Received 17 June 2024, accepted 9 July 2024, date of publication 15 July 2024, date of current version 25 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3427668

RESEARCH ARTICLE

Development Life Cycle for Using a DNN to Complete the Autonomous Aerial Refueling Task

DANIELLE CLEMENT[1], SARAH MOTTINO[1], AND DONALD H. COSTELLO III[2]
[1]Aeronautics Division at the Lockheed Martin Corporation, Fort Worth, TX 76108, USA
[2]Weapons, Robotics, and Control Engineering Department, United States Naval Academy, Annapolis, MD 21402, USA

Corresponding author: Donald H. Costello III (dcostell@usna.edu)

This work was supported by the Office of Naval Research.

ABSTRACT The future of aviation is unmanned and ultimately autonomous. As part of this effort the Office of Naval Research, in partnership with the Naval Air Systems Command, has initiated the advanced autonomous air-to-air refueling system (A4RS) future naval capability (FNC). The A4RS FNC intends to set the interface requirements for any uncrewed aerial system to receive fuel from a United States Navy (USN) aircraft. Additionally, the A4RS FNC will be the first time that a system will be authorized to complete autonomous behavior without a human in or on the loop. However, a method to certify this behavior safe for flight does not currently exist. This paper details the method that has been proposed as part of the FNC to the naval flight certification authorities for approving a deep neural network to complete the aerial refueling task.

INDEX TERMS Autonomy, autonomy certification, DNN.

I. INTRODUCTION

United States naval aviation is becoming increasingly uncrewed. It is anticipated that the number of uncrewed aircraft systems (UAS) in a carrier airwing will dramatically increase over the next decade [1]. Leadership will require that carrier-based UASs have the ability to aerial refuel. In an effort to enable autonomous aerial refueling by a UAS, the Office of Naval Research (ONR) has sponsored the FY24 new start advanced autonomous air-to-air refueling system (A4RS) future naval capability (FNC). Part of the FNC deals with the certification of a system to perform tasks when a human is not in or on the loop. This paper was generated in response to tasking from the ONR and defines an approach to develop, assure, and generate representative evidence in support of airworthiness for a DNN to perform object detection in an aerial refueling scenario.

At the IEEE 2023 International Automated Vehicle Validation Conference (IAVVC), the authors presented a high-level discussion on certifying a deep neural network (DNN) to build situational awareness for an autonomous vehicle to complete the aerial refueling task through object detection verification and validation (V&V) [2]. This work expands on the IAVVC work. In particular, it details the developmental life cycle that has been proposed to certification officials. This process has been briefed to the National Airworthiness Council Artificial Intelligence Working Group and a group within the Naval Air Systems Command (NAVAIR) that has been tasked to identify a method for certifying a DNN to complete the Aerial Refueling task.

Naval aviation has a history of finding ways to innovate and expand the capabilities of its deployed air wing. One example of this is aerial refueling, which is considered a high-gain task requiring a skilled pilot to complete. Aerial refueling allows a pilot to extend the endurance and range of their aircraft. Before a naval aviator (pilot) can be ready to deploy with their squadron they must first qualify in aerial refueling. The aerial refueling certification process for an aircraft is designed to ensure that a pilot can safely maneuver their aircraft providing they maintain multiple parameters (i.e., closure rate, off-center engagement, lateral drift at contact. During the last 10 feet of an intercept, the naval aviator of

The associate editor coordinating the review of this manuscript and approving it for publication was Seifedine Kadry.

the receiving aircraft uses their experience and situational awareness (SA) to complete the task. They constantly update their interpretation of the situation by assessing closure rates and updating the predicted contact point during the refueling process. The refueling process is described in more detail in Section II-B.

The future of naval aviation is unmanned and ultimately autonomous. Before autonomous aircraft are fielded, a path needs to be defined to certify the SA their sensors build to ensure they are adequate for making sound aeronautical decisions [3]. To complete the autonomous aerial refueling task an autonomous system will require some form of computer vision to assess what is happening around the vehicle as it attempts to make contact. The system will use the sensors to build SA during the autonomous aerial refueling task.

Currently, there are multiple efforts underway to field various levels of machine learning algorithms in United States naval aviation systems. In particular, multiple papers have been published documenting the process of allowing a machine learning algorithm, a deep neural network (DNN), to make safety-critical decisions without a human in or on the loop in the autonomous aerial refueling task [4], [5], [6]. While this work appears to be giving promising results, a standard or method of compliance does not exist for airworthiness officials to use for assuring the decision made by the DNN is a sound risk decision. This includes questions such as: how is assurance provided that the system is accurately detecting the aerial refueling objects? How is assurance provided that the bearing and range the system will pass to the flight controller are accurate enough to enable the system to make contact and safely refuel?

One of the main obstacles in developing an assurance plan for a DNN to perform without a human in or on the loop is the lack of understanding of how a DNN works, is trained, or verified by the airworthiness community. While neural networks have been the best-of-breed solution in the object detection domain for the past 20 years and are used in several applications from self-driving cars [7], [8], [9], [10] to inspection robots [11], [12], established processes and techniques for assuring software and systems development do not translate well to these types of systems.

A critical difference between developing software and developing a neural network model is that when you develop a model, the functional software requirements are not implemented in the code – they reside in the curated datasets used to train the model and the associated weights. From an assurance perspective, the source code that runs on the platform only executes the matrix math to repeatedly apply the activation functions to the weighted inputs. This means that standard software processes assurance steps such as code reviews, code coverage, and unit tests are assuring the mechanics of inference (are the weights loaded correctly, is the math implemented correctly), and not the component functionality (are the right objects detected in the right

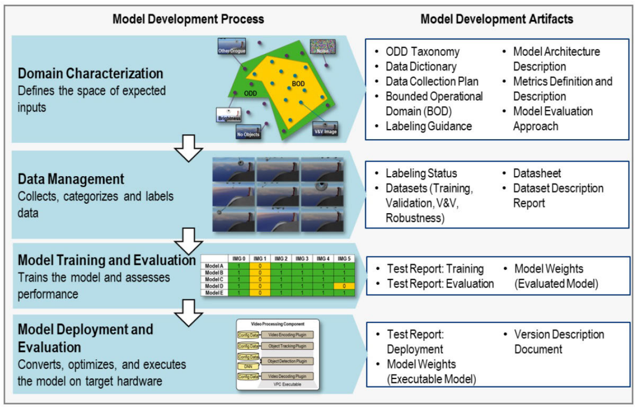

Model Development Process
Model Development Artifacts
Domain Characterization
Defines the space of expected inputs
Other Drogue
Noise
ODD
BOD
Brightness
No Objects
V&V Image
ODD Taxonomy
Data Dictionary
Data Collection Plan
Bounded Operational Domain (BOD)
Labeling Guidance
Model Architecture Description
Metrics Definition and Description
Model Evaluation Approach
Data Management
Collects, categorizes and labels data
Labeling Status
Datasets (Training, Validation, V&V, Robustness)
Datasheet
Dataset Description Report
Model Training and Evaluation
Trains the model and assesses performance
IMG 0
IMG 1
IMG 2
IMG 3
IMG 4
IMG 5
Model A
Model B
Model C
Model D
Model E
Test Report: Training
Test Report: Evaluation
Model Weights (Evaluated Model)
Model Deployment and Evaluation
Converts, optimizes, and executes the model on target hardware
Video Processing Component
Config Data
Video Encoding Plugin
Object Tracking Plugin
Object Detection Plugin
DNN
Video Decoding Plugin
VPC Executable
Test Report: Deployment
Model Weights (Executable Model)
Version Description Document

FIGURE 1. **Model development process and development artifacts available to support airworthiness.**

places). For this reason, alternate approaches for assurance are necessary when incorporating neural networks, including DNNs into systems.

