[dstl] The Science Inside

The Utility of Neural Network Test Coverage Measures

Rob Ashmore, Alec Banks

SafeAI, February 2021

Ministry of Defence

UK OFFICIAL

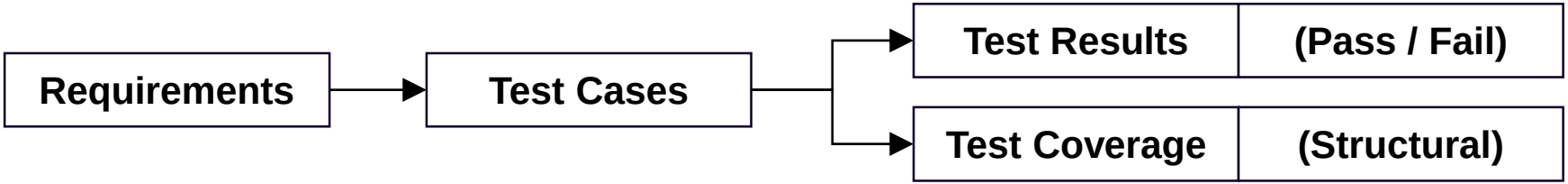
Disclaimer

- *This presentation is an overview of UK MOD sponsored research and is released for informational purposes only. The contents of this presentation should not be interpreted as representing the views of the UK MOD, nor should it be assumed that they reflect any current or future UK MOD policy. The information contained in this presentation cannot supersede any statutory or contractual requirements or liabilities and is offered without prejudice or commitment.*

Contents

Traditional Safety-Related Software Development

Requirements-Based Testing: Overview



- We want assurance that the software behaviour is fully described by the requirements
- We use the ***Requirements*** to derive ***Test Cases***, which are executed dynamically
- We measure both the result of each test and how the collection of all tests covers the software structure
  - For example: Statement, Branch, Modified Condition / Decision Coverage [1]

- Incomplete code causes test failure; incomplete requirements may not be detected
- If tests pass, but do not achieve full coverage then, either:
  - Software behaviour is correct, requirements are incomplete
  - Requirements are correct, software includes additional unnecessary behaviour
  - Software behaviour and requirements are both correct, test set is incomplete

Requirements-Based Testing: Discussion

- 100% Requirements-Based Testing (RBT) for traditional software is based on:
  - A ***complete set of requirements***, which are (for example) accurate, consistent, unambiguous and verifiable
  - Which can be ***decomposed in a hierarchical, traceable manner***, so that low-level test results can confidently aggregate to high-level software behaviour
  - To “units”, where ***behaviour is closely linked to code structure***, behaviour that is strongly data-dependent should be captured as a specific requirement

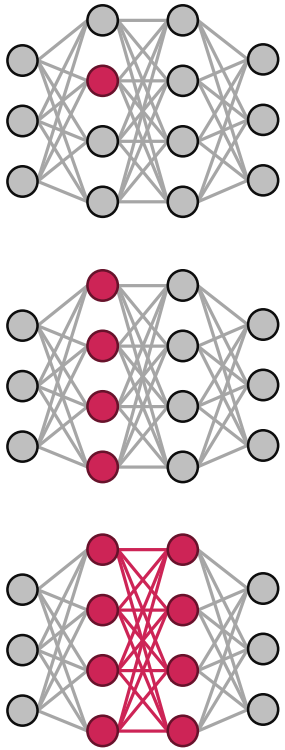
The combination of RBT and structural test coverage provides confidence that the requirements suitably describe the software's behaviour, both what it does and what it does not do

  - Note: 100% RBT and full structural coverage does not guarantee error free code

Neural Network Coverage Measures

Neural Network Coverage Measures

- Typical neural network coverage measures are related to neuron activations
- Historically, there has been a progression:
  - From measures that consider neurons in isolation [2]
  - Through measures that consider neurons within each layer [3], [4]
  - To measures that consider how neurons in one layer affect neurons in the subsequent layer [5]
- These approaches are reminiscent of traditional software coverage measures, with the neuron as the “low-level behavioural unit”



