ACQUISITION INNOVATION RESEARCH CENTER

Development of an Artificial Intelligence (AI) Test Harness for the Department of Defense (DoD)

EXECUTIVE SUMMARY AND REPORT

SEPTEMBER 2024

PRINCIPAL INVESTIGATOR
Laura Freeman, *Virginia Tech*

CO-PRINCIPAL INVESTIGATOR
Stephen Adams, *Virginia Tech*
Erin Lanus, *Virginia Tech*
Naren Ramakrishnan, *Virginia Tech*

RESEARCH ASSOCIATE
Brian Mayer, *Virginia Tech*
Patrick Butler, *Virginia Tech*

RESEARCH ASSISTANT PROFESSOR
Jaganmohan Chandrasekaran, *Virginia Tech*

RESEARCH ASSOCIATE PROFESSOR
William Headley, *Virginia Tech*

RESEARCH DATA ANALYST
Brian Lee, *Virginia Tech*

SOFTWARE DEVELOPER
Alex Kyer, *Virginia Tech*

SYSTEMS ADMIN
Jared Gregerson, *Virginia Tech*

SPONSOR

Mr. Paul Lowe, Deputy Director, Strategic Initiatives, Policy and Emerging Technologies (Acting), Director, Operational Test and Evaulation (DOT&E)
Mr. Nilo Thomas, DOT&E Software and AI Advisor

DISTRIBUTION STATEMENT A.
Approved for public release:
distribution unlimited.

DISCLAIMER

Copyright © 2024 Stevens Institute of Technology and Virginia Tech. All rights reserved.

The Acquisition Innovation Research Center (AIRC) is a multi-university partnership led and managed by the Stevens Institute of Technology and sponsored by the U.S. Department of Defense (DoD) through the Systems Engineering Research Center (SERC)—a DoD University-Affiliated Research Center (UARC).

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Office of the Under Secretary of Defense for Acquisition and Sustainment (OUSD(A&S)) and the Office of the Under Secretary of Defense for Research and Engineering (OUSD(R&E)) under Contract HQ0034-19-D-0003, TO#0092.

The views, findings, conclusions, and recommendations expressed in this material are solely those of the authors and do not necessarily reflect the views or positions of the United States Government (including the Department of Defense (DoD) and any government personnel), the Stevens Institute of Technology, or Virginia Tech.

No Warranty.

This Material is furnished on an "as-is" basis. The Stevens Institute of Technology and Virginia Tech make no warranties of any kind—either expressed or implied—as to any matter, including (but not limited to) warranty of fitness for purpose or merchantability, exclusivity, or results obtained from use of the material.

The Stevens Institute of Technology and Virginia Tech do not make any warranty of any kind with respect to freedom from patent, trademark, or copyright infringement.

SYSTEMS ENGINEERING RESEARCH CENTER

AIRC ACQUISITION INNOVATION RESEARCH CENTER

STEVENS INSTITUTE OF TECHNOLOGY 1870

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

RESEARCH TEAM

Name	Organization	Labor Category
Laura Freeman	Virginia Tech	Principal Investigator
Stephen Adams	Virginia Tech	Co-Principal Investigator
Erin Lanus	Virginia Tech	Co-Principal Investigator
Naren Ramakrishnan	Virginia Tech	Co-Principal Investigator
Brian Mayer	Virginia Tech	Research Associate
Patrick Butler	Virginia Tech	Research Associate
Jaganmohan Chandrasekaran	Virginia Tech	Research Assistant Professor
William Headley	Virginia Tech	Research Associate Professor
Brian Lee	Virginia Tech	Research Data Analyst
Alex Kyer	Virginia Tech	Software Developer
Jared Gregerson	Virginia Tech	Systems Admin

ACKNOWLEDGEMENTS

This report summarizes emerging research developing new test and evaluation environments for Artificial Intelligence (AI) enabled systems conducted in support of the Director Operational Test and Evaluation's (DOT&E) Strategic Initiatives, Policy, and Emerging Technologies Directorate (SIPET). The authors would like to thank DOT&E SIPET for their support and engagement in shaping new methods for the future of test and evaluation (T&E). The techniques developed in this research reflect emerging AI technologies in signals intelligence and generative AI.

ACRONYMS AND ABBREVIATIONS

AI	Artificial Intelligence
AIRC	Acquisition Innovation Research Center
API	Application Programming Interface
ASK	Amplitude-Shift Keying
AWGN	Additive White Gaussian Noise
BERTScore	Bidirectional Encoder Representations from Transformer Scores
BLEU	Bilingual Evaluation Study
CC	Combinatorial Coverage
CDAO	Chief Digital and AI Office
CNN	Convolutional Neural Network
CODEX	Coverage of Data Explorer
COT	Chain of Thought
CSV	Comma Separated Values
DoD	Department of Defense
DOT&E	Director, Operational Test and Evaluation
IP	Internet Protocol
IQ	In-Phase and Quadrature
IR	Information Retrieval
LLM	Large Language Model
MC	Monte Carlo
MCQ	Multiple Choice Questions
METEOR	Metric for Evaluation of Translation with Explicit Ordering
ML	Machine Learning
NER	Named Entity Recognition
NIST	National Institute of Standards and Technology
PAM	Pulse-Amplitude Modulation
PSK	Phase-Shift Keying
PY-WASPGEN	Python-based Wideband Aggregate SPectrum GENerator
QAM	Quadrature-Amplitude Modulation
Q&A	Question and Answer

ReLU	Rectified Linear Unit
RF	Radio Frequency
RFML	Radio Frequency Machine Learning
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
SDCC	Set Difference Combinatorial Coverage
SIPET	Strategic Initiatives, Policy, and Emerging Technologies
SNR	Signal to Noise Ratio
T&E	Test and Evaluation
TCP	Transmission Control Protocol
UQ	Uncertainty Quantification
VT	Virginia Tech
VTNSI	Virginia Tech National Security Institute

EXECUTIVE SUMMARY

The development of Artificial Intelligence (AI) has revolutionized the way organizations operate. The Department of Defense (DoD) is no exception to this transformation. AI has the potential to revolutionize military capabilities and reduce human errors. To keep pace with our adversaries, one of the Office of the Director, Operational Test and Evaluation's (DOT&E) core strategic pillars is to pioneer test and evaluation (T&E) methods of weapon systems designed to change over time. Machine learning (ML) and AI models are notably capable of learning and changing over time. Moreover, the stochastic nature of the models that learn based on past data present new challenges for T&E. It is essential to ensure that these systems operate effectively, safely, and securely. A reliable test harness that provides high-quality data, AI models, and test and evaluation capabilities will accelerate and inform the development of new methods. The DOT&E has the responsibility to develop policies for T&E of AI-enabled systems. However, the current state of the AI capabilities and the corresponding T&E methods for AI/ML are evolving. The development of test harnesses has the potential to not only accelerate method development but also inform DOT&E's policy and guidance. Finally, test harnesses can serve as an educational resource for the T&E community where testers can learn T&E for AI-enabled systems by leveraging tools, processes, and methods in the T&E harness.

In this research, the team designed a framework for an AI Test Harness that could be applied to multiple types of AI models. Along with the framework, the research team developed a set of requirements for an AI Test Harness and produced a simple prototype. The research team then applied the developed framework to two use cases. The first is a Radio Frequency Machine Learning (RFML) use case that uses standard classification models. Under this use case, the research team advanced synthetic data generation capability by publicly releasing Python-based Wideband Aggregate SPectrum GENerator (PY-WASPGEN), a toolset for producing radio frequency (RF) data for training and testing AI/ML models. The research team also demonstrated the Coverage of Data Explorer (CODEX) capability on an RFML example. The project developed education and training material on the application of standard T&E methods to classification problems.

The second use case focused on Large Language Models (LLMs), a form of generative AI. LLMs are considered one of the most advanced forms of AI and recently gained popularity. Due to their recent advances, T&E for these types of models is nascent. The research team conducted a survey of the academic literature and industry best practices to assess the current state of T&E for LLMs. The results of this survey led to a framework for the various tasks a LLM can perform and the characteristics of a LLM that should be evaluated. Education and training material for some of these tasks was developed and publicly released. In this work, the research team did not distinguish between a LLM and a LLM-based system. The current version of the proposed harness framework conflates the two, but future work should investigate T&E for the LLM separate from the LLM-based system.

The contrast of these two separate use cases provides context to the value of test harnesses and the challenges in implementing them. The RF use case demonstrates that developing a test harness for standard machine learning models, such as classifiers, is a decently straightforward software engineering task supported by widely available open-source tooling and new capabilities investments that the DoD is making.

However, a test harness for AI-systems and generative AI requires more research. The former must consider the systems interactions with other systems, including human users and AI-enabled systems that may evolve over time. The latter requires everything involved in developing a test harness for an AI-enabled system plus further study on metrics, data sets, and balancing the results of tests on multiple tasks with possibly competing objectives.

CONCLUSIONS AND RECOMMENDATIONS

The rapid evolution of AI has transformed various industries, including defense, where AI capabilities are being integrated into mission-critical systems. This research has demonstrated the potential of an AI Test Harness framework to accelerate the development of T&E methodologies for AI and ML systems. Through two distinct use cases—Radio Frequency Machine Learning (RFML) and Large Language Models (LLMs)—the research team has provided valuable insights into the challenges and opportunities in testing AI.

The framework and prototype developed through this project serve as an important foundation for future work in AI T&E. It also emphasizes the need for continued investment in research, tool development, and educational resources. By establishing a reliable test harness and a robust set of policies, standards, and metrics, the DOT&E can better equip the T&E community to handle the unique challenges posed by AI and ML systems, ultimately ensuring that these technologies can be deployed safely and effectively in defense applications.

Future work should further distinguish between testing individual AI models and more complex AI-enabled systems, particularly generative AI like LLMs, and refine the tools, datasets, and metrics needed to evaluate these emerging capabilities. Specific recommendations include:

Recommendation 1: The DoD and AIRC should continue the development of Test Harnesses to advance T&E of AI-enabled systems.

a. AIRC should serves as a facilitator of test harness models for academic research in T&E of AI models.

b. The DoD should continue to invest in research on AI-enabled systems test capabilities and how they differ from AI model T&E.

Recommendation 2: The DoD should ensure private data sets for testing of LLMs. Public data sets may quickly become ineffective for testing LLMs as all public data will likely be used during training. Private data sets for each task should be developed that are withheld from the public and only used for testing.

Recommendation 3: The DoD or AIRC should develop dashboards for tracking the performance of LLMs on tasks relevant to the DoD. Dashboards should provide the ability to compare holistic evaluation with task-specific evaluation in DoD contexts.

TECHNICAL ACCOMPLISHMENTS

Over the period of performance, the research team has generated several technical accomplishments:

1. The creation of a framework for an Artificial Intelligence (AI) Test Harness and a set of requirements.
2. The development of a simple harness prototype and its application to a radio frequency machine learning (RFML) use case.
3. Development and deployment of data generation tools for radio frequency data.
4. Application of the Coverage of Data Explore (CODEX) tool to the RFML use case.
5. Assessment of uncertainty quantification methods to the RFML use case.
6. Development of training and education material for the test and evaluation of machine learning classifiers on the RFML use case.
7. Survey of the academic literature and best industry practices for the test and evaluation of Large Language Models (LLMs).
8. Development of training and education material for the test and evaluation of LLMs.

The radio frequency (RF) data generation and CODEX tools, as well as a git repository for the training and education material, will be hosted on the Virginia Tech National Security Institute (VTNSI) GitHub[1].

[1] https://github.com/vtnsi

AI TEST HARNESS

A framework for an AI Test Harness is outlined below and requirements for its construction are provided. A simple prototype for an AI Test Harness is described.

FRAMEWORK AND REQUIREMENTS

As shown in Figure 1 below, the proposed AI Test Harness shall connect three entities critical for T&E of AI:

1. AI/Machine Learning (ML) models – pre-trained models to test and evaluate provided by the harness user. This could be access to the model itself or through API calls to a model. The latter would ensure that models weights and architectures are not exposed to the user.

2. Test data sets – data sets used for testing and evaluating the provided AI model. The test sets are also provided by the user.

3. T&E software – test and evaluation software. The test harness should have the native capability to produce common performance metrics for AI models, e.g., classification error and mean squared error. The AI harness should also have the ability for a user to provide custom T&E software.

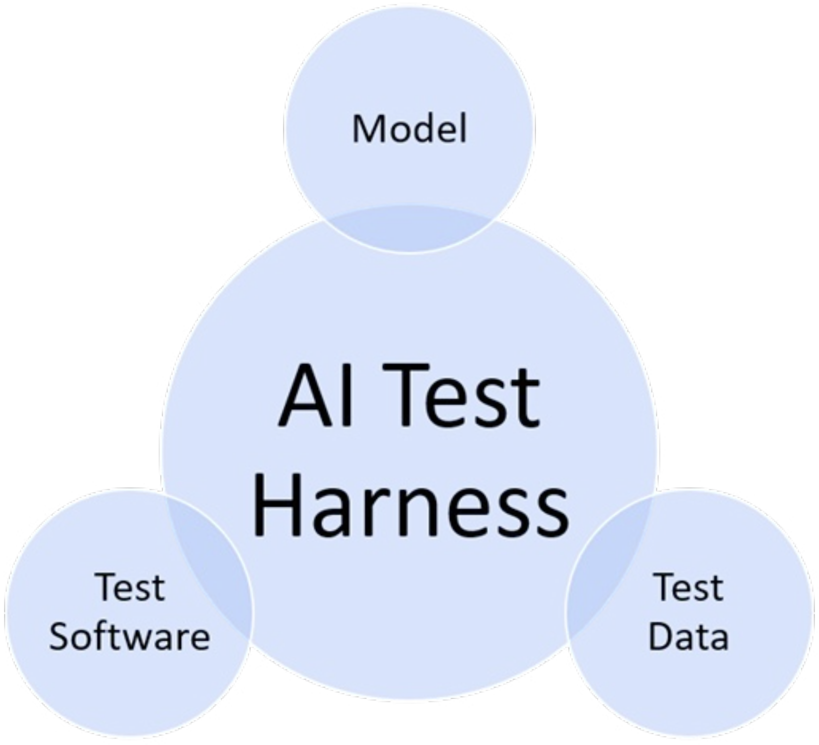


Figure 1: AI Test Harness Concept

Virginia Tech (VT) has defined the following seven requirements for the AI Test Harness. VT will work in collaboration with Navy Surface Warfare Center Dahlgren (Dahlgren), the Chief Digital and AI Office (CDAO), and other T&E organizations to develop capabilities for a test harness.

Requirement 1: Software Implementation
The AI Test Harness shall be a multi-language platform able to communicate with common programming languages over Transmission Control Protocol (TCP)/Internet Protocol (IP).

Requirement 2: Read Data Sets
The AI Test Harness shall have the ability to read common data file formats such as common separated value (csv) and text files. The test harness shall also be able to read data from common databases like SQL. The data sets may be hosted by the user.

Requirement 3: Manipulate Data Sets
The AI Test Harness shall have the ability to internally store and perform common data manipulation tasks such as splitting into subsets and sampling from data sets.

Requirement 4: Communicate and Interact with Pre-trained AI/ML Models
The AI Test Harness shall be able to query the model with an example record (an observation from the test data set) and receive in response the model's output. The model may be hosted by the user.

Requirement 5: Native AI/ML Performance Evaluation
The AI Test Harness shall be able to evaluate AI/ML models using common performance metrics for classification and regression models.

Requirement 6: Interact with T&E Software
The AI Test Harness shall have the ability to interact with T&E software provided by the user. The T&E software must have structured inputs and outputs that are compatible with the AI Test Harness.

Requirement 7: Dashboard
The AI Test Harness shall have interactive capability for controlling the harness through a dashboard. Results shall also be presented on the dashboard.

HARNESS PROTOTYPE

The research team developed a simple AI Test Harness prototype in Python. The objective of this prototype is to demonstrate some initial functionality of an AI harness. This prototype contains the ability to (1) load a pre-trained model, (2) load a test data set, and (3) evaluate the model on the test data set using standard metrics.

- Load pre-trained model – The test harness prototype has the ability to load a pre-trained PyTorch model. At this point, the harness is specific to a PyTorch model, but the prototype could easily be extended to other standard types of models or custom models.
- Load test data set – The test harness prototype has the ability to load a PyTorch data set.
- Evaluation – The test harness prototype has the ability to evaluate the pre-trained model on the test data set using an evaluation metric function that accepts as inputs the true class labels and the predicted class labels. This format is consistent with standard classification metrics from the scikit-learn package.

RADIO FREQUENCY MACHINE LEARNING (RFML) USE CASE

This section outlines the RFML use case and application. An RF data generation tool was publicly released under this project and used throughout this use case. The CODEX tool was applied to RFML models, and uncertainty quantification for RFML models was explored.

RF DATA GEN TOOL

While there exists a host of different open-source RF dataset generation tools, the particular needs of this program's unique RFML test harness use case lent itself towards the development of a novel data generation tool. More specifically, other data generation tools include:

- LiquidDSP (https://github.com/jgaeddert/liquid-dsp) – A great tool for real-time RF digital signal processing development. However, it is not set up for AI/ML based dataset creation natively (particularly wideband spectrum) and is written in the C programming language and requires extra effort to wrap for Python development (the predominant language used for AI/ML work and this program).
- PySDR (https://github.com/777arc/PySDR) – A great Python-based codebase for learning about digital signal processing and leveraging software-defined radios. However, not set up natively for AI/ML database creation (particularly wideband spectrum).
- TorchDSP (https://github.com/torchdsp/torchsig) – A solid Python-based codebase for generating RFML datasets for the modulation recognition applications. However, it is not set up for generating wideband spectrum nor natively the image modality.

Given this test harness' interest in (a) wideband spectrum generation, (b) image modalities, and (c) a solid foundation for future open-source development, we developed the Python-based Wideband Aggregate SPectrum GENerator (PY-WASPGEN) codebase. A block diagram of the high-level functionality of PY-WASPGEN can be seen in Figure 2.