To address these differences, this approach uses a combination of traditional and non-traditional techniques to generate a series of artifacts to support airworthiness for the DNN. The Video Processing Component (VPC) – a software package that runs on the receiving aircraft and executes the DNN – is developed via a traditional software development process. The DNN – a configuration file that is provided to the VPC – is developed via the model development process shown in Figure 1. Major steps in the model development process include: domain characterization, which scopes and bounds the input domain for the aerial refueling application, data management, which focuses on the data used to train the model, model training and evaluation, which establishes the process to build the DNN from the data and assess its performance, and model deployment and evaluation, which structures the process used to transform the model into the data file that executes onboard the vehicle and the evaluation of the system in the target environment.

The major contribution of this paper is a definition of a baseline model development process to aid the airworthiness community in their generation of a risk-based decision to allow a DNN to participate in the autonomous aerial refueling task. If the process is not developed and vetted with the cooperation of the safety of flight clearance authorities prior to the acquisition process, it will be highly unlikely for the autonomous system to be successful when an attempt is made for safety of flight certification.

II. BACKGROUND

A. APPLICABLE DOCUMENTS

A major challenge in airworthiness certification for DNNs is that standards for airworthiness such as the Department of Defense Handbook Airworthiness Certification Criteria, MIL-STD-516C (Reference [13]), reference processes and best practices contained in safety standards such as

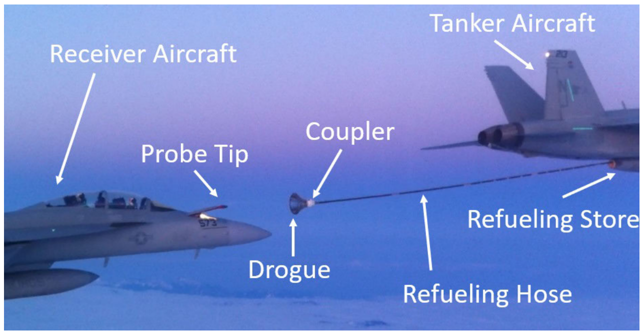


FIGURE 2. **Probe-and-drogue aerial refueling of an EA-18G from a F/A-18E with key components labeled.**

Department of Defense Standard Practice: System Safety, MIL-STD-882E (Reference [14]), and software specific standards such as the Software Considerations in Airborne Systems and Equipment Certification, DO-178C (Reference [15]), as of this writing have not been updated to address DNN systems.

While this means alternative methods for use of DNNs are required, the documents listed are still valuable to understand this plan.

B. AERIAL REFUELING USE CASE

This approach focuses on an aerial refueling use case [6], [16], [17], specifically for a probe-and-drogue system (shown in Figure 2). In probe-and-drogue refueling, the probe is mounted on the receiving aircraft and the drogue is a conical object attached via a coupler to a hose, which connects to an aerial refueling store on a tanker aircraft. During the refueling process, the receiving aircraft maneuvers the probe tip into the drogue to make a connection with the coupler and start the flow of fuel.

The drogue is free-floating in the airstream behind the tanker aircraft, making the precise position of the drogue incredibly challenging to compute with any degree of accuracy, requiring real-time information about platform speeds, hose length, turbulence, etc. This provides an excellent use case for a computer vision object detection neural network which can provide an effective means to detect and localize the drogue object visually.

Object detection neural networks are an instance of supervised learning, which uses labeled datasets to train algorithms to classify data and/or predict outcomes [18], [19], [20]. The neural network for the aerial refueling use case is provided with labeled instances (images with objects of interest identified via bounding boxes, as shown in Figure 3). Based on those labeled instances, the network learns a mapping between input images and where the objects are located. The model then produces a set of bounding boxes (that enclose an object), labels (that identify the class of the object), and confidence scores (that reflect the DNN's confidence that the object is of the identified class). Detecting these refueling objects using an object detection neural network provides a way to localize them more precisely in a dynamic environment.

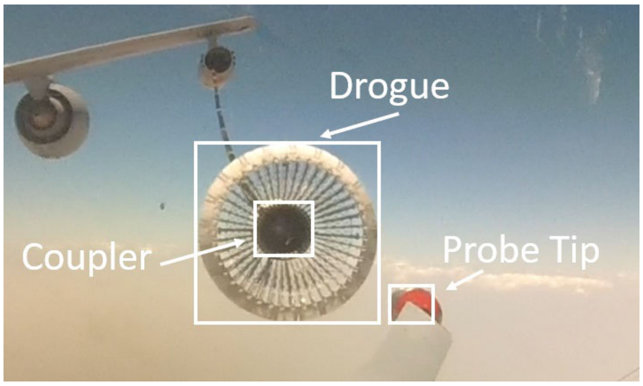


FIGURE 3. **Classes of aerial refueling objects labeled in an image of an EA-18G refueling from a KC-10.**

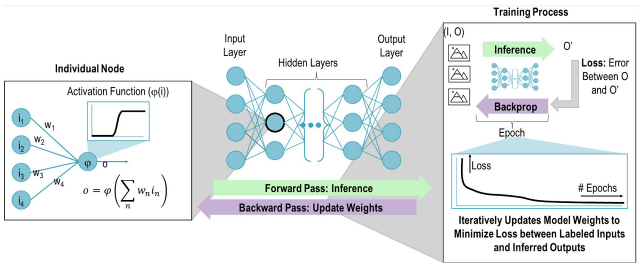

Individual Node
Activation Function (φ(i))
i_1
i_2
i_3
i_4
w_1
w_2
w_3
w_4
φ
o
$o = \varphi\left(\sum_n w_n i_n\right)$
Input Layer
Hidden Layers
Output Layer
Forward Pass: Inference
Backward Pass: Update Weights
Training Process
(I, O)
Inference
O'
Loss: Error Between O and O'
Backprop
Epoch
Loss
# Epochs
Iteratively Updates Model Weights to Minimize Loss between Labeled Inputs and Inferred Outputs

FIGURE 4. **DNN structure and training process.**

C. DNN OVERVIEW

A DNN (also called a model) is made up of layers of nodes (or neurons). Each node receives inputs from one or more other nodes in the network. Each of those inputs is weighted. The input data starts at the input layer and is then passed through a series of hidden layers until the data reaches the outer layer. At each individual node, a similar processing activity occurs: the weighted sum of the node's inputs determines if a node will "activate" or not, based on the activation function (such as ReLU or sigmoid) for that node.

