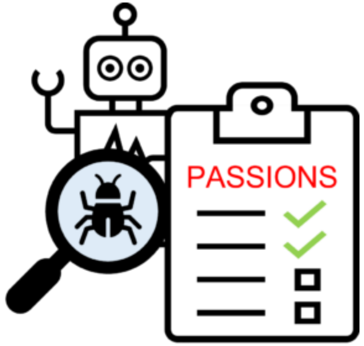
Product Assurance for On-Board Learning-Enabled Software

Iulia Dragomir
GMV

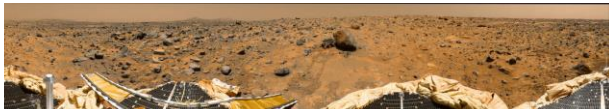
Software Product Assurance Workshop
September 27th, 2023



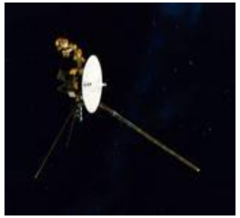
AUTONOMY IN SPACE SYSTEMS

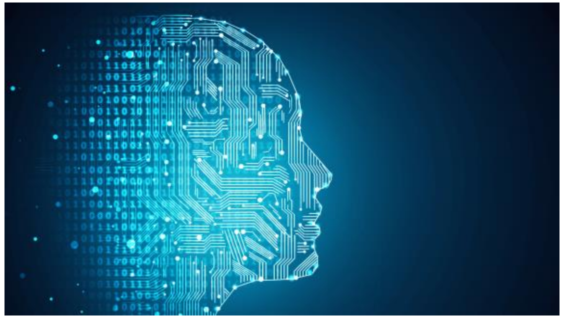
- Future space missions need to address several challenges:
  - Scientifical, technical and performance such as mission/scientific return, communication delays, environment uncertainty, responsiveness, availability, and reliability
  - Complexity in design, development, AIT, and operation, with elevated costs at ground segment both for personnel and equipment
- **Autonomy** is considered a must to improve on the criteria above
  - It is defined as the capability to act without human or other systems intervention
- ECSS-E-70 defines 4 autonomy levels: E1 (telecommand), E2 (time-tag), E3 (event-driven) and E4 (goal-commanding)
- More missions develop and employ autonomy mostly at function-level, with the system/mission-level in view, and mainly at E3 (e.g., OBCPs)

#	Mission Type	Environment uncertainty	TC-TM Links	Plan complexity	Autonomy
1	Orbital mission	Low	Almost permanent	Medium-High	E1,E2,E3,E4 (i.e., EO-1)
2	Deep space	Medium	High latency	Medium	E1,E2,E3,E4 (i.e., DS1)
3	Planetary exploration	High	High latency / Communication windows	High	E1,E2,E3 (i.e., Curiosity/ Opportunity)


