NAWCWD TP 8864

Level of Rigor for Artificial Intelligence Development

by
Bruce Nagy
Mission Systems Acquisition and Integration (MSA&I) Division
Software and Mission Systems Integration Department

prepared for
Systems Engineering Department

APRIL 2022

NAVAL AIR WARFARE CENTER WEAPONS DIVISION
CHINA LAKE, CA 93555-6100

Naval Air Warfare Center Weapons Division

FOREWORD

This Level of Rigor (LOR) for Artificial Intelligence (AI) Development technical publication was prepared at the request of the Naval Ordnance Safety and Security Activity (NOSSA), Indian Head, MD, and funded by a contract from the Office of the Secretary of Defense (OSD) in support of the Naval Air Warfare Center Weapons Division (NAWCWD) D51 Systems Engineering Department. This project was funded for 1.5 years, over fiscal years (FYs) 2021 and 2022, for the purpose of supporting NOSSA in creating guidelines and policies for any AI technology being used within a weapons system. Over this period of time, various parts of this content have been presented in AI-based government and commercial symposiums and workshops allowing for the evolvement of the content. Additionally, for the latter 6 months of this effort, Department of Defense (DoD) data science communities, including academic and contracting support groups, have provided detailed reviews/feedback. As a result, the suggestions contained in this document have supported the Joint AI System Safety Working Group (SSWG) and the AI Safety Working Group Test and Evaluation, Verification and Validation (TEVV) Subgroup in the development of their guidelines and policies.

This report was reviewed for technical accuracy by Kenneth Chirkis, Code D510000.

Approved by
K. R. CHIRKIS, *Head*
Systems Engineering Department
22 April 2022

Under authority of
W. S. DILLON
RDML, U.S. Navy
Commander

Released for publication by
D. A. CARREÑO
Executive Director

NAWCWD Technical Publication 8864

Published by ..... Technical Communications and Graphics Branch
Collation ..... Cover, 36 leaves
First printing ..... 3 paper, 5 electronic media

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
22-04-2022	Final	1 October 2020 – 31 September 2021

4. TITLE AND SUBTITLE
Level of Rigor for Artificial Intelligence Development (U)

5a. CONTRACT NUMBER
N/A

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
N/A

6. AUTHOR(S)
Bruce Nagy

5d. PROJECT NUMBER
N/A

5e. TASK NUMBER
N/A

5f. WORK UNIT NUMBER
N/A

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Air Warfare Center Weapons Division
1 Administration Circle
China Lake, California 93555-6100

8. PERFORMING ORGANIZATION REPORT NUMBER
NAWCWD TP 8864

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Ordnance Safety and Security Activity
3817 Straus Avenue
South Potomac Indian Head, Maryland 20640-5151

10. SPONSOR/MONITOR'S ACRONYM(S)
NOSSA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
N/A

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
None.

14. ABSTRACT
(U) See reverse.

15. SUBJECT TERMS
Acquisition, AI Confidence, AI Quality Control, AI Trust, AI, Algorithms, Architecture, Artificial Intelligence, Code, Dataset, Design, Level of Rigor, LOR, Machine Learning, ML Requirements, Program Management, Risk Analysis, Sandbox Development, System Engineering, System Safety, Test and Evaluation, Test, Training Data

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Bruce Nagy
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR	66	19b. TELEPHONE NUMBER *(include area code)* (760) 608-7285

14. ABSTRACT (CONTD.)

(U) Throughout the Department of Defense (DoD), artificial intelligence (AI)/machine learning (ML) integration is currently 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? To make this determination, AI/ML product development/acquisition needs a unique evaluation process compared to other traditional software development/acquisition projects. In response, the Naval Ordnance Safety and Security Activity (NOSSA) funded the following research to investigate unique policies, guidelines, tools, and techniques to assess safety concerns within identified AI/ML critical functions. From that effort, 14 key Level of Rigor (LOR) tasks were developed and applied across five stages: (1) Requirements, (2) Architecture, (3) Algorithm Design, (4) Algorithm Code, and (5) Test and Evaluation (T&E). The 14 LOR tasks involve best practice discussions, definitions, measurements, justification documentation, and hazard analysis formats specific to AI/ML systems. These 14 LOR tasks also provide clarity as to why AI/ML software development requires special consideration by the acquisition community. Moreover, this research has the potential of affecting the acquisition community with regard to how they define requirements, create architecture, produce AI/ML algorithm designs, develop AI/ML algorithm code, and perform T&E. The need for the 14 LOR tasks across five stages of development became evident during the development of an "acquisition sandbox" that researched a route planner deploying AI/ML autonomous systems, as well as having those systems deliver packages with a focus on assessing safety in critical function behavior. The sandbox was designed using a DoD Architecture Framework (DoDAF) and Unified Modeling Language (UML) diagrams that incorporate a variety of AI/ML techniques. When faced with this level of complexity and/or uncertainty, the 14 LOR tasks represent a cohesive set of questions/considerations that provide focus in response to current naval AI/ML acquisition issues. These guidelines also provide organizations involved with safety, such as NOSSA and Airworthiness, as well as Acquisition Professionals, including Program Managers and System Engineers, a step-by-step "how-to" evaluation approach that ensures the creation of quality AI-embedded products.

CONTENTS

Introduction 3

Background 4

LOR Tasks 5
Interpretation of Level of Confidence Table 8
Stage 1: During Operational and Systems Requirements Discussions/ Reviews or Equivalent Stage in Development 8
Stage 2: During Architecture Discussions/Reviews (If Applicable, Must Include Architecture of Any Simulations Creating "Synthetic Data" and/or "Live Data" Collection Composing ML Training Sets) or Equivalent Stage in Development 36
Stage 3: During Algorithm Design Discussions/Reviews for AI/ML Functional Candidates Using UML Diagrams/Documents or Equivalent Stage in Development 38
Stage 4: During Algorithm Code Discussions/Reviews for AI/ML Functional Candidates Using Design Pattern Diagrams/ Documents or Equivalent Stage in Development 44
Stage 5: During T&E Discussions/Reviews for AI/ML Functional Candidates Using Test Scripts/Procedures or Equivalent Stage in Development 46

AI Definitions 47

References 61

Bibliography 63

Acronyms 65

Figures:

1. Operational Use Case 5
2. Example of Modality 1 Receiving Attribute Values From Various Sources 14
3. Examples of Modality 2 Training Requiring Images (a, b, and c) or Fused Attribute Data (d) 14
4. Example of Modality 3, Combination of Modality 1 and 2 15
5. TSAT 24
6. StAR-n Order Matrix 27
7. Basic Confusion Matrix 38
8. Typical Receiver Operator Characteristic (ROC) Plot (for Balanced Classes) 39
9. Typical Precision Recall (PR) Plot (for Imbalanced Classes) 39
10. Sweet Spot: Balance Between Enough Bias and Variance to Increase the Predictive Success Rate 41
11. Comparison Illustration of What Was Determined During Development as Compared to What is Determined During T&E Process 44
12. Example of CNN 48
13. Example of DNN 48
14. Examples of Markov Decision Process 50
15. Example of Supervised Learning 51
16. Example of Unsupervised Learning 52
17. ML AI 57
18. Handcrafted Knowledge AI 59

Tables:

1. LOR Tasks 6
2. Ensuring Levels of Confidence 7
3. Training Data Attributes Table 31
4. Attribute Instances by Significant Grouping Instances Table 32
5. Attribute Instances by Significant Grouping Table 32
6. Majority and Minority Class Analysis Table 34

INTRODUCTION

Throughout the Department of Defense (DoD), artificial intelligence (AI)/machine learning (ML) integration is currently 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? To make this determination, AI/ML product development/acquisition needs a unique evaluation process compared to other traditional software development/acquisition projects. In response, the Naval Ordnance Safety and Security Activity (NOSSA) funded the following research to investigate unique policies, guidelines, tools, and techniques to assess safety concerns within identified AI/ML critical functions. From that effort, 14 key Level of Rigor (LOR) tasks were developed and applied across five stages: (1) Requirements, (2) Architecture, (3) Algorithm Design, (4) Algorithm Code, and (5) Test and Evaluation (T&E). The 14 LOR tasks involve best practice discussions, definitions, measurements, justification documentation, and hazard analysis formats specific to AI/ML systems. These 14 LOR tasks also provide clarity as to why AI/ML software development requires special consideration by the acquisition community. Moreover, this research has the potential of affecting the acquisition community with regard to how they define requirements, create architecture, produce AI/ML algorithm designs, develop AI/ML algorithm code, and perform T&E. The need for the 14 LOR tasks across five stages of development became evident during the development of an "acquisition sandbox" that researched a route planner deploying AI/ML autonomous systems, as well as having those systems deliver packages with a focus on assessing safety in critical function behavior. The sandbox was designed using a DoD Architecture Framework (DoDAF) and Unified Modeling Language (UML) diagrams that incorporate a variety of AI/ML techniques. When faced with this level of complexity and/or uncertainty, the 14 LOR tasks represent a cohesive set of questions/considerations that provide focus in response to current naval AI/ML acquisition issues. These guidelines also provide organizations involved with safety, such as NOSSA and Airworthiness, as well as Acquisition Professionals, including Program Managers and System Engineers, a step-by-step "how-to" evaluation approach that ensures the creation of quality AI-embedded products.

This report provides detailed guidelines for the acquisition and development of systems incorporating AI functions. The guidelines allow the user to create varying degrees of confidence in the behavior of the AI function during the challenges of operational deployment. The degree of confidence is determining which of the 14 LOR tasks are being applied across the five stages. Each LOR task provides questions and/or considerations that allow developers to objectively evaluate the safety and reliability of the AI/ML function. When reviewing each LOR task, the "Ref ID" that follows the LOR task number (and related stage) refers to a corresponding identification (ID) within the document used to develop the questions and/or considerations. The four documents begin with the titles (1) Operational View (OV), (2) Systems View (SV), (3) Data Set Design, and (4) Algorithm Design. Examples of LOR task "Ref ID" nomenclature are Ops1, Sys1, Alg1, and Dat1. In the examples, each ID relates to one of the four documents, where the

number "1" indicates the first LOR task described within the document. Using the "Ref ID" within each document supports traceability to the research, including mathematics.

BACKGROUND

The 14 LOR tasks across five stages of development were derived from an "acquisition sandbox" AI development project. The project involved two autonomous robots delivering packages, using an intelligent route planning system, that allowed them to choose from challenging routes involving crime, weather conditions, and human factors. This program was used to mock-up an AI development process for the purpose of creating AI system of systems (see Figure 1).

- **AI Systems/Products Under Development.** The platforms deploying AI within this fictitious project are (1) two autonomous delivery robots (AI system) and (2) intelligent route planners (AI system).
- **Operational Scenario/Products Purpose.** Two robots are each delivering a package. The two robots are carried part way to their destination via two pickup trucks. After the pickup trucks arrive at their destination, each truck unloads their robot. At the unload point, the robots will walk and arrive synchronously to deliver their packages to the intended recipient. The two robots are able to walk long distances using special Global Positioning System (GPS) navigation. As the robots get closer to the package recipient, a Convolutional Neural Network (CNN) image recognition algorithm, using three-dimensional (3D), color spectrum images from one infrared (IR) sensor, (no data curation) will take over navigation. NOTE: The special GPS navigation was loaded with waypoints produced by an AI trained route planner tool. Additionally, the two robots must deliver each of their packages at the same time.
- **Environmental Requirements.** Robots must be able to perform in all weather, i.e., on a rainy day, and be able to determine houses and buildings amongst background clutter. Robots must be able to navigate around vehicles on the road, other robots, and people walking, complex highway systems, and city-like sidewalks and walkways. Robots must be able to identify the package recipient from many other people, some looking very similar to the recipient's profile. Additionally, they must certain people do not receive the package by mistake.

