i with grade "m" indicates a_i (m) is the most important attribute based on operational needs. Additionally, attribute grading range is (m-n+1) to m, consecutively, where lowest grade indicates least operationally important (possibly DOE analysis and/or SME determination).
3. Identify the n attributes that occur the most times in the training data. Using the same scale "m," grade attributes b_1 to b_n based which attribute occurred the most often within the training set (this can be a statistical number, e.g., 70% of the time b_i attribute was used in simulations or 70% of the samples/instances were collected, e.g., images, that contained attribute b_i). Again, grade "m" indicates b_i occurred the most and (m-n+1) indicates b_j occurred the least within the training set.
4. Perform $k = \sum_{i=m-n+1}^{m}(\mathrm{i})$ and $\beta = \sum_{i=1}^{n}\left(\frac{a_i(grade)}{k}\right) * bi(grade) \leq m$
5. Perform $\left(\frac{\beta}{m}\right) * 100 = \alpha\% \geq 50\%$ as a constraint

Section 2: Dataset Quantity

Once attributes are ranked in terms of priority, the next question should be, "Does the ranking indicate a grouping of attributes based on the importance and availability of data during a mission?" In other words, can ranking from *(1 to m)* represent primary attributes or more specifically, are the key attributes that the algorithm depends on used? If so, then attributes ranked *(m + 1) to n* represent secondary attributes. When some of the primary attributes are missing, secondary attributes may be used as input for the algorithm to produce a more successful categorization rate. This also means that a mix of primary and secondary attributes are needed as part of the training data. It should be noted that primary attributes should occur more often than secondary in the training data, based on what is most important for the algorithm to learn.

Primary, secondary, tertiary, and etc., will be based on how often a grouping of attributes are expected as input to the algorithm during deployment. If they were all considered primary, what happens when there is missing and sparse data issues during deployment? Missing and sparse data, by definition, means primary attributes were not available. Therefore, to support realism, should secondary and tertiary attributes be considered? If considered, what should be the ratio of primary to secondary and tertiary attributes? Can this be a requirement?

- Specifically, how will the priority and ratio of a grouping of attributes be determined and how will it be used for testing?

As an aid to determining priority and ration of a grouping of attributes, a process might be to create operational scenarios looking at nominal and extreme cases. Warning, this is only an optional starting point. The focus is on impact to the input attributes to the algorithm, i.e., mission or sparse data events. Manually developing even dozens of scenarios would not be enough. Each scenario manually developed would likely needs fifty or more variations in attribute values for algorithm training. For creating training data, total of all scenarios developed should consist of ten to many hundred thousand variations, balanced by class, as will be discussed. These variations need to be either collected, instances itemized attribute by attribute or instances synthetically created with the goal of creating the desired ratios. For synthetic generation, a Generative Adversarial Network (GAN) inspired approach provides a randomness associated with the creation of large sets of data covering a range of potential issues that the algorithm might encounter during deployment.

The reason why randomness is important is because of the inability to predict future deployed engagement events. Randomness of attribute values within scenarios ensures greater readiness to handle unknown future events. It is because tens of thousands of variations increase the likelihood of the algorithm being trained to handle unanticipated deployed situations. These attribute combinations associated with classes need to be assessed based on their ranking of importance determined earlier.

As an example, the Source to Attribute Ratios for 1, 2, 3 (nth) (StAR-n) Order Matrix approach can support the development of requirements based on attributes (primary, secondary, or tertiary groupings, e.g., a third order matrix) representing highest significance (priority/rating defined in TSAT). Higher priority grouping, e.g., primary, should occur in greater numbers of instances within the training set, by class, than a lower significance grouping of attributes, e.g., secondary. The comparison of numbers can be analyzed as ratios.

Why should developers verify that primary instances have greater numbers than secondary, and so on?

- With live data collection, it might be difficult to find all the training data that includes the appropriate noisy environments that might cause missing and sparse data issues; and
- With synthetic data creation, simulation may be too ideal, not representing the appropriate noisy environments. (Remember that there's most likely an infinite number of possibilities in terms of training data variations and simulation time to meet schedule demands might get strained.) What should be the priority when considering or designing your synthetically created training data?

The StAR-n Order Matrix focuses on the quantity of how often attributes occur, by their grouping, within the training set. StAR-3 looks at the ratios of primary, secondary and tertiary attributes, as they are defined through requirements.

