RBT4DNN: Requirements-based Testing of Neural Networks

Nusrat Jahan Mozumder[1,2][0009−0003−1802−5150], Felipe Toledo[1,3][0000−0002−5632−7518], Swaroopa Dola[1,4][0000−0002−9831−2744], and Matthew B. Dwyer[1,5][0000−0002−1937−1544]

[1] Department of Computer Science, University of Virginia, Charlottesville, Virginia, USA
[2] nm8tm@virginia.edu
[3] ft8bn@virginia.edu
[4] sd4tx@virginia.edu
[5] matthewbdwyer@virginia.edu

Abstract. Testing deep neural networks (DNNs) is challenging due to the difficulty of formalizing functional requirements, limiting the applicability of traditional requirements-based testing. We propose RBT4DNN, a requirements-driven approach that leverages natural language specifications and a glossary-defined semantic feature space to express preconditions as logical combinations of semantic features. These preconditions guide the selection of training data to fine-tune generative models for producing targeted test inputs, enabling comparison between model outputs and expected postconditions. RBT4DNN supports both fault detection and requirement-guided exploration of model behavior. Our evaluation shows that generated tests are realistic, diverse, and aligned with requirement preconditions, enabling effective validation and analysis of model generalization.

Keywords: test input generation, neural network, functional requirements, structured natural language

1 Introduction

As deep neural networks (DNNs) become increasingly integrated into software systems, determining whether a trained model is fit for deployment remains a critical challenge. The most common evaluation method is test accuracy, which measures performance on a labeled test set [1]. While analogous to traditional test oracles [2], this approach has significant limitations: it relies on input-specific labels, fails to capture broader acceptable behaviors, and does not support targeted testing of specific input conditions.

Requirements-based testing in traditional software addresses these issues by defining functional requirements with preconditions and postconditions, enabling systematic test generation and validation [3,4,5,6,7,8,9,10]. However, applying

such methods to learned components (LCs) is challenging due to the difficulty of formalizing semantic features in high-dimensional input spaces. Despite prior efforts [11,12,13,14,15,16], no broadly applicable approach exists.

We propose RBT4DNN, a requirements-based testing method that leverages structured natural language (SNL) requirements expressed over semantic features [17,4,5,18]. Preconditions are defined using glossary terms representing domain-specific semantic features [19,20,21], and training data is automatically labeled using techniques such as auto-labeling and scene understanding [22,23,24,25].

To generate test inputs satisfying requirement preconditions, RBT4DNN leverages latent representations in modern generative models [26]. Instead of prompting alone, which produces unrealistic samples, we fine-tune models using low-rank adaptation (LoRA) [27] on data matching the precondition. This enables generation of realistic, diverse, and precondition-consistent inputs, improving test validity and coverage [28,29,30,31].

RBT4DNN supports both fault detection and exploratory analysis of model behavior, enabling developers to evaluate DNNs against requirement-defined expectations. By bridging the gap between informal natural language requirements and executable test generation, it provides a practical and scalable pathway for applying requirements-based testing to complex learned systems. Furthermore, by focusing testing on semantically meaningful input regions, RBT4DNN facilitates deeper insights into model generalization, uncovers systematic failure modes, and supports more reliable deployment of DNN-based systems in safety-critical and real-world applications.

2 Approach

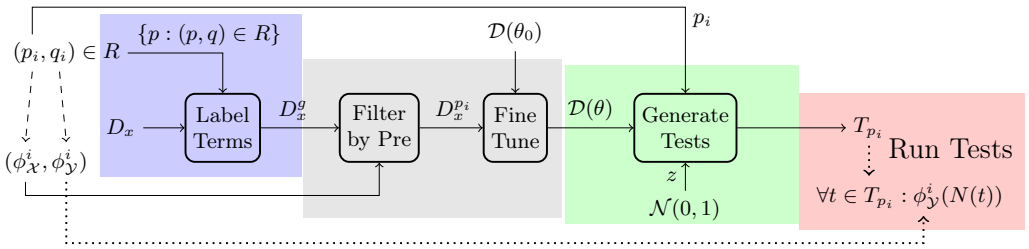

Fig. 1: The phases of RBT4DNN: (1) glossary term labeling (blue), (2) training a requirements conditioned generative model (gray), (3) generating a precondition-specific test suite (green), and (4) running tests to check the postcondition oracle (red).