1) MODEL TRAINING AND EVALUATION

The basic concepts behind training a neural network using a supervised learning process are summarized in Figure 4. To train a neural network, a set of inputs with labeled outputs are provided to the network. For this use case, the inputs are images, and the labeled outputs are the named bounding boxes around the objects of interest associated with each image. The network calculates its outputs using a process called inference, or a "forward pass" through the network. The difference between the calculated outputs and the known outputs is called the loss or the error. A measure such as Mean Squared Error can be used as a loss function. The weights in the neural network are updated using a process called backpropagation (or a "backwards pass") that attempts to minimize the loss across all the provided input samples. One training cycle of inference and backpropagation is called an epoch, and over the course of many epochs, the error between the true labels and the predicted labels is expected to approach 0. The variables used to orchestrate the training process, for example: the number of layers of the network, the learning rate (how big an adjustment is made to reduce the error in each backwards pass), and the activation functions are called hyperparameters. Hyperparameters are adjusted during the model development process to improve the quality of the model output.

2) MODEL DEPLOYMENT

For the aerial refueling use case, when the model is used operationally (deployed), it will only perform inference (the forward pass). All the model training happens prior to deployment, and the certification will be for the combination of the deployed model and the software and configuration data used to execute that model.

III. SYSTEM OVERVIEW

This document describes the planned certification approach for a computer vision object detection function embedded in a vendor-provided video processing pipeline (the Video Processing Component (VPC)) that is part of the larger Drogue Tracking System (DTS), and the process used to generate and evaluate the performance of a DNN for object detection. This approach covers the unique aspects of certification associated with the VPC. This component is responsible for the detection of key aerial refueling objects using a DNN and runs on the receiving aircraft. The plan for certification described here is intended to be applicable to any receiving aircraft platform that might integrate this component.

The VPC is deployed as a part of a larger DTS, as shown in Figure 5. The deployed DTS contains processing hardware, including a graphics processing unit (GPU) and central processing unit (CPU) resources and storage, a camera, and additional platform-specific software. The VPC communicates video and data through the platform adaptation software to other system components. The VPC is made up of several software subfunctions, notably: Video Decoding, Object Detection, Object Tracking, and Video Encoding. Configuration data and the DNN are loaded into the VPC from local storage at startup. The platform-specific adaptation software and other aircraft system components are out of scope for this document. They are included here to provide some general system context and to call out any expectations for their behavior or capabilities as they relate to the DNN.

The VPC is a software component that has two major parts: (1) the VPC executable source, which is developed by the vendor and (2) the VPC configuration data. This VPC configuration data defines the input, output and behavioral properties of each of the VPC subfunctions. A special instance of the configuration data for the Object Detection subfunction is the executable model, which is also referred to broadly as the DNN.

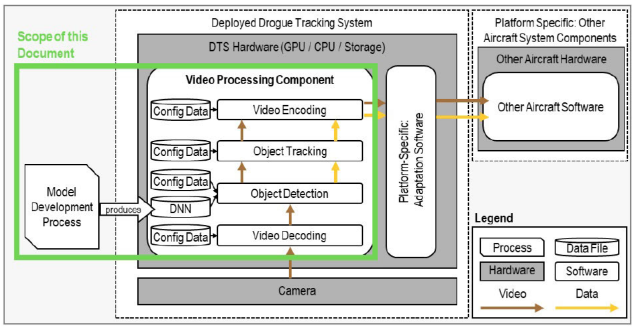


FIGURE 5. High-level deployment approach.

The development of a DNN begins with assessing datasets and selecting an appropriate model architecture for training. Once this selection is made, the model is trained on the datasets, metrics are captured, and if model performance meets the established criteria to deploy, the model is prepared for deployment. The resulting executable DNN is used as a configuration file for the video processing object detection subfunction, which executes the model on input frames extracted from video streams. Process assurance is applied to the data used to train the model.

It is important to reiterate that the DNN is configuration data provided to a video processing component: the DNN requires a software application (the VPC) in order to execute. This DNN "configuration file" differs from many traditional software configuration files, in that it is not human-readable or human-editable. The proposed deployment approach for certification is to test and certify the deployed VPC on its GPU hardware combined with static configuration data (including a "frozen" DNN).

If the DNN is retrained in the future, this would result in a different input file to the VPC, which would trigger an evaluation and potential recertification of the VPC. Recertification could be required because the object detection functionality is maintained in the DNN configuration file, not in the software. Changes to that file could change the system functionality, which would require an evaluation. Updates to the VPC are scheduled and coordinated based on the release tempo for that software package and in line with planned airworthiness evaluations. Each platform that uses the VPC will determine if and when a software update should be deployed per their defined software maintenance and update approaches.

IV. DEVELOPMENT LIFE CYCLE

A. ORGANIZATION

There are 3 groups responsible for the development of the VPC: the model development team, the model integration team, and the Data Set Control Board (DSCB). The model development team is made up of data scientists and artificial intelligence (AI)/machine learning (ML) developers focused on the accuracy and performance of the model. The model integration team is a more traditional software development team focused on configuring the VPC and integrating the developed model with the VPC. The DSCB is an oversight body responsible for assuring the data development lifecycle.

The model development team is responsible for developing the Operational Design Domain (ODD) based on the requirements and the concept of operations (ConOps), collecting and managing the data, and developing and analyzing the performance of the model. The model development team is focused on data analysis and the construction of a model that best fits the aerial refueling use case. The model development team is made up of individuals with AI, data science and computer vision backgrounds. Work instructions for job-specific activities such as data labeling will need to be provided, and detailed descriptions of the training process and tools used for model training are included in the DNN Training Plan.

The model integration team is responsible for integrating the developed model with the VPC and the other aircraft components. This team is focused on traditional software integration and test. Members of this team have a software engineering or computer science background.

The DSCB is a multifunctional review board that acts as a quality cross-check for data management activities. The DSCB is responsible for coordinating and controlling the datasets associated with model training and evaluation. The DSCB has approval authority for all aspects of data management, including ODD definition, data collection and preparation plans, data quality requirements, all datasets (training, validation, and test), the metrics definition used to evaluate the quality of the training process, and the model evaluation approach. The board's purpose is to involve key stakeholder communities, including safety and airworthiness, in the data management process.

The mapping of high-level process steps to different teams is reflected in Figure 6, and details on the process steps are provided in the following sections. All teams and the DSCB are involved in the planning process. The model integration team is responsible for the software development of the VPC, while the model development team focuses on managing the data and training the DNN. The planning and model development process phases are informed by feedback and interactions with the DSCB. All process steps address configuration management (CM), quality assurance (QA), and verification.

