收 件 记
日期： 收件人：

申　请　书

申请人姓名：孙昌爱

所在单位：　北京科技大学

申请课题：　智能软件系统的数据驱动式的测试技术与方法

申请金额：

年限起止：

填报日期：　2018 年 11 月 13 日

申 请 课 题 介 绍

一、研究的意义、预期成果、应用前景及国内外发展综述

1. 研究的意义及国内外发展综述

智能软件系统是指能够产生人类智能行为的软件系统。近年来，随着计算机的计算能力日益强大和机器学习算法不断完善，智能软件系统通过学习或者自适应等方式获得处理问题的规则并在诸多任务（例如：驾驶汽车[1]、图像识别[2-3]、语音识别[4]等）中表现良好甚至达到了人类的水平。强大的学习能力以及在多个任务中的良好表现使得智能软件在很多系统（例如：自动驾驶汽车[1]、检测恶意软件[5]、飞机避碰[6]）中扮演重要角色。在这些系统中尤其是安全攸关的系统（例如无人驾驶系统），智能软件的准确性和可靠性受到极高的关注，其原因是智能软件系统不正确的行为或者微小的偏差可能造成致命的事故。例如，谷歌的无人驾驶汽车和一辆公共汽车相撞，原因是它希望公共汽车在一系列罕见的条件下停车，然而实际上公共汽车不可能停止[7]；特斯拉的一辆无人驾驶汽车和一辆拖车相撞，原因是拖车的外表颜色和天空相近并且底盘较高[8]。由此可见，如何有效地保证智能软件系统的正确性以及预测的准确性已经成为智能软件应用、开发领域的一个亟待解决的问题。

软件测试是一种广泛采用的软件质量保证手段[9]，通过运行有限的测试用例，比较测试用例的输出与预期输出是否一致来检测软件中潜藏的故障。与测试传统软件相比，测试智能软件系统更具有挑战性：（1）**测试充分性无法保证：**智能软件系统的输入空间十分巨大，例如无人驾驶汽车通过各种传感器获取无数种可能的外界信息。目前的测试技术主要通过两种方式获得测试数据：（a）**随机生成：**可以生成大量的数据，但是不能发现大多数系统不正确的行为并且会出现数据不具备实际意义的现象（例如在测试人脸识别系统中生成“紫色眼睛”的图片）；（b）**人工生成：**该方式通过在普通的测试数据的上做微小改动来获得新的测试数据，并且需要测试人员判断每一个新测试数据对应的系统行为；（2）**测试用例不可靠：**智能软件系统的测试数据可能不准确并且基于学习的智能软件系统的学习模型经常出现过拟合的现象，出现不正确或者不被期待的行为；（3）**难以判定系统行为是否正确：**目前主要的判定有两种：（a）将系统的行为与执行数据对应的标签对比；（b）测试人员判断。这两种方式都需要人工参与，然而在长时间的工作下，资深测试工程师的准确性也不能得到保证。（4）**传统软件的测试技术不可用：很多**智能软件的运行逻辑通过训练带标签的数据获得（例如基于学习的智能软件系统）而不是程序开发人员指定，这使得传统的软件测试技术测试智能软件系统的效果可能不好。由于上述问题，智能软件测试在测试充分性评估、测试数据生成、测试结果判定等方面面临新的问题与挑战。

近年来，围绕智能软件系统的测试用例生成、测试结果判定、测试方法等问题，研究者提出了一系列的智能软件测试技术。部分代表性研究工作包括：

（1） **测试用例生成方面：**测试用例生成是软件测试的一个重要步骤，直接影响软件测试方法的故障检测能力。智能软件的测试用例生成技术主要有：（a）**随机生成测试用例：**随

批注 [付1]：还需要再总结提炼一下
输入空间巨大所以就难以保证充分性了么？
我从你的描述中，我感觉到测试充分性难以保证的原因是：已有的测试用例生成技术难以实现某种测试准则的覆盖（如你提到的“不正确行为的发现”，这有点类似于变异测试中的“变异覆盖”），再深一点，**缺乏覆盖准则指导（驱动）的测试用例生成技术**。

批注 [付2]：我现在有点不太明白这一点，为什么说测试用例不可靠会导致测试困难呢？是不是因为这些不可靠的测试用例不能用于测试呢，或是因为这些不可靠的测试用例难以寻找，