As stated earlier, training data is key to the AI/ML algorithms development and therefore, the question becomes, "How much of the training data consists of primary vs. secondary vs. tertiary attributes that are dependent on data sources that will be available in the field?" Again, the issue becomes dealing with missing and sparse data during deployed operations.

StAR-n provides confidence that there's an adequate quantity of training data, whether generated from live events or synthetically created to train the algorithm. StAR-n represents these ratios in the form of a matrix, consisting of three colors, to relate the amount of justification needed to support the training data quantity required. This is similar to a risk matrix coloring scheme. Once the matrix is defined, the actual training data ratios of primary, secondary, tertiary, etc. can be placed within the matrix to determine rigor documentation needed.

The color zones are:

- Zone Green: Evidence of data by showing appropriate n-th order groups of training sets, collected from "live" data or generated by the simulations, including success rates as well as the TSAT results.
- Zone Yellow: Zone Green evidence plus justification of why the n-th group priority can still handle the unexpected and provide acceptable success rates.
- Zone Red: Zone Green and Yellow evidence as to how the algorithm is going to be supervised or monitored when operationally unexpected events occur.

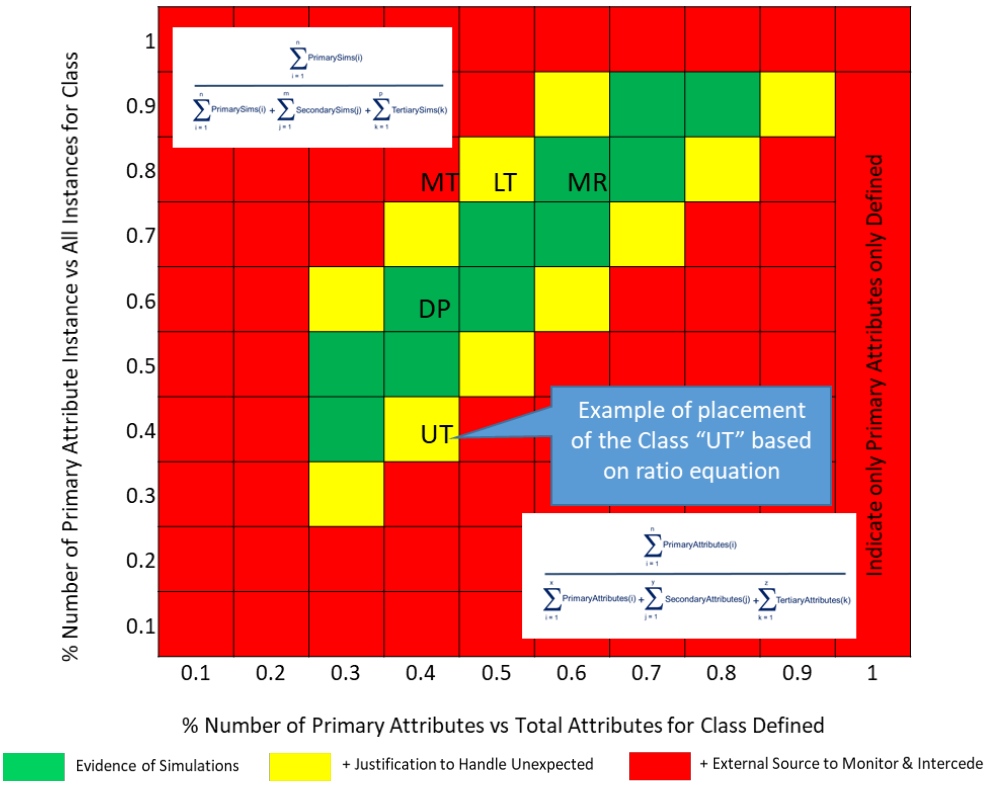

Figure 6. Source to Attribute Ratios for 1, 2, 3, . . . nth (StAR-n) Order Matrix

Above is an example of a StAR-n Order Matrix focused on Primary Attribute ratios. Matrices can be created for primary, secondary, and tertiary attributes, not just primary. Again, zone placement is based on operational needs. Zones can also be changed in terms of how much justification is needed or added ("+"). When measuring actual ratios, placement of the actual ratios would determine what cell the class will be placed, therefore what justification, green, yellow, or red, is needed to support that ratio.