B. PLANNING

The planning process defines the primary structure and overall standards for the model development process. The planning phase generates the following artifacts that describe in more detail the approach for implementing and deploying the DNN model within the VPC:

- *CM Plan* describes how data, models, hyperparameters, and evaluation results will all be configured and traced to one another. It will specify the overall requirements for the Version Description Document (VDD). It will include information about how third-party software

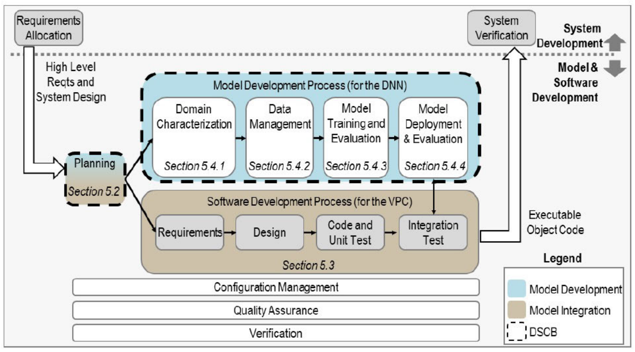


FIGURE 6. DNN development process mapped to organizations.

packages will be controlled for both the training environment and the deployment space. If a higher-level CM plan exists that supports the unique aspects of DNN development, those plans will be adopted.
- *Information Security Approach* is a critical aspect of the process. There are unique attack vectors for neural network training and deployment environments compared to traditional software. The Information Security Approach will provide an assessment of the threat vectors specific to this project and methods to mitigate those threats. It will describe how access to data, models, and results will be protected and secured.
- *Software Bill of Materials* (SBoM) will define by version the software used in (a) model training and (b) model deployment, including all dependencies. The SBoM is different from a VDD in that the SBoM defines the tooling environment for training and deployment, while the VDD includes more detailed information about specific configuration details associated with model deployment. The SBoM is expected to remain mostly static during the duration of the project, while the VDD will be updated with every development increment.
- *Data Preparation Description* documents the process of converting collected video data into images and then labeling those images with metadata and bounding boxes. This will include information about flight data preparation, lab/ground data preparation, and synthetic image generation as required.
- *Data QA Approach* will be used to review and verify the quality of labeled images (primarily: the accuracy of the bounding boxes). This includes the identification of any support tooling or requirements for the Data QA process. For example, it may be preferred to use a unique person or process to conduct a final review of the data, to increase the likelihood of identifying errors. Expectations such as those will be defined in the Data QA Approach.
- *DNN Training Plan* describes in detail the planned process to train the model, including relevant training pipelines and toolchains, training environment specifications, criteria for configuring the training process, and the approach for mechanizing CM, experiment tracking,

and training repeatability. If tools are used for model optimizations (such as model pruning) those tools are identified and the process to maintain those tools is established.

- *Deployment Approach* specifies the deployment hardware environment, software stack for execution, any containerization or partitioning approaches, and configuration specifics of third-party tooling, including the VPC configuration. This document also includes details about the software load process for VPC software and associated configuration data.

To support CM, the CM Plan defines the approach for storing and maintaining DNN-unique artifacts. The planning artifacts described above will be baselined and configuration-controlled.

To maintain QA, the artifacts developed in this phase are reviewed by the team for completeness and alignment with the overall effort's goals.

Verification. The plans are reviewed for alignment with best practices of DNN development.

C. SOFTWARE DEVELOPMENT PROCESS (VPC)

The VPC is the software application that will execute the DNN onboard the receiving aircraft. The VPC software is a executable with plugins that implement each of the VPC subfunctions. Plugin source code can be extended or modified if needed. Each plugin is independently parameterized via human-readable configuration data, and the executable model created by the model development process is a binary configuration file for the VPC Object Detection plugin.

The structure of the non-DNN configuration files is defined by a Interface Control Description (ICD) and the corresponding parameter settings are populated by the system integrator. These configuration files contain parameters that shape the behavior of the VPC plugins, for example, information about the format of the input video stream, the rate at which video frames are extracted for object detection, and the number of frames missing the object before the track is dropped. The process used to populate these files will include requirements generation and testing of the VPC component, configuration settings, and plugins to achieve the desired system video processing performance. As part of the software integration process, a set of functional VPC requirements, a Requirements Verification Cross-Reference Matrix (RVCM) and a Test Plan are developed for the VPC software.

The products developed in this step, including the VPC executable, the plugin source code, and the configuration data, are maintained per the CM Plan.

The requirements, RVCM, and Test Plan generated in this step are reviewed by the team for consistency with the System Requirements.

The requirements developed for the VPC are inspected for: alignment with system-level requirements, accuracy and consistency, and compatibility with the target hardware. The RVCM and Test Plan are inspected for traceability to the requirements, accuracy, and consistency.

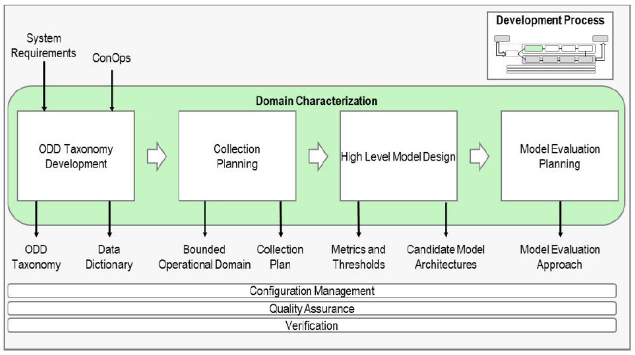


FIGURE 7. Domain characterization process.

D. MODEL DEVELOPMENT PROCESS (DNN)

The following four subsections describe the model development process in more detail and provide insight into how different artifacts are created to support learning process assurance and airworthiness.

1) DOMAIN CHARACTERIZATION

The domain characterization step establishes the approach for data management, model development and evaluation in the context of the system requirements (Figure 7). The first step in domain characterization is to define the ODD through an ODD Taxonomy. The system requirements and ConOps are used to develop the ODD Taxonomy for the system as well as a Data Dictionary that defines the different cases for ODD parameters. These documents are used to plan data collection activities and create the Bounded Operational Domain (BOD) which is the subset of that ODD that focuses on current capabilities of interest. Based on that work, a high-level model design is completed. This includes establishing key metrics for evaluating model performance as well as identifying potential model architectures that are well-suited to the current problem. As the final step in domain characterization, the approach to evaluate the model is defined.

The ODD Taxonomy (example shown in Figure 8) defines the overall relevant features of the ODD through relative categories of interest for the dataset. The ODD is the space of potential operationally relevant inputs. Defining the ODD helps to bound and scope the input set, as well as provide design documentation of the dataset assumptions built into the model. The ODD has its roots in the Automotive Safety community who realized that defining requirements for a perception system involved a complex set of state variables and a relatively unlimited set of input states. The National Highway Traffic Safety Administration published a report [21] using the ODD to bound and define test cases for automated driving.