Towards Traditional Coverage for Neural Networks

Traditional Software -vs- Neural Networks

[dstl] The Science Inside

Traditional Software Aspects	Neural Network Context
There is a complete set of requirements	Neural networks solve “open” problems; there is not a set of requirements that is accurate, complete, unambiguous and verifiable
Requirements can be hierarchically decomposed	Some level of decomposition may be possible, but requirements are not decomposed to a level that can directly be coded against
Behaviour is closely linked to code structure	The link between structure (i.e. neuron activation) and behaviour is indirect and not well-understood
Coverage can be measured efficiently	*Information on individual neuron activations may not be readily available from highly optimised hardware*

None of the aspects that make test coverage “work” for traditional software apply to neural networks

Partially Meeting the Test Coverage Aims

[dstl] The Science Inside

Software satisfies requirements:

- Robustness to adversarial examples [6]
- Disciplined approach to requirements specification [7]
- Transformation of abstract domains through typical activation functions [8]
- Encoding logic constraints as part of the network training process [9]

Requirements cover behaviour:

- Tests cover input, operational, failure and adversarial domain spaces [10]
- Detection and mitigation of backdoor attacks [11]
- Automatic (formal) inference of network properties [12]

Current techniques allow the two main aims of requirements-based test coverage to be partially met for neural networks

Utility of Neural Network Coverage Measures

Utility of Neural Network Coverage Measures

- Neural network coverage measures provide additional utility, beyond the aims of requirements-based test coverage for traditional software:
  - They could be used to ***optimise training data*** (e.g. whether there is value in collecting additional training data)
  - They could be used to ***compare training and (independent) verification data sets***, specifically from the network's perspective
  - They could be used to ***generate additional inputs***, which could meaningfully extend the training data set
  - They could be used to ***choose between different neural networks***, for similar levels of performance, select the better covered network
  - They could be used to ***monitor runtime behaviour*** (e.g. by benchmarking activation patterns)

Conclusions

Conclusions

- Current neural network test coverage measures ***cannot provide the same level of confidence*** as requirements-based test coverage does for traditional software

- But, there are some approaches that ***can partially meet the aims of traditional coverage*** in the context of neural networks

- And, the neural network test coverage measures are ***valuable in other ways***

References

[1] RTCA. 2011. DO-178C: Software Considerations in Airborne Systems and Equipment Certification.

[2] Pei, K.; Cao, Y.; Yang, J.; and Jana, S. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. arXiv.

[3] Ma, L.; Juefei-Xu, F.; Sun, J.; Chen, C.; Su, T.; Zhang, F.; Xue, M.; Li, B.; Li, L.; Liu, Y.; Zhao, J.; and Wang, Y. 2018. DeepGauge: Comprehensive and Multi-Granularity Testing Criteria for Gauging the Robustness of Deep Learning Systems. arXiv 1803.07519.

[4] Ma, L.; Juefei-Xu, F.; Xue, M.; Li, B.; Li, L.; Liu, Y.; and Zhao, J. 2019. DeepCT: Tomographic Combinatorial Testing for Deep Learning Systems. In Proceedings of the 26th IEEE International Conference on Software Analysis, Evolution and Reengineering, 614–618. IEEE.

[5] Sun, Y.; Huang, X.; Kroening, D.; Sharp, J.; Hill, M.; and Ashmore, R. 2019. Structural Test Coverage Criteria for Deep Neural Networks. ACM Transactions on Embedded Computing Systems (TECS) 18(5s): 94.

[6] Huang, X.; Kwiatkowska, M.; Wang, S.; and Wu, M. 2017. Safety Verification of Deep Neural Networks. In Proceedings of the 29th International Conference on Computer Aided Verification, 3–29.

[7] Banks, A. and Ashmore, R., 2019. Requirements Assurance in Machine Learning. In SafeAI@ AAAI.

[8] Singh, G.; Gehr, T.; P¨uschel, M.; and Vechev, M. 2019. An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages 3(POPL): 1–30.

[9] Fischer, M.; Balunovic, M.; Drachsler-Cohen, D.; Gehr, T.; Zhang, C.; and Vechev, M. 2018. DL2: Training and querying neural networks with logic.

[10] Ashmore, R.; Calinescu, R.; and Paterson, C. 2019. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. arXiv preprint 1905.04223.

[11] Wang, B.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; and Zhao, B. Y. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In 2019 IEEE Symposium on Security and Privacy (SP), 707–723. IEEE.

[12] Gopinath, D.; Converse, H.; Pasareanu, C. S.; and Taly, A. 2019. Property Inference for Deep Neural Networks. arXiv 1904.13215v2.

[dstl] The Science Inside

Discover more

UK OFFICIAL