The steps discussed below for taking a StAR-n measurement are groups by requirements/architecture and algorithm code stages:

During the requirements and checked during architecture review:

- First Step: Create a ten by ten matrix, labeling each axis from zero to 1.

- Second Step: Label the horizontal axis “% Number of Primary Attributes vs. Total Attributes for Class” and the vertical axis “% Number of Primary Attribute Instances vs. All Instances for Class.”
- Third Step: Determine a three-color zone scheme (as in the example), where green indicates that the ratio fell within acceptable limits, yellow indicates ratio is boarder line acceptable, and red color zone indicated ration is outside expected limits. The color of the zone should indicate how well training data reflects operational environment. Based on color zone, determine evidence justification. Examples (used for guidance only) are described below:
    - Zone Green: Evidence of data by showing appropriate n-th order groups of training sets collected or generated by the simulations, including success rates as well as the TSAT results.
    - Zone Yellow: Zone Green evidence plus justification on why n-th group precedence can still handle the unexpected and provide acceptable success rates.
    - Zone Red: Zone Green and Yellow evidence as to how this algorithm is going to be supervised or monitored when operationally unexpected events occur.

During Algorithm code review when the training set is produced:

- Fourth Step: Calculate the σ and δ (as in the example) ratios. Each ratio should be less than 1. The example below is for primary attributes, but can be done for any n-th order attributes:
    - σ (by Class) = (Number of Primary Attributes / Number of All Attributes) ≤ 1.
    - δ (by Class) = (Number of all Primary Instances / Number of All Instances) ≤ 1
- Fifth Step: Plot (x, y) using (σ, δ) pair of numbers and assess where the pair fall within the color zones to determine support action. An example is provided in the example.
    - Zone Green: Evidence of data by showing appropriate n-th order groups of training sets collected or generated by the simulations, including success rates as well as the TSAT results.
    - Zone Yellow: Zone Green evidence plus justification on why n-th group precedence can still handle the unexpected and provide acceptable success rates.
    - Zone Red: Zone Green and Yellow evidence as to how this algorithm is going to be supervised or monitored when operationally unexpected events occur.

Section 3: Dataset Measurement Review

How do you know if the quality and quantity of Training Data is sufficient? Quality refers to the correct number of attributes (including primary, secondary, etc. mixes) that are representative of the deployed operational environment, including noise factors. Quantity refers to the amount of data/instances used for training, with consideration to mix ratios, underfitting, overfitting and majority/minority classes.

- How do you assess the operational limits described by the training data? (Consider the “You don’t know what you don’t know” issue.)
- Did the training set include enough noise/clutter for each class (in this case, less significant attributes determined by SMEs for a particular meta-model class) to ensure

that the function works properly when deployed? Are there sparse data and/or mission data issues? How is the bias of the training set and variance of the test results determined?

For simulation generation of the training data:

- How would you ensure synthetic or live data configurations work, i.e., is the training data covering the real-world experiences? (Optimizing bias [how well it fits the training set] and variance [how well it predicts using the test set], including considerations of overfitting/under-fitting).
- What quality of synthetic or live training data, i.e., attribute composition on each instance, and how many of these various compositions are really enough to train an algorithm?

Test and Evaluation Readiness Report

Has a process been identified to ensure that randomly selected T&E data is available for testing from the curated training data before any developer uses it? If not, why not?

Will model versioning control be used to track model drift or data drift? Will the versioning control support troubleshooting of any AI model issues that might occur later? Data versioning supports the ability to version multiple sets of data against many different compiled algorithms and then rollback/forward to different training data sets depending on need?

- **Model drift.** Is a form of model decay caused by not keeping the model current with significant attribute changes in the training data, e.g., boys face evolves to a man's face but never updated in the training data.
- **Data drift.** Is undocumented changes to data structures, semantics, and infrastructure, e.g., undocumented modification to the API causing the model to view that part of the input as missing or sparse data.

The model versioning control process should include positive control over who, what, where, and when transactions occur involving the creation of the training data composition.

As an example, the need to use positive control over a training set would be when a T&E set of training data from a k-fold cross validation approach is identified. If live data is limited in terms of quantity available, it is recommended that T&E training data needs take priority and that possibly all live data be set aside just for T&E testing. In either case, there must be a separate amount of training data, randomly selected from a pool of training data that is untouched/unviewed by the developer and specifically focused on supporting T&E.

