Towards Behavior-Space Testing of Critical ML-Enabled Systems

Eunho Cho
ehcho@kaist.ac.kr
Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, Korea

Abstract

In safety-critical machine learning (ML)-enabled systems, the internal logic of ML component is inherently opaque, and keep evolving by retraining, or architectural modification. While these changes undermine the validity of existing test results, black-box testing alone is insufficient to capture these shifts in internal logic. We propose a novel testing approach that estimates the behavior-space geometry, an abstract geometric space representing the internal behavioral similarity that the ML component forms among data. We seek to estimate behavior-space by combining signals from the ML comopnent's learning process, and define test coverage based on the geometry. When the behavior-space is altered by ML evolution, we plan to estimate geometric changes to detect regions of coverage gaps in the existing test suite and select supplementary test suite.

CCS Concepts

• **Software and its engineering** → **Software verification and validation**; • **Computing methodologies** → *Machine learning algorithms*.

Keywords

Behavior-Space, ML-Enabled System Testing, Safety-Critical ML-Enabled Systems, MLOps, SE4AI, Software Testing

ACM Reference Format:
Eunho Cho. 2026. Towards Behavior-Space Testing of Critical ML-Enabled Systems. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE-Companion '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3774748.3787661

1 Introduction

Testing Machine Learning (ML)-based systems is fundamentally challenging, because their behavior emerges from stochastic patterns formed by ML model during learning and the ML component that encapsulates both the model and dependent code [9]. Most prior works treat ML components as black-boxes, and develop methods that assess reliability without accessing internal information, such as partitioning feature space and checking behavioral consistency within each region [10].

For safety-critical ML-enabled systems, however, such black-box approaches may be insufficient. As black-box testing observes only inputs and outputs, it may struggle to identify the root causes of errors or the shifts in internal logic. In safety-critical ML-enabled systems, this opacity of internal behavior acts as a critical obstacle to reliability assurance [2, 8]. Moreover, ML-enabled systems continuously evolve. ML components continuously evolve through processes such as retraining, architectural modification, and optimization. As ML components evolve continuously, test results from previous versions may no longer be valid for a new version [1, 4]. This necessitates an evolution-aware testing approach for ML-enabled systems capable of recognizing internal logic changes.

To address these challenges, we focus on the phenomena manifested during the ML component's learning process. An ML component implicitly forms internal logic by repeatedly differentiating data. Therefore, learning signals, such as loss, gradients, and the Jacobian, serve as indirect representations of how data is processed and distinguished within the ML component. We assume that data points exhibiting similar signals also behave similar, while those with dissimilar signals behave by different logic. Consequently, if the signal-based behavioral similarity can be expressed as a geometric relationship among data, we can indirectly reconstruct the ML component's opaque internal logic as a quantifiable geometry.

We define the behavior-space geometry as the projection of the ML component's internal logic, as captured through these diverse signals. This behavior-space is an abstract space where data points are arranged according to the component's learned logic: behaviorally similar data is positioned closely, while dissimilar data is placed far apart. This space enables a new testing approach, one that moves beyond treating the ML component as a simple black box and instead targets its internal logic.

Based on this idea, we aim to achieve three main objectives: (1) To estimate the geometry of the behavior-space. (2) To redefine test coverage based on this geometry. (3) To identify coverage gaps caused by evolution and to select supplementary tests.

2 Related Work

Recent papers on testing ML systems largely treat the ML component as a black box, assessing reliability based solely on input-output relationships. A prevalent approach, for instance, involves partitioning the feature space and verifying behavioral consistency within each resulting region [8, 10]. Another papers focuses primarily on achieving output diversity or discovering fault-eliciting inputs [3, 5]. These approaches can be limited in capturing how the model differentiates data, or how its internal behavior shifts.

Data-centric papers also attempt to quantify the influence of data on the learning process to analyze performance contribution or data selection [6, 7]. They do not extend to organizing these insights into a global behavior-space for testing.