批注 [付3]：这一部分有点类似于我写的内容中的测试结果难以判定，我打算**把你这一部分和我的测试结果判定部分进行融合**。

批注 [付4]：对应上面提到的第一点

批注 [付5]：这个是对应第二点？

批注 [付6]：以下部分我将纳入智能软件测试的研究现状中

机地在测试用例集中挑选测试用例或者随机地生成测试数据具有简单、易用的特点。该方法被谷歌等世界著名公司使用来生成无人驾驶车的测试数据[10-12]。然而，随机的方式没有利用任何系统内部信息以及测试过程信息，使得很多智能系统不正确的行为不能被发现[13-14].**（b）人工生成测试用例：**该方式通过在普通测试数据的基础上做微小改动来得到新的测试数据[15-16]，然后测试人员判断每一个新测试数据对应的行为。这种方式成功地运用在多种智能软件系统中[17-21]。随机生成和人工生成测试用例的方式都没有考虑智能软件系统的内部结构，不能覆盖智能软件系统的大部分逻辑，导致了这些测试用例只能发现少量的系统异常行为。Pei 等人通过经验研究发现：随机生成一个测试用例几乎能够覆盖无人驾驶车的所有代码，但是只激活了不到 10%的神经元。基于上述观察，Pei 等人提出了神经元覆盖指标，评估测试用例集激活的神经元数目[14]。Tian 等人利用神经元覆盖指标，生成测试用例，激活智能系统大多数的神经元，并通过经验研究的方式验证该方法生成的测试用例集可以发现更多智能系统不正确的行为。

（2） **测试结果判定方面：**目前智能软件测试工作验证系统的行为是否正确的方法大致可以分为两类：（a）将系统行为与测试用例对应的预期进行对比或者测试人员判断系统的行为是否正确[22-28]。该方法严重依赖测试人员的参与，并且在测试用例或者系统复杂时测试人员极有可能出现误判的情况。（b）一些智能软件测试的研究者通过采用间接的方式判断系统的行为是否正确，减少测试的时间开销，避免出现测试人员误判的情况：Pei 等人利用交叉引用技术[29-33]，将同一个测试数据在不同的无人驾驶车上测试，观察所有无人驾驶车的行为，如果其中一辆车的行为与其它不一样，说明其中一辆车的行为可能是错误的[14]。Tian 等人将蜕变关系[34-35]作为一种判断系统行为是否正确的依据，如果两个测试数据的行为不符合蜕变关系，说明系统中存在缺陷[36]。

参考文献

[1] V. Rausch, A. Hansen, E. Solowjow, C. Liu, E. Kreuzer, J. K. Hedrick. Learning a Deep Neural Net Policy for End-to-End Control of Autonomous Vehicles. *Proceedings of the 2017 American Control Conference (ACC'17)*, IEEE Computer Society, 2017, pp. 4914-4919.

[2] K. He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'16)*, IEEE Computer Society, 2016, pp. 770-778.

[3] M. A. O. Vasilescu, D. Terzopoulos. Multilinear Image Analysis for Facial Recognition. Proceedings of the 16th In International Conference on Pattern Recognition (ICPR'02), IEEE Computer Society, 2002, pp. 511-514.

[4] W. Xiong, J. Droppo, X. Huang, F. Seide, M. L. Seltzer, A. Stolcke, G. Zweig. Toward Human Parity in Conversational Speech Recognition. *Proceedings of the IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP'17)*, IEEE Computer Society, 2017, pp. 2410-2423.

[5] Z. Yuan, Y. Lu, Z. Wang, Y. Xue. Droid-Sec: Deep Learning in Android Malware Detection. *Proceedings of the In ACM SIGCOMM Computer Communication Review (CCR'14)*, ACM