During the development of the ODD Taxonomy, the Data Dictionary is defined to establish metadata conventions

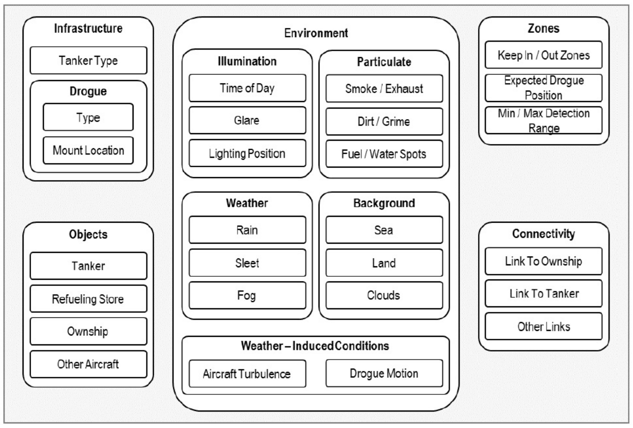


FIGURE 8. An example ODD taxonomy for aerial refueling.

and guidance for applying metadata to the frames as they are processed. For example, the "Dirt/Grime" category in the ODD Taxonomy can be one of five categories: None, Minimal, Minor, Major, or Extreme. The Data Dictionary provides definitions for each of those categories and visual examples of images in those categories, to help the humans that are assigning the images to parts of the ODD. Other metadata about the datasets is also defined in the Data Dictionary, for example, file naming conventions and the structure used to store collection dates and times. Establishing this consistency early in the model development process helps with the analysis of the datasets in future model development process phases.

The BOD and the Collection Plan are developed based on the system requirements and the ODD Taxonomy. A graphical example of the ODD and its mapping to the BOD is shown in Figure 9. While the ODD describes the complete potential operating environment for the DNN, as the DNN is developed, it may only cover a subset of the ODD – for example, it may only train on data about a small selection of the potential tanker types or only a few environmental conditions – this subset is called the BOD. The BOD reflects the part of the ODD that will be used for training and evaluated for performance in a relevant operational environment for a given DNN. While the ODD Taxonomy is expected to be static across all iterations of the DNN, the capabilities of the DNN will expand over time through additional iterative training cycles, and the BOD is the specification of the to-be-implemented capabilities for a given iteration. For example, an early version of the BOD might use only daytime images for training, which would result in (most likely) better performance of the DNN on daytime images than for nighttime images. In the next iteration of DNN development, nighttime capability could be added to the BOD, impacting that iteration's Collection Plan so that nighttime images would be collected and the DNN could be updated.

The Collection Plan uses the ODD Taxonomy and the BOD to define the approach to collect data to train and verify the computer vision component. It includes information about

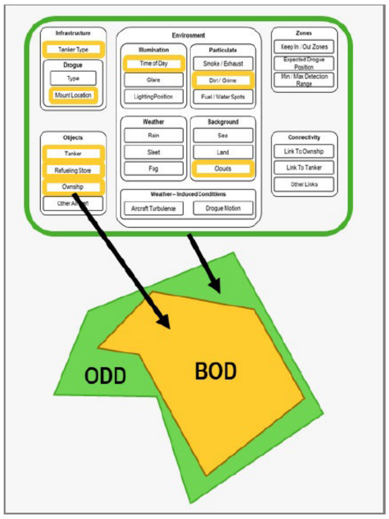


FIGURE 9. Example of operational design domain relationship to bounded operational domain.

how much data in each category should be collected, the types and properties of the data, and information about collection environments.

The system requirements have an impact on the potential model architectures used in training. For example, some model architectures are better at finding small objects than others, meaning they would be more performant when identifying objects at far distances. The time it takes to execute a model can also vary by architecture type. Once the system requirements have been established, they are used to identify which object detection architectures might be best suited to the problem at hand. These candidate model architectures and the design rationales in selecting them are defined in the Model Architecture Description. In addition to identifying potential model architectures, the model design step establishes which metrics will be used to assess model performance and what the thresholds are for those metrics. A model is evaluated over a suite of images, resulting in statistical performance outcomes, as opposed to a binary pass/fail. The thresholds that are established in this step define what ranges constitute a "pass" across the V&V test suite of images. These metrics and thresholds are defined in the Metrics Definition and Description.

Model evaluation planning establishes the approach to evaluate model performance through the definition of test cases (image and video test cases). This includes test cases that represent what the model should expect to see in the conditions that it was trained for, as well as robustness cases to evaluate model performance in conditions it may rarely (or never) encounter. These unlikely inputs are included in testing to characterize the boundaries and limits of the model's performance.

Figure 10 extends Figure 9's representation of the ODD and the BOD by including a depiction of these test cases. Test cases selected from within the BOD (yellow shape in Figure 10) make up the primary V&V set. These V&V

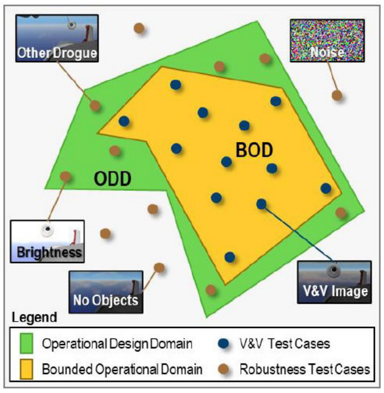


FIGURE 10. Operational design domain alignment to test cases.

test cases come from a similar dataset as the training data but are not used in model training. Figure 10 uses synthetic images as an example, however, in the DNN development process all V&V images will be from real-world environments.

Additional robustness test cases are selected from within the ODD but outside the BOD (green shape in Figure 10) – these may be test cases against untrained-on drogue types, for example. These robustness cases serve to characterize the potential operating capabilities of the DNN outside the BOD boundaries. For example, the DNN may be able to detect drogue types it has not been trained upon but may do so with less localization accuracy than the trained-on objects, potentially impacting the ranging function implemented by other system components. Images that map to identified sensor failure modes or environmental degradations, for example, overly bright or dark images, images with dead pixels, or pixelated images, are also included as robustness cases. Understanding how the DNN responds to these cases is important to characterize what the impacts might be if users choose to employ the system in "out-of-BOD" environments.

Finally, an additional set of robustness cases is selected from outside the ODD (outside both the green and yellow shapes in Figure 10). These could include images with no drogue or tanker in the field of view, to ensure that the DNN does not detect objects where there are no objects to be found. Purely solid images and pure noise images are also used as a test baseline to ensure that objects are not found where they do not exist. Some lab tests may also fall into this category of robustness testing, as the environmental characteristics of the lab environment (artificial lighting, for example) may not map directly to the ODD. Where test cases can be mapped to the ODD, traceability is maintained. In robustness cases where no mapping can be made (the pure noise case, for example), those test cases are categorized for ease of search and retrieval and to aid in identifying performance trends. The overall model evaluation strategy and approach, including the definition of planned robustness cases, is described in the Model Evaluation Approach.

Developed documentation is maintained in CM.

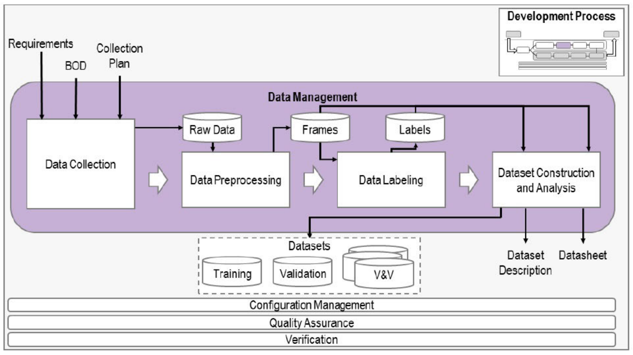


FIGURE 11. Data management process.