ICSE-Companion '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2296-7/ 2026/04
https://doi.org/10.1145/3774748.3787661

3 Prior Research and Key Results

Our prior work proposed two complementary approaches to quantitatively analyze the inter-data relationships formed by ML components during training. Our initial work, through loss-based estimation, quantified the degree of learning interference between data, demonstrating that the co-learnability of data pairs varies with the ML component architecture. Subsequent work utilized gradient signals during training to define a learning dynamics vector for each data, enabling the interpretation of the behavior-space's cohesion.

The core result of this prior research was providing common empirical evidence that the behavior-space as a projection for internal logic of ML component can be quantified by observing signals at the loss- or gradient-levels, which serves as the foundation for the proposed method. Building upon this foundation, our proposed research integrates multiple learning dynamics signals to estimate the geometric relationships between data inside the behavior-space. Based on this, we plan to develop an behavior-space testing method to assess coverage over the ML-enabled system's behavior.

4 Proposed Work: Behavior-Space Testing

This section introduces the conceptual flow of our proposed behavior-space testing method. The method will consist of three steps: (1) estimating the behavior-space geometry, (2) defining test coverage on that geometry, and (3) identifying coverage gaps caused by ML component evolution and selecting supplementary tests.

Assumption 1. Given an ML component architecture A, there exists a behavior-space geometry G_A that is dependent on A.

G_A is an abstract that reflects the relationships among data, positioning data that exhibit similar responses during learning or testing closer together, and those with dissimilar responses farther apart. It encapsulates all geometric relationships between any two arbitrary data within the behavior-space.

4.1 Estimating Behavior-Space Geometry

Hypothesis 1. G_A can be approximately constructed by leveraging signals from learning process, g_A.

In this step, we seek to estimate G_A by leveraging various signals, such as loss, gradients, and the Jacobian. As an initial approach, we quantify behavioral similarity between data by combining these learning signals. However, it remains unclear how reliably these signals reflect the actual behavior-space relationships. The contribution of each signal may vary depending on the training phase or the ML component architecture, and the complementary interactions among signals have not been systematically analyzed. Therefore, we plan to experimentally compare the impact of different signal combination methods on behavior-space estimation and to evaluate the consistency and interpretability of the geometric relationships.

Hypothesis 2. The set of subspaces, S_A, obtained by partitioning the behavior-space based on G_A, is expected to exhibit high cohesion by aggregating data that induce similar responses.

Next, we aim to decompose the behavior-space based on estimated G_A. Rather than relying on label- or feature-based heuristics, we plan to extract the density structure from geometric relationships derived from the estimated behavior-space geometry. We can define the resulting partitions as behavior-subspaces, which signify decomposed regions of the behavior-space where distinct learning and validation units can be addressed without interference. Specifically, we will construct a proximity graph and compare clustering methods, such as HDBSCAN and K-means, to identify densely populated regions as distinct subspaces. We also intend to investigate how the choice of clustering algorithm impacts the cohesion of the resulting set of subspaces, S_A.

4.2 Defining Behavior-Space Coverage

Hypothesis 3. By synthesizing both subspace- and point-level behavior-space coverage in test design, the internal logic of the ML component can be systematically evaluated.

In this step, we plan for defining test coverage by leveraging the identified subspace S_A and the behavior-space geometry G_A. Our initial idea to define behavior-space coverage has two complementary levels: macro (subspace-level), and micro (point-level).

Definition 1 (Subspace-level Behavior-Space Coverage)**.** Subspace-level Behavior-Space coverage is defined as the proportion of subspaces in S_A for which at least one test input exists.

Subspace-level behavior-space coverage macroscopically assesses the extent to which the test set spans the entire behavioral space of the ML-enabled system. As each subspace corresponds to a behavioral unit formed by the ML component during the learning process, the fundamental metric for coverage is the extent to which the test set achieves a balanced inclusion of these subspaces. Consequently, this coverage assesses whether the test suite reaches all primary regions of the behavior-space.