Press, 2014, pp. 371-372.
[6] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, M. J. Kochenderfer. Policy Compression for Aircraft Collision Avoidance Systems. Proceedings of the 14th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC'16), IEEE Computer Society, 2016, pp. 1-10.
[7] http://www.theverge.com/2016/2/29/11134344/google-selfdriving-car-crash-report, A Google Self-Driving Car Caused a Crash for The Frst Time.
[8] https://electrek.co/2016/07/01/understandingfatal-tesla-accident-autopilot-nhtsa-probe/, Understanding the Fatal Tesla Accident on Autopilot and the NHTSA Probe..
[9] G. J. Myers, C. Sandler, T. Badgett. The Art of Software Testing. John Wiley and Sons, 2011.
[10] https://www.dmv.ca.gov/portal/wcm/connect/946b3502-c959-4e3b-b119-91319c27788f/GoogleAutoWaymodisengagereport2016.pdf?MOD=AJPERES, Google Auto Waymo Disengagement Report for Autonomous Driving.
[11] https://www.theatlantic.com/technology/archive/2017/08/insidewaymos-secret-testing-and-simulation-facilities/537648/, Inside Waymo's Secret World for Training Self-Driving Cars.
[12] I. H. Witten, E. Frank, M. A. Hall, and J. P. Christopher. Data Mining:Practical Machine Learning Tools and Techniques. *Journal of Management Science, Ubon Ratchathani University*, 2005, 3(6): 92-96.
[13] http://www.cleverhans.io/security/privacy/ml/2017/06/14/verifcation.html, The Challenge of Verifcation and Testing of Machine Learning.
[14] K. Pei, Y. Cao, J. Yang, S. Jana. Deepxplore: Automated Whitebox Testing of Deep Learning Systems. *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP'17)*, ACM Press, 2017, pp. 1-18.
[16] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, A. Swami. The Limitations of Deep Learning in Adversarial Settings. *Proceedings of the 1st IEEE European Symposium on Security and Privacy (EuroS&P'16)*, IEEE Computer Society, 2016, pp. 372-387.
[17] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, A. Criminisi. Measuring Neural Net Robustness with Constraints. *Proceedings of the 10th Annual Conference on Neural Information Processing Systems (NIPS'16)*, 2016, pp. 2613-2621.
[18] N. Carlini, D. Wagner. Towards Evaluating the Robustness of Neural Networks. *Proceedings of the IEEE Symposium on Security and Privacy (SP'17)*, IEEE Computer Society, 2017, pp. 39-57.
[19] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, A. Criminisi. Measuring Neural Net Robustness with Constraints. *Proceedings of the International Conference on Machine Learning (ML'17)*, IEEE Computer Society, 2017, pp. 2613-2621.
[20] S. Gu, L. Rigazio. Towards Deep Neural Network Architectures Robust to Adversarial Examples. *Proceedings of the International Conference on Learning Representations (ICLR'15)*, IEEE Computer Society, 2015, pp. 777-780.
[21] X. Huang, M. Kwiatkowska, S. Wang, M. Wu. Safety Verification of Deep Neural Networks. *Proceedings of the International Conference on Computer Aided Verification (CAV'17)*, Springer, 2017, pp. 3-29.

[22] J. H. Metzen, T. Genewein, V. Fischer, B. Bischoff. On Detecting Adversarial Perturbations. *Proceedings of the International Conference on Learning Representations (ICLR'17)*, IEEE Computer Society, 2017, pp. 701-713.

[23] N. Papernot, P. McDaniel. Extending Defensive Distillation, *arXiv preprint arXiv:1705.05264*.

[24] N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami. Distillation as A Defense to Adversarial Perturbations Against Deep Neural Networks. *Proceedings of the IEEE Symposium on Security and Privacy (SP'16)*, IEEE Computer Society, 2016, pp. 582-597.

[25] U. Shaham, Y. Yamada, S. Negahban. Understanding Adversarial Training: Increasing Local Stability of Neural Nets Through Robust Optimization, arXiv preprint arXiv:1511.05432.

[26] W. Xu, D. Evans, Y. Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks, arXiv preprint arXiv:1704.01155.

[27] Y. Zhang, C. Mu, H. W. Kuo, J. Wright. Toward Guaranteed Illumination Models for Non-Convex Objects. *Proceedings of the IEEE International Conference on Computer Vision (ICCV'13)*, IEEE Computer Society, 2013, pp. 937-944.

[28] S. Zheng, Y. Song, T. Leung, I. Goodfellow. Improving the Robustness of Deep Neural Networks Via Stability Training. *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'16)*, IEEE Computer Society, 2016, pp. 4480-4488.

[29] G. Argyros, I. Stais, S. Jana, A. D. Keromytis, A. Kiayias. SFADiff: Automated Evasion Attacks and Fingerprinting Using Black-Box Differential Automata Learning. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (CCS'16), ACM Press, 2016, pp. 1690-1701.