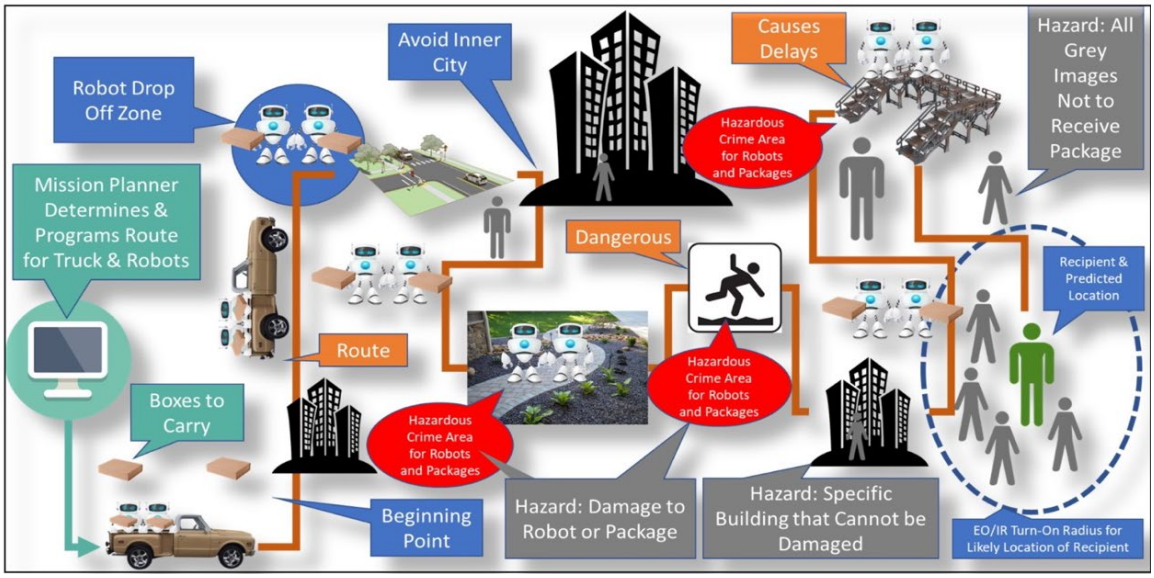

FIGURE 1. Operational Use Case.

From this sandbox approach, (1) DoDAF and UML diagrams were created that identified Message (MSG,) Application Programming Interface (API), and Structured Query Language (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 previously described—this allowed for training data to be acquired consisting of 5 classes and 17 attributes. Using the sandbox development approach allowed for the creation of LOR tasks appropriate for each of the five stages. Through this analysis and development process, the 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 Code, and (5) T&E.

LOR TASKS

The 14 LOR tasks are applied across five stages of AI/ML development: (Stage 1) Requirements, (Stage 2) Architecture, (Stage 3) Algorithm Design, (Stage 4) Algorithm Code, and (Stage 5) T&E. In this report, each stage in the LOR provides considerations and questions to ensure confidence in the development of an AI-based system. In this report, each stage provides questions and considerations for review. In the LOR, examples of tables and measurements will be provided to support detailed discussions with managers and data scientists. Employing the LOR, by ensuring the sandbox followed each stage of development, allows for insights to achieve an AI quality product within a software or hardware system/product. The LOR provides guidance

in terms of what questions and considerations to address and how they should be adequately answered by managers and data scientists. Table 1 provides a reference as to how the 14 LOR tasks are applied across the appropriate stages of AI/ML development process.

TABLE 1. LOR Tasks.

Task No.	LOR Task Type	Stages Involved	Requirement (Stage 1)	Architecture (Stage 2)	Algorithm Design (Stage 3)	Algorithm Code (Stage 4)	T&E (Stage 5)
1	AI Function Type Definition	1	X				
2	Justifying AI Function Needed	1	X				
3	Justifying AI Function Needs to be Autonomous or Semi-autonomous	1	X				
4	Best Practice Discussion About AI/ML Development	3	X	X			X
5	Training Set Alignment Test (TSAT)	3	X	X		X	
6	Sources to Attribute Ratios for 1, 2, or 3 (nth) (StAR-n) Order Matrix	3	X	X		X	
7	Best Practice Discussion on Data Set Generation	3	X	X		X	
8	k-Fold Variation	2	X				X
9	Missing and Sparse Data (MSD) Class Requirements/Actual Tables	3	X		X		X
10	Confusion Matrix Creation	2			X		X
11	Receiver Operator Characteristic (ROC) Area Under the Curve (AUC) or Precision Recall (PR) AUC Analysis	3			X	X	X
12	Bias, Variance, and the Sweet Spot	1			X		
13	Subsystem Hazard Analysis Format	3			X	X	X
14	Best Practice Discussion About AI/ML Algorithm Development	3			X	X	X

Most LOR tasks have multiple parts across the five stages. For example, a list of training data requirements developed in Stage 1 might require a task to compare that list to actual results to ensure the performance of the algorithm was achieved. The various parts of the LOR task between stages support traceability. The majority of task descriptions, related definitions, terms, and explanations occur in Stage 1, when the initial LOR task is introduced. From stage to stage, detailed explanation is reduced to avoid repetition with the exception of “Best Practice” questions, repeated in several stages. LOR tasks build

upon discussions from previous stages. Therefore, it is recommended that the Requirements section, Stage 1, be reviewed first to become familiar with terminology, before moving to other sections. This is especially true when reviewing the final stage, Stage 5, T&E, written in a "bottom-line" format.

The 14 LOR tasks, applied across five stages of development, are designed to raise confidence in the behavior of the AI function when deployed. Table 2 suggests five levels of confidence, from high to minimal. The highest level of confidence, "Highly Confident," involves all 14 LOR tasks applied across all five stages. On-the-other-hand, the lowest level of confidence, "Minimally Confident," involves the first four LOR tasks applied only in Stage 1.

TABLE 2. Ensuring Levels of Confidence.

Confidence Level	Meaning	Stages					LOR Tasks
		1	2	3	4	5	
1	Highly Confident	X	X	X	X	X	All
2	Most Confident	X	X	X		X	All
3	Confident	X	X			X	1 to 9, 11
4	Somewhat Confident	X				X	1 to 9, 11
5	Minimally Confident	X					1 to 4

Table 2 is provided to support the developer in determining how much confidence is needed in the development process, and therefore, how much effort will be involved with raising confidence. An AI-related project might experience time and money constraints that could cause a need to do less, and therefore, the developer might need to perform fewer LOR tasks, creating less confidence in the AI's behavior when deployed. Another option would be to have the sponsor or developing agency mix and match the 14 LOR tasks across five stages using a custom combination to suit specific needs. Again, the table offers a straightforward approach by simply adding stages, and therefore, LOR tasks to raise confidence in the AI product's behavior when deployed.

System safety practitioners have a formal hazard analysis approach to identify their LOR "levels," which corresponds to one of the five levels of confidence described. The most critical, autonomous level for system safety, their LOR 1, would equate to "Highly Confident" in Table 2, using related stages and tasks specified. The other four system safety assessment categories, i.e., LOR categories, would correspond to other confidence levels and tasking.

INTERPRETATION OF LEVEL OF CONFIDENCE TABLE

This section discusses the LOR tasks as they pertain to the five stages of AI development. The bullets below represent another way to interpret Table 2. For example, to achieve Level 1, all 14 LOR tasks must be performed as listed by its 5 sub-bullets. To reiterate, when performing all 14 tasks across the 5 stages, acquisition/development rigor results in a "Highly Confident" approach. This is because the level of confidence is related to a level of control over the development process. Level 1 means that the AI-based systems (or systems of system) will follow a highly controlled process designed to yield a high-quality product.

- Level 1 – Highly Confident (All Development Stages, All LOR Tasks)
  - Stage 1: Recommend Using LOR Tasks 1 to 9
  - Stage 2: Recommend Using LOR Tasks 4 to 7
  - Stage 3: Recommend Using LOR Tasks 9 to 14
  - Stage 4: Recommend Using LOR Tasks 5 to 7, 11, 13, 14
  - Stage 5: Recommend Using LOR Tasks 4, 8 to 11, 13, 14
- Level 2 – Most Confident (Stages 1, 2, 3, 5, All LOR Tasks)
  - Stage 1: Recommend Using LOR Tasks 1 to 9
  - Stage 2: Recommend Using LOR Tasks 4 to 7
  - Stage 3: Recommend Using LOR Tasks 9 to 14
  - Stage 5: Recommend Using LOR Tasks 4, 8 to 11, 13, 14
- Level 3 – Confident (Stages 1, 2, 5, LOR 1 to 9, 11 Tasks)
  - Stage 1: Recommend Using LOR Tasks 1 to 9
  - Stage 2: Recommend Using LOR Tasks 4 to 7
  - Stage 5: Recommend Using LOR Tasks 4, 8 to 11
- Level 4 – Somewhat Confident (Stages 1, 5, LOR 1 to 9, 11 Tasks)
  - Stage 1: Recommend Using LOR Tasks 1 to 9
  - Stage 5: Recommend Using LOR Tasks 4, 8 to 11
- Level 5 – Minimally Confident (Stages 1, 5, LOR 1 o 4 Tasks)
  - Stage 1: Recommend Using LOR Tasks 1 to 4

Each task in each stage provides the specific questions and considerations that when addressed increases control of the development process, thereby having a direct effect on the quality of the product being produced. This is how a high confidence can be gained in the development process. It is gained because there is more control over how the AI is developed. This greater control results in the AI technology's deployed, operational behavior having a greater likelihood to perform as expected. This is in contrast to Level 5, where only the first four tasks in Stage 1 are listed (one sub-bullet). This means that there is very little control in development process. Level 5 represents a "Minimally Confident" development approach. Lack of control, meaning less confidence in the development process, represents a high degree of risk in AI performance when deployed.

Stage 1: During Operational and Systems Requirements Discussions/Reviews or Equivalent Stage in Development

For AI/ML functional candidates using OVs and use case documents:

LOR Task 1 (Stage 1 – Ref Ops1). 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, LOR Tasks 1 to 13 are recommended to follow to address those unique properties.

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

1. 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 that 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 (EUFs). If an algorithm uses a payoff table to make a decision, the EUF approximation might cause decision errors.
2. 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.

Data approximations (Criteria 1) and data samples (Criteria 2) can be considered two sides of the same coin, both describing a representation or sample of a larger population.

AI algorithms are specifically designed to support these limited input conditions and still provide reliable answers. This makes AI a very powerful approach to handling high levels of complexity beyond traditional techniques. Because of this uniqueness, both Criteria 1 and 2 indicate a need for the AI algorithm to be assessed using unique LOR tasking, i.e., questions and considerations specific to AI characteristics. If either criteria is met for a determined function, identifying it as an AI Type Function, then the LOR tasking within this document provides direction as to how to properly assess the algorithm during the development process to establish greater confidence in its functional behavior before it experiences "realistic" deployment issues (Reference 1).

This LOR begins the initial review process is to determine if an AI algorithm is contained within a software function or module. To determine if the AI LOR will be used, the above criteria assesses the function. If any of the criteria is satisfied, the software function is determined as an AI Type Function. The next step is to separate out the AI algorithms from the function, and for each algorithm, identify its inputs and outputs. Once identified, each AI algorithm within the function must be reviewed following the AI LOR. The other software functions need to follow the standard/traditional software LOR approach.

- AI Type Function. An AI Type Function is defined as a function or a module containing one or more AI models. If this function is determined as an AI Type Function, two types of LOR need to be applied: (1) AI rigor and (2) traditionally developed code rigor.
- Traditionally Developed Code. Users coding the algorithm directly into the function, being the translator between the data and the code structure. In contrast, the AI algorithm does the translation between data and code, where the user creates constructs affecting the translation.

For our purposes,

- AI. Machines having the ability to perform functions that mimic levels of human intelligence. The software or hardware functionality is based on algorithm training of data collected, acquired human knowledge, or direct feedback that creates a model/mathematical function to perform analytical evaluation (e.g., assessment, recognition or categorization of an object) and/or determine a numeric solution (e.g., numeric approximation, forecast in a series or regression to define a number or group of numbers).

  Recent advances have created subgroups of AI. The subgroups are (1) ML, (2) neural networks (as subgroup of ML), (3) deep learning (a subgroup of neural networks).

- AI Algorithm. Derived from some form of training data, a mathematical process, or set of rules defined in software or hardware to be followed in a calculation that performs analytical evaluation (e.g., assessment, recognition or categorization of an object), and/or determine a numeric solution (e.g., numeric approximation, forecast in a series or regression to define a number or group of numbers).

- AI Model. An AI algorithm that has been trained to perform analytical evaluation (e.g., assessment, recognition, or categorization of an object) and/or determine a numeric solution (e.g., numeric approximation, forecast in a series or regression to define a number or group of numbers).

  When referring to AI, most times AI algorithm and AI model are used interchangeably. That is why the two definitions are very similar. In general, models, trained algorithms, represent functions. Functions create a subsystem or system. Therefore, an AI Type Function can contain both AI and non-AI algorithm components.

LOR Task 2 (Stage 1 – Ref Ops2). 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:

1. 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 need 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.

2. For Criteria 2, data samples: Can the 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.

LOR Task 3 (Stage 1 – Ref Ops3). 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.

1. 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.
2. If checks and balances are limited, provide documentation as to operational limitations by
   (a) Describing 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?
   (b) Determining how the training data is being generated, e.g., truth, synthetic, combination. Are these sources valid? Why?
   (c) 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.)
   (d) Will an outside independent source review the training, validation, and test data created? Why or why not?

(e) 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?

For AI/ML functional candidates using SVs/documents:

LOR Task 4 (Stage 1 – Ref Sys1). Review of Best Practices to document various discussions that need to be translated into needed requirements.

- Basic AI/ML Requirements:

1. What ML 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 to ensure adequate development of the 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.