Planetary Exploration Mission

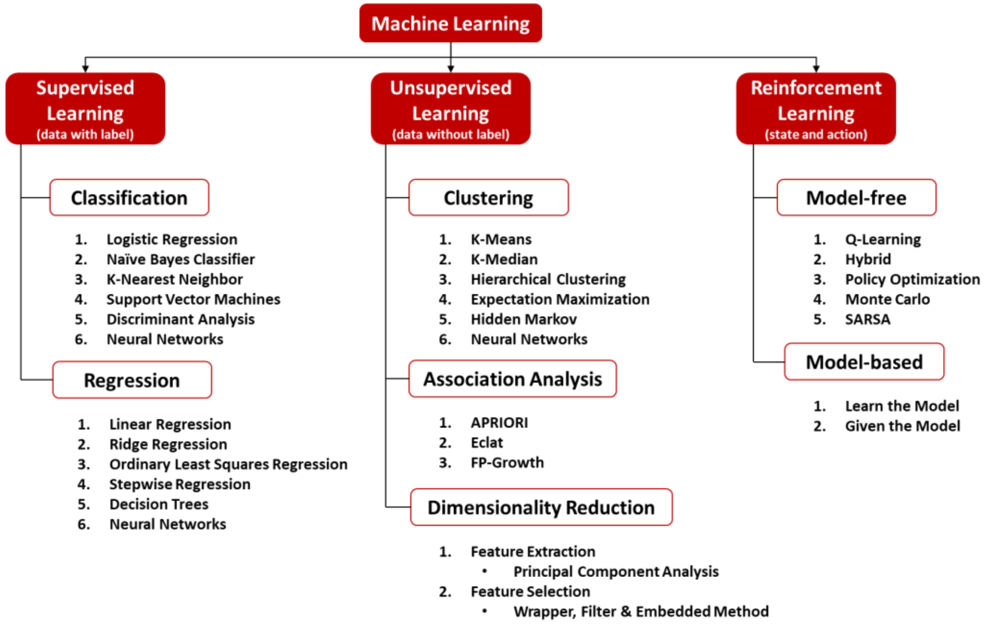
Orbital Mission


Deep Space Mission



MACHINE LEARNING IN ON-BOARD SOFTWARE

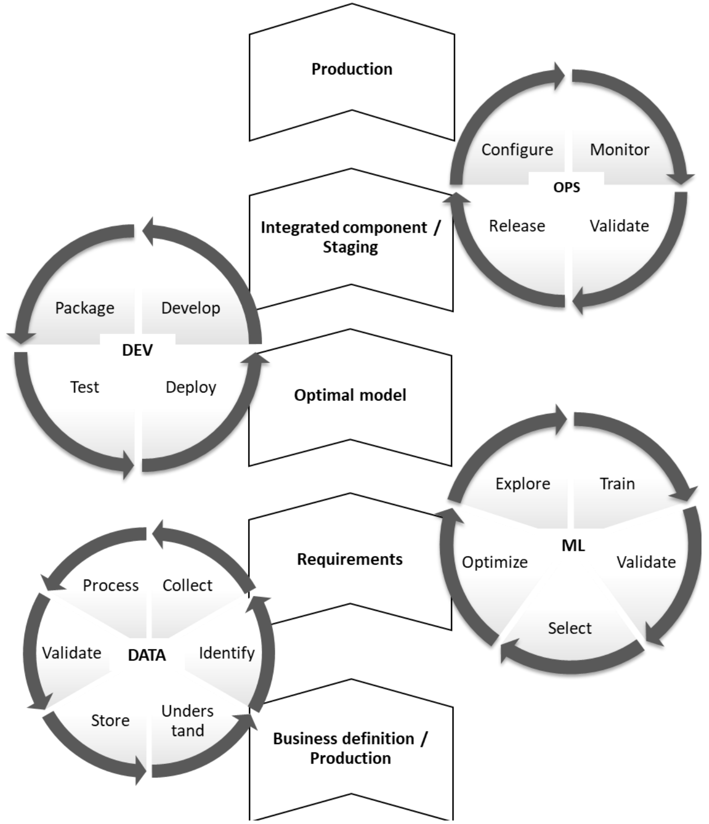
- Autonomy can be achieved through standard algorithms or Artificial Intelligence (symbolic, **Machine Learning**, and integrative)
- Three paradigms for ML: **supervised**, **unsupervised** and **reinforcement learning**
- Compute the estimation of a function *f*, given a set of inputs and possibly known set of outputs (e.g., reinforcement learning)
- Use in space systems for:
  - Detection and classification (e.g., Φ-sat-1/2, MoonNet)
  - Navigation (e.g., ArgoMoon)
  - Anomaly detection and prediction (e.g., ArgoMoon)
  - Control optimization
- Drawbacks:
  - Black-box component
  - Data quality dependent
  - Non-deterministic (stochastic) behaviour
  - Ill-suited at system level



A state-of-the-art of ML techniques

CHALLENGES FOR ON-BOARD ML (1/2)