[30] C. Brubaker, S. Jana, B. Ray, S. Khurshid, V. Shmatikov. Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations. *Proceedings of the 35th IEEE Symposium on Security and Privacy (S&P'14)*, IEEE Computer Society, 2014, pp. 114-129.

[31] Y. Chen, T. Su, C. Sun, Z. Su, J. Zhao. Coverage-Directed Differential Testing of JVM Implementations. *Proceedings of the ACM SIGPLAN Notices*, ACM Press, 2016, pp. 85-99.

[32] Y. Chen, Z. Su. Guided Differential Testing of Certificate Validation in SSL/TLS Implementations. *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering (FSE'15)*, ACM Press, 2015, pp. 793-804.

[33] W. M. McKeeman. Differential Testing for Software. *Digital Technical Journal*, 1998, 10(1): 100-107.

[34] T. Y. Chen, S. C. Cheung, S. M. Yiu. Metamorphic Testing: A New Approach for Generating Next Test Cases. Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, 1998.

[35] S. Segura, G. Fraser, A. B. Sanchez, A. R. Cortes. A Survey on Metamorphic Testing. *IEEE Transactions on software engineering*, 2016, 42(9): 805-824.

[36] Y. Tian, K. Pei, S. Jana, B. Ray. Deeptest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars. *Proceedings of the 40th International Conference on Software Engineering (ICSE'18)*, ACM Press, 2018, pp. 303-314.

二、研究采取的技术路线与方法

1. 研究内容

在智能软件系统测试领域前沿研究成果基础上，围绕智能软件系统测试中测试充分性评估、测试用例生成与测试结果判定等关键问题开展研究，探索智能软件系统的测试框架与优化技术。主要研究内容如下：

（1） **智能软件系统的数据驱动式的测试框架**

智能软件测试的主要步骤包括测试用例生成、测试执行与测试结果判定。测试用例生成与测试结果判定是智能软件测试领域尚未有效解决的开放问题。智能软件系统要求具有类似人类的智能行为，且大多数智能软件需要和外界进行频繁的交互，导致智能软件系统的输入域异常庞大、系统复杂性较高。生成有效的、多方位测试智能软件系统的测试用例是一项具有挑战性的工作。由于智能软件系统的执行逻辑复杂，测试结果判定更加困难，在某些情况下甚至无法确定测试预期。现有的智能软件测试技术利用随机或人工的方式生成测试用例并且主要依靠测试人员验证系统行为，导致测试效率可能不高。另一方面，现有的智能软件测试工作忽略了测试过程对测试效率的影响。在测试资源有限的情况下，利用数据驱动的方式（基于测试的历史信息），控制测试过程，提高测试效率。**研究智能软件系统的数据驱动式测试时，首先需要建立面向智能软件系统的数据驱动式测试框架**。研究如下问题：

- **智能软件系统获取执行逻辑的方式与故障特点：**与传统软件相比，智能软件系统的执行

  逻辑不是由开发人员指定，并且不同智能软件系统获取执行逻辑的方式有可能不同，相应地，故障类型与特点也不同。
- **智能软件系统的故障检测机制：**测试用例生成是智能软件系统测试的关键步骤。在测试过程中，为了增加发现系统行为异常的概率，生成的测试用例应当考虑系统的内部结构。智能软件系统的故障检测不仅要关注测试用例的生成方式，还要考虑如何尽可能地覆盖不同的执行逻辑。

（2） **智能软件系统的高效、多样性测试用例生成**

在真实世界中，外界环境的多样性、多元化以及多变性等特征导致智能软件系统的输入空间异常庞大。例如，谷歌和特斯拉等知名公司通过随机或者人工的方式生成测试用例，探索无人车的输入空间。然而研究者认为随机生成测试用例的方式没有利用任何智能软件系统的内部结构和测试过程信息，并且由于生成数据的过程是无指导、不受控制的，产生的数据可能不具有实际意义。因此，该方式产生的测试用例集不可能发现很多系统的异常行为，并且只能探索较少的智能系统的规则。人工生成测试用例是在普通测试用例的基础上增加微小改动。由于测试资源有限，这种严重依赖测试人员的方式生成的测试用例数目有限，并且也只能探索较少的智能软件系统的规则。研究以下内容：

- **如何生成高效的测试用例集：**增加测试用例集的多样性有助于更有效地检测智能软件系统的异常行为。为了提高智能软件的故障检测效率，探索面向智能软件系统的覆盖准