Figure 1 outlines RBT4DNN, which takes structured natural language (SNL) requirements R for a learned component (LC) $N : X \to Y$ and generates test inputs T_{p_i} satisfying requirement preconditions p_i. These inputs are used to evaluate postconditions $\phi_{\mathcal{Y}}^i$, where violations indicate faults.

Requirements are represented as pairs (p, q), where both preconditions p and postconditions q are logical combinations of glossary terms G, encoding semantic features [21].

RBT4DNN operates in four phases:

1. **Glossary Term Labeling:** Annotate dataset D_x with semantic features (glossary) to obtain D_x^g.
2. **Precondition-Guided Training:** Filter data using p_i and fine-tune a generative model $\mathcal{D}(\theta_0)$.
3. **Test Generation:** Generate inputs via $\mathcal{D}(\theta)(p_i, z)$.
4. **Oracle Checking:** Evaluate $\forall t \in T_{p_i} : \phi_{\mathcal{Y}}^i(N(t))$.

The goal is to generate inputs that are *realistic*, *precondition-consistent*, and *diverse*.

2.1 Requirements

RBT4DNN supports SNL-based requirements expressed as logical combinations of semantic features. Terms correspond to Boolean features (e.g., "has feathers"), while continuous properties are discretized into Boolean ranges. Logical expressions are formed using conjunction, disjunction, and negation, aligning with requirements engineering practices [11,17,4,5,18].

Feature-based Robustness RBT4DNN supports semantic feature robustness, extending beyond pixel-level perturbations [32,33,34]. It allows expressing robustness as invariance of model outputs under variations of semantic features, either globally or under input constraints. In this work, we focus on conditional robustness, where robustness is evaluated over specific subsets of inputs defined by preconditions.

Semantic Feature Functional Requirements (SFFR) RBT4DNN also supports functional requirements that define acceptable input-output relations [11]. Preconditions describe semantic properties of inputs, while postconditions constrain outputs (e.g., class labels or control signals). This enables validation of both correctness and expected behavior across domains.

2.2 Glossary Term Labeling and Annotation

Glossary term labeling maps inputs for a given glossary G to semantic features as $gt : X \to 2^{E \times 2^G}$, where E is the set of entities present in the input. RBT4DNN is agnostic to labeling methods and supports multiple strategies:

- **Analytic Methods:** Use domain-specific feature extraction (e.g., morphometric analysis [35], scene graphs [36]).
- **VQA-based Labeling:** Use prompts (e.g., "Does the object have **g**?") and compute:

$$gt(x) = \{g \in G \mid vqa(prompt(g), x) = \text{yes}\}$$

These approaches enable scalable annotation across diverse datasets.

2.3 Training for RBT4DNN

Given the labeled dataset D_x^g, RBT4DNN constructs a precondition-specific training set:

$$D_x^{p_i} = \{(x, p_i) \mid (x, gt(x)) \in D_x^g \wedge \phi_{\mathcal{X}}^i(gt(x))\}$$

A text-conditional generative model is then fine-tuned using this filtered dataset. To improve efficiency, RBT4DNN adopts low-rank adaptation (LoRA) [27], which reduces training cost while maintaining generation quality.

This strategy enables learning from small, precondition-specific datasets, aligns training with requirement semantics, and improves consistency of generated inputs. As a result, RBT4DNN produces realistic, diverse inputs that satisfy requirement preconditions and support effective testing.

3 Evaluation

We evaluate RBT4DNN along two dimensions: (1) the quality of generated test inputs, and (2) their applicability for assessing model behavior and detecting faults. The evaluation spans four datasets (MNIST [37], CelebA-HQ [38], SGSM [39], and ImageNet [40]), multiple requirement types, and several learned components (LCs) with strong baseline performance.

To assess quality, we examine three properties: consistency with requirement preconditions, realism, and diversity. Results show that RBT4DNN consistently generates inputs that satisfy preconditions at a high rate, outperforming baseline test generation methods that do not target requirements. Realism is evaluated using distribution-based metrics, demonstrating that fine-tuned models produce more realistic samples compared to prompt-based generation, particularly for domains not well represented in pre-trained models. Diversity is measured via feature distributions, indicating that generated inputs closely match the diversity of training data under the same preconditions.