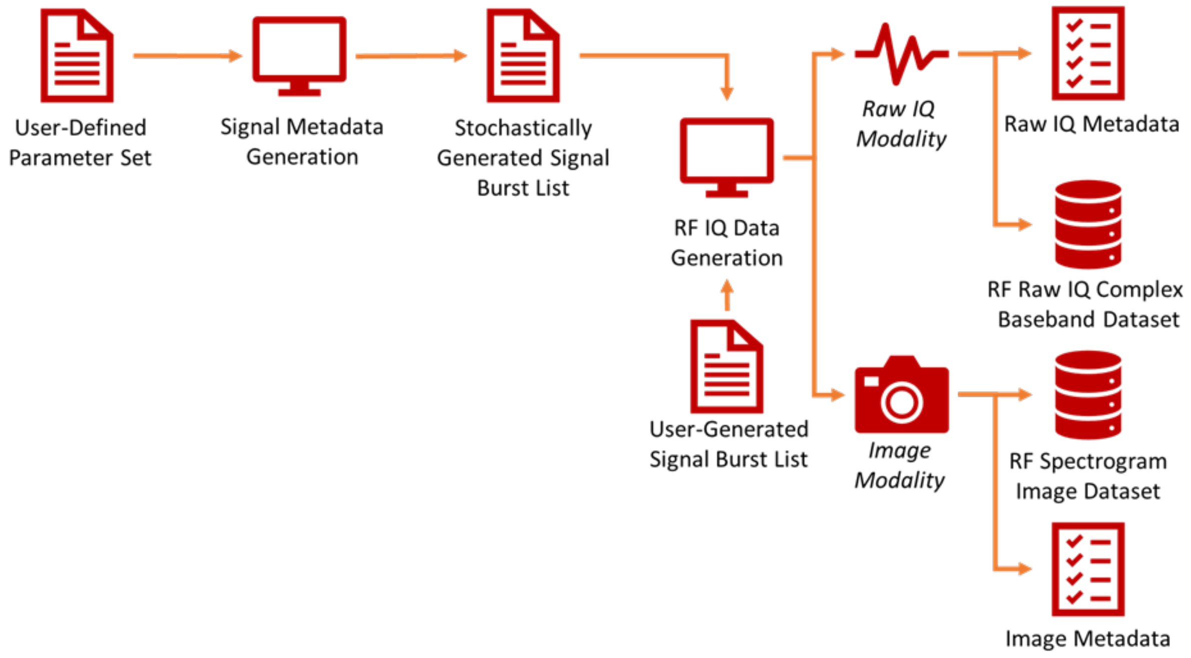

User-Defined Parameter Set
Signal Metadata Generation
Stochastically Generated Signal Burst List
RF IQ Data Generation
User-Generated Signal Burst List
Raw IQ Modality
Image Modality
Raw IQ Metadata
RF Raw IQ Complex Baseband Dataset
RF Spectrogram Image Dataset
Image Metadata

Figure 2: RFML Data Generation

The PY-WASPGEN consists of three primary stochastic data generation functions, namely:

1. Metadata Generator – Based on a user-defined configuration file, the metadata generator randomly creates “bounding boxes” for the RF data to be placed in the wideband spectrum. These bounding boxes are output in the form of burst definitions that are organized as such: [unique identifier, center frequency, bandwidth, start, duration, signal type, signal metadata]. These burst definitions can be thought of as the basic “recipes” for where and how to generate the RF data in the spectrum. A visual representation of an example set of these bounding boxes can be seen in Figure 3. *Note that the user can also generate these burst definitions manually to create desired signals without using the stochastic metadata generation process.*

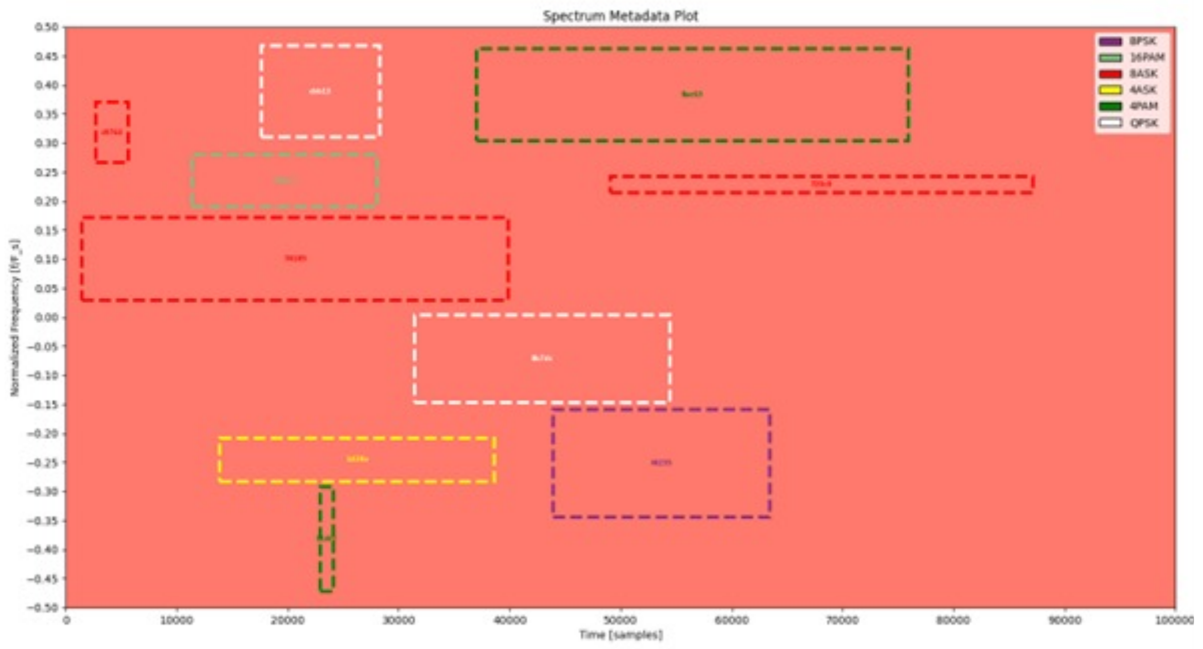


Figure 3: Example Metadata Generation Output

2. In-Phase and Quadrature (IQ) Data Generator – The IQ data generator takes in burst definitions (either created manually by the user or output from the metadata generator) and user-defined RF-specific parameters in the configuration file to create the actual IQ data. More specifically, the RF data generated by PY-WASPGEN is complex-baseband data (i.e., IQ data) as is common for RF work in simulation in order to ease memory and processing requirements (see IQ Complex Tutorial - GNU Radio for a good overview). Based on the signal type of the burst definition, additional signal metadata must be generated, or provided by the user, to generate the IQ data. For example, for most digital waveforms, a pulse shaping function needs to be defined to set the "shape" of the signal in frequency. Finally, the propagation environment for each of the signals to be generated, as well as the receiver characteristics must be provided. Note that in the current iteration of PY-WASPGEN only the receiver noise (as Additive White Gaussian Noise or AWGN) is implemented with other options to be added as future integration. The output of this generator is (a) the IQ data and (b) an updated set of burst definitions with the generated signal specific metadata appended. A visual representation of the spectrogram (Spectrogram - Wikipedia) of an example output IQ data generated can be seen in Figure 4.

   - The IQ modality provided by the IQ data generator is particularly useful for testing and evaluating (a) signal isolation algorithms, and (b) signal identification and classification problems. Given that the IQ data is the complete representation of the underlying signals, as opposed to the image modality which losses information such as phase information, it is much more useful when trying to solve these harder problems that require more features of the underlying signals. Note however, in practice, typically the signals must be isolated individually from the wideband data and processed individually to constrain the amount of data and ML-architecture sizes and training times to practical sizes.

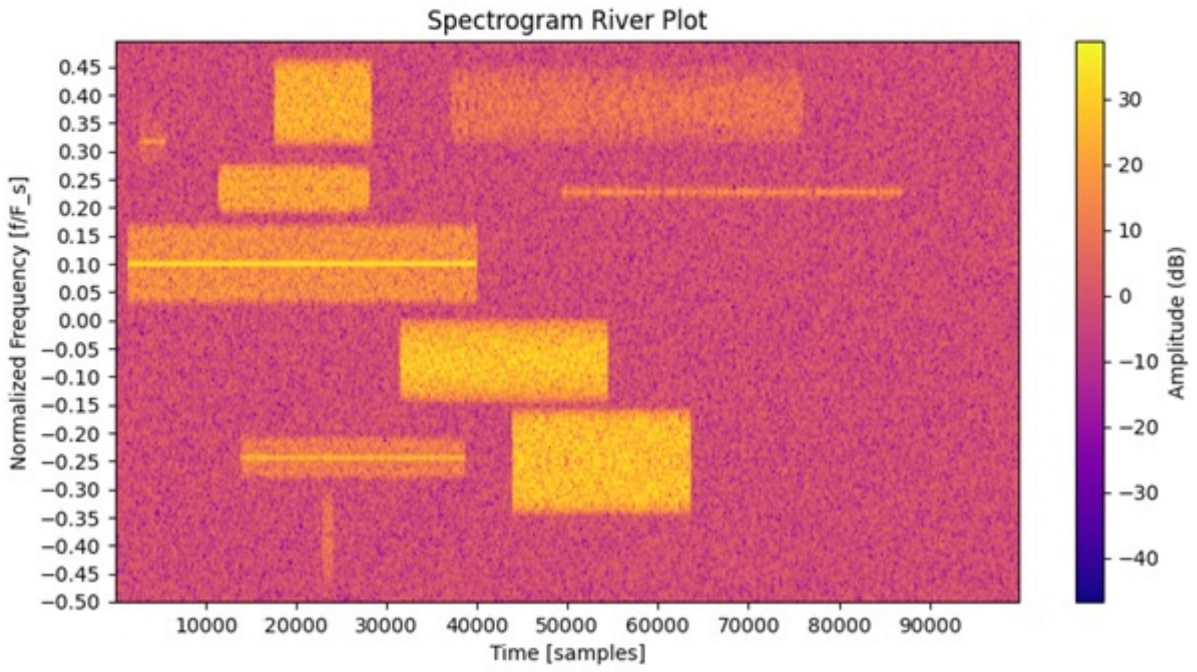


Figure 4: Example IQ Generation Output (as a spectrogram)

3. Image Data Generator – The image generator simply takes the IQ data and burst definitions output from the IQ data generator in order to create a spectrogram image of the data and the bounding box dimensions (or ground truth) of the spectrogram for labeling purposes. Figure 5 shows an example of the spectrogram with overlaid ground truth bounding boxes.

   - The image modality provided by the image data generator is particularly useful for testing and evaluating RFML approaches that aim to perform signal detection in wideband IQ data. More specifically, utilizing spectrogram images within object detection ML-algorithms (such as THU-MIG/yolov10: YOLOv10: Real-Time End-to-End Object Detection (github.com)) is a popular approach to finding the bounds of signals (i.e., the frequency and time extent of a signal) in a congested environment, and is usually done as a precursor before more exquisite processing on each individually detected signal.

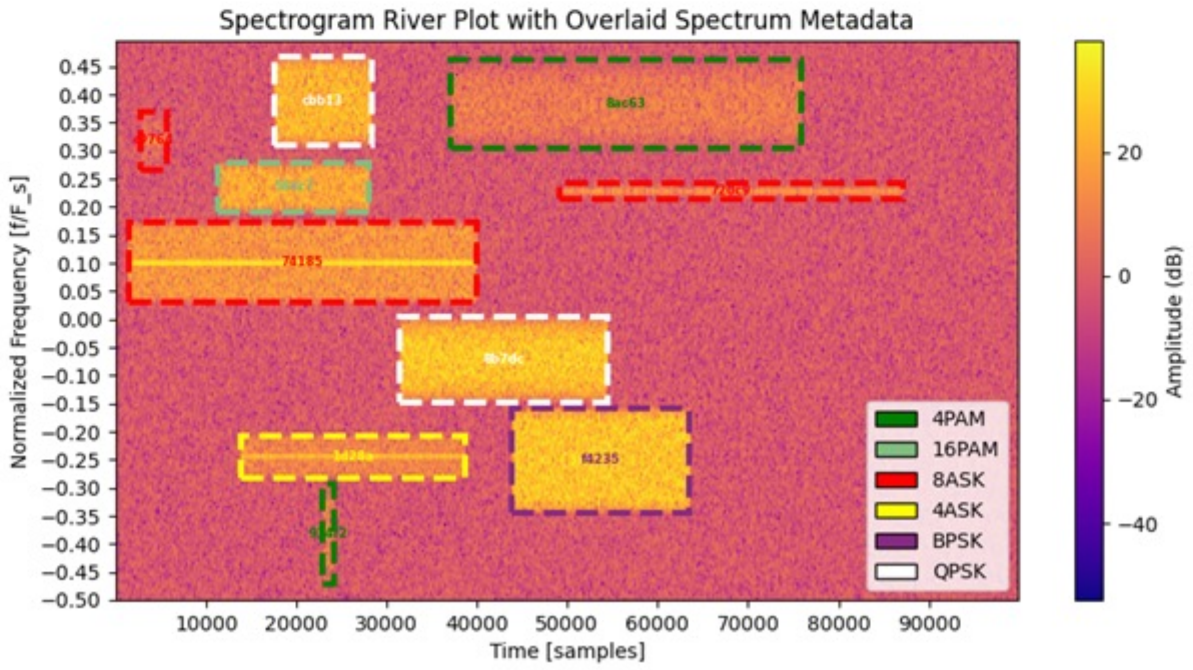


Figure 5: Example Image Generator Output (Spectrogram with Overlaid Bounding Box Ground Truth)

CODEX DEMONSTRATION ON RFML USE

CODEX is a Python package that implements the combinatorial coverage (CC) and set difference combinatorial coverage (SDCC) metrics for machine learning datasets [Lanus et al (2021)], see AIRC WRT-1070 final report for more details on the CODEX project [Freeman et al (2024)]. Ahead CODEX's planned public release, its viability as a testing suite for various datasets and data domains is extended through its application to generated RFML datasets. Tools and metrics offered by CODEX are applicable to tables of metadata containing properties of each generated sample, including center frequency, bandwidth, signal-to-noise ratio, and signal duration, which act as a surrogate for RF IQ data generated. Many concerns in dataset coverage, balance, and dataset characterization are also present in the RF domain too.

The dataset evaluation mode computes combinatorial coverage of a dataset for a defined universe. CC is the proportion of appearing t-way interactions out of all possible interactions for that defined universe for a given t. In whatever environment that a model operating on RF data is deployed in, the dataset on which it trains should ideally cover the whole space not only for features and levels, but interactions between those features as well. With a defined universe, dataset evaluation aims to measure a dataset for completeness.

Consider an environment in which RF signals are each expected to have a center frequency between [-0.1, 0.1], bandwidth between [0.1, 0.5] and signal-to-noise ratio (SNR) between [5, 25]. A universe can be defined in these ranges to bound and describe the operating environment. Then consider a low-coverage dataset $\mathcal{A}$ consisting of samples generated from subsets of ranges of each generation parameter, and dataset $\mathcal{B}$ consisting of samples generated from the full ranges of values in the universe.

Table 1: Dataset Parameters for CODEX RFML Example

Dataset ID	*Number of Samples*	*Center Frequency Range*	*Bandwidth Range*	*SNR Range*	*Train-val-test-split*
$\mathcal{A}$	1,100,000	[-0.05,0.05]	[0.1,0.2]	[5,20]	N/A
$\mathcal{B}$ (full range)	1,100,000	[-0.1,0.1]	[0.1,0.5]	[5,25]	N/A
$\mathcal{C}$	1,100,000	[0.05, 0.1]	[0.325, 0.5]	[15,25]	0.81-0.9-0.9
$\mathcal{D}$	1,100,000	[-0.05, 0.05)	[0.1,0.275]	[5,15)	0.81-0.09-0.09

As a result of the different RF generation methods, CODEX's dataset evaluation can be used to numerically and visually characterize these datasets for completeness. With the defined universe and a dataset, dataset evaluation results can be obtained. The binary coverage plot in Figure 6 demonstrates that $\mathcal{A}$ is missing portions of the defined universe for the 2-way interaction level even with the same number of samples. CODEX results specify the missing interaction for each value of t in the dataset as well. Meanwhile, $\mathcal{B}$ has every 2-way interaction present in the dataset. In this example, $\mathcal{A}$ has CC value of 0.175 (Figure 6), while $\mathcal{B}$ has a full CC value of 1.0 at the t=2 level (Figure 7).

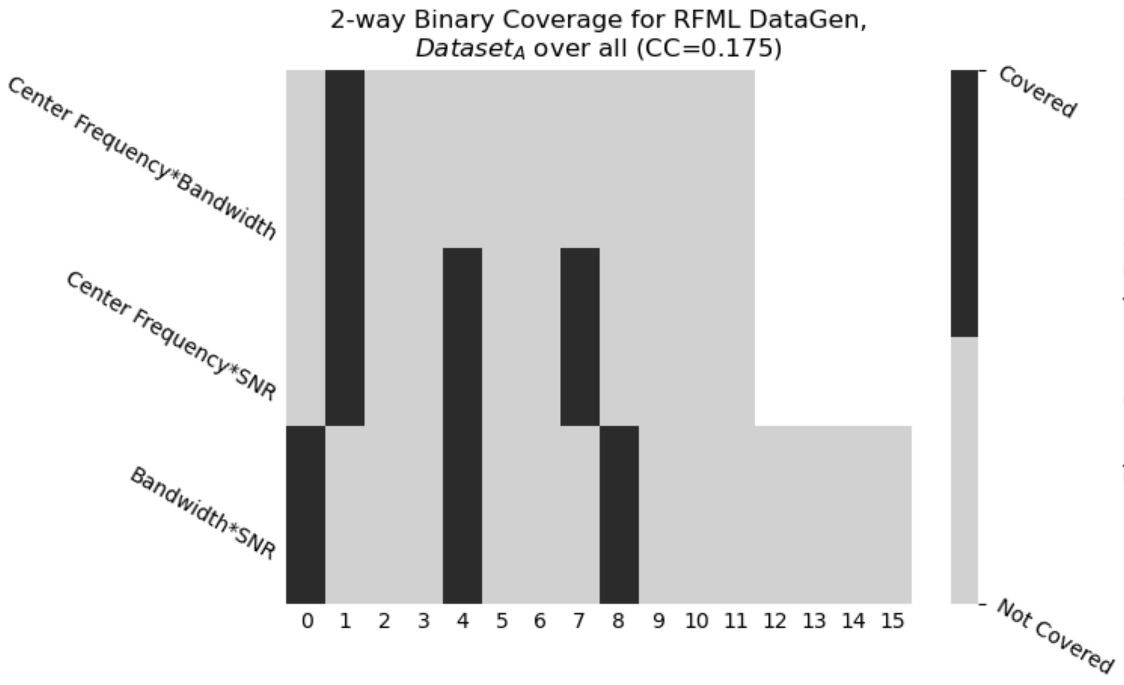


Figure 6: Binary coverage map of 2-way interactions for $\mathcal{A}$ with 0.175 CC. $\mathcal{A}$ was constructed with a modified range of possible center frequncy values.