   (a) ML Training Data Modality 1 (Figure 2). This modality 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.

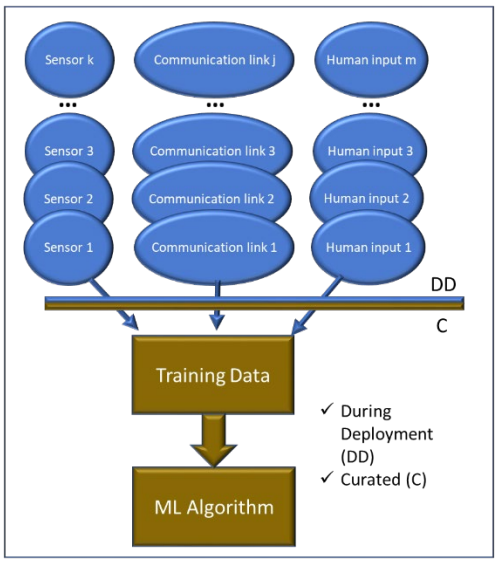

FIGURE 2. Example of Modality 1 Receiving Attribute Values From Various Sources.

(b) ML Training Data Modality 2 (Figure 3). Training data sets that are based on an operational environment from a single data source, where the single data source contains multiple data attributes. 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.

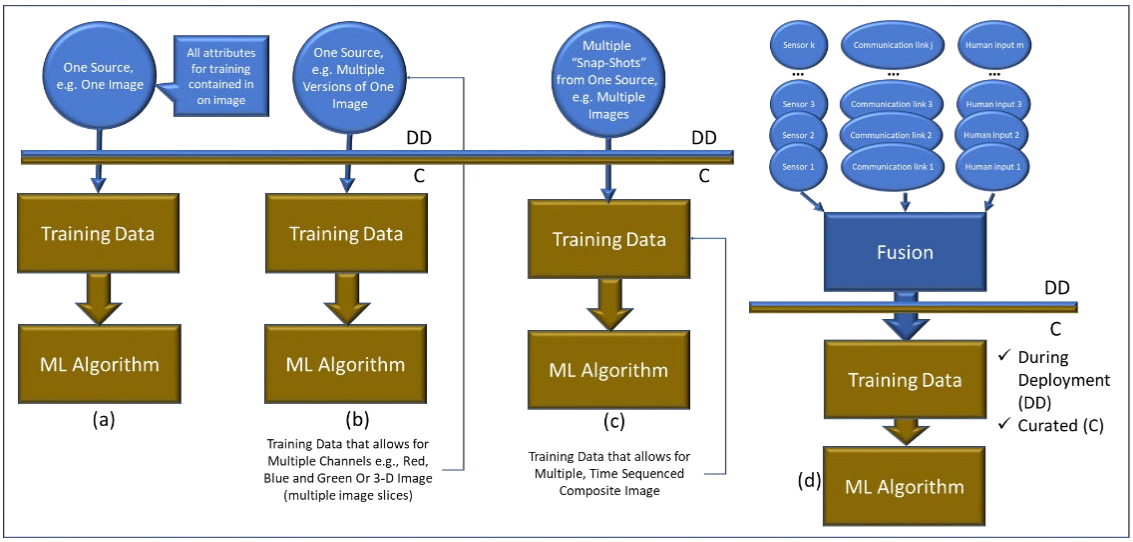

FIGURE 3. Examples of Modality 2 Training Requiring Images (a, b, and c) or Fused Attribute Data (d).

(c) ML Training Data Modality 3 (Figure 4). Training data sets that are based on an operational environment from a combination of multiple data sources; 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.

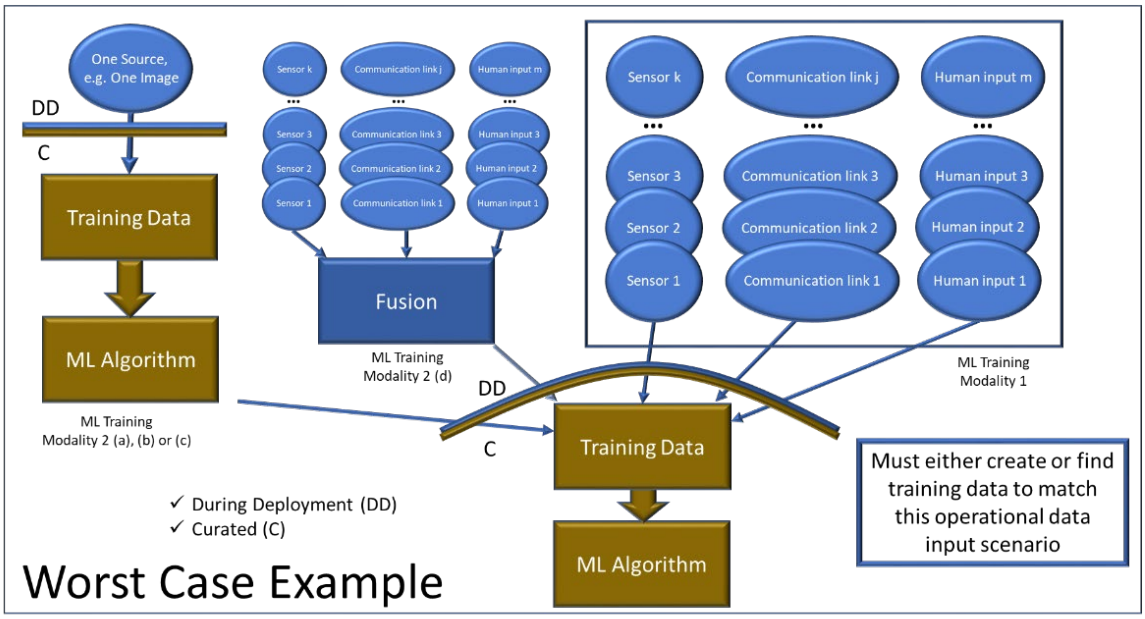

FIGURE 4. Example of Modality 3, Combination of Modality 1 and 2.

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 report, 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.

2. 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.

   (a) 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 is 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.

3. 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?

   (a) 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.

(b) Underfit. Indicates that there is a quantity issue with the training data used. Underfit occurs when there is an insufficient amount of training data that 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.

4. How do we know that the synthetic or live data creating the training data is aligned with the mission parameters?

(a) Was a traceability study performed to support adequate coverage?

(b) Have statistics been shown on the number of configurations available and the number trained using data sources?

(c) How are we avoiding overfitting and underfitting based on training mixes and sets?

(d) 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 previous question 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.

(a) How many operational use cases were created and then translated into training data requirements?

(b) Were the data sources feeding the input to the AI adequately assessed and how does anyone know?

(c) 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.

(a) 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 1,000 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.

(b) 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 does not 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 is 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.

5. 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.

   (a) Is analysis of the algorithm's success rate a function of attribute availability within its anticipated operational environment?

   (b) 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?

6. 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 (Reference 2).

   Requirement content needed for an IA should include

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

   (b) Identification of performers, both machine and human, involved with that function.

   (c) Identification of the method(s) used to ensure the interaction between the human and the machine in terms of observability, predictability, and directability.

(d) Description of the multiple paths through the key function where applicable.

(e) Description of how necessary metrics can be obtained to objectively support any subjective determinations to reduce autonomy discovered through the IA process.

(f) IA failure walk through, including any failure modes associated with AI/ML-enabled functions.

7. 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 or live data, that it will perform as defined.

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

(a) 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?

(b) 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?

(c) For Modality 3: Are combinations of modalities 1 and 2, regarding training of the algorithm, able to deal with "real" operational issues?

8. 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.

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

   (b) 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 is not 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.

   (a) 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?

   (b) 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?

   (c) 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?

9. 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 ML goal, where the organization is based on the classification or regression needs of the algorithm.

   (a) How does your data curation approach avoid “garbage in, garbage out”?

   (b) 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?

10. How well does the particular AI/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).

    (a) "Known-knowns" (facts) – 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.

    (b) "Known-unknowns" (assumptions) – 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.

    (c) "Unknown-knowns" (absent-data) – factors that cause a participant to be "absent of data," sometimes decision critical information; these factors can include human mistakes, sensor failures, and communication issues.

    (d) "Unknown-unknowns" (surprises) – factors that will "surprise" participants during the engagement; these include unforeseen technology and anything not anticipated in the previous three categories.

    Trust that the training data is sufficient begins by completing the LOR tasking. For example, understanding the modality of the training data (as facts) or as will be described in follow-on LOR descriptions, conducting Training Set Alignment Test (TSAT) and Source to Attribute Ratios for 1, 2, 3 (nth) (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 (Reference 3).

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

(a) Can we trust that the training data *factually* represents the real world when deployed, e.g., use of correct attributes/features, noise/background, etc.?

(b) 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?

(c) Can we trust that the *absence of data* when needed to the AI model has been adequately anticipated and compensated to maintain success rate?

(d) 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 is 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.

(a) How will this complexity be considered when synthetically creating or finding data to use for training?

(b) How will the requirements be defined with regard to complexity?

(c) Will guard rails/gates be used?

11. Is there a plan for the AI algorithm to earn trust from its human users? If so, what is the AI trust plan?

    Consider an AI trust plan might use the following architype (1) having the AI learn from subject matter experts (SMEs), where its learning can be continually tested/validated, thereby proving performance and (2) having the AI be involved with "real" technology, learning from firsthand experience what systems can and cannot do, where its learning can be continually tested/validated, thereby proving performance. A final key aspect to using this architype is ensuring that any human involved with the training of the AI receive value, i.e., his or her "What's In It For Me" (WIFM) factor is also filled during the process (References 4 and 5).

12. Is there a need for a special AI Development Plan (AIDP) to be included in the Contract Deliverable Item List (CDRL) that addresses the questions and concerns, i.e., LOR Tasks, described in this report? (Reference 6)

LOR Task 5 (Stage 1 – Ref Dat1). 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 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 the algorithm is using a quality training set?

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 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 (Reference 5).

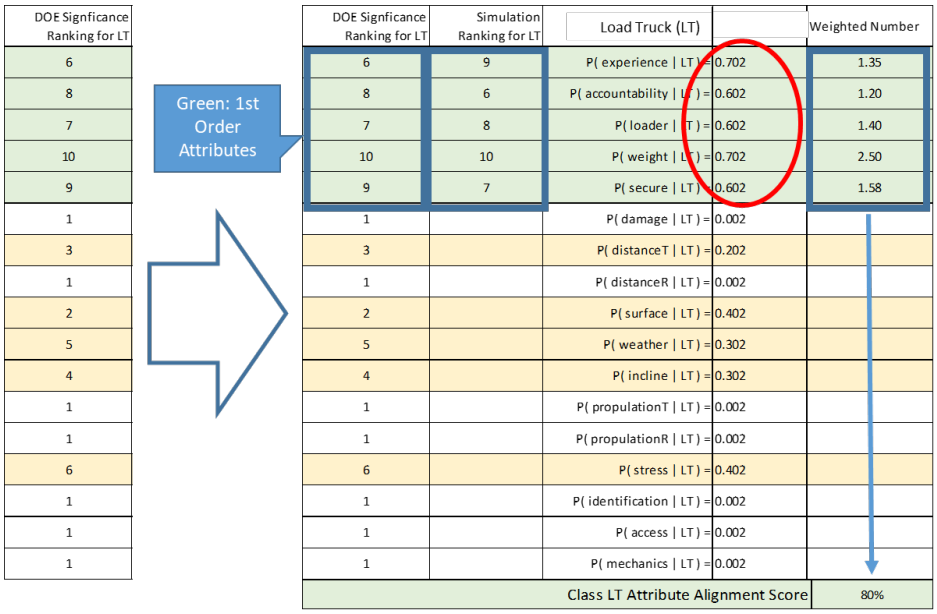

FIGURE 5. TSAT.

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 $n \leq r$ and $n \leq m$, 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

LOR Task 6 (Stage 1 – Ref Dat2). 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, 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?

(a) 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 50 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 (Reference 7).

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 StAR-n Order Matrix approach can support the development of requirements based on attributes (e.g., 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 is 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 is 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: Zones Green and Yellow evidence as to how the algorithm is going to be supervised or monitored when operationally unexpected events occur.

Figure 6 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.