To assess applicability, we evaluate the ability of generated tests to detect faults and analyze model behavior. Results show that RBT4DNN effectively identifies requirement violations with low false positive rates and provides meaningful insights into model decision behavior. Unlike existing approaches, it enables targeted evaluation aligned with requirement semantics, improving the reliability of testing outcomes.

Further details of the evaluation and results can be found in [41].

4 Conclusion

RBT4DNN introduces the first requirements-driven test generation approach for neural networks, leveraging semantic feature-based preconditions and corresponding postconditions for validation. Our evaluation shows that it generates realistic, diverse, and precondition-consistent inputs that effectively reveal faults and unexpected behaviors in learned components. This work establishes semantic feature-based testing as a practical framework for validating learned systems and enabling requirement-guided analysis of model behavior.

References

1. I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, Deep learning. MIT press Cambridge, 2016, vol. 1.
2. E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," IEEE transactions on software engineering, vol. 41, no. 5, pp. 507–525, 2014.
3. M. Unterkalmsteiner, R. Feldt, and T. Gorschek, "A taxonomy for requirements engineering and software test alignment," ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 23, no. 2, pp. 1–38, 2014.
4. E. Uusitalo, M. Raatikainen, T. Männistö, and T. Tommila, "Structured natural language requirements in nuclear energy domain towards improving regulatory guidelines," in 2011 Fourth International Workshop on Requirements Engineering and Law. IEEE, 2011, pp. 67–73.
5. A. Veizaga, M. Alferez, D. Torre, M. Sabetzadeh, and L. Briand, "On systematically building a controlled natural language for functional requirements," Empirical Software Engineering, vol. 26, no. 4, p. 79, 2021.
6. J. Cleland-Huang, O. C. Gotel, J. Huffman Hayes, P. Mäder, and A. Zisman, "Software traceability: trends and future directions," in Future of software engineering proceedings, 2014, pp. 55–69.
7. S. Rayadurgam and M. P. E. Heimdahl, "Test-sequence generation from formal requirement models," in Proceedings Sixth IEEE International Symposium on High Assurance Systems Engineering. Special Topic: Impact of Networking. IEEE, 2001, pp. 23–31.
8. C. Nebut, F. Fleurey, Y. Le Traon, and J.-M. Jezequel, "Automatic test generation: A use case driven approach," IEEE Transactions on Software Engineering, vol. 32, no. 3, pp. 140–155, 2006.
9. M. W. Whalen, A. Rajan, M. P. Heimdahl, and S. P. Miller, "Coverage metrics for requirements-based testing," in Proceedings of the 2006 international symposium on Software testing and analysis, 2006, pp. 25–36.
10. C. Pecheur, F. Raimondi, and G. Brat, "A formal analysis of requirements-based testing," in Proceedings of the eighteenth international symposium on Software testing and analysis, 2009, pp. 47–56.
11. S. A. Seshia, A. Desai, T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, S. Shivakumar, M. Vazquez-Chanlatte, and X. Yue, "Formal specification for deep neural networks," in Automated Technology for Verification and Analysis: 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings 16. Springer, 2018, pp. 20–34.
12. M. Rahimi, J. L. Guo, S. Kokaly, and M. Chechik, "Toward requirements specification for machine-learned components," in 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW). IEEE, 2019, pp. 241–244.
13. A. Vogelsang and M. Borg, "Requirements engineering for machine learning: Perspectives from data scientists," in 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW). IEEE, 2019, pp. 245–251.
14. B. C. Hu, L. Marsso, K. Czarnecki, R. Salay, H. Shen, and M. Chechik, "If a human can see it, so should your system: Reliability requirements for machine vision components," in Proceedings of the 44th International Conference on Software Engineering, 2022, pp. 1145–1156.