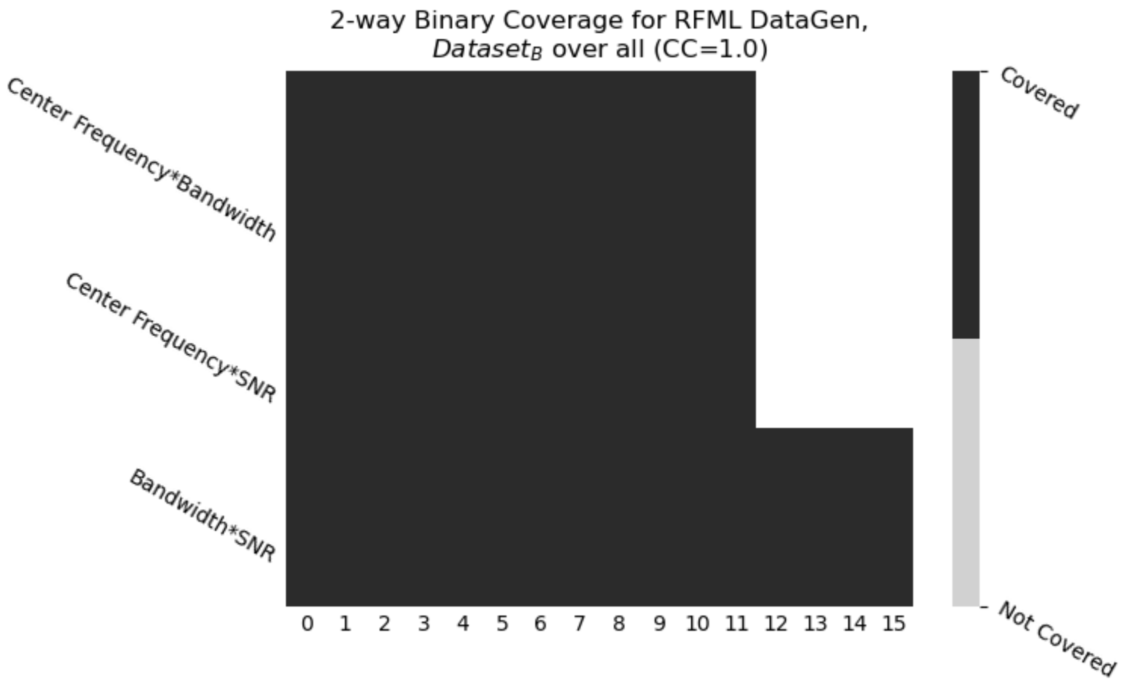


Figure 7: Binary coverage map of 2-way interactions for $\mathcal{B}$ with 1.0 CC. $\mathcal{B}$ was constructed with the full range of possible values.

The intersection of RF and ML would mean that dataset splits and how they are constructed are part of the test and evaluation process. While random sampling prevents samples from appearing in more than one split, there is no guarantee of how the t-way interactions appearing in the dataset are distributed between sets of data or how to characterize the differences between datasets. Dataset split evaluation, unlike dataset evaluation, computes set difference combinatorial coverage between splits of data with respect to a defined universe for a given t. SDCC between sets T and S is the proportion of interactions in one set, T, do not appear in another, S, notated as $T - S$.

Consider two datasets constructed from regions of the RF space that are not only disjoint in their samples, but in all 2-way interactions that appear in them. These two datasets, $\mathcal{C}$ and $\mathcal{D}$,are dissimilar. With the same defined universe, a dataset, and a split designation, dataset split evaluation results can be obtained.

From these results on RF-generated data for 2-way interactions, the user can identify differences in the appearance of interactions between the two sets. The datasets can be split into train and test sets designated using subscripts $\mathcal{C} = \{\mathcal{C}_{train}, \mathcal{C}_{test}\}$ and $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{test}\}$. Numerical results in the computed SDCC between $\mathcal{C}_{test} \backslash \mathcal{C}_{train}$ (Experiment 1) and between $\mathcal{D}_{test} \backslash \mathcal{C}_{train}$ (Experiment 2), displayed in Figure 8 and Figure 9 suggest that $\mathcal{D}_{test}$ is far from $\mathcal{C}_{train}$ (SDCC = 1), while $\mathcal{C}_{test}$ is close to $\mathcal{D}_{train}$ (SDCC = 0). This difference in SDCC is expected from how these sets were constructed. As can be seen in Figure 9, every single interaction in $\mathcal{C}_{train}$ also appears in $\mathcal{C}_{test}$, while no interaction in $\mathcal{D}_{test}$ appears in $\mathcal{C}_{train}$. This is confirmed in Figure 10, whose CC plot of $\mathcal{D}_{test}$ shows that each interaction exists in the set difference. CODEX's characterization and enumeration of the set differences between splits can allow the user to construct test sets that are representative of the trained environment - to test for adequate learning - or test sets that are challenging - to test for model robustness.

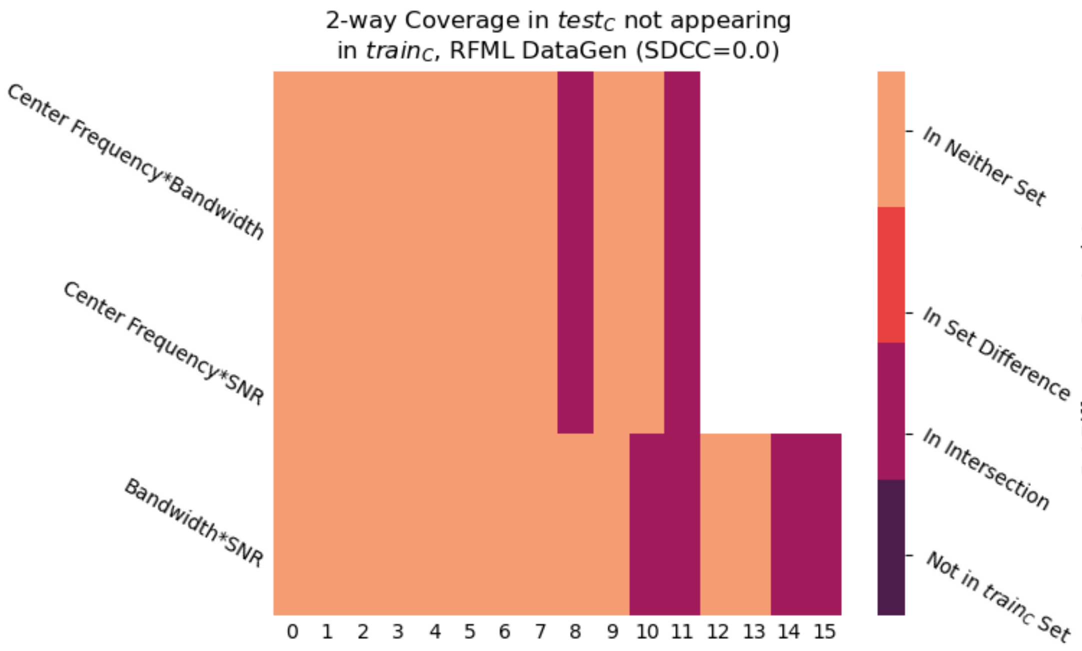


Figure 8: Set difference combinatorial coverage map for 2-way interactions for Experiment 1. Note this plot indicates that there are no interactions in the set difference.

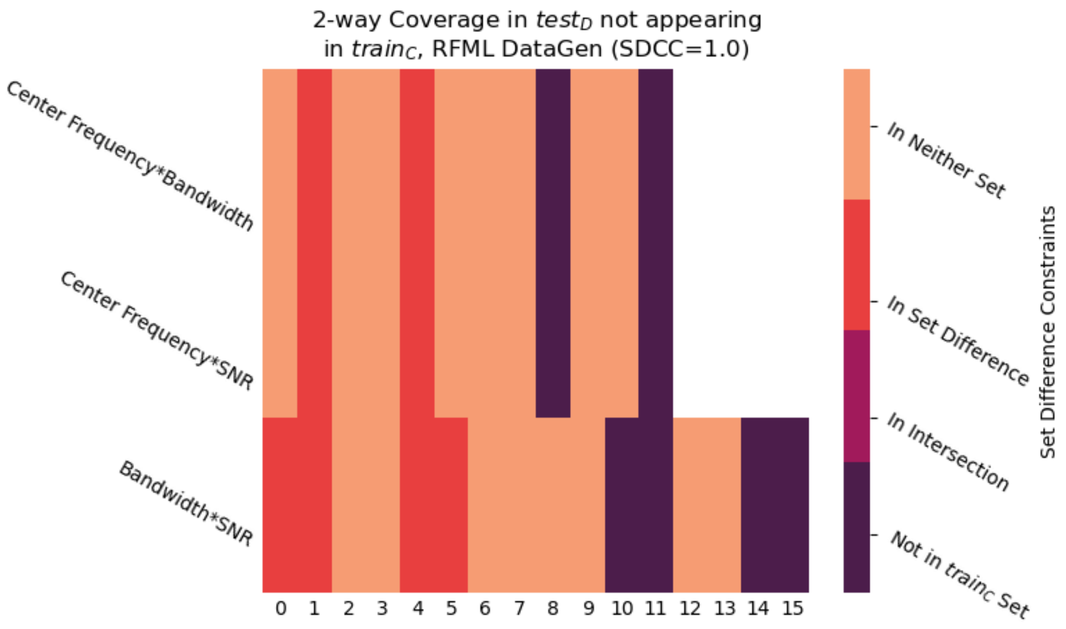


Figure 9: Set difference combinatorial coverage map for 2-way interactions Experiment 2, which was constructed in a different region of the dataset. Note this plot indicates that there are interactions in the set difference.

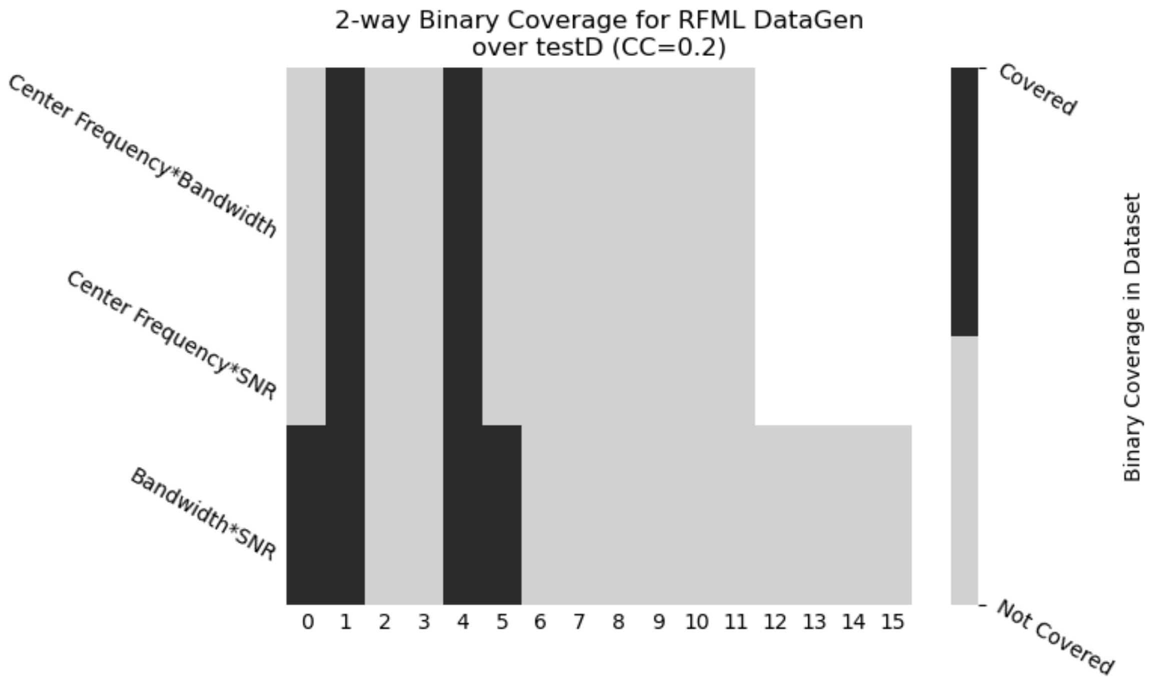


Figure 10: 2-way coverage plot of the test set for $\mathcal{D}$.

CODEX can also use user-generated performance files of a model on the splits to uncover any correlation between SDCC values between splits and model performance with a related dataset split comparison mode.

From these results plotting SDCC against performance, the user can further investigate a correlation between SDCC and performance. Figure 11 suggests that a model that encounters signals dissimilar from the operating envelope on which it was trained will see degraded model performance, given that the universe accurately reflects the real-life environment of possible RF signals to be encountered.

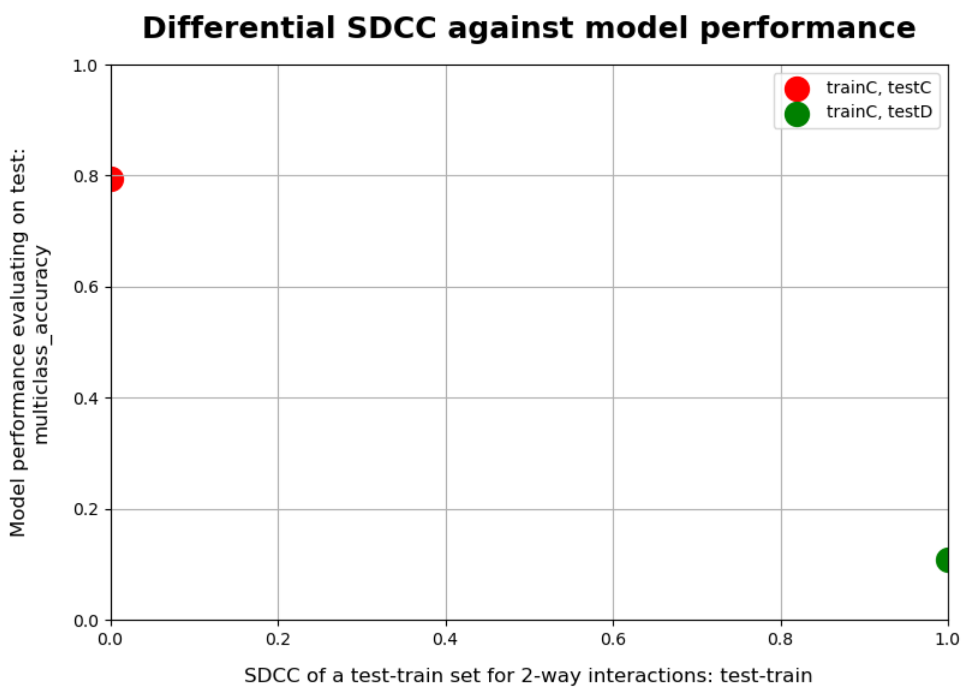


Figure 11: Differential performance of a model trained on $\mathcal{C}_{train}$ and tested on $\mathcal{C}_{test}$ (low distance between splits) vs one trained on $\mathcal{C}_{train}$ and tested on $\mathcal{D}_{test}$ (high distance between splits).

CODEX was demonstrated live at the Combinatorial Testing for AI-Enabled Systems workshop hosted at Virginia Tech's Arlington Research Center on September 4, 2024. Dr. Jaganmohan Chandrasekaran and Dr. Erin Lanus from the AIRC research team organized the event. They presented research on applying combinatorial testing to machine learning. Mr. Brian Lee developed and delivered the Jupyter Notebook for the CODEX live demonstration. Collaborators from the National Institute of Standards and Technology (NIST), Mr. Rick Kuhn, Dr. Raghu Kacker, and Dr. M S Raunak, presented material on the software foundations of combinatorial testing. Dr. Laura Freeman gave the welcome address. The workshop had 29 registrants with 21 attending from 14 organizations across government, federally funded research centers, industry, and academia. One outcome from the workshop was the creation of a repository of research hosted on the NIST website: https://csrc.nist.gov/Projects/combinatorial-testing-for-ai-enabled-systems.

UNCERTAINTY QUANTIFICATION (UQ) FOR RFML

For general multi-class classification problems, uncertainty quantification (UQ) attempts to model the predictive probability

$$p_i = \mathbb{P}(y = c|x, \theta),$$

where $x \in \mathbb{R}^d$, $y \in \{1, \dots, C\}$, and θ represents the model parameters. Ideally, the predictive probability is high (e.g., close to 1) for the true class and low for all other classes. However, there may be examples that are difficult to classify do to several reasons including increased noise and lying near the decision boundary. In those situations, the predictive probability for the true class should decrease and the uncertainty of the prediction should increase, i.e., the model should be less confident in its predictions.