- **Development lifecycle**:
  - Different paradigm focusing on data and models, rather than SW
- **Validation and testing**:
  - Data correctness, since ML model correctness depends on it
  - ML model correctness criteria, since such components are often black-boxes lacking a formal specification
  - ML model prediction uncertainty and quantification of the error impact
  - ML model coverage criteria and adequate data for testing
- **Verification**:
  - ML model correctness criteria formalization
  - Inherent complexity of the verification approach
  - Scalability of state-of-the-art techniques



MLOps development approach

CHALLENGES FOR ON-BOARD ML (2/2)

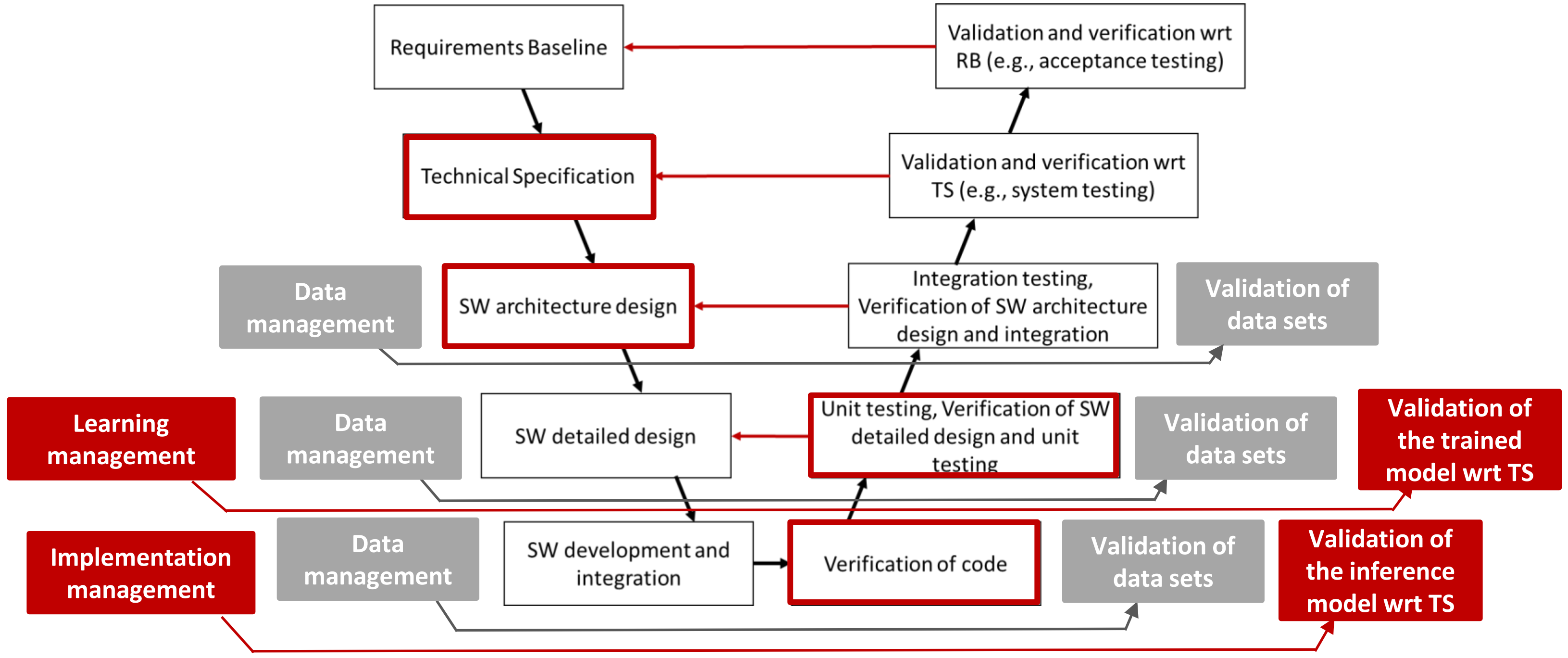
- **Safety and dependability assurance**:
  - ML model prediction uncertainty,
  - Complexity of explainability
- Relevant impact on the **system RAMS and criticality analysis**
- **Product assurance**:
  - Process for assured ML model development and deployment,
  - Quality model regarding the main features of ML models