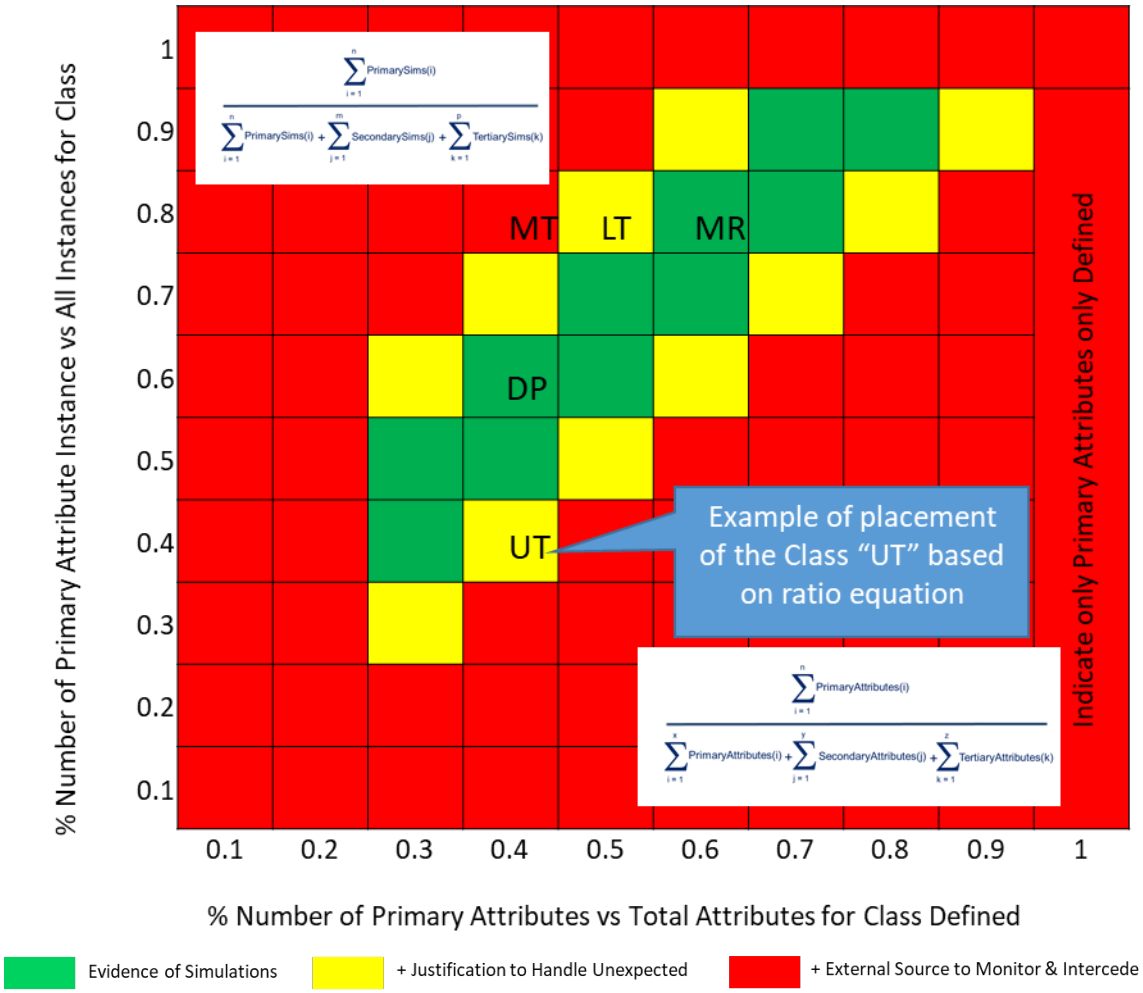

FIGURE 6. StAR-n Order Matrix.

The following steps discuss taking a StAR-n measurement (Reference 8).

During the Requirements Stage 1 and checked during Architecture Stage 2:

1. First Step: Create a ten-by-ten matrix, labeling each axis from zero to 1.
2. 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."
3. 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 show how well training data reflects operational environment. Based on color zone, determine evidence justification. Examples (used for guidance only) are described below:
   (a) 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.
   (b) Zone Yellow: Zone Green evidence plus justification on why n-th group precedence can still handle the unexpected and provide acceptable success rates.
   (c) Zone Red: Zones 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:

4. 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:
   (a) σ (by Class) = (Number of Primary Attributes / Number of All Attributes) ≤ 1.
   (b) δ (by Class) = (Number of all Primary Instances / Number of All Instances) ≤ 1.
5. 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.
   (a) 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.
   (b) Zone Yellow: Zone Green evidence plus justification on why n-th group precedence can still handle the unexpected and provide acceptable success rates.

(c) Zone Red: Zones Green and Yellow evidence as to how this algorithm is going to be supervised or monitored when operationally unexpected events occur.

LOR Task 7 (Stage 1 – Ref Dat3). Review of Best Practice to document discussions on data set generation or data set collection (from "live" data to identify needed requirements.

1. How do you know if the quality and quantity of Training Data is sufficient?
   (a) Quality refers to the correct number of attributes (including primary, secondary, etc. mixes) that are representative of the deployed operational environment, including noise factors.
   (b) Quantity refers to the amount of data/instances used for training, with consideration to mix ratios, underfitting, overfitting, and majority/minority classes.
2. How do you assess the operational limits described by the training data? (Consider the "You don't know what you don't know" issue.)
3. 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?
4. For simulation generation of the training data:
   (a) 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/underfitting).
   (b) 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?

LOR Task 8 (Stage 1 – Ref Alg1). 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?

1. 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?

(a) 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., boy's face evolves to a man's face but never updated in the training data.

(b) 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.

2. 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.

3. As an example, the need to use positive control over 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 the 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 (Reference 9).

   (a) 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.

LOR Task 9 (Stage 1 – Ref Alg2). Will a Missing and Sparse Data (MSD) Class Requirements 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 in this section 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: Create a table or list by class 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 3 is a Section 1 example of a *Training Data Attributes Table* 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.

TABLE 3. Training Data Attributes Table.

Class	No. of Max Data Source for a Decision	Attributes			±% Compared to Actuals
		Primary	Secondary	Tertiary	
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%

- Section 2: 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 4 is a Section 2 example of an *Attribute Instances by Significant Grouping Instances Table* 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.

TABLE 4. Attribute Instances by Significant Grouping Instances Table.

Attribute Instances to Total Instances	Class a	Class b	Class c	…	Class n	±% 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%

- Section 3: 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 5 is a Section 3 example of *Attribute Instances by Significant Grouping Table* 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.

TABLE 5. Attribute Instances by Significant Grouping Table.

Test Results by Class	Data Source/Attributes			±% Compared to Actuals
	Primary	Secondary	Tertiary	
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 2nd order testing success rate (emphasis 1st order)	2%
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)	2%
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)	2%

- Section 4: 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.

  (a) 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 Order), may be the only consideration when analyzing each class.

  Table 6 is a Section 4 example of a *Majority and Minority Class Analysis Table* 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.

TABLE 6. Majority and Minority Class Analysis Table.

Possible Combinations of Attributes Based on Operational Needs	Expected Instances for Training					Balanced
	Class a	Class b	Class c	…	Class n	
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)	?	?	?	…	?	

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.

Stage 2: During Architecture Discussions/Reviews (If Applicable, Must Include Architecture of Any Simulations Creating "Synthetic Data" and/or "Live Data" Collection Composing ML Training Sets) or Equivalent Stage in Development

For AI/ML functional candidates using system views and design documents:

LOR Task 4 (Stage 2 – Ref Sys1). Review of Best Practice documented discussions to ensure compliance to requirements identified in Stage 1. This is to ensure traceability from requirements to architecture. As a review,

- Basic AI/ML Requirements:
    1. What ML Training Data modality type are you representing in your deployed system and your data generation process?
    2. Does the synthetic or live data represent Classes for the ML process? If not, how are the Classes represented?
    3. Does each Class have a sufficient number of attributes that can be learned by the algorithm for that Class? Has overfitting and underfitting been considered for each Class with regard to the quantity of attributes simulated and do they reflect real-world operations?
    4. How do we know if the synthetic or live data that is creating the training data (defined at truth) is aligned with the mission parameters? Was a traceability study performed to support adequate coverage? Have statistics been shown on how many configurations are available, and how many were 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 type or numbers of attributes that can adequately represent missing and sparse data occurrences from related sources?
    5. How are we ensuring that the algorithm being deployed provides the correct answer even when data input issues occur? Is analysis of the algorithm's success rate a function of attribute availability within its anticipated operational environmental?
    6. Can other control entities (such as a human operator) be inserted into the loop to reduce autonomy of the function?
        (a) Does the synthetic or live data used to create training data represent sparse and missing data occurrences?
        (b) Does the architecture, design, and code support sparse and missing data management, specifically filter or provide a selection of less significant attributes to do the calculations?
        (c) Does the data management support filtering to ensure that the ML algorithm provides an accurate data input avoiding "garbage in, garbage

out"? Has what has been constituted as "garbage in, garbage out" been defined?

(d) How well does ML algorithm support increased complexity, and how does that affect sparse and missing data issues?

LOR Task 5 (Stage 2 – Ref Dat1). Based on ML Training Data Modality being implemented, review congruence between deployed architecture and the data set generation process; specifically review the priority/rating of attributes defined in TSAT to document compliance to with requirements identified in Stage 1.

LOR Task 6 (Stage 2 – Ref Dat2). Based on ML Training Data Modality being implemented, review congruence between deployed architecture and the data set generation process; specifically review to the categorized ratios of attributes defined in the StAR-n to document compliance with requirements identified in Stage 1.

LOR Task 7 (Stage 2 – Ref Dat3). Review of Best Practice documented discussions to ensure compliance to requirements identified in LOR 7, Stage 1. This is to ensure traceability from requirements to architecture. As a review,

1. How do you know if the quality and quantity of Training Data is sufficient?
   (a) Quality refers to the correct number of attributes (including primary, secondary, etc. mixes) that are representative of the deployed operational environment, including noise factors.
   (b) Quantity refers to the amount of data/instances used for training, with consideration to mix ratios, underfitting, overfitting, and majority/minority classes.
2. How do you assess the operational limits described by the training data? (Consider the "You don't know what you don't know" issue.)
3. 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?
4. For simulation generation of the training data:
   (a) 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/underfitting).
   (b) 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?

Stage 3: During Algorithm Design Discussions/Reviews for AI/ML Functional Candidates Using UML Diagrams/Documents or Equivalent
Stage in Development

Code Design for an AI/ML is different from traditional software development design. It normally consists of mathematics that will translated into software. An example of a design describing the mathematical equations that will be coded in software using combination of AI/ML and game theory is provided in Reference 10. Reference 10 is provided for an appreciation and comparison to the software equivalent only. AI/ML algorithm design is different. It is not necessary to understand the algorithm design to complete the following LOR tasking.

For Stage 3, the following LOR tasks will apply. Some of the LOR tasking is a follow-on for traceability from previous Stages, other LOR tasks are new, specific to algorithm design.

LOR Task 9 (Stage 3 – Ref Alg2). Create a MSD Class Actuals Table, including Stages 1, 2, and 3 Data Source, Success Rate, and Majority/Minority Class Analysis Requirements.

LOR Task 10 (Stage 3 – Ref Alg3). Review justification for the algorithm's design through improved performance in terms of confusion Matrix parameters, e.g., sensitivity, specificity, precision.

See Figure 7 for an example of a Basic Confusion Matrix.

Sensitivity = True Positive Rate = True Positives / (True Positives + False Negatives)
Specificity = True Negative Rate = True Negatives/ (True Negatives + False Positives)
(1 – Specificity) = False Positive Rate = False Positives/ (True Negatives + F Positives)

Confusion Matrix	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

FIGURE 7. Basic Confusion Matrix.

Notes:

1. When value of Sensitivity is high, algorithm is trained to be good at predicting true positives.

2. When value of Specificity is high, algorithm is trained to be good at predicting true negatives.

Points plotted represent many Confusion Matrices with a hyperparameter having different values, e.g., thresholds to determine an optimal threshold to use when making a decision regarding accurate recognition of class (Figure 8).

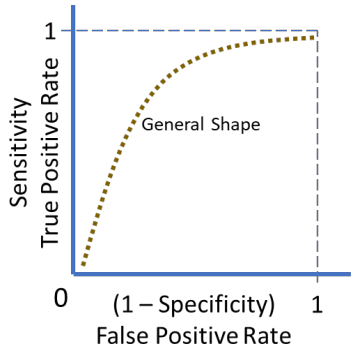

FIGURE 8. Typical Receiver Operator Characteristic (ROC) Plot (for Balanced Classes).

Imbalanced classes mean, one class has more training samples/instances and is significantly larger than the others. The classes with the larger number of instances are called the majority class and those with the smaller number of instances are the minority class (Figure 9).

Precision = Positive Predictive Power = True Positives / (True Positives + False Positives)
Sensitivity = Recall = True Positives / (True Positives + False Negatives)
(1 – Precision) = False Positives / (True Positives + False Positives)
Accuracy = (True Positives + True Negatives) / T Pos + F Pos + F Neg + T Neg)

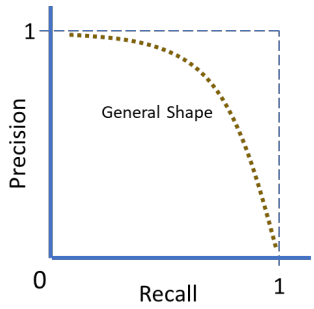

FIGURE 9. Typical Precision Recall (PR) Plot (for Imbalanced Classes).