Since the training data drives the composition of the algorithm during training, it is important that the creation of the data, part of which will become the T&E test set, has strong oversight, in addition to versioning. An approach to provide this oversight, especially when the data is coming from many diverse sources, with multiple touch points, is a technology called blockchain.

Blockchain is a type of distributed e-ledger (similar to what an accountant would use to keep track of financial transactions). It is designed to be a form of tamper-resistant, decentralized documentation that provides proof of transaction involving physical or intellectual assets, in this case T&E training data. It ensures confidence that only people allowed to access the data, from its origin to a T&E facility (separating this test set from the development test set), have access to the data.

By using a blockchain approach, policy enforcement can be ensured and that the rules for accessing the data are followed. A blockchain architecture documents the who, what, where,

and when user access (transactions) involving the creation of the data set composition, data attribute transfer to location for T&E random selection, ownership of the T&E test set and integrity of the data.

Missing and Sparse Class Analysis

Will a Missing and Sparse Data (MSD) Class Table, consisting of four sections, be used? The MSD Class Requirements Table provides requirements/guidance for developers to deal with missing and sparse data issues. As part of the requirements, within the table, you can indicate a plus or minus percentage for meeting the numbers listed. As an example, you can have four sections focused on various aspects of missing and sparse data.

If this table is created, an equivalent MSD Class Actuals Table must also be created to be filled in during the development process and then compared to ensure listed requirements are being met.

Using the sandbox use case involving robots delivering package, the tables below provide a five class, 17 attribute example of a package delivery system involving trucks loaded with robots (LT), driving to a drop off location (MT), unloading the robots (UT), having the robots move to the desired location (MT), and deliver packages to the recipients (DP).

Section 1: Class Representations in Dataset

Create a table or list by class for the expected training data quantities/numbers based on ML Training Data. The headings need to describe the "Number of Max Data Sources Allowed for a Decision," "Number of Primary Attributes Based on Data Source Availability," "Number of Secondary Attributes Based on Data Source Availability," "Number of Tertiary Attributes Based on Data Source Availability," and so on. The goal is to have an understanding of data source availability during deployment and the number of attribute inputs (from those data sources) that will feed the algorithm/model.

Table 1. Training Data Attributes Table

Class	Number of Max Data Source for a Decision	Primary Attributes	Secondary Attributes	Tertiary Attributes	+/- % Compared to Actuals
Load Truck (LT)	h1 + h2 + h3	h1	h2	h3	x%
Move Truck (MT)	i1 + i2 + i3	i1	i2	i3	x%
Unload Truck (UT)	j1 + j2 + j3	j1	j2	j3	x%
Move Robot (MR)	k1 + k2 + k3	k1	k2	k3	x%
Deliver Package (DP)	l1 + l2 + l3	l1	l2	l3	x%

Table 1 is an example listing variables h, I, j, k and l that would be converted to numbers supporting requirements for each class. The x% represents the acceptable variance allowed when compared to actual results.

Section 2: Class Ratios of Classes

Create a table or list that describes, within the training set, an expected percentage of how often primary attributes occur in an instance/sample compared to the total number of instances being used for training. Also create percentages for instances of secondary attribute occurrences to the total number of instances, as well as tertiary attributes, etc. These percentages should be defined for each class.

Table 2. Attribute Instances by Significant Grouping Instances Table

Attribute Instances to Total Instances	Classa	Classb	Classc	…	Classn	+/- % Compared to Actuals
Primary to Total	a1%	b1%	c1%		n1%	y%
Secondary to Total	a2%	b2%	c2%		n2%	y%
Tertiary to Total	a3%	b3%	c3%		n3%	y%

Table 2 is an example listing variables a, b and c that would be converted to numbers supporting requirements for each class per priority grouping. The y% represents the variance allowed as acceptable when compared to actual results.

Section 3: Success Rates

Create a table or list that describes the expected success rate when combining attributes from various priority groups of the algorithm (e.g., as a percentage). They can then be measured using the T&E set created from the k-fold cross validation approach described in LOR 8. This description should list the required test results by primary, secondary, and tertiary priority groupings and when mixing groups, e.g., primary only success rate, primary with secondary success rate (with primary as the majority of attributes in the instance), primary with tertiary success rate, secondary with primary (with secondary as the majority attributes in the instance), and so on.