15. B. C. Hu, L. Marsso, K. Czarnecki, and M. Chechik, "What to check: Systematic selection of transformations for analyzing reliability of machine vision components," in 2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2022, pp. 49–60.
16. F. Toledo, D. Shriver, S. Elbaum, and M. B. Dwyer, "Deeper notions of correctness in image-based dnns: Lifting properties from pixel to entities," in Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2023, pp. 2122–2126.
17. A. Mavin, P. Wilkinson, A. Harwood, and M. Novak, "Easy approach to requirements syntax (ears)," in 2009 17th IEEE International Requirements Engineering Conference. IEEE, 2009, pp. 317–322.
18. K. Großer, M. Rukavitsyna, and J. Jürjens, "A comparative evaluation of requirement template systems," in 2023 IEEE 31st International Requirements Engineering Conference (RE). IEEE, 2023, pp. 41–52.
19. A. v. Lamsweerde, Requirements engineering: from system goals to UML models to software specifications. John Wiley & Sons, Ltd, 2009.
20. K. Pohl, Requirements Engineering : Fundamentals, Principles, and Techniques. Springer, 2010.
21. C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Automated extraction and clustering of requirements glossary terms," IEEE Transactions on Software Engineering, vol. 43, no. 10, pp. 918–945, 2016.
22. N. Das, S. Chaba, R. Wu, S. Gandhi, D. H. Chau, and X. Chu, "Goggles: Automatic image labeling with affinity coding," in Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 2020, pp. 1717–1732.
23. A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: rapid training data creation with weak supervision," The VLDB Journal, vol. 29, no. 2, pp. 709–730, 2020.
24. A. V. Malawade, S.-Y. Yu, B. Hsu, H. Kaeley, A. Karra, and M. A. Al Faruque, "Roadscene2vec: A tool for extracting and embedding road scene-graphs," Knowledge-Based Systems, vol. 242, p. 108245, 2022.
25. R. Li, S. Zhang, D. Lin, K. Chen, and X. He, "From pixels to graphs: Open-vocabulary scene graph generation with vision-language models," in 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, 2024, pp. 28 076–28 086.
26. M. Kwon, J. Jeong, and Y. Uh, "Diffusion models already have a semantic latent space," in 11th International Conference on Learning Representations, ICLR 2023, 2023.
27. E. J. Hu, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen et al., "Lora: Low-rank adaptation of large language models," in International Conference on Learning Representations, 2022.
28. S. Dola, M. B. Dwyer, and M. L. Soffa, "Distribution-aware testing of neural networks using generative models," in 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, 2021, pp. 226–237.
29. D. Berend, X. Xie, L. Ma, L. Zhou, Y. Liu, C. Xu, and J. Zhao, "Cats are not fish: Deep learning testing calls for out-of-distribution awareness," in Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, 2020, pp. 1041–1052.
30. S. Dola, M. B. Dwyer, and M. L. Soffa, "Input distribution coverage: Measuring feature interaction adequacy in neural network testing," ACM Transactions on Software Engineering and Methodology, 2022. [Online]. Available: https://doi.org/10.1145/3576040

31. S. Dola, R. McDaniel, M. B. Dwyer, and M. L. Soffa, "Cit4dnn: Generating diverse and rare inputs for neural networks using latent space combinatorial testing," in Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, 2024, pp. 1–13.
32. Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in Proceedings of the 40th international conference on software engineering, 2018, pp. 303–314.
33. D. Demir, A. Betin Can, and E. Surer, "Test selection for deep neural networks using meta-models with uncertainty metrics," in Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis, 2024, pp. 678–690.
34. D. Huang, Q. Bu, Y. Fu, Y. Qing, X. Xie, J. Chen, and H. Cui, "Neuron sensitivity-guided test case selection," ACM Transactions on Software Engineering and Methodology, vol. 33, no. 7, pp. 1–32, 2024.
35. D. C. Castro, J. Tan, B. Kainz, E. Konukoglu, and B. Glocker, "Morpho-mnist: Quantitative assessment and diagnostics for representation learning," Journal of Machine Learning Research, vol. 20, no. 178, pp. 1–29, 2019.
36. T. Woodlief, F. Toledo, S. Elbaum, and M. B. Dwyer, "S3c: Spatial semantic scene coverage for autonomous vehicles," in Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, 2024, pp. 1–13.
37. Y. LeCun, "The mnist database of handwritten digits," http://yann. lecun. com/exdb/mnist/, 1998.
38. C.-H. Lee, Z. Liu, L. Wu, and P. Luo, "Maskgan: Towards diverse and interactive facial image manipulation," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.
39. F. Toledo, T. Woodlief, S. Elbaum, and M. B. Dwyer, "Specifying and monitoring safe driving properties with scene graphs," in 2024 IEEE International Conference on Robotics and Automation (ICRA), 2024, pp. 15 577–15 584.
40. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," International Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, 2015.
41. N. J. Mozumder, F. Toledo, S. Dola, and M. B. Dwyer, "Rbt4dnn: Requirements-based testing of neural networks," arXiv preprint arXiv:2504.02737, 2025.