LOR Task 11 (Stage 3 – Ref Alg4). Ask whether ROC Area Under the Curve (AUC) or PR AUC graph will be used, with respect to the potential of Majority and Minority classes, based on training sets to determine Sweet Spot and Best Algorithm selection – variance (functional performance measured through cost/loss/error functions) vs. bias (functional fit/modification to training set composition). The goal is to have the model functionally perform with just enough bias to cause a low output variance associated with the data set, i.e., making the model not too complex or simple (References 11 and 12).

- The sweet spot is achieved when you adjust the algorithm's hyperparameters to have a high True Positive rate and a low False Positive rate.
- Bias measurements determines how much the algorithm function has been modified to fit the training data. An example of regression bias is how the curve fit function is defined. An example of categorization bias is the selection of threshold for categorization. Zero bias, the difference between the function result and the truth result are identical, meaning it has been precisely fit and likely will cause a high degree of variance with the test data.
- Variance measurements determines how the ML algorithm function performs. An example of regression variance is the measurement of least mean squared error or in the case of logistic regression, it optimizes the sum of the likelihoods because it is based on maximum likelihood to find the best fitting line based on the training data. For categorization, the variance is measured at the difference between the error rate and the bias. (Note: Error rate is the probability that the algorithm will misclassify for at given input.)

How is underfitting being considered, in other words, showing a high bias and low variance result? Are any of the techniques below to reduce underfitting being used?

- Increase model complexity.
- Increase number of features, performing feature engineering.
- Remove noise from the data.
- Increase the number of epochs or increase the duration of training to get better results.

How is overfitting being considered, in other words, showing a high variance and low bias result? Are any of the techniques below to avoid overfitting being used?

- Increase training data.
- Reduce model complexity.
- Early stopping during the training phase (Monitor the loss function results over the training period. As soon as loss begins to increase, stop training).

- Ridge Regularization and Lasso Regularization.
- Use dropout for neural networks to tackle overfitting.

LOR Task 12 (Stage 3 – Ref Alg5). For the algorithms considered, review Best Practices (industry approaches specific to selected algorithms) regarding bias, variance, and "sweet spot" determination to document various discussions for reference when implementing this LOR (Figure 10).

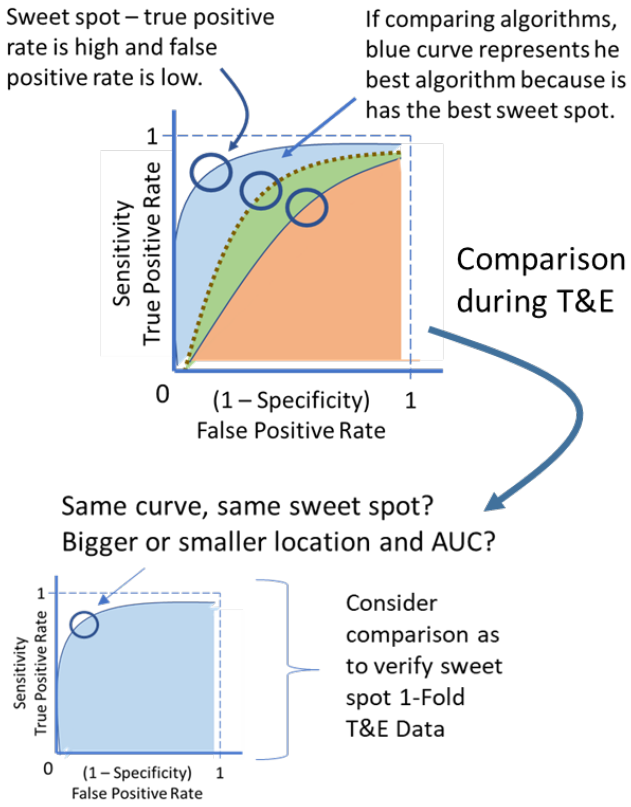

FIGURE 10. Sweet Spot: Balance Between Enough Bias and Variance to Increase the Predictive Success Rate.

Note that the goal of the ML algorithm developer is to find the "Sweet Spot:" In other words, create balance between enough bias and variance to increase the predictive success rate. Techniques to achieve predictive success: popular approaches are regularization – modifying bias using additional factors (e.g., Ridge and Lasso for Logistic Regression), bagging – adding modeling complexity to reduce variance (e.g., for Random Forest), boosting – reducing variance and bias by combining weak learners to make a strong learner (e.g., AdaBoost, XG boost).

LOR Task 13 (Stage 3 – Ref Alg6). Create a Subsystem Hazard Analysis Format to Assess Quality and Quantity of the Training Data by following thread and mapping to issues that might cause a failure to the AI. Here are some questions to consider:

1. Would the corruption of API/MSG/SQL/Other affect data variations and require additional training of the Target Algorithm? This is a yes or no answer.
2. If yes, will the quality (composition/complexity/structure) of Training Data significantly increase? Will it affect the ML Training Modality? Explain specific to the API/MSG/SQL/Other interfaces to the algorithm. Corruption might mean a need to add secondary or tertiary sources. It might also affect how data is collected from various sources, potentially changing ML Training Modality.
3. Will these variations be part of the analysis for selecting the "best" algorithm? Explain. ROC sweet spot analysis might be used with hyperparameter changes based on the type of variation.
4. Because of this issue, will the quantity (more instances) of Training Data significantly increase? The answer needs to be specific to the API/MSG/SQL/Other interfaces to the algorithm.
5. Will creating/finding enough training data that replicates the corruption be an issue? Explain. If it is synthetic, it may not be an issue depending on the model. If it comes from "live" data, can this type of corruption be collected and would there be enough of this type of issue collected to support the training of the algorithm?
6. Are you confident that any additional data created/found will adequately represent the effects associated with replicating the corruption? Explain. This is an important statement related to the quality (composition/complexity/structure) of the Training Data.

LOR Task 14 (Stage 3 – Ref Alg7). Review of Best Practice Topics about AI/ML Algorithm Development to document discussions on data set generation to identify needed requirements. This is to ensure traceability from Architecture to Algorithm design. As a review,

1. General Questions to Consider if ML Algorithm is appropriate:
   (a) How do you trust the behavior in the real world for this ML function? Success Rate? Quality of Training Data?
   (b) Was ML function the correct selection for this function within the system? How did you know?
   (c) Was choice based on what gives you the best operational performance and understanding of operational limits? How did you determine that?

(d) How reliable will the function perform in the real world in terms of defined operational parameters? How do you assess the operational limits of this ML function? (Don't know what you don't know.)

(e) How do you know that the hyperparameters were optimally used?

2. Are there other potential best practices specific to the algorithm selected? For example, in the case of using a Naïve Bayes approach, the following questions may be posed:

   (a) Is MAP (Maximum A Posteriori) or Maximum Likelihood better?

   (b) Is the size of the alpha (a Naïve Bayes hyperparameter) correct? In this type of Naïve Bayes application: What attributes do you choose to simulate and the number of times (based on what the developer/engineer deems important will determine how this algorithm learns to distinguish between classes)? Would a Gaussian approach be more effective? (Yet, even using a Gaussian approach, the number of simulations will make a difference in performance.)

   (c) How do you know that the attributes used in Naïve Bayes are really independent?

   (d) Are there proven statistical methods that can be used to investigate training data sufficiency, e.g., z-value analysis of Logistics Regression algorithm using Wald's test per our use Case example?

Stage 4: During Algorithm Code Discussions/Reviews for AI/ML Functional Candidates Using Design Pattern Diagrams/Documents or Equivalent Stage in Development

LOR Task 5 (Stage 4 – Ref Dat1). Review of ML training data set priority/rating in the rating tool, TSAT, to ensure compliance to requirements identified in LOR 5 Stage 1.

LOR Task 6 (Stage 4 – Ref Dat2). Review of ML training data set ratios in the ratios tool, StAR-n Order Matrix, to ensure compliance to requirements identified in LOR 6 Stage 1.

LOR Task 7 (Stage 4 – Ref Dat3). Review of Best Practice documented discussions to ensure compliance to requirements identified in LOR 7 of Stage 1.

LOR Task 11 (Stage 4 – Ref Alg4). Review ROC AUC or PR AUC graph and analysis that determined Sweet Spot and Best Algorithm selection (Figure 11).

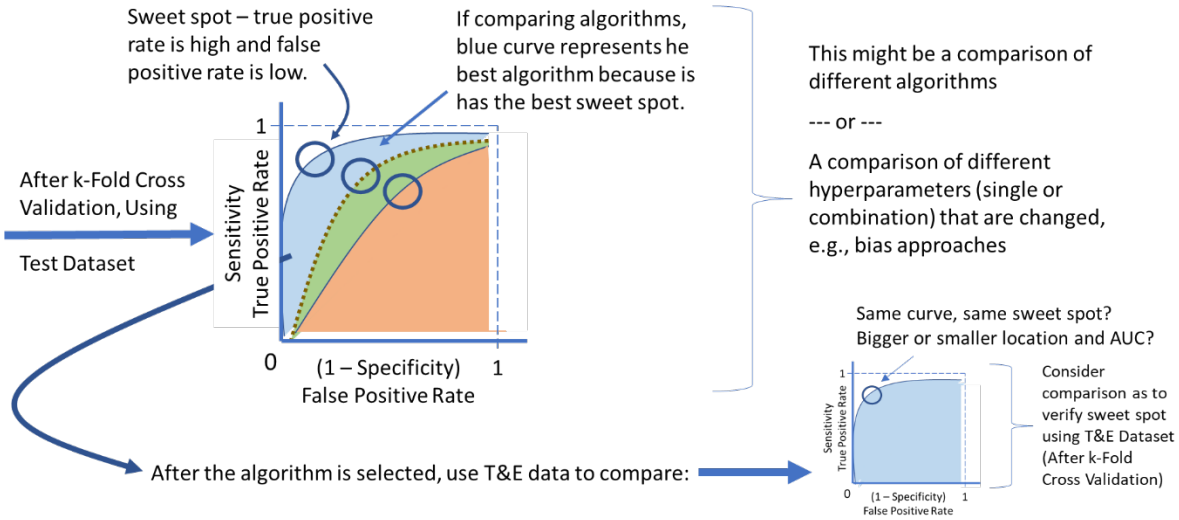

FIGURE 11. Comparison Illustration of What Was Determined During Development as Compared to What is Determined During T&E Process.

LOR Task 13 (Stage 4 – Ref Alg6). Document that the Subsystem Hazard Analysis Format to Assess Quality and Quantity of the Training Data was part of Code Discussions/Reviews.

LOR Task 14 (Stage 4 – Ref Alg7). Review of Best Practice Topics (see Alg7 questions) about AI/ML Algorithm Development to ensure compliance.

1. General Questions to Consider if ML Algorithm is appropriate:
   (a) How do you trust the behavior in the real world for this ML function? Success Rate? Quality of Training Data?
   (b) Was ML function the correct selection for this function within the system? How did you know?
   (c) Was choice based on what gives you the best operational performance and understanding of operational limits? How did you determine that?
   (d) How reliable will the function be to perform in the real world in terms of defined operational parameters? How do you assess the operational limits of this ML function? (Don't know what you don't know.)
   (e) How do you know that the hyperparameters were optimally used?
2. Other Potential Best Practice specific to the algorithm selected in terms of operational performance, for example using Naïve Bayes. Example questions below associated with Naïve Bayes Algorithm Implementation. (Suggest that a collection questions, like the ones below, are gathered from an independent team familiar with the algorithm being developed.)
   (a) Is MAP or Maximum Likelihood better?
   (b) Is the size of the alpha (a Naïve Bayes hyperparameter) correct? In this type of Naïve Bayes application: What attributes do you choose to simulate and the number of times (based on what the developer/engineer deems important will determine how this algorithm learns to distinguish between classes)? Would a Gaussian approach be more effective? (Yet, even using a Gaussian approach, the number of simulations will make a difference in performance.)
   (c) How do you know that the attributes used in Naïve Bayes are really independent?
   (d) Are there proven statistical methods that can be used to investigate training data sufficiency, e.g., z-value analysis of Logistics Regression algorithm using Wald's test per our use Case example?

Stage 5: During T&E Discussions/Reviews for AI/ML Functional Candidates Using Test Scripts/Procedures or Equivalent Stage in Development

LOR Task 4 (Stage 5 – Ref Sys1). Review of Best Practice documented discussions (see Sys1 questions previously discussed) to ensure compliance to requirements identified in Stage 1. This has been repeated several times and there should be traceability from the requirements, through architecture, through design and finally, to testing.