SOFTWARE PRODUCT ASSURANCE FOR AUTONOMOUS ON-BOARD SOFTWARE (PASSIONS)

- **Aim:** Study the impact of using ML-based on-board software in the software product assurance activities and propose guidelines enabling the safe and reliable deployment of autonomous on-board software
  - Cover the use of autonomy in ongoing and upcoming space missions and identify the needs of such systems to correctly operate
  - Cover completed and ongoing standardisation activities that aim to provide guidelines for the development of autonomous systems
  - <u>Provide a definition of correct autonomous systems in terms of safety and dependability and associated development, verification and validation techniques that are representable through quality metrics</u>
  - Provide a set of guidelines and requirements in the form of amended ECSS standards that are immediately usable for space missions
- **Consortium:** GMV Spain (prime), Fondazione Bruno Kessler
- **Duration:** 01/12/2022 – 31/11/2023

DEVELOPMENT LIFECYCLE

- Based on the ECSS described activities
- Data lifecycle with dedicated activities for data sets production and validation
- Model lifecycle with dedicated activities for trained and inference model production and validation


Requirements Baseline
Validation and verification wrt RB (e.g., acceptance testing)
Technical Specification
Validation and verification wrt TS (e.g., system testing)
Data management
SW architecture design
Integration testing, Verification of SW architecture design and integration
Validation of data sets
Learning management
Data management
SW detailed design
Unit testing, Verification of SW detailed design and unit testing
Validation of data sets
Validation of the trained model wrt TS
Implementation management
Data management
SW development and integration
Verification of code
Validation of data sets
Validation of the inference model wrt TS

DATA LIFECYCLE

- Produces the training, validation and testing data sets and ensures their quality for the obtention of an adequate ML model
- Starts from the Technical Specification definition, which includes the specification of the data sets requirements
  - For example, data needed, origin, annotation process, completeness and representativeness, dependency relations, accuracy, precision, etc.
- Data management includes the main activities for the (i) collection, (ii) labelling, (iii) preparation and (iv) splitting of the data sets
- Validation of the data sets includes checking the data sets quality and the other requirements set in the Technical Specification
- Each step of the data management and validation shall be documented together with the quality results obtained in a dedicated deliverable, e.g., Data Design Document
- The data management and validation are performed for PDR, with new iterations for the next milestones if any changes occur (e.g., new data becomes available from operations)

DATA QUALITY MODEL

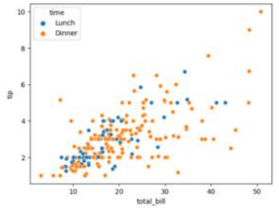
- Based on available standards such as ISO/IEC 25012, ISO/IEC 25024, ISO/IEC CD 5259-2 and guidelines from ESA and EASA
- Some **characteristics**:
  - *Representativeness*: statistically accurate for the data distribution
  - *Completeness*: includes values for all attributes and related entities
  - *Consistency*: free of contradictions and coherent with other data
  - *Relevance*: suited for the context it is used in
  - *Balance*: distribution of data for all dimensions
  - *Accuracy*: correctly represents the true value
  - *Precision*: exact representation needed
- The quality model shall be **tailored** according to the scope of the ML-based system, including the measurable characteristics, appropriate metrics and target values

DATA TECHNIQUES

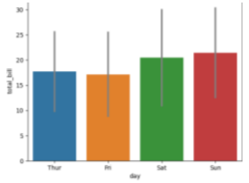
- **For quality**:
  - **Data quality improvement**: combination of data profiling, cleaning validation and other processes to ensure that models and further analysis are based on accurate and reliable information
  - **Data visualization**: graphical representation of data through charts, plots, etc. allowing to understand complex data relationships and validate statistical data sets properties (e.g., representativeness, completeness, balance)