Table 3. Attribute Instances by Significant Grouping Table

Test Results by Class	Primary data source/attributes	Secondary data source/attributes	Tertiary data source/attributes	+/- % Compared to Actuals
Primary data source/attributes	Actuals: 1st Order Only testing success rate (emphasis 1st Order)	Actuals: 1st and 2nd Order testing success rate (emphasis 1st Order)	Actuals: 1st and 3rd Order testing success rate (emphasis 1st Order)	z%
Secondary data source/attributes	Actuals: 1st and 2nd Order testing success rate (emphasis 2nd Order)	Actuals: 2nd Order Only testing success rate (emphasis 2nd Order)	Actuals: 2nd and 3rd Order Only testing success rate (emphasis 2nd Order)	z%
Tertiary data source/attributes	Actuals: 1st and 3rd Order testing success rate (emphasis 3rd Order)	Actuals: 2nd and 3rd Order Only testing success rate (emphasis 3rd Order)	Actuals: 3rd Order Only testing success rate (emphasis 3rd Order)	z%

Table 3 is an example listing variables a, b, and c that would be converted to numbers supporting requirements for each class per priority grouping. The y% represents the variance allowed as acceptable when compared to actual results.

Section 4: Class Balance

Create a table or list that provides an expected majority or minority class analysis of how balanced (equal quantities) the classes are with each other. This is done to avoid data bias

- Data Bias. Occurs when the training data does not equally represent all of the environment where deployed but focuses on a subset. A form of data bias is imbalanced classes. Imbalanced classes means that one class has more training samples/instances and is significantly larger than the others. The class with the larger number of instances is called the majority class and the smaller number of instances is the minority class.

The table or list needs to describe the expected average number of instances, within the training set, for each class. The list should be divided based on the priority grouping of attributes.

As an example of reviewing combinations:

- 1st order only
- 1st and 2nd order (emphasis/more of 1st order)
- 1st and 3rd order (emphasis 1st order)
- 1st and 2nd order (emphasis 2nd order)
- 2nd order only (emphasis 2nd order)
- 2nd and 3rd order only (emphasis 2nd order)

Therefore, focus is on determining what class is a majority or minority class. In most cases, 1st order only, 1st and 2nd order (emphasis 1st rder), may be the only consideration when analyzing each class.

Table 4. Majority and Minority Class Analysis Table

Possible Combinations of Attributes based on Operational Needs	Expected Instances for Training*					Balanced
	Classa	Classb	Classc	...	Classn	
1st Order Only	a1	a2	a3		an	?
1st and 2nd Order (emphasis 1st Order)	a1	a2	a3		an	?
1st and 3rd Order (emphasis 1st Order)	a1	a2	a3		an	?
1st and 2nd Order (emphasis 2nd Order)	a1	a2	a3		an	?
2nd Order Only (emphasis 2nd Order)	a1	a2	a3		an	?
2nd and 3rd Order Only (emphasis 2nd Order)	a1	a2	a3		an	?
1st and 3rd Order (emphasis 3rd Order)	a1	a2	a3		an	?
2nd and 3rd Order Only (emphasis 3rd Order)	a1	a2	a3		an	?
3rd Order Only (emphasis 3rd Order)	a1	a2	a3		an	?
Total Instances by Class:	Σ	Σ	Σ		Σ	Analysis: Balanced or Unbalanced
If Not Balanced, Majority or Minority Class (Based on Availability of Training Data)	?	?	?		?	

Table 4 is an example listing variables “a1” to “a*n*” to ensure balanced classes, meaning there are no larger instances, i.e., there are no more minority classes. The desired result would be that the number of instances is basically the same for all classes.

Consideration: A key goal of the last four sections is to ensure that the developer demonstrates a detailed understanding of potential deployment issues that could affect the AI algorithm. This understanding is measured by the composition quality of the training data reflecting operational “realism.” When composition quality accurately reflects the deployed operational environment, it results in an improved performance of the AI model under realistic conditions.

The challenge becomes an adversarial network tradeoff. For example, an image recognition system for a smart phone is trained on key facial features. If the owner is wearing a headband, the smart phone may be stumped until the AI is trained to recognize the owner wearing the headband. However, the phones initial inability to recognize unexpected/surprise variations in facial features, e.g., wearing a headband, ensured others were denied unwanted access to phone.