QA is provided by the DSCB. The DSCB reviews the products generated in this step for consistency with the ConOps and System Requirements.

The products developed in this phase are reviewed for completeness and consistency. A review event is held to gather stakeholder feedback and concurrence on the ODD and the relevant sections of the Data Dictionary. The BOD is reviewed and approved by the DSCB.

2) DATA MANAGEMENT

The Data Management Process (Figure 11) begins with data collection. Informed by the ODD, the requirements, and the collection plan, data collection is the gathering of data from synthetic, lab, and real-world environments. This process is iterative, for example, many frames of synthetic data might be generated and processed before real-world data is available. As raw data is collected, it is stored and tagged for future reference.

The Data Preprocessing activity is the process of converting the raw collected data into images that can be used for training. In addition to extracting frames from raw video data, data preprocessing includes tracing each image to the ODD Taxonomy and assigning metadata as described in the Data Dictionary. The preprocessing steps are outlined in the Data Preparation Description.

Data labeling is the process of identifying objects in the prepared images. This is typically done by hand using a labeling tool, however, there are auto-labeling tools available that can support the humans doing the labeling task. Direction for labelers is provided in the Labeling Guidance, which includes work instructions specific to the labeling tool as well as overall labeling design decisions (for example, to only label objects that are at least 60% visible). If auto-labeling is used, specific instructions for the auto-labeling software are defined. This includes a description of how the auto-labeler operates, what its limitations are, and alignment with human labeling processes.

Once the images have been labeled, datasets are constructed based on these images. These datasets include: (a) training sets, used to train the model; (b) validation sets, used to evaluate the model's performance during the training process; and (c) V&V sets, which are used after training to evaluate the model's performance on unseen data. Each image is a part of only one type of dataset (i.e. training images are not used for validation or for V&V). The V&V sets are typically made up of only real-world data, but can also include images that have been augmented or altered (i.e. brightness changed, noise added) in support of robustness testing. The Dataset Construction activity also produces a Datasheet aligned with the Datasheets for Datasets [22] template to describe the source, preprocessing completed, and intended use of the datasets, as well as a Dataset Description Report which outlines key statistical properties of the data within each dataset (alignment to the ODD, image and object counts, etc.). The datasheet, which includes source information, may be common across multiple datasets (for example, data collected over multiple data collections), but each individual dataset should have its own description to outline the statistical properties of that dataset.

The raw data, images, and labels are maintained in a CM system as per the CM Plan. Raw data (collected video) and images are not subject to frequent alteration and are not well-suited for text-based CM tooling, so those files are maintained in repo-based storage. The mapping of images to the ODD Taxonomy as well as labeling data is maintained in a CM system more suitable for text. Image hashes are maintained separately from the images in CM.

As per the Data QA Approach, labels applied to images (ground truth bounding boxes as well as metadata and object tagging), are reviewed for quality in a multi-step process, initially as a part of the label assignment process, and then verified by an independent review process. When automated image augmentations are used which change the location of the boxes (for example, image resizing, rotation, or flipping) a representative sample of the augmented images and labels will be reviewed for quality.

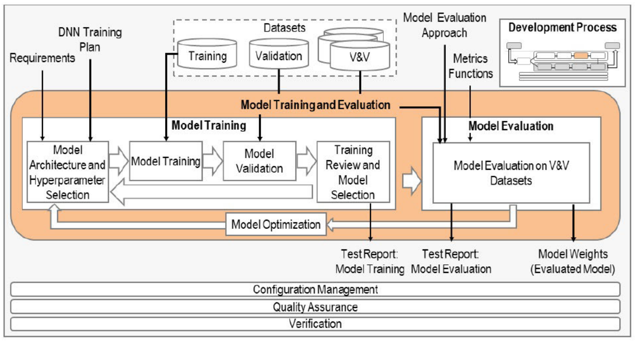
The DSCB reviews the products generated in this step for consistency with the ConOps and System Requirements. This includes a review of the Collection Plan, ODD Taxonomy, Data Dictionary, Dataset Description, and BOD. The datasets themselves (and the determination of which data is held out for V&V) are also subject to DSCB review. Since the datasets can be large, it is expected that the DSCB reviews a representative sample (as opposed to every image and every label).

3) MODEL TRAINING AND EVALUATION

The Model Training and Evaluation process, shown in Figure 12, begins with the selection of a model architecture and hyperparameters. There are many model architectures that are optimized for different purposes (detecting small objects, minimizing the time required to execute the model, etc.), and the overall component requirements will guide that selection. In addition, different models require different parameters for shaping both the learning and the inference process, and those hyperparameters need to be set. If the models are determined to not meet the metrics established in the Metrics Description and Definition, hyperparameters can be changed, the model architecture can be adjusted, or more data can be added to the training process, and the model can be retrained. The DNN Training Plan describes in more detail the approach used to select and evaluate the model and the hyperparameters.

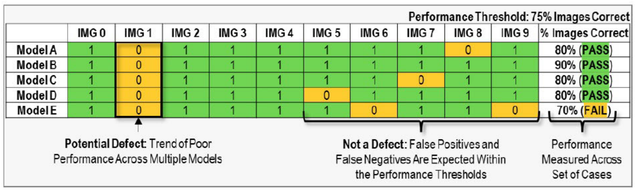
DNNs are trained using an iterative approach. During the training process, labeled images from the training dataset are processed through the model architecture, resulting in output bounding boxes. Those outputs are compared with the known labels, the difference between the known boxes and the resulting boxes is calculated, and the resulting error (called loss) is backpropagated to update the model weights. After a set number of epochs, the model is saved, the labeled images from the validation dataset are fed through the model, and the outputs are captured. The difference between the validation set inputs and the outputs is calculated and that loss is used to score the model for that epoch. When the validation loss stops significantly changing for a certain number of epochs, the model is considered "trained". Any version of the model generated and saved in the iterative model training cycle can be selected as the model to evaluate, but models are typically chosen based on their performance on the validation set during training. The criteria used to select the model(s) to advance to the evaluation step is detailed in the DNN Training Plan. The selected model and associated hyperparameters used for training, including the training loss curves and other training-specific metrics, are documented in the Model Training Report (a type of test report).

In the evaluation process, the selected model performs inference on the labeled images in the V&V datasets. The resulting outputs are then scored via the metrics functions identified in the Domain Characterization step. These metrics include operationally relevant metrics derived from the system requirements and ConOps as well as more traditional



DNN Training Plan
Requirements
Datasets
Training
Validation
V&V
Model Evaluation Approach
Metrics Functions
Development Process
Model Training and Evaluation
Model Training
Model Evaluation
Model Architecture and Hyperparameter Selection
Model Training
Model Validation
Training Review and Model Selection
Model Evaluation on V&V Datasets
Model Optimization
Test Report: Model Training
Test Report: Model Evaluation
Model Weights (Evaluated Model)
Configuration Management
Quality Assurance
Verification

FIGURE 12. Model training and evaluation process.