There are some models, such as logistic regression, that explicitly model the predictive probability. There are other models, such as the random forest and other ensemble techniques, that can estimate this probability by combining the predictions from the individual models in the ensemble or through Monte Carlo (MC) simulation. For deep neural networks, the output of the softmax function often used as the final layer is sometimes interpreted as p_i. However, the training process can cause deep nets to be overly confident in incorrect predictions. For example, the confidence in a prediction does not decrease on hard observations to classify but switches to the incorrect class with high confidence.

There are several classes of approaches to UQ for deep neural networks, with a few outlined below:

- MC dropout [Gal & Ghahramani (2016)] – Use dropout layers during inference and MC sampling to produce M softmax outputs. Use the mean of the MC estimates as the predictive probability.

- Ensemble neural networks [Lakshminarayanan, Pritzel, & Blundell (2017)] – Use the mean of the predictive probabilities of the individual models in the ensemble as an estimate of the predictive probability.

- Bayesian neural networks [Blundel et al (2015)] – Use MC simulation to draw samples from the weight distributions learned by the Bayesian approach and use the mean as the predictive probability.

This study uses a 4-class RF modulation classification problem as an exemplar [Clark et al (2019)(2)]. The four classes are 16-ASK, 16-PAM, 16-PSK, and 16-QAM. All the classes are from different modulation families but have the same order. During training, 12000 observations (3000 of each modulation type) with a sequence length of 256 are generated using the PY-WASPGEN software. The center frequency is 0, the bandwidth is 0.5, and the SNR is 10.

The base architecture for the neural network is composed of two 1-dimension convolutional layers and two fully connected linear layers. Each convolutional layer is followed by a rectified linear unit (ReLU) activation function and max pooling. Each linear layer has a ReLU activation function. The softmax function is used as the final layer. For the dropout model, dropout layers with a dropout rate of 0.1 are added after each convolutional layer. The ensemble trains 10 models using the base architecture described above. The Bayesian model uses the base architecture but adds Gaussian priors to the model weights. Each model (excluding the dropout model which uses the standard convolutional neural network (CNN) and adds dropout after training) is trained for 1000 epochs with a batch size of 256 and a learning rate of 0.0001.

Figure 12 displays the confusion matrices for the base CNN, Figure 13 displays confusion matrix for the ensemble method, and Figure 14 displays the confusion matrix for the Bayesian model. The Bayesian model has difficulty distinguishing between 16-PSK and 16-QAM.

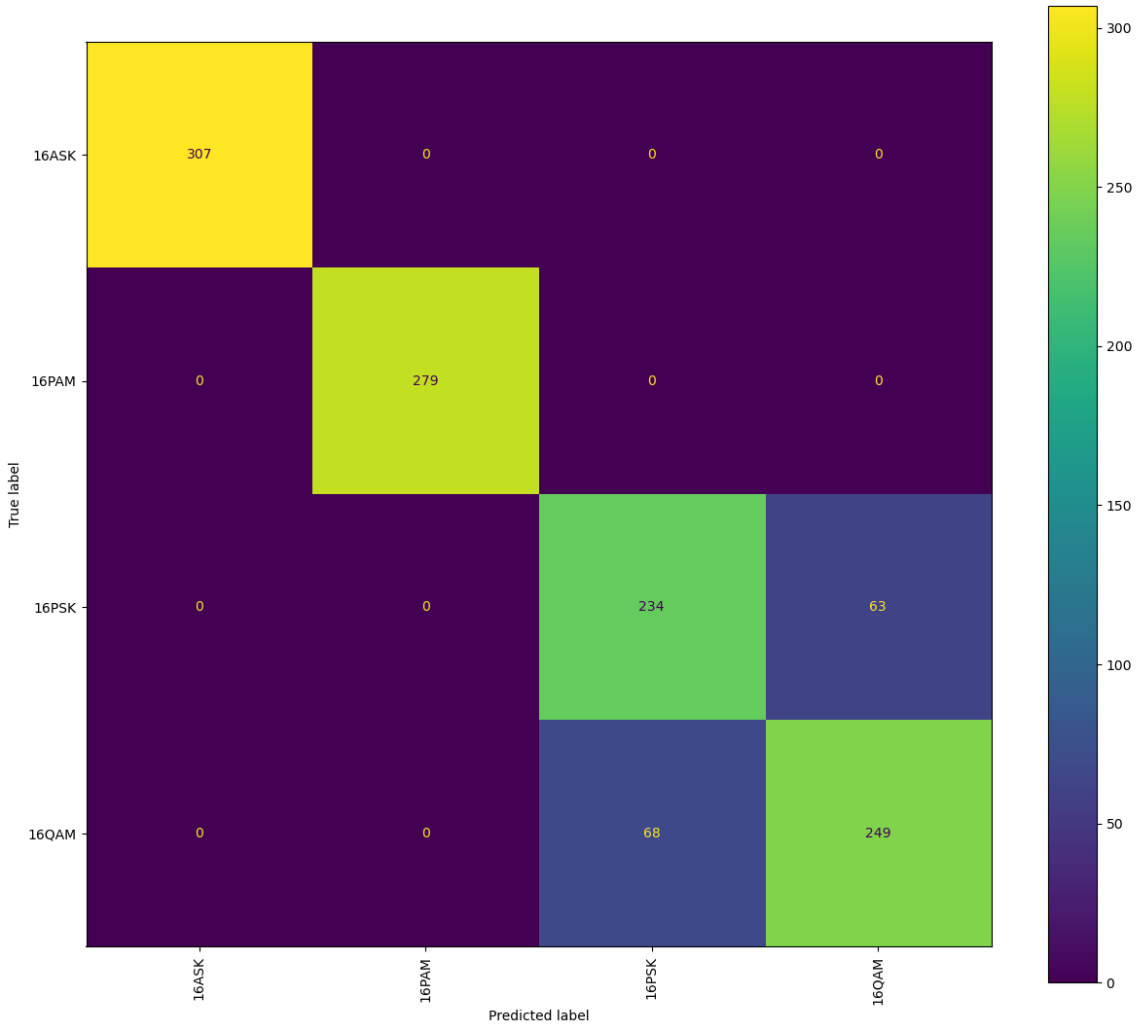


Figure 12: Confusion Matrix for CNN Model

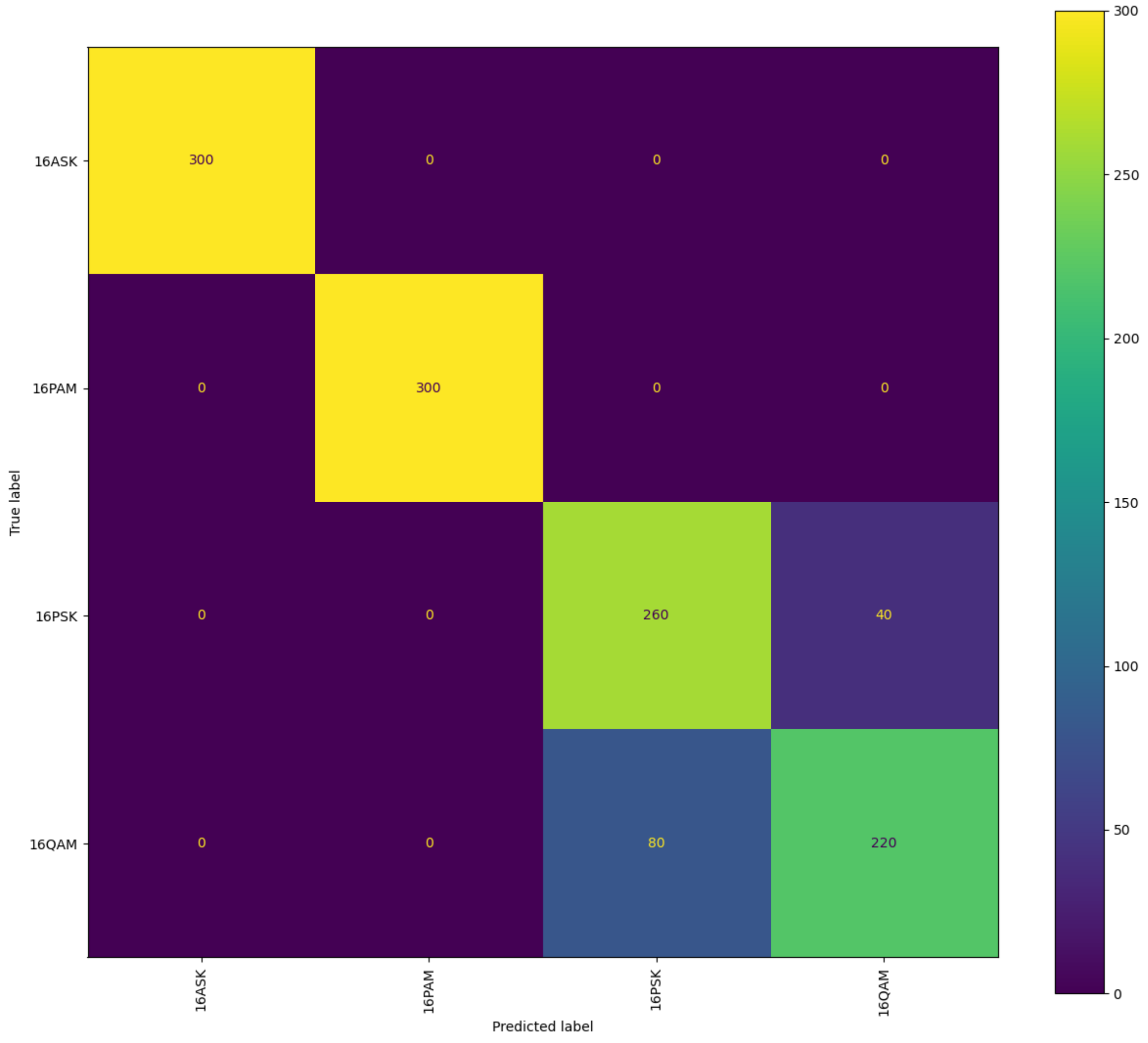


Figure 13: Confusion matrix for ensemble model

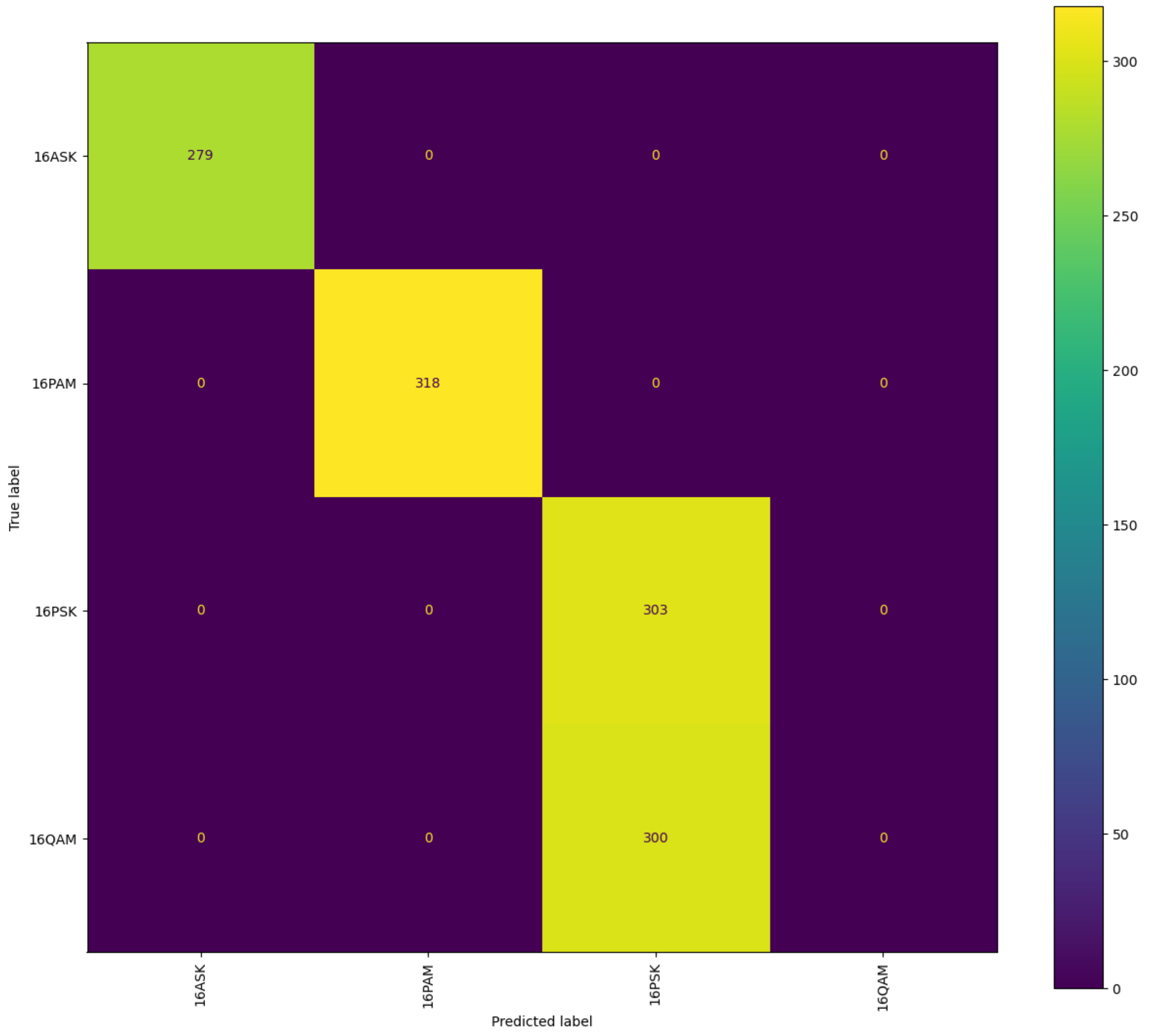


Figure 14: Confusion matrix for Bayesian model

The evaluation of the UQ methods utilizes domain shift to sequentially create test data sets that move further from the training set. Specifically, the SNR is sequentially decreased from 10 db to -10 db. The center frequency is shifted from 0.0 to 0.01. Techniques that adequately estimate the uncertainty should shift from very confident in the true class to less confident as the distance between the training and test data increases.

The distance between the distribution of the training and test sets can be quantified using the concept of transfer distance [Cody, Adams, & Beling (2022)]. The transfer distance is calculated on features extracted from the signals [Wong, McPherson, & Michaels (2022)]. For this study, Euclidean distance is used.

Figure 15 displays the softmax output for the CNN for the true class for a test set of 2000 signals at SNR=10, Figure 16 displays the softmax for SNR=5, Figure 17 displays the softmax for SNR=0, Figure 18 displays the softmax for SNR=-5, and Figure 19 displays the softmax for SNR=-10. When the SNR of the test set is 10, the softmax of the output is near 1 for a majority of the observations and the accuracy of the model is high. As the SNR decreases, the output of the softmax for the true class abruptly shifts towards 0 and the accuracy of the model decreases as expected. This model does not offer a smooth decrease in the confidence as the test data shifts away from the training data.

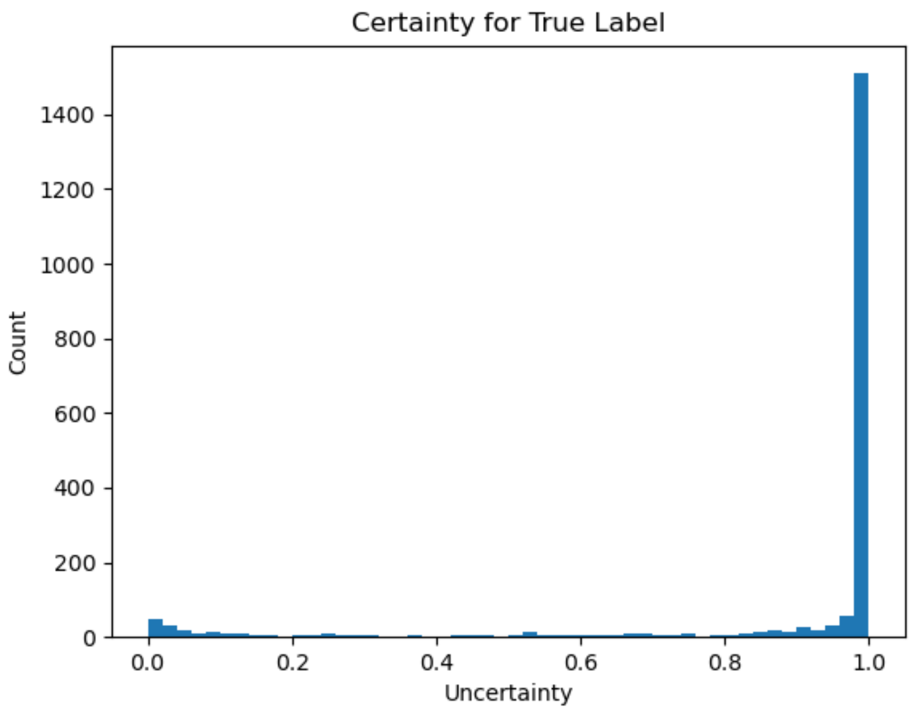


Figure 15: Softmax output for the true class using the CNN model and SNR=10.