LOR Task 8 (Stage 5 – Ref Alg1). Use the k-Fold (T&E Variation) process to conduct functional testing, including those defined LORs 4 through 7, and 10 through 14.

LOR Task 9 (Stage 5 – Ref Alg2). Compare the Requirements Table and Actuals MSD Class Table, including Stages 1, 2, and 3 Data Source, Success Rate, and Majority and Minority Class Analysis Requirements to ensure compliance.

LOR Task 10 (Stage 5 – Ref Alg3). Conduct confusion Matrix testing for parameters defined in Stage 1, e.g., sensitivity, specificity, precision.

LOR Task 11 (Stage 5 – Ref Alg4). Conduct testing to determine ROC AUC or PR AUC graph and analysis to verity Sweet Spot identified in Stage 2 LOR 12 (Stage 5 – Ref Alg4). Conduct testing to determine ROC AUC or PR AUC graph and analysis to verify Sweet Spot identified in Stage 2.

LOR Task 13 (Stage 5 – Ref Alg6). Ensure testing covers identified columns in Subsystem Hazard Analysis Table, i.e., Quality and Quantity related performance issues.

LOR Task 14 (Stage 5 – Ref Alg7). Review of Best Practice Topics (see Alg7 questions previously discussed) about AI/ML Algorithm Development to ensure compliance. This has been repeated several times and there should be traceability from the requirements, through architecture, through design and finally, to testing.

AI DEFINITIONS

AI Algorithm. A mathematical process, or set of rules, defined in software or hardware to be followed in a calculation that uses some form of training data to perform recognition, e.g., categorization of an object and/or problem solving, e.g., assessment or categorization of an object and/or determine a numeric solution, e.g., numeric approximation, forecast in a series or regression to define a number or group of numbers.

AI Bias. Bias is the difference between what the model forecasts and what is truth. Most times bias occurs when the training data does not equally represent all of the deployed environment but focuses on a subset. A form of bias is imbalanced classes. Imbalanced classes mean one class has more training samples/instances than the other classes. The classes with the larger number of instances is called the majority class, and the smaller number of instances is the minority class. Data imbalance could lead to unexpected performance of the AI model. Another form of bias is overfitting and underfitting (see those definitions.) Sometimes the way the data is collected and labeled (or measured) to identify what is truth has errors, causing a poor training set. Too many outliers within the training data can cause bias. Inconsistent labeling can cause bias. Human curation issues, belief systems (fruit vs. vegetable), paradigm focus (this is important, not that), etc. can cause bias. Depending on how bias is used, it can also support more accurate forecasting. As an example, biasing can ensure the training of the model is focused on priority features instead of noise.

Artificial Intelligence (AI). Machines having the ability to perform functions that mimic levels of human intelligence. The software or hardware functionality is based on algorithm training of data collected, acquired human knowledge, or direct feedback that creates a model/mathematical function to perform analytical evaluation (e.g., assessment, recognition or categorization of an object) and/or determine a numeric solution (e.g., numeric approximation, forecast in a series or regression to define a number or group of numbers).

Recent advances have created subgroups of AI. The subgroups are (1) machine learning (ML), (2) neural networks (as subgroup of machine learning, (3) deep learning (a subgroup of neural networks). These definitions are described below along with other AI types (Reference 13).

Typical types/applications of AI:

- **AdaBoost.** Short for Adaptive Boosting, is a statistical classification meta-algorithm. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms (“weak learners”) is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers (Reference 14).

- **Convolutional Neural Network (CNN).** CNNs are a variety of artificial neural networks often used in image processing (Figure 12). A CNN is typically used in image recognition and processing that is specifically designed to process pixel data.

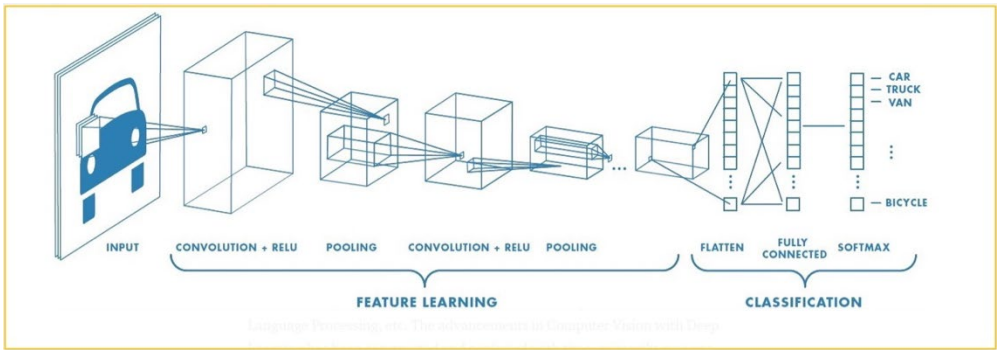

FIGURE 12. Example of CNN.

- **Deep Learning.** ML focused on frameworks to learn.
- **Deep Neural Network (DNN).** An application of AI defined as an advanced form of ML that utilizes big data to create a computational model without manually extractive features. Deep Learning applications can learn by way of supervised, unsupervised, and/or reinforcement learning, and often use an approach of multiple layers of nonlinear processing to extract features and create a transformation for predicting an output based on an input (Figure 13). It could support complex boundary classification problems. (Also, see *Machine Learning*.)

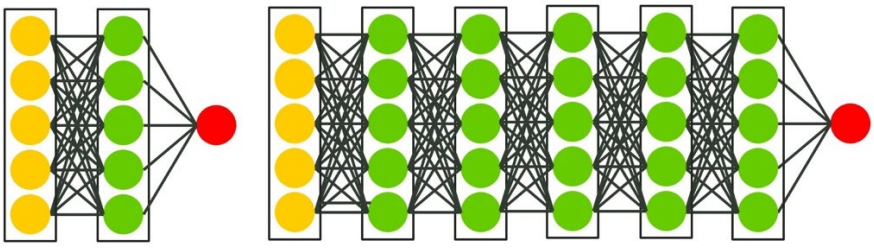

FIGURE 13. Example of DNN.

- **Inference Engine.** Logical rules on a knowledge base to reach a conclusion based on the evidence presenting the knowledge base and reasoning associated with that evidence.
- **K-Nearest Neighbor (KNN).** K-Nearest Neighbor is one of the simplest ML algorithms based on Supervised Learning technique. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

- **Logistic Regression.** Logistic regression is a ML algorithm that is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability. The hypothesis of logistic regression tends it to limit the cost function between 0 and 1. Logistic regression is another technique borrowed by ML from the field of statistics. It is the go-to method for binary classification problems (problems with two class values).
- **Monte Carlo Tree Search.** A heuristic function to efficiently traverse the search space.
- **Naïve Bayes.** Naïve Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naïve Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 centimeters in diameter. A Naïve Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features (Reference 15).
- **Natural Language Processing (NLP).** A computer technology that can process, understand, detect, and identify patterns as language in audio using AI models.
- **Natural Language Generation (NLG).** NLG is defined as an application of AI that produces conversational text or speech understandable by humans.
- **Neural Network (Artificial).** Neural Network(s) is defined as a computer system or application of AI trained by reinforcing connections between inputs, intermediate learning layers, and the outputs of what is learned.
- **Random Forests.** Random Forest or Random Decision Forests are an ensemble learning method for classification, regression, and other tasks that operates by constructing a multitude of decision trees at training time.

Types of ML Paradigms:

- **Reinforcement Learning (RL).** RL is an AI training method based on rewarding desired actions or punishing/no response for undesired actions through a reward function. The RL process, through the reward/punishment approach, creates increased values for successful actions and develops a policy in which highest valued actions, i.e., greatest rewards, are performed from state to state. Therefore, the algorithm learns the "correct" actions for each state. States and actions can be modeled (Markov Decision Process; Figure 14) or model-less (Q-Learning or state-action-reward-state-action [SARSA]). As an example, training data can be represented by state attributes, e.g., dynamic motion to make a decision as to whether an aircraft is friend of foe. "Truth" as to best action to take needs to be

provided using a feedback mechanism to support the algorithm's learning. For example, an expert in the field providing feedback or a measurement grading the action, e.g., state analysis of the dynamic motion to determine friend or foe. The reward function can also assess a string of actions to determine the best sequence with the goal of maximizing reward. When dealing with RL algorithms interacting in "live" environments, the training data set, i.e., state attributes and related action "truth," will likely only represent a sample of the total states available. An approach to limit an infinite number of states is to approximate how each state is characterized. Continuing with the previous example, instead of defining a state with a specific altitude, the elevation might be approximated to be either above or below a set altitude. A collection of all the characterized states is considered a training data set.

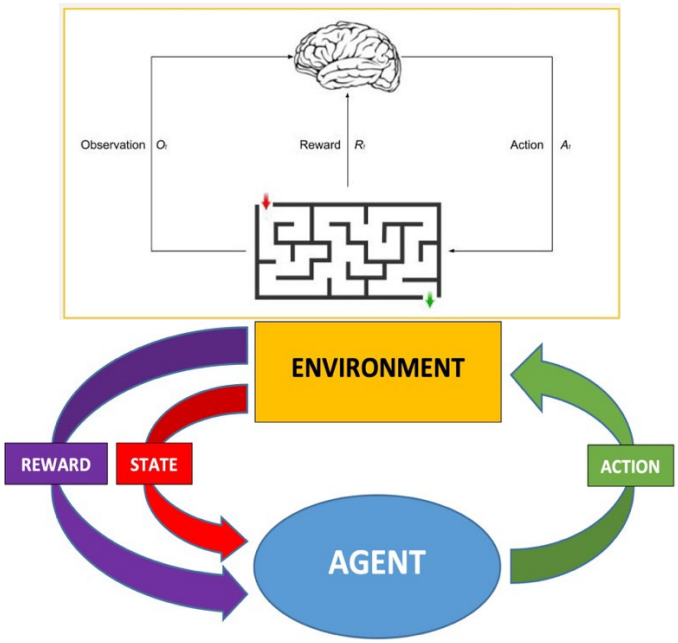

FIGURE 14. Examples of Markov Decision Process.

- **Supervised Learning.** A learning technique in which the algorithm is developed to create a function that performs a nonlinear task, such as recognizing an image, by using a feedback mechanism that optimizes its performance (Figure 15). Therefore, the training data input needs to be labeled for the algorithm to use as feedback to accurately tune the function's performance using cost/loss/error values.

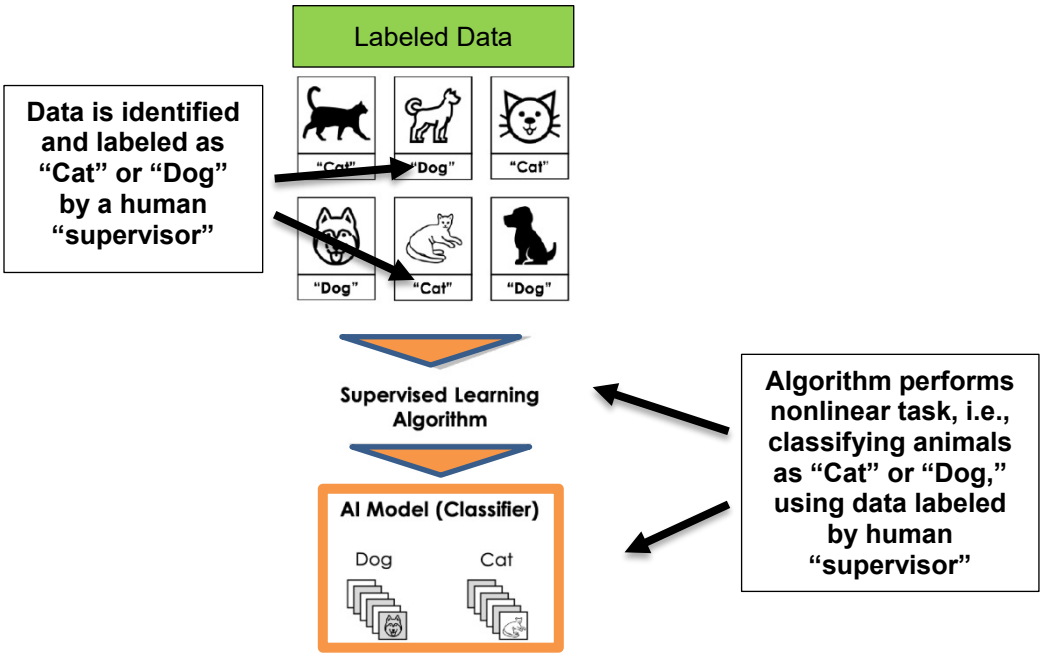

FIGURE 15. Example of Supervised Learning.