Performance Threshold: 75% Images Correct

	IMG 0	IMG 1	IMG 2	IMG 3	IMG 4	IMG 5	IMG 6	IMG 7	IMG 8	IMG 9	% Images Correct
Model A	1	0	1	1	1	1	1	1	0	1	80% (PASS)
Model B	1	0	1	1	1	1	1	1	1	1	90% (PASS)
Model C	1	0	1	1	1	1	1	0	1	1	80% (PASS)
Model D	1	0	1	1	1	0	1	1	1	1	80% (PASS)
Model E	1	0	1	1	1	1	0	1	1	0	70% (FAIL)

Potential Defect: Trend of Poor Performance Across Multiple Models

Not a Defect: False Positives and False Negatives Are Expected Within the Performance Thresholds

Performance Measured Across Set of Cases

FIGURE 13. Example model evaluation for a single metric.

machine learning metrics. Robustness test cases are similarly evaluated. The results of the model evaluation are provided in the Model Evaluation Report (a type of test report), and the Evaluated Model is stored in CM. An example of model evaluation for a single metric is shown in Figure 13.

The performance metrics used to evaluate the model are defined and described in the Metrics Definition and Description, and the simplified example shown here is for illustration only. In this example, five models are evaluated against the same V&V dataset containing 10 images. The performance metric used for evaluation in this example is the percentage of the images in the V&V set where the model correctly identifies the drogue (the percent of true positives). A correct detection is shown as a 1 in the table, and a miss or incorrect detection is shown as a 0. A model is considered to have achieved sufficient performance (a PASS) if the combined performance of the V&V dataset is above the threshold for that metric (ex. 75%). As shown in the example, no evaluated model detects the drogue objects in all of the V&V images, however, Models A-D all exceed the performance threshold. Also shown in the figure is a trend of consistent misses across all models in V&V Image 1. Trends across images within a model could also be defects – for example, some models may not perform well detecting small objects or may perform less well in very bright images. This type of performance trend could reflect a defect in either the training or validation datasets, the model architectures, or the training process, and will be documented and investigated.

The intermittent misses shown for images 5-9 are most likely not defects. It is expected that there will be some false positives (where objects are mistakenly detected) and false negatives (where existing objects are not found) in the normal performance of the system along with the runtime assurance monitoring approaches other system components could implement to detect their occurrence. The metrics definition process will identify the system-specific metrics thresholds for acceptable levels of false positives and false negatives. The model evaluation process identifies performance trends, compares performance across multiple model options, tracks and investigates potential defect trends, and uses these insights to down-select models for optimization and deployment.

Once a model has been assessed to meet performance, additional steps can be taken to optimize the model to run more efficiently on the target hardware. An example of one of those optimizations is model pruning, where model weights that are near zero (and therefore not contributing significantly to the model decision-making) are removed to make the model smaller, and therefore more time-efficient to execute. Once the model is pruned, it is retrained and re-evaluated to ensure that the performance of the model continues to meet thresholds, following the same training and evaluation process described above. These optimizations are applied only to models that are identified as valuable to move forward to deployment.

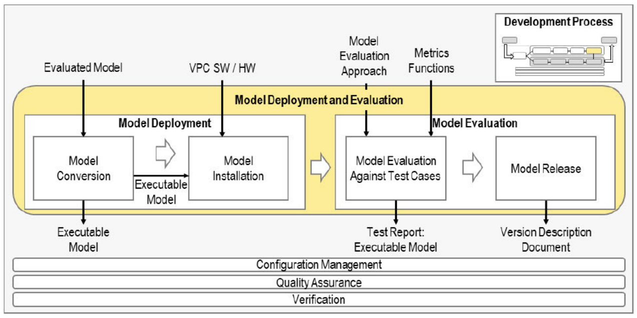


FIGURE 14. Model deployment and evaluation process.

The selected model and associated hyperparameters are maintained in a CM system as per the overall CM Plan. Software tooling that facilitates model training and evaluation (for example, training notebooks or containers) are maintained CM as defined by the CM Plan. Experiment tracking is used to maintain traceability between the model, the datasets, and hyperparameters used in training and the model results.

As per the DNN Training Plan, the architecture, training process, hyperparameters and outputs of the learning process are reviewed during the model review and selection phase.

The model is verified against the V&V and robustness cases as described in the Model Evaluation Approach. The generated test reports and their associated analyses are reviewed.

4) MODEL DEPLOYMENT AND EVALUATION

Once a model has been selected for deployment, the model is converted to a format that can be consumed by the VPC. Once the executable model has been created, it is installed on the target hardware as a configuration file for the VPC's Object Detection subfunction. The VPC (and integrated DNN) is then executed against a suite of captured test videos to evaluate the model performance against those known cases. Once it has been determined that the model meets performance as defined in the Metrics Definition and Description, the model and VPC are released with a corresponding VDD to system test.

As shown in Figure 14, the model deployment process is the process used to create a model that can be consumed and executed in a video processing pipeline. The process is described in detail in the Deployment Approach. To begin deployment, the selected model is converted into a configuration file that can be consumed by the VPC's Object Detection subfunction and that is optimized to run on the GPU hardware. Once the executable model has been created, it is installed on the GPU hardware with the video processing software and associated configuration parameters. The executable model is evaluated on the same V&V datasets as the evaluated model to ensure that the conversion process has not impacted model performance.

The installed executable model is then evaluated as defined in the Model Evaluation. This approach includes test videos from lab collections and videos collected from real-world environments, some of which were extracted into frames and labeled to form the V&V image data set. The evaluation process for the executable model uses video inputs (as opposed to individual frame data), as it is run as a part of the VPC. Since there is typically not labeled truth data for videos, the metrics for evaluating the executable model running on the hardware will be focused on how the model performs on video inputs: including the number of frames per second performance achievable by the model, the number of dropped tracks when tracks are in frames, and the overall processing utilization and throughput of the VPC. Models that reach this step have already achieved the model performance thresholds defined for single image detection, but this evaluation is the first time in the process that runtime-relevant metrics such as frames per second can be captured. There are different configuration settings for the VPC that impact those metrics, and the evaluation process includes updating those settings to ensure that performance requirements are met. The metrics used to assess the performance of the executable model are defined in the Metrics Definition and Description. As the model is evaluated against the established test cases, test reports are maintained.

Finally, once the model has met all thresholds, the model is released for use. A VDD referencing the baseline information for the data used in model training and testing, hyperparameter and architecture selection, configuration information for the VPC, and references to CM components and their archival locations is created. The Deployment Approach defines the build and release process for the VPC and the configuration files that are required for its use.

The executable model, configuration data, and VPC software are maintained in a CM system as per the CM Plan. An experiment tracking tool is used to maintain traceability between the model, the datasets, and hyperparameters used in training and the model results.

The VDD and Test Reports are reviewed for consistency and completeness.

To verify that the executable model produces the same results as the evaluated model, the executable model runs inference against the same data as the evaluated model, to ensure that the results are in alignment.