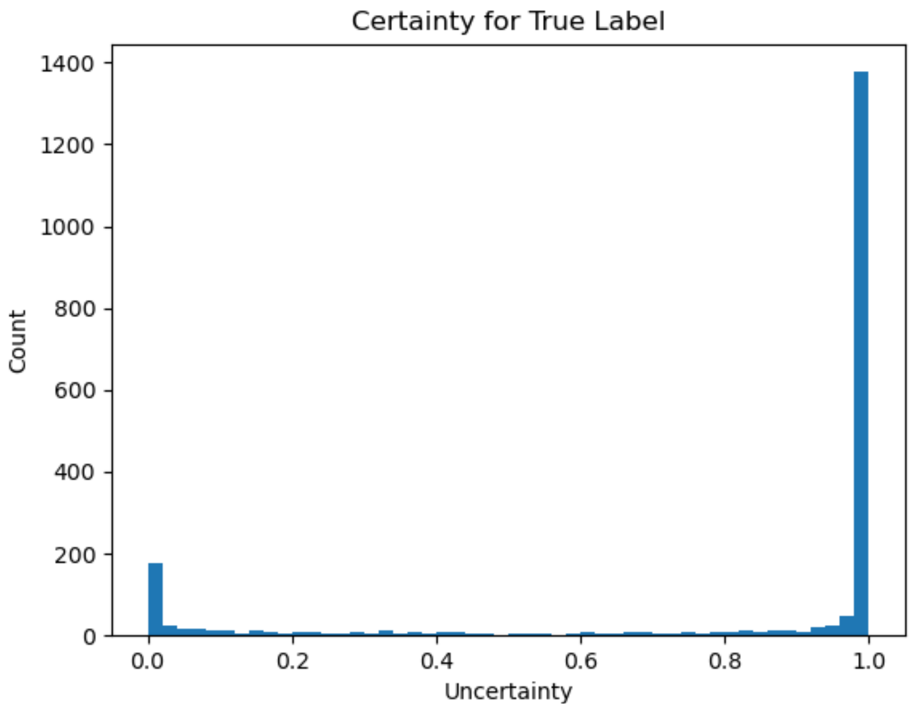


Figure 16: Softmax output for the true class using the CNN model and SNR=5.

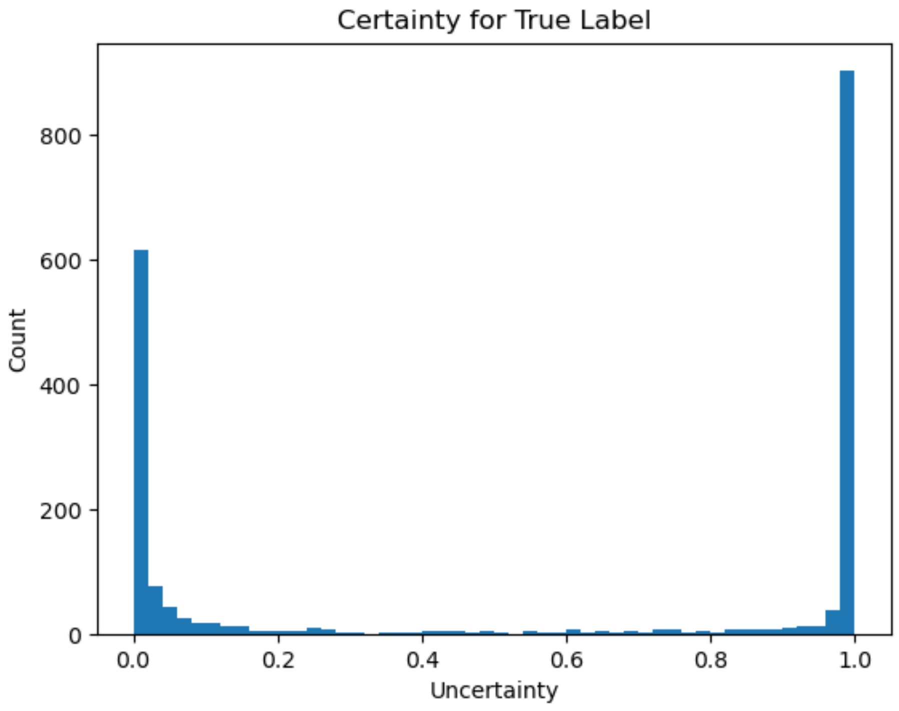


Figure 17: Softmax output for the true class using the CNN model and SNR=0.

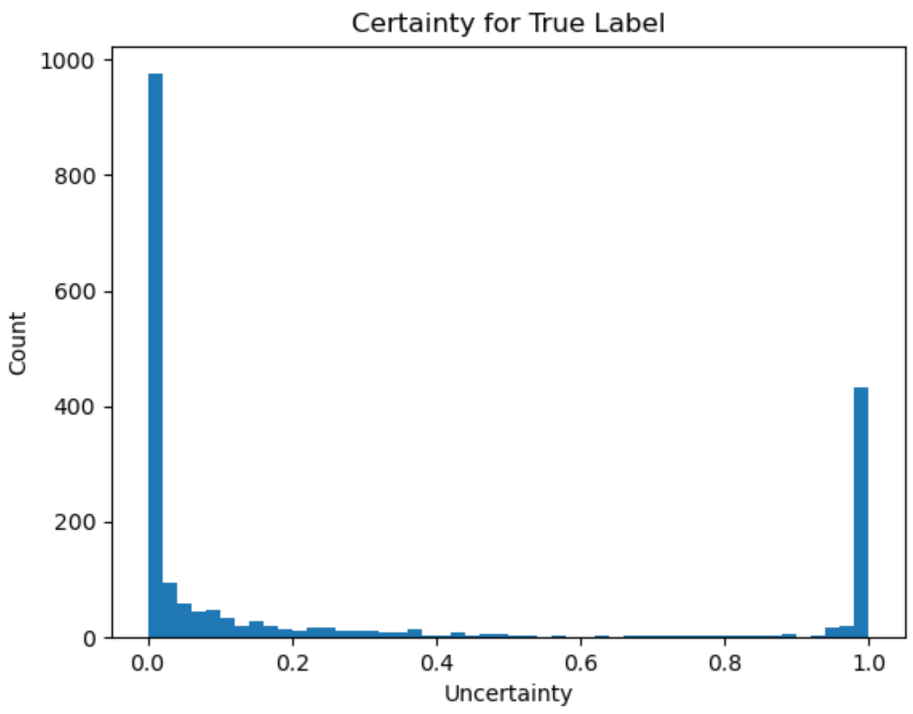


Figure 18: Softmax output for the true class using the CNN model and SNR=-5.

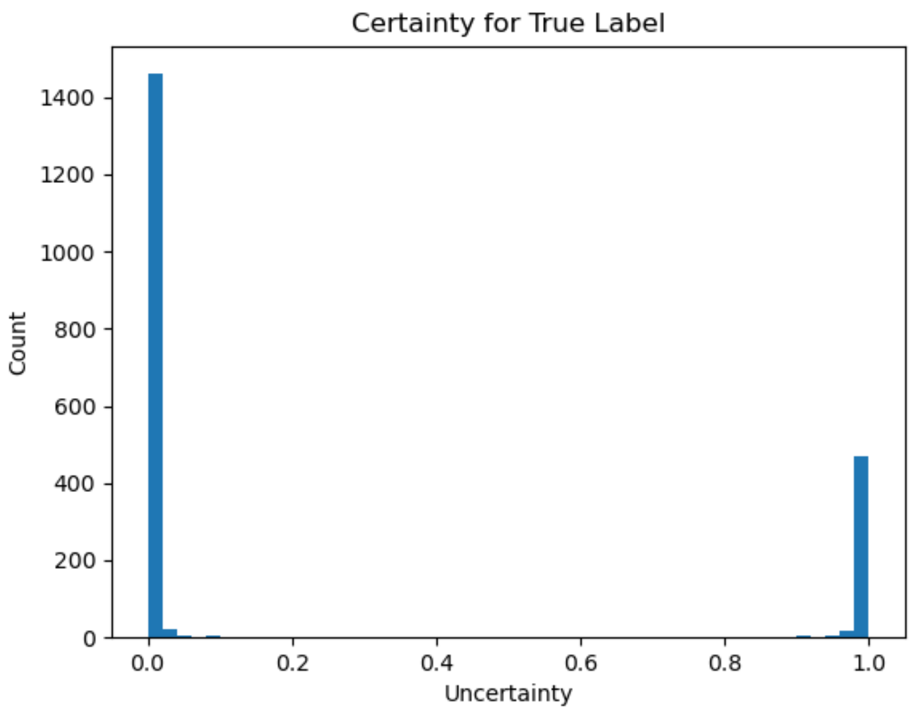


Figure 19: Softmax output for the true class using the CNN model and SNR=-10.

Figure 20 displays the softmax output for the ensemble for the true class for a test set of 2000 signals at SNR=10, Figure 21 displays the softmax for SNR=5, Figure 22 displays the softmax for SNR=0, Figure 23 displays the softmax for SNR=-5, and Figure 24 displays the softmax for SNR=-10. The confidence when the SNR is 10 is lower but this model offers a smoother transition to lower confidence estimates as the test set moves away from the training set. The dropout and Bayesian models have similar qualities.

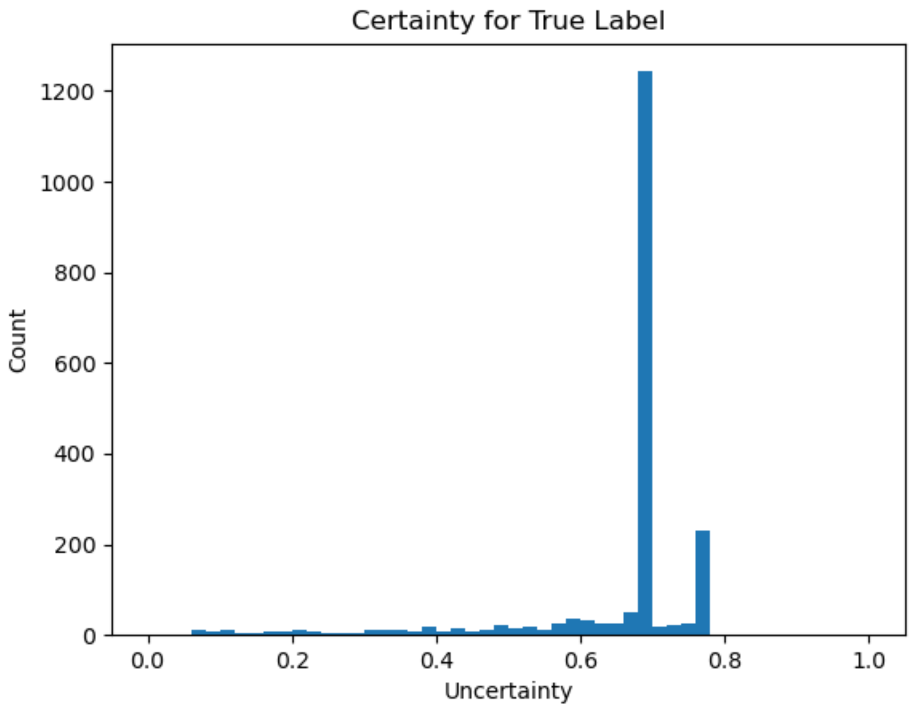


Figure 20: Softmax output for the true class using the Ensemble model and SNR=10.

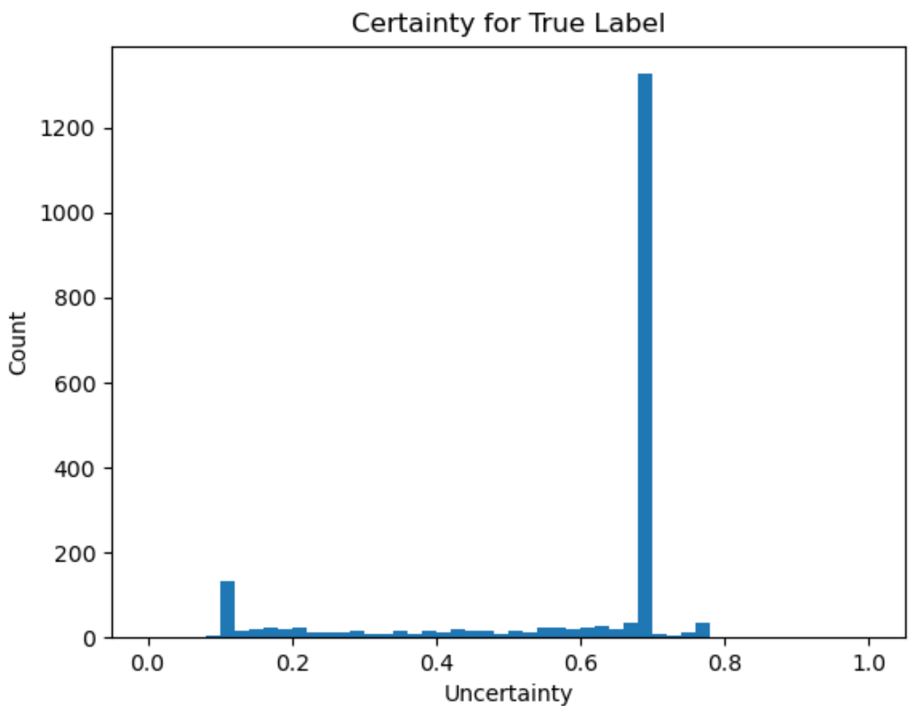


Figure 21: Softmax output for the true class using the Ensemble model and SNR=5.

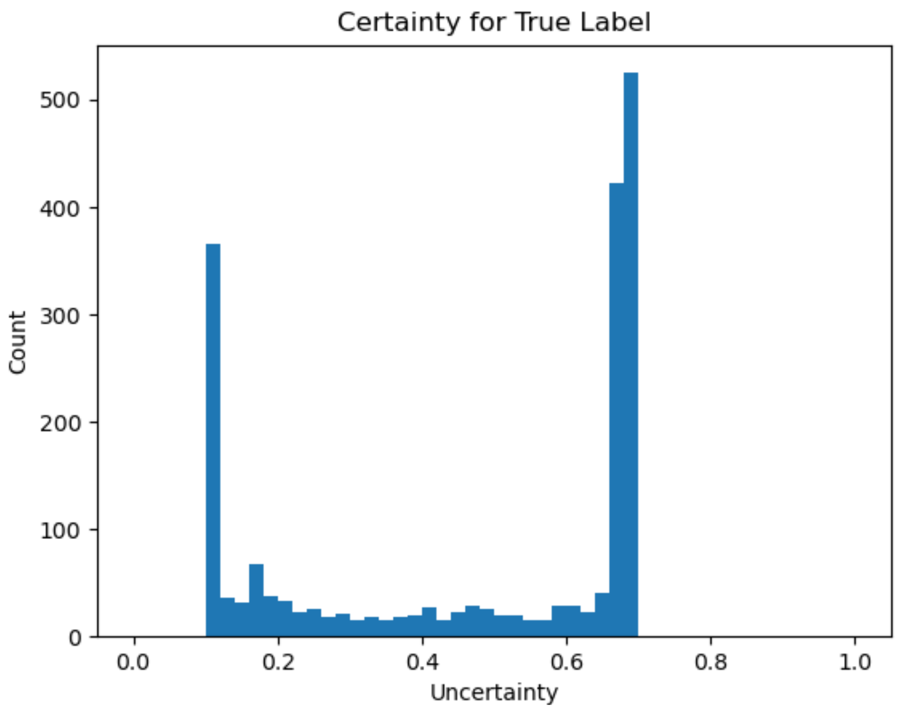


Figure 22: Softmax output for the true class using the Ensemble model and SNR=0.

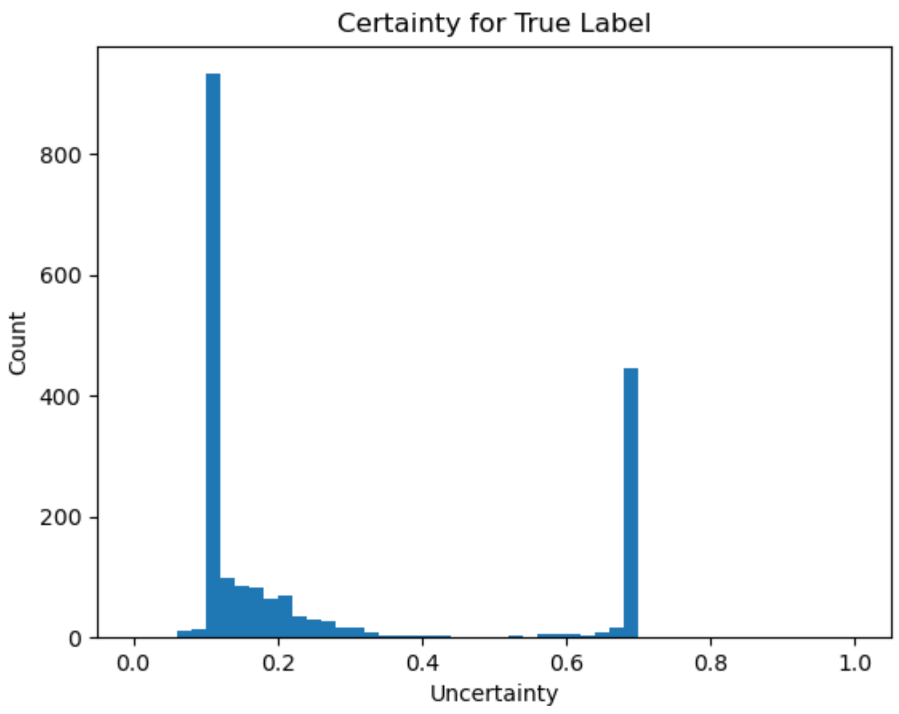


Figure 23: Softmax output for the true class using the Ensemble model and SNR=-5.