- **For improved ML performances**:
  - **Feature engineering**: selecting, creating or transforming features through pre-processing such that they are suitable for the chosen algorithm
  - **Data augmentation**: creating new data through predefined transformations (e.g., rotation, scaling, translation, flipping, cropping, noise injection) of the original data in order to improve the generalization and robustness of the model
  - **Dimensionality reduction**: data reduction to lower dimensionality (most useful features) without losing general information

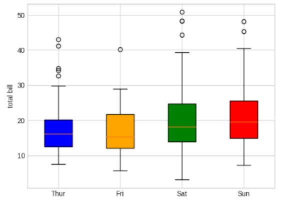
Scatter plot



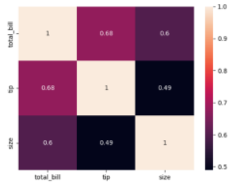
Bar plot

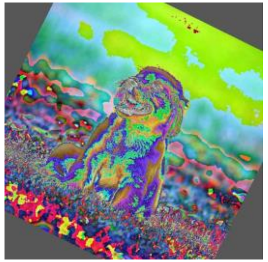


Box plot



Heatmap

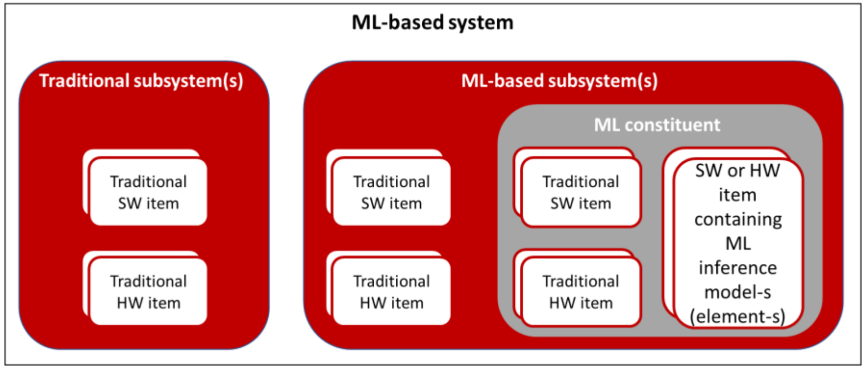




Data augmentation with rotation, flip, and Gaussian filter

MODEL LIFECYCLE (1/2)

- Produces in several steps the inference model part of the ML constituent and ensures its quality and Technical Specification satisfaction
- Distinguishes between the **trained model** obtained through the training and optimization phase, and the **inference model** obtained through the implementation phase on the trained model (i.e., the executable model targeted to the deployment environment)
- The **ML constituent** embeds the inference model together with pre- and post-processing components and it is considered as part of the SW