In the four sections previously described, groupings of secondary and tertiary attributes show that the AI model is being trained to handle deployment variations associated with missing and sparse data. These deployment variations are equivalent to training the smart phone to recognize the user when wearing a headband. The concern is whether these types of approaches to increase the quality of data, i.e., using training data to support unexpected/surprise variations in deployment conditions, are also making the AI model more susceptible to adversarial network attacks.

Is the developer considering the balance between versatility, handling variations, and security? If so, there should be formal analysis associated with identifying this balance. The analysis should describe how versatility is generating greater security issues.

If the developer creates the training set as described in each of the four sections, what are the balance considerations between versatility and security? Are they being considered? Balance is obviously an important analysis and emphasizing either "too much or too little" can possibly lead to an issue in the confidence of the behavior of the AI model or the security of the system. This discussion of balance and any related analysis applies to all LOR focused on ensuring quality training data composition

Summary

The development of advanced artificial intelligence (AI)/machine learning (ML) systems for deployment by/throughout the Department of Defense (DoD) is a reality. AI/ML integration into DoD is in the form of upgrades to existing programs or new program acquisitions. How do we know these AI/ML-enabled systems will perform as intended? This paper presented an approach in the form of an AIDP.

Subject "Project Overmatch," Memo October 1, 2020 from the CNO begins, "The Navy's ability to establish and sustain sea control in the future is at risk." In the memo, he goes on to explain that we need to catch up to our competitors in autonomy and AI. The fourth paragraph starts, "Bring me your initial plan within 60 days, and update me every 90 days thereafter." This was a direction to all RDT&E centers to increase acquisition of new AI and autonomous technology. This puts a burden on the acquisition community to ensure reliable AI systems for deployment.

With regard to the DoD's AI deployed systems, there are no policies, guidelines, or tools that ensure reliability is met during the unique AI aspects of software requirements, architecture, design, code, or test. For example, consider that training data creates ML algorithms. There are currently no constraints on how training data is created too learn from "live" operational environments. Current work in this area demonstrates that training data issues can result in significant redo of work, cost overruns, and schedule delays. Addressing this deficiency is necessary. Because of this deficiency gap in AI policies, guidelines, and tools, system safety practitioners as well as the acquisition community members, including SE and PM, have no support/direction in assessing confidence in the behavior of current WD weapons programs that are implementing AI upgrades. Currently, AI is being developed for integration into critical systems within DoD programs of record. Because this deficiency of support/direction is consistent throughout the DoD, many working groups are attempting to understand the unique aspects of AI software requirements, architecture, design, code and test without any suitable safety guidelines. The five types of documentation comprising the AIDP provides some insight into what the developing agency should provide to support reliable, quality development of AI.

1. AI Justification Report—3 Sections: The "AI Justification Report" asks the requestor to establish that the AI requirements being requested are truly a candidate for AI and cannot be met by traditional software or hardware. The process requires the requestor

perform tasks in three sections: Section 1, the requestor conducts an AI Type Function analysis to determine if there is an AI algorithm meeting at least one of two criteria. Section 2, the requestor documents the justification for the proposed use of the AI algorithm vs. using a more traditional software, firmware or hardware technique. And, Section 3, document a justification for the AI algorithm's level of autonomy.