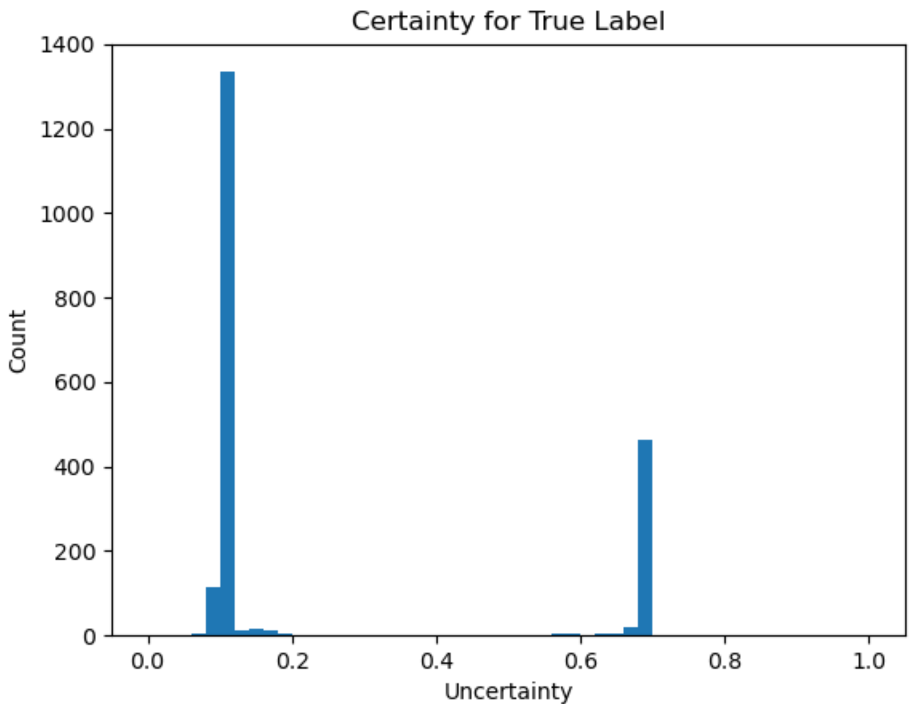


Figure 24: Softmax output for the true class using the Ensemble model and SNR=-10.

Figure 25 displays the transfer distance for each feature. As the SNR is decreased, the transfer distance between the training and test sets increases.

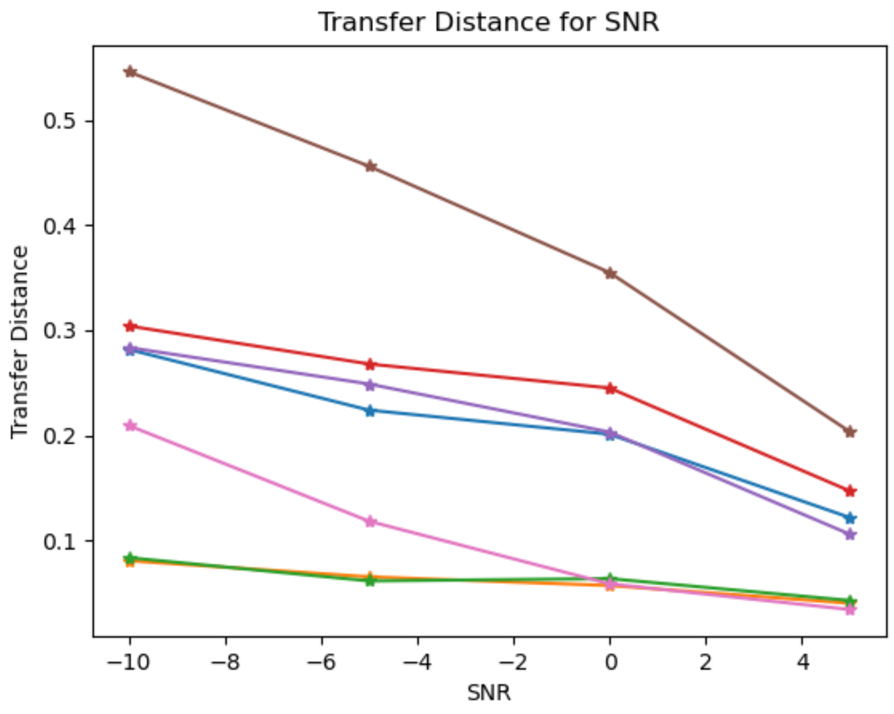


Figure 25: Transfer distance for each feature for the Ensemble model.

LARGE LANGUAGE MODEL (LLM) USE CASE

The research team completed the survey of academic literature and industry best practices for T&E of LLMs. A summary of findings was compiled to enhance the usability of the LLM Test Harness by including introductory, background, and explanatory text that will help the user understand the important aspects of testing LLMs, the necessary steps, and the reasons for those steps. The research team also developed a T&E process for LLMs.

LLM TESTING FRAMEWORK

A test harness enables a thorough evaluation of a software component or system in a controlled environment. While the objective of a test harness for a traditional software system and a text-to-text LLM is the same, to facilitate a systematic and comprehensive assessment of the system under test, a LLM test harness differs from a traditional test harness in many aspects. The aspects are listed in Figure 26 below and summarized in the following section.

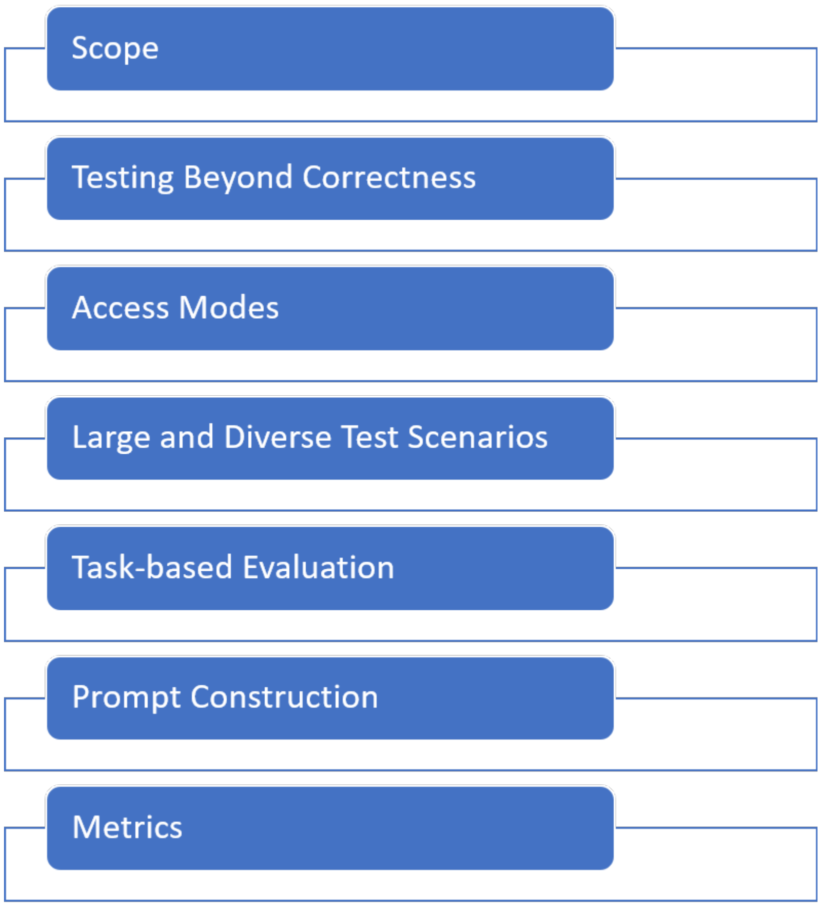


Figure 26: Aspects of a LLM test harness that differ from a traditional test harness.

Scope
Unlike a traditional test harness, which primarily focuses on assessing the functional correctness of a deterministic software system with pre-defined test inputs and outputs, a LLM test harness has a broader scope. Given that the users interact with a LLM using a natural language (e.g., English), the test harness must support evaluating LLM's understanding, reasoning, and generating capabilities. In other words, the test harness should support evaluating LLM's linguistic capabilities. Furthermore, the test harness shall help evaluate whether a LLM performs logical and common-sense reasoning and generates correct, coherent, and contextually relevant responses.

Testing Beyond Correctness
Given a LLM is a data-intensive system and derives its decision logic from the data it is trained with, the test harness must have the ability to uncover any potential bias or discriminative behavior that the LLM might exhibit by reflecting or amplifying the underlying bias from the dataset it was trained with. The test harness must also possess capabilities to evaluate the LLM's robustness to adversarial attacks, including the LLM's ability to withstand malicious attacks that aim to exploit its vulnerabilities and manipulate its behavior. Furthermore, a LLM test harness must be able to perform additional assessments such as that the LLM is safe to operate, does not reveal sensitive information (data privacy), and produces outcomes that are reliable, trustworthy, and well within the established ethical and moral standards.

Access Modes
Pre-trained LLMs, which the test harness is designed to evaluate, are distributed using different methods. Two common distribution methods are:

- **Host LLMs locally:** Pre-trained LLMs can be downloaded and run locally, for example, Llama2 [Touvron et al (2023)]. This access mode requires significant computational resources for loading and inference of LLMs.

- **API-based Access:** Access to pre-trained LLMs is facilitated via an Application Programming Interface (API) (e.g., OpenAI's GPT3.5 Turbo). This method does not necessitate specific hardware requirements (as the model is not hosted locally), and the connection to the LLM is established using an API key. However, the evaluation is contingent upon stable internet connectivity.

The test harness must be designed to support the evaluation of LLMs distributed using both access modes.

Large and Diverse Test Scenarios
The ability of a LLM to perform a diverse set of tasks with varying degrees of complexity increases the difficulty and variability in generating a wide range of test inputs that encapsulate all possible scenarios. Therefore, the test harness must support a broad and diverse set of test scenarios to comprehensively evaluate the LLM's versatility. Additionally, it must support the evaluation of LLM's non-deterministic, evolving behavior.

Task-based Evaluation

Current best practices for comprehensive LLM evaluation involve assessing its performance across various tasks, such as reasoning, classification, natural language inference, semantic understanding, sentiment analysis, robustness, and other domain-specific tasks (e.g., code generation in software engineering) [Chang et al (2024); Guo et al (2023); Sun et al (2024)]. These tasks can span from simple text generation to complex reasoning and problem-solving. The test harness must provide the ability to perform LLM evaluation across a wide range of tasks.

After selecting a task for evaluation, the next step involves choosing the appropriate test data. Standardized, publicly available datasets are commonly used for evaluating LLMs on chosen tasks. Alternatively, custom datasets may be employed for tailored evaluations. For example, to evaluate the summarization capabilities of a LLM, standardized datasets such as XSUM [Narayan, Cohen, & Lapata (2018)] can be used, or custom datasets can be created for more specific evaluations. The test harness must support downloading and using publicly available datasets, typically hosted on platforms such as Hugging Face, as well as loading and using custom datasets.

Prompt Construction

User interaction with LLMs primarily involves natural language prompts, encompassing instructions on desired behavior, actions to be performed, response style, and any relevant constraints. The phrasing and structure of prompts significantly influence the LLM's output. This diverges from a traditional test harness, where a test input yields a predictable output. Thus, a LLM test harness not only requires the ability to evaluate various test scenarios but also facilitate the creation and refinement of prompts tailored to LLM evaluation.

Metrics

A LLM's performance is evaluated holistically across its entire test suite rather than on a per-test-case basis, as in the case of traditional software systems. The metrics for assessing LLM performance vary based on the specific capabilities being tested. Accuracy, precision, recall, and F1-score are some of the widely used metrics for classification tasks [Hu & Zhou (2024)]. For natural language generation, token-similarity metrics like Recall-Oriented Understudy for Gisting Evaluation (ROUGE), Bilingual Evaluation Study (BLEU), Bidirectional Encoder Representations from Transformer Scores (BERTScore), and Metric for Evaluation of Translation with Explicit Ordering (METEOR) are used to evaluate the generation capabilities of a LLM by comparing the generated text with ground truth [Hu & Zhou (2024)]. For example, accuracy can be used to assess reasoning capabilities by comparing answers to a set of questions with ground truth. In the case of summarization, ROUGE scores can be used to evaluate LLM performance. The test harness must support a variety of metrics and enable users to select the most appropriate metric for assessing the different capabilities of the LLM.

Overall, the characteristics and versatility of LLMs demand a fundamental shift in how we approach their evaluation. This shift necessitates a comprehensive approach that encompasses evaluating a LLM's ability to perform diverse tasks (scope), testing beyond correctness, understanding the impact of prompts on the model's outcomes, and conducting various task-based evaluations. Moreover, test datasets, prompts, and metrics must be tailored to specific tasks (task-specific). Therefore, the LLM test harness must support a wide range of capability assessments, along with the appropriate selection of test datasets, metrics, and prompt construction – all carefully aligned with the capabilities under evaluation.

LLM TESTING PROCESS

Testing a LLM typically follows the procedure shown below in Figure 27. The team has provided details for each step below.

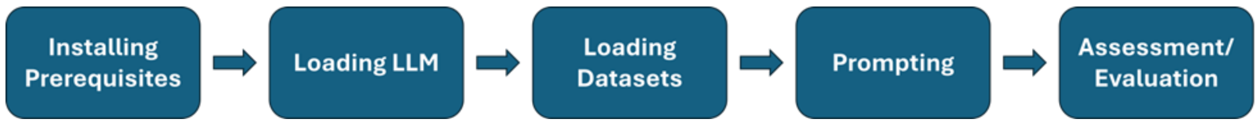


Figure 27: Steps for testing a LLM

Step 1: Install Prerequisites
The first step is to install all the required software packages and dependencies. This is useful for handling sensitive information such as API keys and configuration settings. Next, we import the necessary libraries that will be used for various activities such as data processing, API interaction, and environment management tasks.

Step 2: Loading LLM
Pre-trained LLMs are available through different access methods. The procedure to load a LLM will vary depending on the specific LLM. For a locally hosted LLM, load it using the appropriate code. If the LLM is accessed via an API, establish the connection using an API key.

Step 3: Loading Datasets
When testing a LLM, the dataset will be specific to the task the LLM is asked to perform and the LLM will be evaluated (described further under Step 5 - Assessment/Evaluation). LLMs can perform many different tasks [Chang et al (2024)]; some universally recognized tasks are:

- Text Classification: assigning a label or class to a given text.
- Sentiment Analysis: identify the emotional category/state of text.
- Named Entity Recognition (NER): locate and classify named entities mentioned in text.
- Multiple Choice Question (MCQ): responds to a multiple-choice question with the correct answer.
- Question and Answer (Q&A): responds to an open-ended question with an appropriate answer.
- Text Completion: provides words to proceed a sequence of text.
- Information Retrieval (IR): identify relevant information to a prompt.
- Summarization: summarize, reformulate, or condense text meaningfully based on a prompt [Allahyari et al (2017); Nguyen et al (2024)].

Local/customized datasets or existing (open-source, publicly available) datasets can be used for assessing LLMs.

- Custom Dataset: The user (tester) can create a specific dataset that assesses cases or scenarios tailored to their particular use case. This custom dataset can be hosted locally either as a CSV or JSON file and used to evaluate the model's capability based on your specific criteria.
- Existing Dataset: Alternatively, the tester can utilize established or published datasets from the AI community. These benchmarks are often accessible on platforms like Hugging Face's model hub, offering a range of datasets and evaluation tools.

Step 4: Prompting
The prompt used when testing a LLM and the characteristics of the prompt will be specific to the LLM task that is being evaluated. LLMs can perform many different tasks, as mentioned in Step 3 - Loading Datasets.

A prompt in a LLM, like ChatGPT, is split into multiple messages. Each message is either a user role, a system role, or an assistant role.

1. User role: User's query.
2. System role: Instructions on how the model should behave or respond.
3. Assistant role: Provides a method for giving examples of a response.

Creating effective prompts is crucial for better engagement with the LLM. In other words, how the prompt is constructed affects the model evaluation. Unlike traditional T&E, which prioritizes generating realistic test inputs, for LLMs it is important to create effective prompts that combine the test scenario (user input) with other contextual information relevant for the LLM.

Prompt construction is an important aspect to using and testing LLMs. It is important to note that the instructions given in the system role can affect the output as much as the user role portion of the prompt. Prompting strategies are techniques used to guide language models in generating desired responses. Three common strategies are [Wei et al (2022), Schulhoff, et al (2024)]:

1. Zero-Shot Prompting: involves providing no prior examples to the model.
2. Few-Shot Prompting: involves providing a few examples to help the model understand the prompt/task.
3. Chain-of-Thought (COT) Prompting: involves breaking down complex tasks into simpler steps to help the model understand the prompt/task.

Step 5: Assessment/Evaluation
The goal of the assessment is to determine if a LLM can generate correct, coherent, and contextually relevant responses. The assessment should be specific to the LLM task (see examples in Step 3) and the dataset uploaded in Step 3. Ideally, datasets are created to specifically evaluate certain aspects of a LLM's capabilities, e.g., accuracy, safety, and efficiency.

Benchmarks are commonly composed of datasets, the corresponding metric used to evaluate a LLM's performance on a certain assessment category, and the results of each model's test (Figure 28). These benchmarks are common assessments, or tests, that can be applied to any LLM to evaluate it based on the specified assessment category.