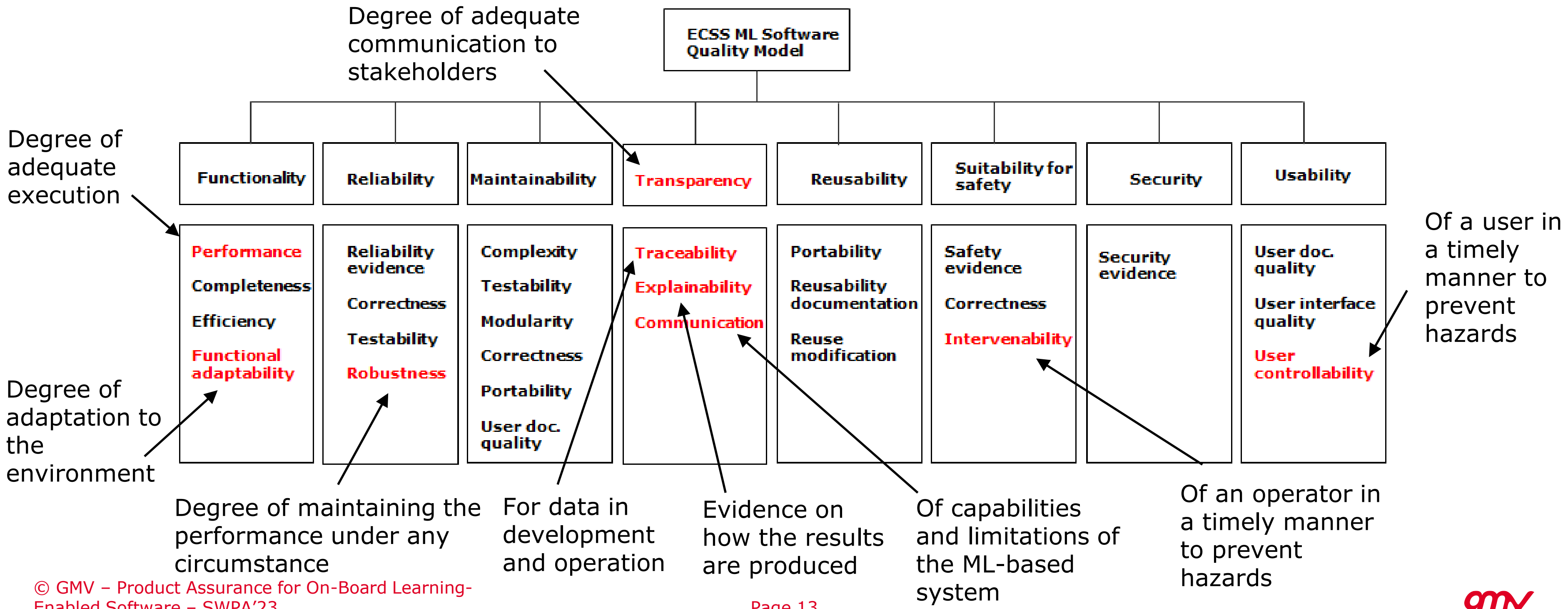
Architecture of an ML-based system

MODEL LIFECYCLE (2/2)

- Starts from the Technical Specification definition which includes the specification for the trained model, inference model and ML constituent
- Learning management includes the main activities for obtaining the trained model: (i) architecture(s) implementation, (ii) training with the training data set, (iii) validation with the validation data set, (iv) selection of the best model according to cost/loss function, and (v) optimization of the selected model
- Implementation management includes the main activities for obtaining the inference model: (i) implementation of the trained model for the target architecture, and (ii) validation with the testing data set
- Validation of the trained/inference model wrt Technical Specification includes the verification and validation activities for ensuring the trained/inference model quality and satisfaction of set requirements
- Learning/implementation management and the corresponding model validation are performed for Model Design Review/TRR, accordingly
- Each step of the learning/implementation management and validation shall be documented together with the quality results obtained in a dedicated deliverable, e.g., Model Design Document

EXTENDED SW QUALITY MODEL

- Based on available standards such as ISO/IEC 25010, ISO/IEC DIS 25059 and guidelines from EASA
- Metrics are defined for the new characteristics, but also existing ones such as efficiency and reliability
- The quality model shall be tailored according to the scope of the ML-based system



VERIFICATION AND VALIDATION

- Can be applied at component or system-level to check the desired properties, such as safety, liveness, reachability, robustness, stability, monotonicity, etc.
- Formal verification consists of methods in which the component and property are formalized in a mathematical model and state-of-the-art solvers and algorithms are used for checking the satisfaction (e.g. SMT or MILP solvers)
- Validation includes a plethora of techniques such as testing, simulation, or statistical analysis
- Several tools were explored and analysed:
  - Maturity issues
  - Usability issues in terms Of applicability to other type of case studies (tailoring of interfaces, lack of updates to latest technologies)