- **Unsupervised Learning.** A learning technique in which the algorithm is using a mechanism to show how features within the training data are grouped/clustered by their displayed similar patterns (Figure 16). This gives insight/discovery into how to manage or analyze the input data. An example would be to use unsupervised learning on communication signals to identify similar patterns in various wavelengths and amplitudes or even hidden modulation approaches.

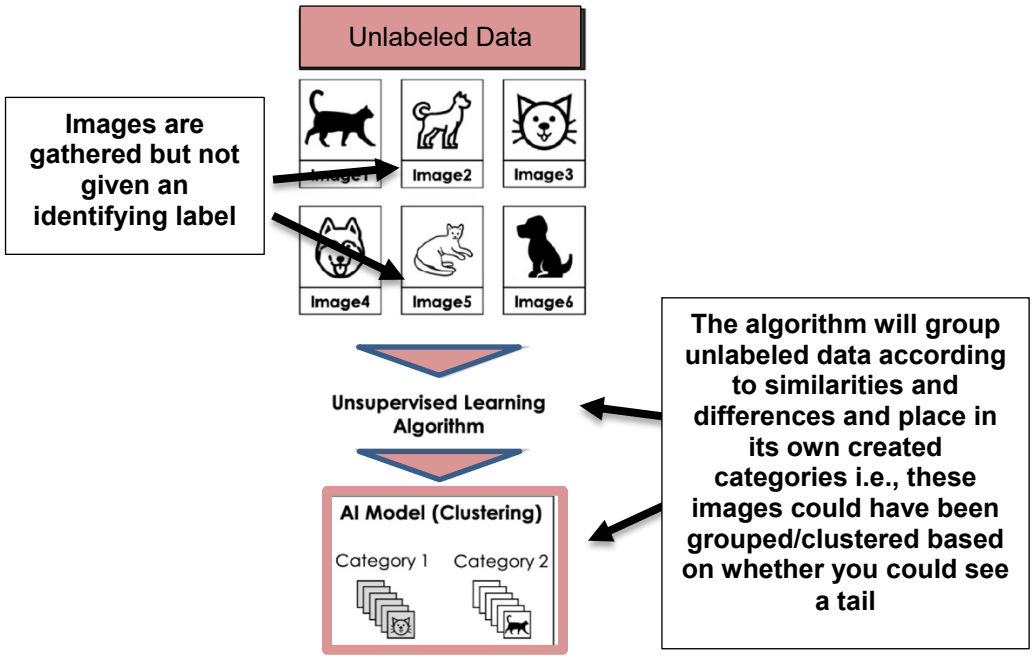

FIGURE 16. Example of Unsupervised Learning.

AI Model. An AI Algorithm that has been trained to perform recognition, e.g., categorization of an object and/or problem solving, e.g., assessment or categorization of an object and/or determine a numeric solution, e.g., numeric approximation, forecast in a series or regression to define a number or group of numbers.

AI Type Function. A function or module containing one or more AI/ML models. The function can be a combination of traditional software/firmware/complex electronics and AI trained/built models implemented in software/firmware/complex electronics.

AI Type Function Classification. For system safety concerns, an AI Type Function means that an algorithm will be developed (1) from using data approximations to build its model from the algorithm, e.g. from simulations and synthetic data vs. an equation that accurately represents real world physics, and/or (2) when data samples used to build its model from the algorithm is a subset of the actual population size, e.g., training data

samples from population to support ML, training data samples requiring clutter backgrounds.

AI Safety-Significant Data Set. An AI Safety-Significant data set is used to train, validate, and test AI-enabled systems that have been identified as performing safety-significant functions that could cause or contribute to one or more system level hazard(s) or mishap(s).

Autonomy – Levels of Functionality:

- **Autonomous.** Function exercises control authority over potentially safety-significant hardware systems, subsystems, or components without the possibility of predetermined safe detection and intervention (without any interlocks) by an independent control entity to preclude the occurrence of a mishap or hazard.
    - Function that displays safety-significant information that does not allow time for the operator (time is critical) to execute any action that would prevent or eliminate the occurrence of a mishap or hazard.
    - No functioning interlock that would prevent or eliminate the occurrence of a mishap or hazard.
- **Autonomous** (alternate definition). AI software function(s) that has total control over an operational system and/or subsystem(s) to execute a task(s) without human intervention.
- **Semi-Autonomous.** Function exercises control authority over potentially safety-significant hardware systems, subsystems, or components, allowing time for predetermined safe detection and intervention by an independent control entity to preclude the occurrence of a mishap or hazard.
    - Function that displays safety-significant information requiring the operator (with sufficient time) to execute an action (that the operator has been trained to perform) for mitigation or control over a mishap or hazard.
    - At least one functioning interlock that would prevent or eliminate the occurrence of a mishap or hazard.
- **Semi-Autonomous (Control in the loop).** AI software function(s) that has control over an operational system and/or subsystem(s) but requires an element of human or external control at some point to complete the task.
- **Semi-Autonomous (Control on the loop).** AI software function(s) that has control over an operational system and/or subsystem(s) to complete a task but is monitored by a human or external control to interrupt if deemed necessary.
- **Redundant Fault Tolerance.** Function that issues commands over safety-significant hardware systems, subsystems, or components but requires a safety control entity to complete the command function. The system must provide the safety control entity sufficient notification of a failure or potential unsafe state.

The system must also include one or more interlocks that would preclude the occurrence of a mishap or hazard.

- Function that generates information or display of a safety-significant nature used by a safety control entity to make safety-significant decisions. The system includes two or more interlocks that would preclude the occurrence of a mishap or hazard.
  - In the case of function failure, the system includes two or more independent interlocks that preclude the occurrence of a mishap or hazard.
- Influential. Function generates information of a safety-related nature used to make decisions by the operator but does not require operator action to avoid a mishap.
  - In the case of function failure, the system includes three or more independent interlocks that preclude the occurrence of a mishap or hazard.
- No Safety Impact. Function does not possess command or control authority over safety-significant hardware systems, subsystems, or components and does not provide safety-significant information. Function does not provide safety-significant data or information that requires control entity interaction. Function does not transport or resolve communication of safety-significant data.

Attribute/Features. 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 certain region of the image, which has certain properties as opposed to another popular definition of a feature as 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.

Blockchain. Blockchain is a type of distributed e-ledger (like an accountant would use to keep track of financial transactions). It is designed to be a tamper-resistant decentralized documented approach that provides proof of transaction involving physical or intellectual assets. In this case, T&E training data for the LOR, it ensures confidence that only people with permission to use the training data, from origin to T&E separated test set, are granted access. By using a blockchain approach, policy enforcement can also ensure that rules for accessing the data are followed. A blockchain architecture can document the who, what, where, and when transactions involve: the creation of the training data composition, data attribute transfer to location for T&E random selection, managing ownership of a T&E test set and the integrity of the data.

Class. Also referred to as target, label, or categories. A class is what a categorization algorithm labels an input. For instance, when an image of a cat or dog is an input to a

categorization algorithm, i.e., trained to recognize a cat or dog, then the algorithm has two classes, a cat and a dog. These classes are in the form of labeling the input being classified as a cat or a dog.

Classification. A technique used in AI and ML, in which the application determines what group or categories an output or observation belongs in, using a basis training set data containing output or observations whose group category is already known.

Clustering. Clustering algorithms let machines group data points or items into groups with similar characteristics.

Convolutional Neural Network (CNN). CNNs are a variety of artificial neural networks often used in image processing.

Data Analytics. Creating derivative data for greater understanding or summarization of the source data

Data Curation. Data curation is the organization and integration of data collected from various sources. This 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 ML goal, where the organization is based on the classification or regression needs of the algorithm.

Data Drift. 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.

Data Set. Data set is a collection of instances/samples where each is comprised of attributes/features supporting the development of the algorithm (also known as a Training Data Set).

Distributed AI. A class of technologies that solve problems by distributing them to independent AIs that interact with each other. Swarm Intelligence is an example of this subset, where collective behaviors emerge from the interaction of decentralized self-organized AIs.

Expert Systems. A form of AI that utilizes preprogramed rules to optimize solutions.

Explainability. Explainability (also called explicability) is a concept in which the mechanism by which an AI system or application determined a specific outcome or recommendation that can be explained.

Forward Propagation. As the name suggests, the input data is fed in the forward direction through a neural network. Each hidden layer accepts the input data, processes it

as per the activation function and passes to the successive layer. This only applies to neural networks.

Gradient Descent. An optimization algorithm used to minimize the cost function, which trains the ML models by minimizing errors between predicted and actual results. Training data helps these models learn over time, and the cost function within gradient descent specifically acts as a barometer, gauging its accuracy with each iteration of parameter updates. Until the function is close to or equal to zero, the model will continue to adjust its parameters to yield the smallest possible error. Once ML models are optimized for accuracy, they can be powerful tools for AI and computer science applications.

Greedy Search. An optimization algorithm that always takes the best immediate, or local, solution while finding an answer. Greedy algorithms find the overall, or globally, optimal solution for some optimization problems, but may find less-than-optimal solutions for some instances of other problems.

Guardrails/Gates. Boundaries that prevent an AI Function from performing an action that will result in an unwanted event.

Inference. The process by which a ML model makes predictions about stimuli. The term inference refers specifically to its role in deployment, when the model is making predictions about novel stimuli.

Instance. A single sample of the training data used for training the algorithm. An instance, as part of the training set of instances, should contain attributes needed by the algorithm to perform its given/required function at the desired success rate.

Labeled Data. A type of data used in AI applications that pairs input data with designed correct output data for each element or example.

Learning. For a given topic, an ability to comprehend so that appropriate action can be taken within an environment of (1) "known-knowns" (facts), (2) "known-unknowns" (assumptions), (3) "unknown-knowns" (absent data), and (4) "unknown-unknowns" (surprises).

For ML,

- "Known-knowns" (facts) are training data, feedback, or knowledge regarding a subject.
- "Known-unknowns" (assumptions) are what we are assuming about how those facts might vary, i.e., techniques to handle variations that are outside training set.
- "Unknown-knowns" (absent data) related to things we expected, but unfortunately do not have, i.e., techniques to handle sparse and missing data.

- "Unknown-unknowns" (surprises) refer to things we never expected, but must still somehow comprehend and take appropriate action.

Level of Rigor (LOR). LOR is the set of analysis and assurance processes taken together to achieve required confidence in a functions performance based on the criticality of the function in a system context and use to determine risk contribution.

Machine Learning (ML). ML is defined as an enabling technology of AI that provides systems, using algorithms, data, tools, and techniques; the ability to learn and change without providing/programming an explicit mathematic model for mapping input to output (Figure 17). (Also, see *Deep Learning*.)

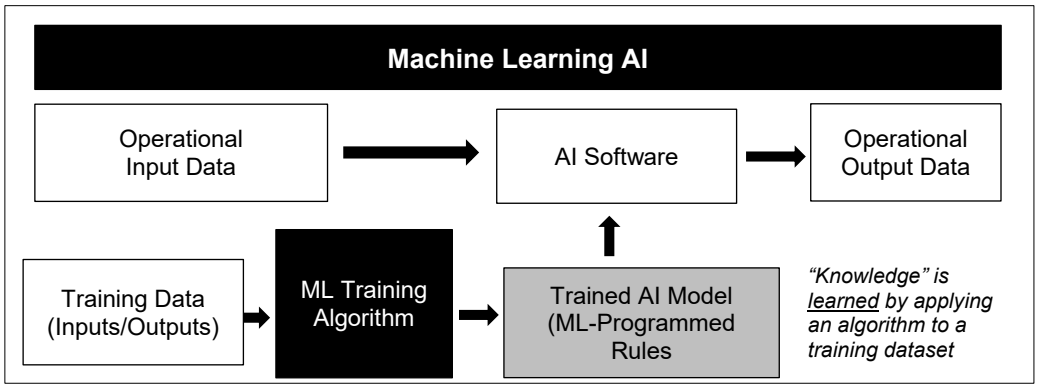

FIGURE 17. ML AI.

Machine Vision. A computer technology that can process, understand, detect, and identify patterns and objects in the content of images and video using AI models. Add also sometimes known as computer vision.

Meta-Model. A meta-model is a model of a model. Where meta-modeling is the analysis, construction, and development of the frames, rules constraints, models, and theories applicable, and useful for modeling a predefined class of problems. In our case, simulation frameworks are created to be captured as input variables/attributes that create statistical results of interest. We create a set of meta-models, where there is a fixed set of variables. The meta-model set represents the same variable but with different numerical input variations. And, the numerical input variations cause different statistical results. A meta-model consists of variables having specific number values that are input to a simulation, resulting in outcomes that are statistically represented in the meta-model. The meta-model has as input variables, their numeric numbers and the resulting statistic (represented at the output to the meta-model). A meta-model set has a collection of meta-models, each having the same variables but with different numerical values, and potentially different statistical outcomes. For example, time to complete a lap around an obstacle course would be the statistic of interest. Changes to the input variables might be numerical

input variations associated with terrain compared to design drag, acceleration, and turning radius of the car being simulated. Terrain, design drag, acceleration, and turning radius are variables/attributes that remain the same, while their values change and each value is captured, along with its statistical result creating as an element within the meta-model set.