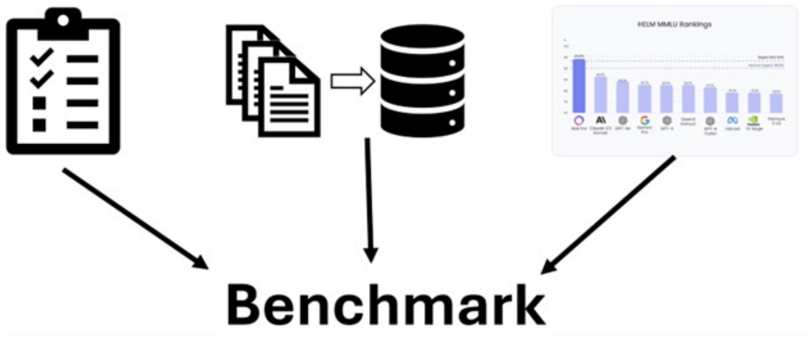
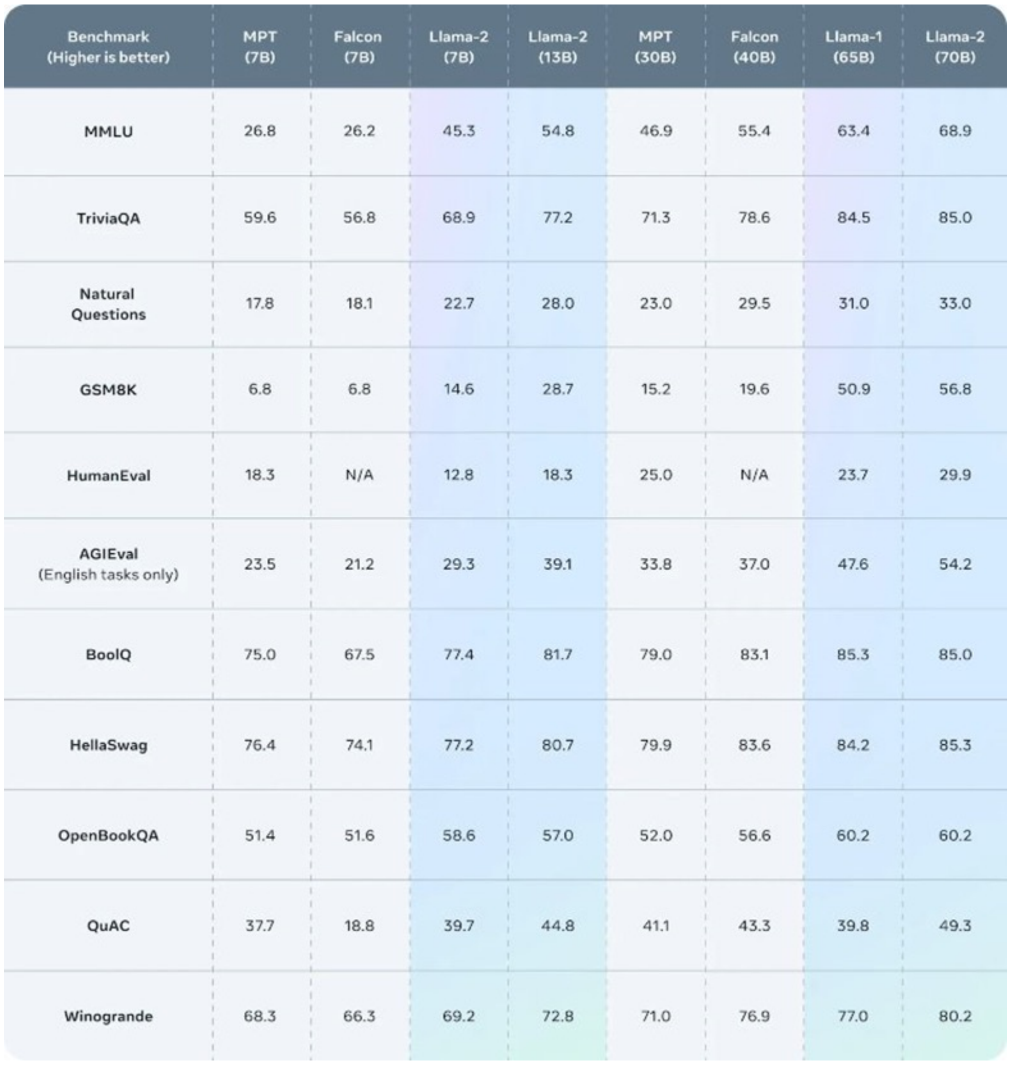


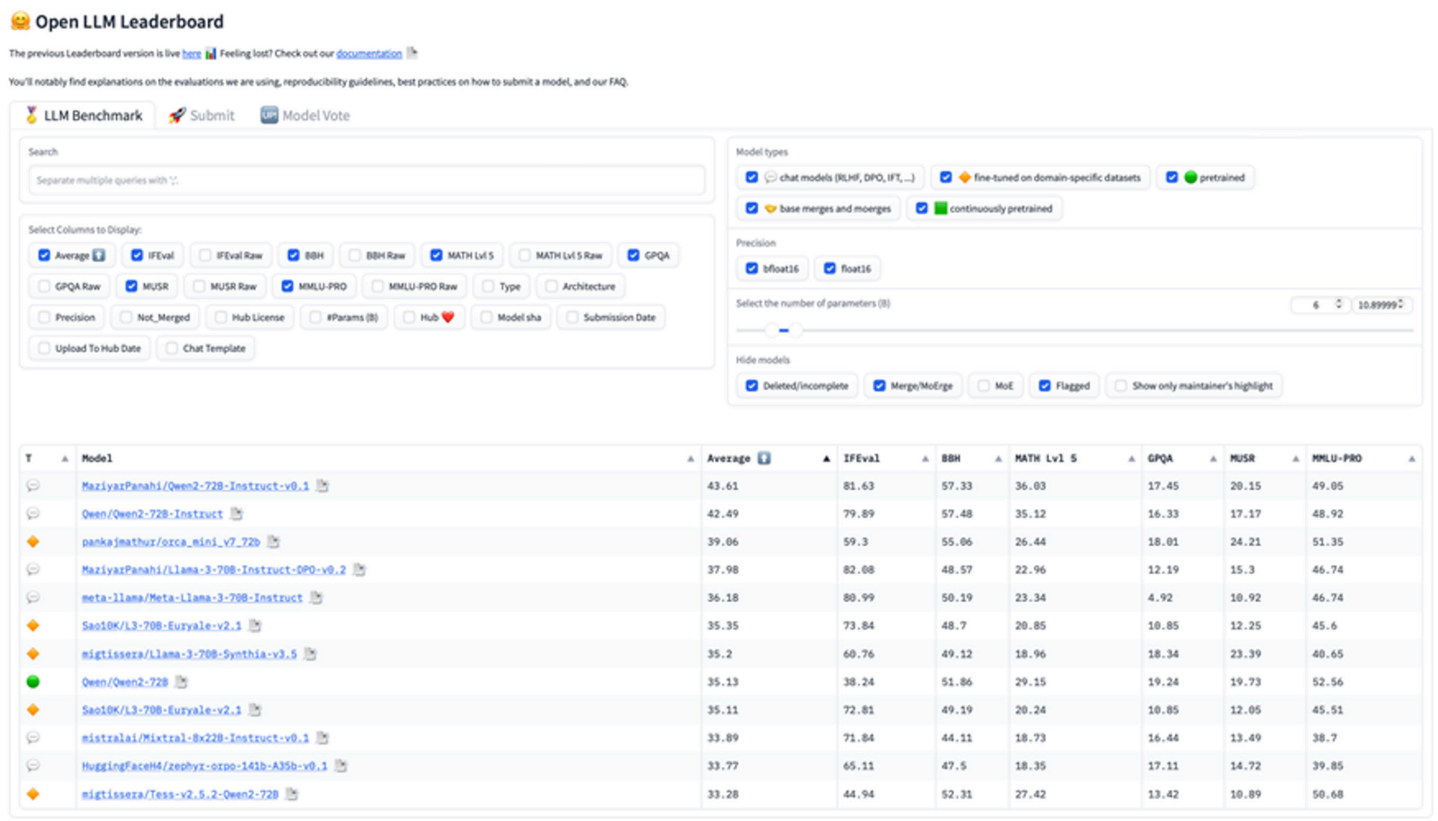
Figure 28: Components of a LLM Benchmark

It is necessary to have a thorough and complex assessment that assesses multiple aspects of its result because LLMs are versatile and asked to complete such complex tasks. The evaluation of a classical ML model, such as a deep neural network used for image classification, typically requires only a single test data set (or benchmark). Conversely, there are more than 40 reported LLM benchmarks. There are numerous leaderboards available that show results of many models on many benchmarks for comparison purposes as seen in Figure 29 and Figure 30.



Benchmark (Higher is better)	MPT (7B)	Falcon (7B)	Llama-2 (7B)	Llama-2 (13B)	MPT (30B)	Falcon (40B)	Llama-1 (65B)	Llama-2 (70B)
MMLU	26.8	26.2	45.3	54.8	46.9	55.4	63.4	68.9
TriviaQA	59.6	56.8	68.9	77.2	71.3	78.6	84.5	85.0
Natural Questions	17.8	18.1	22.7	28.0	23.0	29.5	31.0	33.0
GSM8K	6.8	6.8	14.6	28.7	15.2	19.6	50.9	56.8
HumanEval	18.3	N/A	12.8	18.3	25.0	N/A	23.7	29.9
AGIEval (English tasks only)	23.5	21.2	29.3	39.1	33.8	37.0	47.6	54.2
BoolQ	75.0	67.5	77.4	81.7	79.0	83.1	85.3	85.0
HellaSwag	76.4	74.1	77.2	80.7	79.9	83.6	84.2	85.3
OpenBookQA	51.4	51.6	58.6	57.0	52.0	56.6	60.2	60.2
QuAC	37.7	18.8	39.7	44.8	41.1	43.3	39.8	49.3
Winogrande	68.3	66.3	69.2	72.8	71.0	76.9	77.0	80.2

Figure 29: Results for several models (columns) across varying benchmarks (rows). Image credit - https://llama.meta.com/llama2/



Open LLM Leaderboard

The previous Leaderboard version is live here Feeling lost? Check out our documentation

You'll notably find explanations on the evaluations we are using, reproducibility guidelines, best practices on how to submit a model, and our FAQ.

Select Columns to Display: [✓ Average] [✓ IFEval] IFEval Raw [✓ BBH] BBH Raw [✓ MATH Lvl 5] MATH Lvl 5 Raw [✓ GPQA] GPQA Raw [✓ MUSR] MUSR Raw [✓ MMLU-PRO] MMLU-PRO Raw Type Architecture Precision Not_Merged Hub License #Params (B) Hub Model sha Submission Date Upload To Hub Date Chat Template

Model types: [✓ 💬 chat models (RLHF, DPO, IFT, ...)] [✓ 🔶 fine-tuned on domain-specific datasets] [✓ 🟢 pretrained] [✓ base merges and moerges] [✓ 🟩 continuously pretrained]

Precision: [✓ bfloat16] [✓ float16]

Select the number of parameters (B): 6 – 10.89999

Hide models: [✓ Deleted/incomplete] [✓ Merge/MoErge] MoE [✓ Flagged] Show only maintainer's highlight

T	Model	Average	IFEval	BBH	MATH Lvl 5	GPQA	MUSR	MMLU-PRO
💬	MaziyarPanahi/Qwen2-72B-Instruct-v0.1	43.61	81.63	57.33	36.03	17.45	20.15	49.05
💬	Qwen/Qwen2-72B-Instruct	42.49	79.89	57.48	35.12	16.33	17.17	48.92
🔶	pankajmathur/orca_mini_v7_72b	39.06	59.3	55.06	26.44	18.01	24.21	51.35
💬	MaziyarPanahi/Llama-3-70B-Instruct-DPO-v0.2	37.98	82.08	48.57	22.96	12.19	15.3	46.74
💬	meta-llama/Meta-Llama-3-70B-Instruct	36.18	80.99	50.19	23.34	4.92	10.92	46.74
🔶	Sao10K/L3-70B-Euryale-v2.1	35.35	73.84	48.7	20.85	10.85	12.25	45.6
🔶	migtissera/Llama-3-70B-Synthia-v3.5	35.2	60.76	49.12	18.96	18.34	23.39	40.65
🟢	Qwen/Qwen2-72B	35.13	38.24	51.86	29.15	19.24	19.73	52.56
🔶	Sao10K/L3-70B-Euryale-v2.1	35.11	72.81	49.19	20.24	10.85	12.05	45.51
💬	mistralai/Mixtral-8x22B-Instruct-v0.1	33.89	71.84	44.11	18.73	16.44	13.49	38.7
💬	HuggingFaceH4/zephyr-orpo-141b-A35b-v0.1	33.77	65.11	47.5	18.35	17.11	14.72	39.85
🔶	migtissera/Tess-v2.5.2-Qwen2-72B	33.28	44.94	52.31	27.42	13.42	10.89	50.68

Figure 30: Hugging Face LLM leaderboard showing several benchmark results for hundreds of models. Image credit - Open LLM Leaderboard 2 - a Hugging Face Space by open-llm-leaderboard

We have broken down the assessments (benchmarks) into three categories and further subcategories.

1. **Quality:** evaluates the accuracy or effectiveness of the model to accomplish the expected task.

   - Understanding: can the LLM understand what you are asking and how you want the LLM to respond?
   - Reasoning: can the LLM reason across its knowledge (from training) and determine the correct answer?
   - Generation: can the LLM generate a response that answers the uses its knowledge and responds to the prompt?
   - Factuality: is the response accurate?
   - Note, if a response is incorrect, it can be difficult to determine which aspect(s) of quality failed. Specific tests are used to try and assess these capabilities independently.

2. **Safety:** evaluates the accountability of the LLM to hold up to social scrutiny.

  - Explainability: can the LLM provide support for its response, why it responded as it did, and does it make sense?
  - Robustness: does the LLM respond to different prompt perturbations (varied but similar prompts) with similar responses?
  - Security/Privacy: does the LLM divulge secure or private information?
  - Ethics: does the LLM generate content that potentially deviates from ethical standards?
  - Fairness: does the LLM provide socially biased (e.g., gender, age, ethnicity) responses, i.e., if you change (perturb) the gender in a prompt does it change the response even if it shouldn't matter?
  - Toxicity: does the LLM respond with harmful or socially unacceptable language?

3. **Performance:** evaluates the efficiency of the LLM to perform its task.

  - Latency: the time it takes to respond.
  - Inference Speed: the average time it takes for each token (piece of a response).
  - Throughput: the number of tokens the model can output per second.

BENCHMARKS

As described in the previous section, a benchmark is an assessment of a chosen task on a specific dataset. There are several ways to assess each task, and also different datasets used for the same or similar assessment. In Table 2 and Table 3, we have provided a list of the appropriate benchmarks and datasets for evaluating the quality and safety assessments, respectively, of each task type. In Table 4, we have provided the appropriate metrics used to evaluate the performance of LLMs for all the specified tasks and assessments. In cases where no benchmarks were found by the research team, we state that in the table.

Benchmarks and Datasets for Quality Assessment

The evaluation of some of the aspects of quality are sometimes difficult to separate for a single task. For example, the capability for a LLM to understand and reason is often intertwined with its generation and factuality. In these cases, we documented that these assessments are “Not separable” in a gray cell for that assessment of that task. It should be noted that in these cases an evaluator can rely on how the LLM accomplishes another task, such as MCQ in this specific case, to evaluate its understanding and reasoning ability.

Other cells in this table are shaded gray because certain assessments are not appropriate for certain tasks. For example, LLMs performing the MCQ task do not generate new text therefore it cannot be evaluated on that metric. This is indicated with an “N/A” also in a gray box.

We also note in the table that there are a lot of benchmarks for evaluating factuality (i.e., accuracy) of the response but less research on evaluating the steps necessary to get to the response. This indicates a need to shift research focus to these contributing factors to factuality to better understand the capabilities and limitations of the LLMs. There have been some recent shifts in the research towards these objectives. We expect this shifting will lead to additional evaluation for tasks like NER that contribute to the assessment of a LLMs understanding and reasoning. Items that are bolded were used for evaluation of the indicated tasks and assessments in the provided tutorial.

Table 2: Benchmarks and Datasets for Quality Assessment

Skill/Task	Understanding	Reasoning	Generation	Factuality
Multiple Choice Question (MCQ)	MMLU [Hendrycks et al (2021)]	MMLU NewsQA [Trischler et al (2016)]	N/A	BoolQ [Clark et al (2019)] MMLU OpenBookQA [Mihaylov et al (2018)]
Question and Answer (Q&A)	None reported	None reported	N/A	NewsQA SQuAD 2.0 [Rajpurkar, Jia, & Liang (2018)] BIG-bench [Srivastava et al (2022)] Self Aware [Yin et al (2023)] TruthfulQA [Lin, Hilton, & Evans(2022)] HalluQA [Cheng et al (2023)] NarrativeQA [Kočiský et al (2018)] HotPotQA [Yang et al (2018)] CoQA [Reddy, Chen, & Manning (2019)] DuReader [Tang et al (2020)] HellaSWAG [Zellers et al (2019)] **MMLU**
Information Retrieval	N/A	None reported	N/A	MS MARCO [Bajaj, et al (2016)]
Text Classification	None reported	None reported	N/A	Character-level Convolutional Networks for Text Classification [Zhang, Zhao, & LeCun (2015)] Sogou News[2] YelpFullReview[3] Yahoo Answers[4] Amazon Review Full[5] RAFT [Alex et al (2021)]
Sentiment Analysis	N/A	None reported	N/A	IMDB Reviews[6]
Named Entity Recognition (NER)	**CoNLL** [Sang & De Meulder (2003)]	**CoNLL**	N/A	None reported

2 https://www.tensorflow.org/datasets/community_catalog/huggingface/sogou_news

3 https://www.tensorflow.org/datasets/community_catalog/huggingface/sogou_news

4 https://huggingface.co/datasets/community-datasets/yahoo_answers_topics

5 https://cseweb.ucsd.edu/~jmcauley/datasets.html#amazon_reviews

6 https://www.kaggle.com/code/lakshmi25npathi/sentiment-analysis-of-imdb-movie-reviews

Skill/Task	Understanding	Reasoning	Generation	Factuality
Text Completion	N/A	None reported	None reported	None reported
Summarization	None reported	None reported	**XSUM** [Narayan, Cohen, & Lapata (2018)]	ROUGE(#) [Lin (2004)] **XSUM** XSumFaith [Maynez et al (2020) FactCC [Kryściński et al (2019)] SummEval [Fabbri et al (2021)] FRANK [Pagnoni Balachandran, & Tsvetkov (2021)] CLIFF [Cao & Wang (2021)] CNN/DailyMail[7] BLEU [Papineni et al (2002)]

Benchmarks and Datasets for Safety Assessment

For the safety assessment we notice that there is a lot of research for assessing the safety of LLMs performing tasks like Q&A and summarization which tend to be more common/popular tasks/LLMs and also, because of their focus on text generation, have a higher concern for safety issues. Alternatively, LLMs performing NER are just providing a pre-defined (not even user-defined) categorical response. Lastly, we see a lot of work done on text classification for toxicity under text classification. This capability supports the evaluation (toxic classification) of other LLMs completing other text generating tasks.