2. Best Practices Report—2 Sections: The "Best Practices Report" focuses on the requestor presenting development questions and how the AI algorithm will perform its function(s). This process consists of two sections. Section 1 asks the ML Training Data modality type represented in the deployed system and data generation process. Section 2 places emphasis on Dataset Structures by asking questions in eight parts regarding: Representation (training data needed), Fitness of Data (overfit and underfit considerations), Mission Alignment (is training data mission aligned), Mission Alignment (is algorithm adequately trained), Oversight (guardrails or gate control), Sparse and Missing Data (expected ratio of occurrence), Data Curation (process defined), and Battle Complexity (can algorithm support increased complexity).
3. Measurement Report—3 Sections: The "Measurements Report" answers the question about quality and quantity of data in an objective format. It presents two examples of measuring the quality and quantity of a dataset to support a reference for the type of adequate response. Section 1 provides a measurement approach to examine dataset quality. Section 2 provides a measurement approach to examine dataset quantity. The third section focuses on how the previous quality and quantity measurements are representative of the deployed operational environment, including noise factor.
4. Test and Evaluation Readiness Report: The "Testing Readiness Report" ensures that during the T&E process, separate datasets, not used or seen by the developer, are available for the test engineer to use. It also briefly discusses terms like model drift and data drift associated with ensuring that the dataset used is up to date. In addition, it provides guidance on configuration management with an example of using blockchain.
5. Missing and Sparse Class Tables—4 Sections: The "Missing and Sparse Class Tables" consist of four tables/sections, where a table's goals can be compared to tables with actual results. Mixes of attributes when training each class becomes important because of mission and sparse data issues. Section 1 focuses on how the quantity of classes are represented in the dataset. Section 2 looks at ratios, comparing the quantity of classes to other classes. Section 3 provides a table to capture success rate, again a goals and then actuals. And finally, Section 4 compares the quantities of each class to the other to ensure equal representation for algorithm training.

References

Acquisition Notes (AcqNotes). (2022). *Program management tool for aerospace.* https://acqnotes.com/acqnote/careerfields/contract-data-requirements-list-cdrl

Borysowich, C. (2009). Sample contract data requirements list (CDRL). *ToolBox Tech*. https://www.toolbox.com/tech/enterprise-software/blogs/sample-contract-data-requirements-list-cdrl-

Department of the Army. (2021). *Artificial intelligence requirement guidelines and precepts* [White Paper]. Artificial Intelligence (AI) Safety Working Group and the Office of the Director of Army Safety.

DI-MISC-80508B. (2006). *Data item description: Technical report-study/services.*

DOD-STD-1703. (1987). *Military standard: Software product standards.*

DOD-STD-2167A. (1988). *Military standard: Defense system of software development.*

DOD-STD-7935A. (1988). *Military standard: DoD automation information systems (AIS) documentation standards.*

Kendal, A., Das, A. Nagy, B., & Ghosh, A. (2021). Blockchain data management benefits by increasing confidence in datasets supporting artificial intelligence (AI) and analytical tools using supply chain examples. *Proceedings of the Eighteenth Annual Acquisition Research Symposium*, Naval Postgraduate School.

MIL-STD-498. (1994). *Military standard: Software development and documentation.*

Miller, S., & Nagy, B. (2021). Interdependence analysis for artificial intelligence system safety. *Proceedings of the Eighteenth Annual Acquisition Research Symposium*, Naval Postgraduate School.

Nagy, B. (2021a). *Applying generative adversarial network constructs to mission-based simulations to produce "realistic" synthetic training data for machine learning algorithms, GANs overview* [Presentation slides]. Naval Applications for Machine Learning Symposium. NABN581, Naval Information Warfare Center Pacific.

Nagy, B. (2021b). Increasing confidence in machine learned (ML) functional behavior during artificial intelligence (AI) development using training data set measurements. *Proceedings of the Eighteenth Annual Acquisition Research Symposium, Naval Postgraduate School*.

Nagy, B. (2021c). *Tips for applying artificial intelligence to battle complexity, naval applications for machine learning symposium* [Presentation slides]. Fall Naval Applications for Machine Learning Symposium. Naval Information Warfare Center Pacific.

Nagy, B. (2022a). *Fourteen tips to increase confidence in the performance of artificial intelligence (AI)/machine learning (ML) functions during the five stages of development* [Presentation slides]. 2022 National Fire Control Symposium.

Nagy, B. (2022b). *Level of rigor—How to develop quality AI products* [Presentation]. Naval Applications for Machine Learning, NABN708, Naval Information Warfare Center Pacific.

Space and Naval Warfare Command (SPAWAR). (2005). *Software development plan template* (Code 20203 TM-SPP-02 V2.0). Systems Engineering Process Office. https://www.acqnotes.com/Attachments/Software%20Development%20Plan%20Template%20-%20SPAWAR.pdf

Thiebes, S., Lins, S. & Sunyaev, A. (2021). Trustworthy artificial intelligence. *Electron Markets, 31*, 447–464. https://doi.org/10.1007/s12525-020-00441-4

Acquisition Research Program
Naval Postgraduate School
555 Dyer Road, Ingersoll Hall
Monterey, CA 93943

www.acquisitionresearch.net