Tool	Pros	Cons	Area of applicability
Formal verification			
Marabou	Supports fully connected and convolutional DNNs Supports divide-and-conquer solving mode	Only supports piece-wise linear activation functions Last tool update was in 2021	Neural Networks
Testing coverage			
DeepGauge	Provides a comprehensive understanding of the internal neuron activation patterns. Provides a coverage criterion that is scalable, simple, and generalizable	Does not guarantee good results in general, excepts for adversarial test data for CNNs	Deep Neural Networks
Simulation-based testing			
VIVAS	Can be applied to different domains and simulators. Uses formal methods for test scenario generation (counterexample generation, sampling)	Requires the specification of the system under study as symbolic model	ML-based systems
VerifAI	Provides several features (falsification, systematic testing, parameter synthesis, data set augmentation) based on formal methods	Limited to the automotive domain for simulations	ML-based systems for autonomous driving

Some examples of V&V tools

SAFETY AND DEPENDABILITY ASSURANCE

- Safety and dependability of a system depends on the confidence of ML predictions:
  - Such prediction could be inaccurate or wrong, which could lead to harmful consequences and failure propagation
  - ML techniques are intrinsically subject to epistemic uncertainty (i.e., lack of training data) and aleatoric uncertainty (i.e., inherent stochasticity of observations), and often lack in explainability (i.e., why and how the prediction was made)

- Well-known techniques can be used to reduce the impact of such predictions on the overall system properties:
  - Redundancy to increase the fault/failure tolerance through architectural elements
  - FDIR to detect and isolate the faults early as possible in operation, and eventually apply reconfiguration
  - Safety cages to detect and correct in operation wrongful inputs or outputs of the ML model (e.g., Tempest)

- Novel techniques can be used to quantify the uncertainty and explainability of ML predictions in development to ensure the overall system properties given the assumptions made:
  - Uncertainty can be understood and assessed as random fluctuations part of the ML model (e.g., GluonTS), of the output (e.g., quantile loss) or of the ensembles of models (e.g., bagging, voting)
  - Explainability can be assessed for a single prediction (local) or for each possible prediction (global) in oder to gain confidence in the model correctness (e.g., Explainability Toolbox, GradCam)

REDUNDANCY AND CRITICALITY

- The criticality analysis and category assigned is well-defined in the ECSS standards and handbooks
- The use of ML techniques in the system is orthogonal to the criteria and analysis to be performed, without adding new constraints
- However, the criticality assigned demands different tailoring of the quality models (e.g., higher criticality shall imply possibly more characteristics with higher target values)
- The criticality level can be reduced by applying the ECSS standard and redundancy including compensating provisions

Compensating procedures possible for the mitigation of common cause of errors

Independence Type	Description	Initial Criticality	Compensating provision with SW redundancy for a ML SW critical component	Criticality after compensating
Functional Independence	Minimizes the likelihood of common sources of error associated with common requirement errors / common requirement interpretation errors.	A	Traditional SW (criticality A)	B
		B	Redundant SW component with different implementation (criticality B)	C
		C	Redundant ML SW component with different implementation (criticality C)	D
Dissimilar Technology	Use of dissimilar technology to achieve a common goal, e.g. hydraulic vs. electric	A	Traditional SW (criticality A)	B
		B	ML in a different learning technique (criticality B)	C
		C	ML in a different learning technique (criticality C)	D
Dissimilar implementation	Use of dissimilar operating systems, coding languages, component types, etc.	A	Traditional SW (criticality A)	B
		B	ML component with different model (criticality B)	C
		C	ML component with different model (criticality C)	D

CONCLUSION

- ECSS-based development lifecycle including dedicated data and model lifecycle with milestones and deliverables
- (Formal) Verification and Validation techniques to check the ML-based system (e.g., model, constituent, system) properties (functional, performance, scalability, generalization, reliability, resilience, safety, operational, etc.)
- Safety and dependability techniques to gain confidence in the ML model prediction and on-board correction in case of uncertainty or fault/failure detection
- Approach for criticality analysis and redundancy in system architecture for dependability
- ECSS-based quality model for data and software to ensure the product assurance of the ML-based system development process
- Ongoing definition of guidelines for the process application in practice and adaptation of standards

Thank you!
Questions?

idragomir@gmv.com