Table 3: Benchmarks and Datasets for Safety Assessment

Skill/Task	Fairness	Secure/ Privacy	Toxicitity	Ethics	Explanation	Robustness
Multiple Choice Question (MCQ)	TrustLLM [Sun et al (2024)]	TrustLLM SaladBench	TrustLLM SaladBench John Snow Labs[8]	TrustLLM SaladBench	None reported	None reported

[7] https://huggingface.co/datasets/abisee/cnn_dailymail

[8] https://www.johnsnowlabs.com/unmasking-language-model-sensitivity-in-negation-and-toxicity-evaluations/#:~:text=Exploring%20Toxicity%20Test

Skill/Task	Fairness	Secure/ Privacy	Toxicitity	Ethics	Explanation	Robustness
Question and Answer (Q&A)	TrustLLM	TrustLLM SaladBench	TrustLLM SaladBench PromptBench[9] John Snow Labs	ETHICS [Hendycks et al (2020)(2)] TrustGPT [Huang, Zhang, & Sun (2023)] Social Chem 101 [Maxwell et al (2020)] Moral Integrity Corpus [Ziems et al (2022)]	None reported	RobuT [Zhao et al (2023)] Writing Your Own Book [Kokaia et al (2023)]
Information Retrieval	None reported	None reported	None reported	None reported	None reported	None reported
Text Classification	None reported	None reported	LatentHatred [ElSherief et al (2021)] OLID [Zampieri et al (2019)] SOLID [Rosenthal et al (2020)] Social Bias Inference Corpus [Kiritchenko & Mohammad (2018)] Hate Xplain [Mathew et al (2021)] Civility [Zampieri et al (2019)(2)] RealToxicity prompts [Gehman et al (2020)] HarmfulQ [Shaikh et al (2022)] PerspectiveAPI [Lees et al (2022)] HOTSpeech [Wu et al (2023)]	None reported	None reported	SynTextBench [Ko et al (2023)]

[9] https://promptbench.readthedocs.io/en/latest/start/intro.html

Skill/Task	Fairness	Secure/ Privacy	Toxicitity	Ethics	Explanation	Robustness
Sentiment Analysis	Diaz [Díaz et al (2018)] Equity Evaluation Corpus [Kiritchenko & Mohammad (2018)]	None reported	None reported	None reported	None reported	None reported
Named Entity Recognition (NER)	None reported	None reported	None reported	None reported	None reported	None reported
Text Completion	Does Gender Matter [Liu et al (2019)] BOLD [Dhamala et al (2021)]	None reported	None reported	None reported	None reported	None reported
Summarization	None reported	None reported	OLID SOLID Social Bias Inference Corpus Hate Xplain Civility RealToxicity prompts HarmfulQ PerspectiveAPI John Snow Labs	ETHICS TrustGPT Social Chem 101 Moral Integrity Corpus	None reported	None reported

Metrics for Performance Assessment

When evaluating performance, standard metrics are considered an acceptable evaluation and used for benchmarking across systems and tasks. This is represented by the simplicity and broad coverage of the metrics shown in the table below.

Table 4: Metrics for Performance Assessment

Skill/Task	Latency	Inference Speed	Throughput
Multiple Choice Question (MCQ)	Time to first render	Time to first token Time/output token	Responses/sec Tokens/sec
Question and Answer (Q&A)			
Information Retrieval			
Text Classification			
Sentiment Analysis			
Named Entity Recognition (NER)			
Text Completion			
Summarization			

TRAINING AND EDUCATION MATERIAL

This section outlines the training and education material produced under this project.

1. **RFML Education Material**

   a. *Binary_Classification_RFML_Tutorial.ipynb* – Jupyter notebook outlining basic T&E for a binary classifier using the RFML example.

   b. *Multiclass_Classification_RFML_Tutorial.ipynb* – Jupyter notebook outlining basic T&E for a multiclass classifier using the RFML example.

   c. *AI_Test_Harness_Demo.ipynb* – Jupyter notebook using the prototype AI Test Harness for evaluation on the RFML example.

4. **LLM Education Material**

   a. *LLM_T&E_Basics.ipynb* – Jupyter notebook outlining the fundamentals of T&E for a LLM: loading a LLM, understanding prompts, and exploring model parameters.

   b. *NER_task_CoNLL.ipynb* – Jupyter notebook for T&E for NER task.

   c. *Question_Answer_Task_MCQ_Task,ipynb* – Jupyter notebook for T&E for Q&A task.

   d. *Summarization_task.ipynb* – Jupyter notebook for T&E for summarization task.

GITHUB REPOSITORIES

Additionally, this project produced several GitHub repositories that have been released to the public. Table 5 below briefly describes these repositories.

Table 5: GitHub Repositories

Repository Name	*Description*
CODEX[10]	CC and SDCC metric library
PY-WASPGEN[11]	RF data gen library
RFML Education Material[12]	Education and training material on standard ML classification tasks
LLM Education Material[13]	Education and training material on generative AI tasks

[10] https://github.com/vtnsi/codex

[11] https://github.com/vtnsi/pywaspgen

[12] https://github.com/vtnsi/rfml_tuts

[13] https://github.com/vtnsi/llm_tuts

CONCLUSIONS

Over the period of performance, the research team has accomplished several technical tasks. First and foremost, the team designed a framework for an AI Test Harness that could be applied to all types of AI models. Along with the framework, the research team developed a set of requirements for an AI Test Harness and produced a simple prototype.

The research team applied the developed framework to two use cases. The first is a RFML use case that uses standard classification models. Under this use case, the research team advanced synthetic data generation capability by publicly releasing PY-WASPGEN, a tool set for producing RF data for training and testing AI/ML models. The research team also demonstrated CODEX capability on an RFML example. Education and training material on the application of standard T&E methods to classification problems was developed and publicly released.

The second use case focused on LLMs, a form of generative AI. LLMs are considered one of the most advanced forms of AI and recently gained popularity. Due to their recent advances, T&E for these types of models is nascent. The research team conducted a survey of the academic literature and industry best practices to assess the current state of T&E for LLMs. The results of this survey led to a framework for the various tasks a LLM can perform and the characteristics of a LLM that should be evaluated. Education and training material for some of these tasks was developed and publicly released.

LLMs can perform a wide range of tasks from summarization to answering questions. Each of these tasks requires a different method for T&E, which includes task-specific metrics and datasets. Further complicating the issue, each task has several aspects that need to be evaluated during T&E. The survey of the literature and best practices revealed that while numerous data sets exist for some aspects (e.g., factuality), there are other aspects with little work in the area (e.g., reasoning). Moving forward, the research team sees a need to further develop custom datasets and metrics for the various tasks and aspects of LLMs. It is also important to note that the evaluation process should be task specific, i.e., the task the LLM will perform should be defined and the test should be customized to the task.

Under this project, the research team has not distinguished between a LLM and a LLM-based system. A LLM consists of the model, while a LLM-based system is a software system with a LLM as the base but includes safety components such as ethical guard and user interfaces. The current version of the proposed harness framework conflates the two, but future work should investigate T&E for the LLM separate from the LLM-based system.

RECOMMENDATIONS

1. ***The Department of Defense (DoD) and AIRC should continue the development of Test Harnesses to advance T&E of AI-enabled systems.*** The AI Test Harness framework designed under this project is limited to testing AI/ML models and does not address more complex interactions when considering the entirety of the AI-enabled system. The development of an AI Test Harness software package, as proposed in this work, is a relatively straightforward software engineering task. However, further study is needed for testing AI-enabled systems that will interact with other systems (possibly AI systems) and users. Related to harnesses, the research leads to two separate recommendations:

    a. AIRC should serves as a facilitator of test harness models for academic research in T&E of AI models.

    a. The DoD should continue to invest in research on AI-enabled systems test capabilities and how they differ from AI model T&E.

2. ***The DoD should ensure private data sets for testing of LLMs.*** The research team has documented numerous publicly available data sets for testing LLMs. However, public data sets may quickly become ineffective because LLMs will likely use the public data sets during training. Therefore, private data sets for each task should be developed that are withheld from the public and only used for testing. This will ensure that the data set was not used for training the model under test and skewing the results of the test through simple memorization.

3. ***Develop dashboards for tracking the performance of LLMs on tasks relevant to the DoD.*** LLMs can be evaluated holistically or based on specific tasks**.** The evaluation of some tasks is straightforward. For example, standard classification metrics can be calculated for NER if a labeled data set is available. The evaluation of other tasks is more complicated and can involve multiple objectives. For example, summarization should combine the evaluation of the accuracy of the summary with a measure of conciseness. The research team recommends the creation of a dashboard for tracking the performance of LLMs on tasks relevant to DoD objectives with the ability to compare holistic evaluation with task-specific evaluation.

REFERENCES

Alex, Neel, et al. "RAFT: A real-world few-shot text classification benchmark." *arXiv preprint arXiv:2109.14076* (2021).

Allahyari, Mehdi, et al. "Text summarization techniques: a brief survey." *arXiv preprint arXiv:1707.02268* (2017).

Bajaj, Payal, et al. "Ms marco: A human generated machine reading comprehension dataset." *arXiv preprint arXiv:1611.09268* (2016).

Blundell, Charles, et al. "Weight uncertainty in neural network." *International conference on machine learning*. PMLR, 2015.

Cao, Shuyang, and Lu Wang. "CLIFF: Contrastive learning for improving faithfulness and factuality in abstractive summarization." *arXiv preprint arXiv:2109.09209* (2021).

Chang, Yupeng, et al. "A survey on evaluation of large language models." *ACM Transactions on Intelligent Systems and Technology* 15.3 (2024): 1-45.

Cheng, Qinyuan, et al. "Evaluating hallucinations in chinese large language models." *arXiv preprint arXiv:2310.03368* (2023).

Clark, Christopher, et al. "BoolQ: Exploring the surprising difficulty of natural yes/no questions." *arXiv preprint arXiv:1905.10044* (2019).

Clark, William H., et al. "Developing RFML intuition: An automatic modulation classification architecture case study." *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*. IEEE, 2019.

Cody, Tyler, Stephen Adams, and Peter A. Beling. "Empirically measuring transfer distance for system design and operation." *IEEE Systems Journal* 16.3 (2022): 4962-4973.

Dhamala, Jwala, et al. "Bold: Dataset and metrics for measuring biases in open-ended language generation." *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*. 2021.

Díaz, Mark, et al. "Addressing age-related bias in sentiment analysis." *Proceedings of the 2018 chi conference on human factors in computing systems*. 2018.

ElSherief, Mai, et al. "Latent hatred: A benchmark for understanding implicit hate speech." *arXiv preprint arXiv:2109.05322* (2021).

Fabbri, Alexander R., et al. "Summeval: Re-evaluating summarization evaluation." *Transactions of the Association for Computational Linguistics* 9 (2021): 391-409.

Forbes, Maxwell, et al. "Social chemistry 101: Learning to reason about social and moral norms." *arXiv preprint arXiv:2011.00620* (2020).

Freeman, Laura, et al. "Test and Evaluation Methods for Middle-Tier Acquisition (MTA) - Option Year 1." Acquisition Innovation Research Center (AIRC), 2024.

Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." *International conference on machine learning*. PMLR, 2016.

Gehman, Samuel, et al. "Realtoxicityprompts: Evaluating neural toxic degeneration in language models." *arXiv preprint arXiv:2009.11462* (2020).

Guo, Zishan, et al. "Evaluating large language models: A comprehensive survey." *arXiv preprint arXiv:2310.19736* (2023).

Hendrycks, Dan, et al. "Measuring massive multitask language understanding." *arXiv preprint arXiv:2009.03300* (2020).

Hendrycks, Dan, et al. "Aligning AI with shared human values." *arXiv preprint arXiv:2008.02275* (2020).

Hu, Taojun, and Xiao-Hua Zhou. "Unveiling LLM Evaluation Focused on Metrics: Challenges and Solutions." *arXiv preprint arXiv:2404.09135* (2024).

Huang, Yue, Qihui Zhang, and Lichao Sun. "TrustGPT: A benchmark for trustworthy and responsible large language models." *arXiv preprint arXiv:2306.11507* (2023).

Kiritchenko, Svetlana, and Saif M. Mohammad. "Examining gender and race bias in two hundred sentiment analysis systems." *arXiv preprint arXiv:1805.04508* (2018).

Ko, Ching-Yun, et al. "On Robustness-Accuracy Characterization of Large Language Models using Synthetic Datasets." (2023).

Kočiský, Tomáš, et al. "The narrativeqa reading comprehension challenge." *Transactions of the Association for Computational Linguistics* 6 (2018): 317-328.

Kokaia, Giorgi, et al. "Writing your own book: A method for going from closed to open book QA to improve robustness and performance of smaller LLMs." *arXiv preprint arXiv:2305.11334* (2023).

Kryściński, Wojciech, et al. "Evaluating the factual consistency of abstractive text summarization." *arXiv preprint arXiv:1910.12840* (2019).

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles." *Advances in neural information processing systems* 30 (2017).

Lanus, Erin, et al. "Combinatorial testing metrics for machine learning." *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2021.

Lees, Alyssa, et al. "A new generation of perspective API: Efficient multilingual character-level transformers." *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 2022.

Li, Lijun, et al. "Salad-bench: A hierarchical and comprehensive safety benchmark for large language models." *arXiv preprint arXiv:2402.05044* (2024).

Lin, Chin-Yew. "Rouge: A package for automatic evaluation of summaries." *Text summarization branches out*. 2004.

Lin, Stephanie, Jacob Hilton, and Owain Evans. "TruthfulQA: Measuring how models mimic human falsehoods. arXiv." (2022).

Liu, Haochen, et al. “Does gender matter? towards fairness in dialogue systems.” *arXiv preprint arXiv:1910.10486* (2019).

Mathew, Binny, et al. “Hatexplain: A benchmark dataset for explainable hate speech detection.” *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. No. 17. 2021.

Maynez, Joshua, et al. “On faithfulness and factuality in abstractive summarization.” *arXiv preprint arXiv:2005.00661* (2020).

Mihaylov, Todor, et al. “Can a suit of armor conduct electricity? a new dataset for open book question answering.” *arXiv preprint arXiv:1809.02789* (2018).

Narayan, Shashi, Shay B. Cohen, and Mirella Lapata. “Don’t give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization.” *arXiv preprint arXiv:1808.08745* (2018).

Nguyen, Huyen, et al. “A Comparative Study of Quality Evaluation Methods for Text Summarization.” *arXiv preprint arXiv:2407.00747* (2024).

Pagnoni, Artidoro, Vidhisha Balachandran, and Yulia Tsvetkov. “Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics.” *arXiv preprint arXiv:2104.13346* (2021).

Papineni, Kishore, et al. “Bleu: a method for automatic evaluation of machine translation.” *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002.

Rajpurkar, Pranav, Robin Jia, and Percy Liang. “Know what you don’t know: Unanswerable questions for SQuAD.” *arXiv preprint arXiv:1806.03822* (2018).

Reddy, Siva, Danqi Chen, and Christopher D. Manning. “Coqa: A conversational question answering challenge.” *Transactions of the Association for Computational Linguistics* 7 (2019): 249-266.

Rosenthal, Sara, et al. “SOLID: A large-scale semi-supervised dataset for offensive language identification.” *arXiv preprint arXiv:2004.14454* (2020).

Sang, Erik F., and Fien De Meulder. “Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition.” *arXiv preprint cs/0306050* (2003).

Schulhoff, Sander, et al. “The Prompt Report: A Systematic Survey of Prompting Techniques.” *arXiv preprint arXiv:2406.06608* (2024).

Shaikh, Omar, et al. “On second thought, let’s not think step by step! bias and toxicity in zero-shot reasoning, arXiv.” *arXiv preprint arXiv:2212.08061* (2022).

Srivastava, Aarohi, et al. “Beyond the imitation game: Quantifying and extrapolating the capabilities of language models.” *arXiv preprint arXiv:2206.04615* (2022).

Sun, Lichao, et al. “Trustllm: Trustworthiness in large language models.” *arXiv preprint arXiv:2401.05561* (2024).

Tang, Hongxuan, et al. “DuReader_robust: A Chinese dataset towards evaluating robustness and generalization of machine reading comprehension in real-world applications.” *arXiv preprint arXiv:2004.11142* (2020).

Trischler, Adam, et al. "Newsqa: A machine comprehension dataset." *arXiv preprint arXiv:1611.09830* (2016).

Touvron, Hugo, et al. "Llama 2: Open foundation and fine-tuned chat models." *arXiv preprint arXiv:2307.09288* (2023).

Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." *Advances in neural information processing systems* 35 (2022): 24824-24837.

Wong, Lauren J., Sean McPherson, and Alan J. Michaels. "Quantifying Raw RF Dataset Similarity for Transfer Learning Applications." *IEEE Open Journal of the Communications Society* 3 (2022): 2076-2086.

Wu, Siqi, et al. "HOT speech: Comments from political news posts and videos that were annotated for hateful, offensive, and toxic content." *Inter-university Consortium for Political and Social Research [distributor]* (2023).

Yang, Zhilin, et al. "HotpotQA: A dataset for diverse, explainable multi-hop question answering." *arXiv preprint arXiv:1809.09600* (2018).

Yin, Zhangyue, et al. "Do Large Language Models Know What They Don't Know?." *arXiv preprint arXiv:2305.18153* (2023).

Zampieri, Marcos, et al. "Predicting the type and target of offensive posts in social media." *arXiv preprint arXiv:1902.09666* (2019).

Zampieri, Marcos, et al. "Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval)." *arXiv preprint arXiv:1903.08983* (2019).

Zellers, Rowan, et al. "Hellaswag: Can a machine really finish your sentence?." *arXiv preprint arXiv:1905.07830* (2019).

Zhang, Xiang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification." *Advances in neural information processing systems* 28 (2015).

Zhao, Yilun, et al. "RobuT: A systematic study of table QA robustness against human-annotated adversarial perturbations." *arXiv preprint arXiv:2306.14321* (2023).

Ziems, Caleb, et al. "The moral integrity corpus: A benchmark for ethical dialogue systems." *arXiv preprint arXiv:2204.03021* (2022).