V. CONCLUSION

Before a system will be permitted to allow an AI/ML algorithm to take actions without a human in or on the loop, a method to certify must be established. This paper details the approach that the ONR A4RS FNC has proposed to allow a DNN to complete the aerial refueling task. In particular, it details the development life cycle for the DNN to complete the object detection sub-task required for the high-gain aerial refueling task. This early baseline is focused on supervised learning use cases and is intended as a starting point for process assessment. As AI/ML techniques used in aviation expand and evolve, this method may similarly need to be re-evaluated and extended. Future work that expands this concept is on the critical path to allow AI/ML capabilities within aviation.

ACKNOWLEDGMENT

The authors would like to thank the ONR for their continued support of this effort.

REFERENCES

[1] P. Tucker. (Mar. 2021). *Drones Could One Day Make Up 40 Percent of a Carrier Air Wing, Navy Says*. [Online]. Available: https://www.defenseone.com/technology/2021/03/drones-could-one-day-make-40-carrier-air-wing-navy-says/172799/

[2] D. Clement, S. Mottino, and D. Costello, "Process assurance for object detection through deep neural networks to accomplish the autonomous aerial refueling task," in *Proc. IEEE Int. Automated Vehicle Validation Conf. (IAVVC)*, Oct. 2023, pp. 1–6.

[3] D. H. Costello and H. Xu, "Relating sensor degradation to vehicle situational awareness for autonomous air vehicle certification," *J. Aerosp. Inf. Syst.*, vol. 18, no. 4, pp. 193–202, Apr. 2021.

[4] J. Parry and S. Hubbard, "Review of sensor technology to support automated air-to-air refueling of a probe configured uncrewed aircraft," *Sensors*, vol. 23, no. 2, p. 995, Jan. 2023.

[5] D. Costello and R. Adams, "A framework for airworthiness certification of autonomous systems within United States naval aviation," *J. Aviation*, vol. 7, no. 1, pp. 7–16, Mar. 2023.

[6] D. Costello and H. Xu, "Using a run time assurance approach for certifying autonomy within naval aviation," *Syst. Eng.*, vol. 26, no. 3, pp. 271–278, May 2023.

[7] E. D. Dickmanns, "Developing the sense of vision for autonomous road vehicles at UniBwM," *Computer*, vol. 50, no. 12, pp. 24–31, Dec. 2017.

[8] B. Barua, C. Gomes, S. Baghe, and J. Sisodia, "A self-driving car implementation using computer vision for detection and navigation," in *Proc. Int. Conf. Intell. Comput. Control Syst. (ICCS)*, May 2019, pp. 271–274.

[9] P. Kohli and A. Chadha, "Enabling pedestrian safety using computer vision techniques: A case study of the 2018 Uber Inc. self-driving car crash," 2018, *arXiv:1805.11815*.

[10] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, L. de Paula Veronese, T. Oliveira-Santos, and A. F. De Souza, "Self-driving cars: A survey," *Expert Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 113816.

[11] C.-J. Yang, W.-K. Huang, and K.-P. Lin, "Three-dimensional printing quality inspection based on transfer learning with convolutional neural networks," *Sensors*, vol. 23, no. 1, p. 491, Jan. 2023.

[12] G.-H. Gwon, J. H. Lee, I.-H. Kim, and H.-J. Jung, "CNN-based image quality classification considering quality degradation in bridge inspection using an unmanned aerial vehicle," *IEEE Access*, vol. 11, pp. 22096–22113, 2023.

[13] United States Dept. Defense. (Dec. 2014). *Mil-hdbk-516c: Department of Defense Handbook, Airworthiness Certification Criteria*. [Online]. Available: https://daytonaero.com/wp-content/uploads/MIL-HDBK-516C-from-ASSIST.pdf

[14] United States Dept. Defense. (May 2012). *Department of Defense Mil-Std-882e: Department of Defense Standard Practice, System Safety*. [Online]. Available: https://www.dau.edu/cop/armyesoh/DAU%20Sponsored%20Documents/MIL-STD-882E.pdf

[15] *Software Considerations in Airborne Systems and Equipment Certification*, document DO-178C, RTCA Inc., 2011.

[16] D. Costello, L. DeVries, C. Mauldin, and B. Ross, "DNN based ranging in support of autonomous aerial refueling," *J. Intell. Robotic Syst.*, vol. 109, no. 3, p. 49, Nov. 2023.

[17] D. Costello, "Towards autonomous aerial refueling: Advanced autonomous air-to-air refueling system (A4RS), ONR FY-24 new start future naval capability (FNC)+ USNA support for the FNC, invited presentation," Aerial Refueling Syst. Advisory Group, Orlando, FL, USA, Tech. Rep., 2023.

[18] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proc. IEEE*, vol. 111, no. 3, pp. 257–276, Mar. 2023.

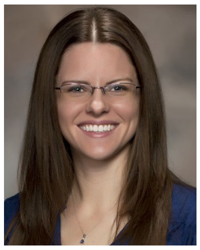
[19] Y. Amit, P. Felzenszwalb, and R. Girshick, *Object Detection*. Cham, Switzerland: Springer, 2020, pp. 1–9.

[20] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *Social Netw. Comput. Sci.*, vol. 2, no. 3, p. 160, May 2021.

[21] Nat. Highway Traffic Saf. Admin., Washington, DC, USA. (2018). *A Framework for Automated Driving System Testable Cases and Scenarios*. [Online]. Available: https://www.nhtsa.gov/document/framework-automated-driving-system-testable-cases-and-scenarios

[22] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III, and K. Crawford, "Datasheets for datasets," 2018, *arXiv:1803.09010*.

DANIELLE CLEMENT received the B.S. degree in computer science from the University of Richmond, in 2000, and the M.S. and Ph.D. degrees in computer science from The University of Texas at Arlington, in 2007 and 2016, respectively. She has nearly 20 years of experience developing mission management and autonomy software applications. She has supported multiple internal and external research development efforts focused on mission management and autonomy. Her Ph.D. research focused on game theoretic team formation algorithms for heterogeneous mobile agents, and her current research interests include hierarchical reinforcement learning algorithms and V&V approaches for autonomy applications.



SARAH MOTTINO received the B.S. degree in biology from California State University, Long Beach, in 2006. She has 17 years of experience in research and development with an emphasis on improving human performance and safety in the cockpit. She is currently an Autonomy and AI/ML Engineer with Lockheed Martin Aeronautics Company, where her work focuses on human performance and the application and certification of artificial intelligence systems in aviation environments.

DONALD H. COSTELLO III was born in San Diego, CA, USA, in 1978. He received the B.S. degree in systems engineering from United States Naval Academy, Annapolis, MD, USA, in 2000, the M.A.S. degree in aeronautical science from Embry-Riddle Aeronautical University, Daytona Beach, FL, USA, in 2005, the M.S. degree in aeronautical engineering from the Air Force Institute of Technology, Dayton, OH, USA, in 2009, the M.S. degree in systems engineering from the Naval Post Graduate School, Monterey, CA, USA, in 2011, and the Ph.D. degree in mechanical engineering from the University of Maryland, College Park, MD, USA, in 2020. He is currently an Assistant/Permanent Military Professor with the Weapons, Robotics, and Control Engineering Department, United States Naval Academy. His work focuses on the certification and development of unmanned autonomous systems for practical use.