Definition 2 (Point-level Behavior-Space Coverage)**.** Point-level coverage is defined as the volume of the union of regions covered by each test input, based on G_A similarity.

At the micro-level, point-level behavior-space coverage evaluates the specific portion of the ML-enabled system's behavioral space covered by each individual test input. Single test input for each subspace may be insufficient for identifying vulnerable zones or sparse regions within the behavior-space. Consequently, G_A-based similarity is employed to evaluate the extent to which each test input covers its surrounding locality in the behavior-space. This coverage, therefore, ascertains whether the test suite is distributed broadly and equitably within the subspaces themselves.

We believe that test design needs to consider both levels of coverage. At the macro-level, the goal is to construct a test set that achieves balanced inclusion of all subspaces. At the micro-level, the process prioritizes the selection of data that have low geometric similarity to the existing test set—that is, data in regions that are currently under-covered. This dual approach can improve the test set to cover the system's entire behavior-space.

The proposed behavior-space coverage is a novel attempt, however, further validation is required. Specifically, the coverage value is potentially sensitive to the setting of the similarity threshold of point-level coverage, or the importance weighting assigned to different subspaces. Therefore, in this research we plan to verify whether the proposed coverage metric maintains a consistent meaning across diverse data distributions and ML component architectures and, if necessary, to explore alternative similarity functions or weighting strategies. Through this, we aim to systematically examine the suitability of the proposed coverage definition for assessing the practical adequacy of behavior-space coverage.

4.3 Identifying Coverage Gaps on Evolution

Hypothesis 4. For similar ML architectures A' and A, it is possible to partially approximate $G_{A'}$ by leveraging the pre-computed G_A.

In this step, we seek to develop a method for exploring how to efficiently select test suite in response to ML component evolution, especially for architectural changes. Evolution of ML component changes in the behavior-space geometry G_A, and recomputing this for all data pairs is computationally expensive. Therefore, rather than precisely reconstructing the entire structure, this research considers an approximate approach that estimates the direction of change relative to the existing G_A.

We expect that if architectures A' and A are sufficiently similar, $G_{A'}$ can be partially reconstructed by leveraging the pre-computed G_A and S_A, thereby avoiding a full recalculation. As an initial approach, we will consider a method that selects a small set of representative data, sparsely computes $g_{A'}$, and then estimates the entire ΔG_A based on the observed differences.

Hypothesis 5. The approximated ΔG_A is valid for identifying regions of the behavior-space that are no longer sufficiently covered by the existing test suite, and it can be utilized to select a complementary test suite targeting these regions.

When the behavior-space is altered by architectural changes, the subspace-level or point-level coverage secured by the existing test set will decrease in some regions. The objective of this stage is to rapidly identify these areas of coverage loss and select test data to compensate for them. By analyzing the estimated ΔG_A, we seek to identify the subspaces with diminished coverage and adding data representative of these regions to the test set. This aims to maintain post-evolution test coverage without incurring cost increases.

However, several critical questions remain. First, it is not yet clear what criteria should be used to select representative data for ΔG_A approximation to reliably capture the direction of change. Furthermore, an appropriate method for extrapolating the partially observed changes (Δg_A) to the entire behavior-space must be investigated. It must also be considered how effective this approximation-based approach will be in scenarios involving substantial architectural divergence, or whether a more conservative selection strategy is required. By systematically addressing these questions, we aim to establish a practical test selection method for maintaining a level of behavior-space coverage, even after ML component evolution.

5 Expected Contributions

The proposed method provides a foundation for interpreting the internal logic of ML components. The behavior-space geometry, as proposed in this research, transparently represents the internal logic formed during the learning process as a geometry. This provides a basis for tracing the root causes of system malfunctions and interpreting test results in an explainable manner.