Min-Max. Min-Max is a decision rule used in AI, decision theory, game theory, statistics, and philosophy for minimizing the possible loss for a worst-case (maximum loss) scenario (Reference 16).

Missing Data. This refers to the data input to the AI Model. 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, such as a radar system. For example, a sensor states a ship is moving at 1,000 knots and therefore has been considered erroneous data. In this case, velocity is considered missing, reported as an empty field in the input stream. Missing data feature comes from some form of data collection failure leaving a blank input response.

Model Drift. 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.

Operational Complexity. A situation described by a series of events, caused by actions between interacting entities, 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). Battle Complexity is a common example, consider an engagement of opposing forces that have access to and use of various forms of environmental challenges, electronic warfare, a large array of missile capabilities, advanced sensor capabilities, manned and unmanned platforms working together, and interoperable, short- and long-range communication, linking organic and inorganic, multidomain assets associated with joint and coalition forces with the purpose of achieving common mission goals.

Overfit/Overfitting. Overfit states that the model has a very low tolerance for input data that is not close enough to the original training data input. Mathematically, the variance between training samples and the predictor is too low. 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. The success rate is very high, but only for training data. An overly complex model can cause overfit.

Regression. Regression is a technique used in AI and ML, in which the algorithm continuously estimates the relationship between outputs through the use of numeric values.

Robust ML. Robust ML typically refers to the robustness of ML algorithms. For a ML algorithm to be considered robust, either the testing error has to be consistent with the training error, or the performance is stable after adding some noise to the dataset.

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 does not 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 is no target to reflect back. Most times sparse data will be represented by a zero where as a blank field represents missing data (Reference 17).

Synthetic Data. Synthetic data is artificially created rather than generated by actual events. It is often created with the help of algorithms and is used for a wide range of activities, including test data, model validation, and AI model training (Reference 18).

Traditionally Developed Code (Handcrafted Knowledge AI). Users coding the algorithm directly into the function, being the translator between the data and the code structure (Figure 18). In contrast, the AI algorithm does the translation between data and code, where the user creates constructs affecting the translation.

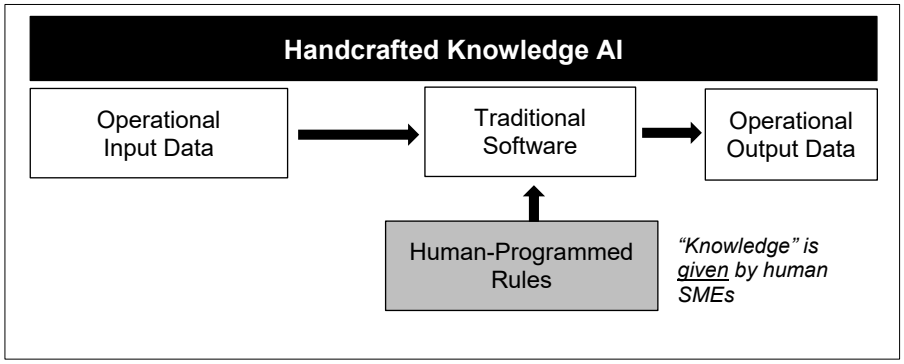

FIGURE 18. Handcrafted Knowledge AI.

Training. The process by which the parameters of a ML model are adjusted such that the model can better perform a task. The model is trained using a data set, which provides example stimuli (e.g., images, a text corpus, audio signals) that the model can learn from through updates to weights and parameters associated with the model (Reference 19).

Training Data Modality. When creating training data, it is important to understand the operational environment being represented in order to ensure adequate development of the 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.

- **ML Training Data Modality 1.** This modality 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.
- **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. 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.
- **ML Training Data Modality 3.** Training data sets that are based on an operational environment from a combination of multiple data sources where each source contains one or more attributes form 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.

Transfer Learning. Specific to DNNs, the process of using an AI Model, adding one or two additional classification layers to create and AI Algorithm, and then retrain the algorithm with a new set of training data to perform a similar a function, becoming a new AI Model supporting a new functional purpose.

Underfit/Underfitting. An indication that there is a quantity issue with the training data used. This occurs when there is an insufficient amount of training data, which causes missed categorizing of the correct class/approximation. A low complex model can cause underfit. Mathematically, the variance between training samples and the predictor is too high.

Variance Measurements. Variance measurements determine how the ML algorithm function performs. An example of regression variance is the measurement of least mean squared error or in the case of logistic regression, it optimizes the sum of the likelihoods because it is based on maximum likelihood to find the best fitting line based on the training data. For categorization, the variance is measured at the difference between the error rate and the bias. (Note: Error rate is the probability that the algorithm will misclassify for at given input.)

REFERENCES

1. Bruce Nagy, Gunendran Sivapragasam, Loren Edwards. "Functional Hazard Analysis and Subsystem Hazard Analysis of Artificial Intelligence/Machine Learning Functions Within a Sandbox Program," *Proceedings of the Eighteenth Annual Acquisition Research Symposium*, Naval Postgraduate School, Monterey, California, 2021.

2. Scot Miller and Bruce Nagy. "Interdependence Analysis for Artificial Intelligence System Safety," *Proceedings of the Eighteenth Annual Acquisition Research Symposium*, Naval Postgraduate School, Monterey, California, 2021.

3. Bruce Nagy. "Tips for Applying Artificial Intelligence to Battle Complexity," *Naval Applications for Machine Learning Symposium*, presentation slides, 27 October 2021.

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

5. Bruce Nagy. "Two gaps that need to be filled in order to trust AI in complex battle scenarios," *Proceedings of the Nineteenth Annual Acquisition Research Symposium*, Naval Postgraduate School, Monterey, California, 2022.

6. Bruce Nagy. "Tips for CDRLs/requirements when acquiring/developing AI-enabled systems," *Proceedings of the Nineteenth Annual Acquisition Research Symposium*, Naval Postgraduate School, Monterey, California, 2022.

7. Bruce Nagy. "Applying Generative Adversarial Network constructs to Mission--based Simulations to produce "Realistic" Synthetic Training Data for Machine Learning Algorithms," *GANs Overview for NAML 2021 NABN581*, Naval Information Warfare Center Pacific, San Diego, California, 2021.

8. Bruce Nagy. "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, Monterey, California, 2021.

9. Anthony Kendal, Arijit Das, Bruce Nagy, and Avantika Ghosh. "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, Monterey, California, 2021.

10. Bruce Nagy. "Using game theory and machine learning to provide white cell automation support in the adjudication of war game," *SPIE Conference on Defense and Commercial Sensing*, Orlando, Florida, 3–7 April 2022.

11. GeeksforGeeks. “ML Underfitting and Overfitting,” accessed March 2022. https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

12. Lasse Schultebraucks. “Underfit-Overfit,” accessed March 2022. https://lasseschultebraucks.com/overfitting-underfitting-ml/ chunml.github.io/ChunML.github.io/tutorial/Underfit-Overfit

13. IBM. “AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What’s the Difference?” by Eda Kavlakoglu, 27 May 2020, accessed March 2022. https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

14. Wikipedia. “AdaBoost,” last edited 17 March 2022, accessed March 2022. https://en.wikipedia.org/wiki/AdaBoost

15. Wikipedia. “Naïve Bayes classifier,” last edited 4 March 2022, accessed March 2022. https://en.wikipedia.org/wiki/Naive_Bayes_classifier

16. Wikipedia. “Minimax,” last edited 12 March 2022, accessed March 2022. https://en.wikipedia.org/wiki/Minimax

17. Oscar Warnling and Johan Bissmark. “The Sparse Data Problem Within Classification Algorithms – The Effect of Sparse data on the Naïve Bayes Algorithm,” Bachelor’s Thesis in Computer Science at CSC, 5 June 2017. https://Kth.diva-portal.org/smash/get/diva2:1111045/FULLTEXT01.pdf

18. Elise Devaux. “Types of synthetic data and 5 real-life examples,” 15 January 2021, accessed March 2022. https://www.statice.ai/post/types-synthetic-data-examples-real-life-examples

19. Consumer Technology Association. *ANSI/CTA Standard – Definitions and Characteristics of Artificial Intelligence (ANSI/CTA-2089)*, Arlington, Virginia, February 2020.

BIBLIOGRAPHY

Black, Paul E. "greedy algorithm," in *Dictionary of Algorithms and Data Structures*. National Institute of Standards and Technology, 2 February 2005, accessed March 2022. https://xlinux.nist.gov/dads/HTML/greedyalgo.html

Brownlee, Jason. "A Gentle Introduction to XGBoost for Applied Machine Learning," Machine Learning Mastery, last updated 17 February 2021, accessed March 2022. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Brownlee, Jason. "How to Fix the Vanishing Gradients Problem Using the ReLU," Machine Learning Mastery, last updated 25 August 2020, accessed March 2022. https://machinelearningmastery.com/how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function/

IBM Cloud Education. "Gradient Descent," 27 October 2020, accessed March 2022. https://www.ibm.com/cloud/learn/gradient-descent

Java T Point. "K-Nearest Neighbor (KNN) Algorithm for Machine Learning," accessed March 2022. https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

Kaur, Jagreet. "Artificial Intelligence Overview and Applications," Xenonstack, 19 January 2020, accessed March 2022. https://www.xenonstack.com/blog/artificial-intelligence

Luhaniwal, Vikashraj. "Forward propagation in neural networks – Simplified math and code version," *Towards Data Science*, 6 May 2019, accessed March 2022. https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250

Marr, Bernard. "What is deep reinforcement learning?" Bernard Marr & Co., 2021, accessed March 2022. https://bernardmarr.com/what-is-deep-reinforcement-learning/

Pant, Ayush. "Introduction to Logistic Regression," *Towards Data Science*, 22 January 2019, accessed March 2022. https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

Salem, Mike. "What is an 'Autonomous System'?" Udacity, 24 September 2018, accessed March 2022. https://www.udacity.com/blog/2018/09/what-is-an-autonomous-system.html

Unnamed. “GOFAI, or Good Old Fashioned Artificial Intelligence,” 12 June 2018. https://blog.ivyglobal.com/index.php/2018/06/12/gofai-or-good-old-fashioned-artificial-intelligence/

Wikipedia. “Backpropagation,” last edited 11 March 2022, accessed March 2022. https://en.wikipedia.org/wiki/Backpropagation

Wikipedia. “Model-free (reinforcement learning),” last edited 28 February 2021, accessed March 2022. https://en.wikipedia.org/wiki/Model-free_(reinforcement_learning)

Wikipedia. “Q-learning,” last edited 2 March 2022, accessed March 2022. https://en.wikipedia.org/wiki/Q-learning

Wikipedia. “Random forest,” last edited 1 March 2022, accessed March 2022. https://en.wikipedia.org/wiki/Random_forest

Wikipedia. “Recurrent neural network,” last edited 17 April 2022, accessed March 2022. https://en.wikipedia.org/wiki/Recurrent_neural_network

ACRONYMS

3D	Three-Dimensional
AI	Artificial Intelligence
AIDP	AI Development Plan
API	Application Programming Interface
AUC	Area Under the Curve
CDRL	Contract Deliverable Item List
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DoD	Department of Defense
DoDAF	DoD Architecture Framework
DOE	Design of Experiments
DP	Deliver Package
EUF	Expected Utility Function
FN	False Negative
FP	False Positive
GAN	Generative Adversarial Network
GPS	Global Positioning System
IA	Interdependency Analysis
ID	Identification
IR	Infrared
LOR	Level of Rigor
LT	Load Truck
MAP	Maximum A Posteriori
ML	Machine Learning
MR	Move Robot
MSD	Missing and Sparse Data
MSG	Message
MT	Move Truck
NLG	Natural Language Generation
NLP	Natural Language Processing
NOSSA	Naval Ordnance Safety and Security Activity

OV	Operational View
PR	Precision Recall
RL	Reinforcement Learning
ROC	Receiver Operator Characteristic
SARSA	State-Action-Reward-State-Action
SME	Subject Matter Expert
SQL	Structured Query Language
StAR-n	Sources to Attribute Ratios for 1, 2, or 3 (nth)
SV	Systems View
T&E	Test and Evaluation
TN	True Negative
TP	True Positive
TSAT	Training Set Alignment Test
UML	Unified Modeling Language
UT	Unload Truck
WIFM	What's In It For Me

INITIAL DISTRIBUTION

1 Defense Technical Information Center, Fort Belvoir, VA

ON-SITE DISTRIBUTION

1 Code DC12100 (file copy)
2 Code D5J4000 (archive copies)
2 Code D510000, Chirkis, K.
2 Code D571000, Nagy, B.