The proposed behavior-space coverage redefines the test coverage standards required in safety-critical industrial domains. This coverage metric, grounded in the behavior-space, directly reflects the actual behavioral patterns of the ML component. This can lead to a new standard for quantitatively assessing how comprehensively tests cover behaviors in fields such as autonomous driving.

The proposed method contributes to the continuous verification mechanisms within MLOps essential for the continuous evolution of ML components. The proposed ΔG approximation and gap-based test selection method selectively add the necessary tests. This approach has the potential to reduce maintenance costs and establish an automated test management that continuously assures quality.

6 Evaluation Plan

We will conduct experiments across multiple ML-enabled system domains, including image classification tasks and autonomous driving scenarios using simulation environments. For behavior-space geometry estimation, we will measure how consistently different combinations of signals reproduce meaningful inter-data relationships. For coverage definition, we will evaluate whether subspace-level and point-level coverage reflect behavioral diversity and identify under-covered regions. For evolution-aware test selection, we will examine whether approximated ΔG_A enables detection of coverage gaps and whether supplementary tests derived from these gaps improve post-evolution coverage with minimal cost.

- 2026 H1: Refine behavior-space geometry estimation techniques, and construct an improved G_A estimation model.

- 2026 H2: Define behavior-space-based test coverage metrics, and evaluate how effectively these metrics reflect test adequacy.

- 2027 H1: Develop a method for approximating ΔG and identifying coverage gaps on ML component evolution.

Acknowledgments

This work was partly supported by the IITP grant funded by the Korea government (MSIT)((RS-2025-02218761, 50%), (RS-2024-00406245, 25%)), and Samsung Electronics(25%).

References

[1] Firas Bayram and Bestoun S Ahmed. 2025. Towards trustworthy machine learning in production: An overview of the robustness in mlops approach. *Comput. Surveys* 57, 5 (2025), 1–35.

[2] Saddek Bensalem, Chih-Hong Cheng, Wei Huang, Xiaowei Huang, Changshun Wu, and Xingyu Zhao. 2023. What, indeed, is an achievable provable guarantee for learning-enabled safety-critical systems. In *International Conference on Bridging the Gap between AI and Reality*. Springer, 55–76.

[3] Christian Birchler, Sajad Khatiri, Bill Bosshard, Alessio Gambi, and Sebastiano Panichella. 2023. Machine learning-based test selection for simulation-based testing of self-driving cars software. *Empirical Software Engineering* 28, 3 (2023).

[4] Satvik Garg, Pradyumn Pundir, Geetanjali Rathee, PK Gupta, Somya Garg, and Saransh Ahlawat. 2021. On continuous integration/continuous delivery for automated deployment of machine learning models using mlops. In *2021 IEEE fourth international conference on artificial intelligence and knowledge engineering (AIKE)*. IEEE, 25–28.

[5] Dong Huang, Qingwen Bu, Yichao Fu, Yuhao Qing, Xiaofei Xie, Junjie Chen, and Heming Cui. 2024. Neuron Sensitivity-Guided Test Case Selection. *ACM Transactions on Software Engineering and Methodology* 33, 7 (2024), 1–32.

[6] Krishnateja Killamsetty, Sivasubramanian Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. 2021. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*. PMLR, 5464–5474.

[7] Maximilian Muschalik, Hubert Baniecki, Fabian Fumagalli, Patrick Kolpaczki, Barbara Hammer, and Eyke Hüllermeier. 2024. shapiq: Shapley interactions for machine learning. *Advances in Neural Information Processing Systems* 37 (2024), 130324–130357.

[8] Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. 2020. Testing machine learning based systems: a systematic mapping. *Empirical Software Engineering* 25, 6 (2020), 5193–5254.

[9] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. 2020. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering* 48, 1 (2020), 1–36.

[10] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. 2023. Efficient and effective feature space exploration for testing deep learning systems. *ACM Transactions on Software Engineering and Methodology* 32, 2 (2023), 1–38.