批注 [付7]：我感觉这三个内容的顺序需要换一换，
1. 测试用例生成
2. 测试结果判定
3. 测试充分性评估

批注 [付8]：对于这一部分，我打算作为研究内容的第一个

但是有一个问题，这里面好像没有体现数据驱动

批注 [付9]：这些是智能软件系统测试的困难与挑战。我感觉可以针对这些问题与挑战去设置针对性的研究内容，而不是作为我们论述“研究智能软件系统测试框架的意义或必要性”的依据。其原因如下：一方面，我们要**说出智能软件系统独有的特点，这些独有的特点是传统软件测试中所从未考虑的。在这种情况下，我们有必要针对于智能软件系统研究新型测试框架**。另一方面，这些问题在传统软件系统的测试中也存在（只不过这些问题在智能软件系统的测试中更加严重），相应的，一定存在一些技术手段来解决这些问题。如果我们没办法说明这些技术不能（或很难）应用于智能软件系统，那我们 “建立测试框架”的研究内容实际上是站不住脚的。所以，**最好是从智能软件系统独有的特点出发，说明建立新型测试框架的重要性**。
Why?
，

批注 [付10]：所以，我觉得这里缺点东西：如果扣着智能软件系统与传统软件系统的区别来叙述，会更好。原因如下：正是因为这些区别与不同的存在，所以，我们要研究针对于（适用于）智能软件系统的测试框架

批注 [付11]：你的意思是不是“学习（或构建）业务逻辑的方式”？

批注 [付12]：我感觉这部分更像是测试用例的改进与优化与测试充分性的提升，而非故障检测机制。原因：我理解的故障检测机制时判断是否有故障的机制，在传统软件测试中，故障检测机制是比较预期输出与实际输出，对于蜕变测试的话，就是看蜕变关系是不是满足。我感觉你的意思是要针对于智能软件系统提出新型故障检测机理，但是下面论述的内容与我理解的故障检测机制不相关。

批注 [付13]：我个人认为在研究内容里面就少提现状及问题吧，因为这些放在“研究的意义及国内外发展综述”比较好。

我建议重点写“针对…问题，依据…思路，研究….，具体如下”

批注 [付14]：我觉得这部分内容分两点写比较好，一个是提出新型覆盖准则（我记得你之前提到过现有的覆盖准则无法满足智能软件系统的测试覆盖度量），之后再写，测试覆盖准则指导的测试用例生成技术（或方法）

则，以及生成满足不同覆盖准则的测试用例生成方法。

（3） **测试用例集充分性的评估指标**

与传统软件相比，智能软件系统的执行逻辑不是由开发人员指定，大部分是通过训练数据获得。这种差异使得传统软件测试的覆盖标准（例如代码覆盖和分支覆盖等）可能不适合智能软件系统测试。另一方面，无人驾驶系统的研究者通过经验研究的方式发现一个随机的输入几乎能够覆盖所有的程序代码。因此，智能软件系统测试用例集的评估指标应结合系统的内部结构和特点，提出更有效的覆盖准则。

（4） **智能软件系统的执行结果的判定方式**

智能软件系统的测试工作主要通过测试人员判断系统行为是否正确，这种方式不仅需要大量的测试资源，还易出现误判的情况（将错误的行为判断为正确的行为）。因此，提出自动化验证系统行为的方法，可以极大提高测试效率与准确度。研究以下问题：

- **提出一种自动或者直观的方式判断系统行为是否正确：**人工验证系统行为的方式不仅需要占用大量的测试资源而且不能保证判断的准确度。因此，提出一种自动地验证系统行为的方式是有必要的。传统软件测试中缓解测试预期问题的技术有很多（例如：N版本、断言等），其中一些技术运用到智能软件测试可能效果不好。例如，断言技术需要在代码中插入一些可以获取执行信息的代码，但很多智能软件系统的执行逻辑不是由程序控制的。考虑将蜕变关系以及交叉验证的的方式作为验证系统行为的机制。
- **（如果利用蜕变关系作为测试预期）如何识别有效的蜕变关系：**识别蜕变关系是蜕变测试的核心问题。智能软件系统的执行逻辑比较复杂，并且在测试的过程中应当尽可能多的覆盖不同的执行逻辑。因此，需要识别可以尽可能多的覆盖不同执行逻辑的蜕变关系。

（5） **智能软件系统与环境实时交互的可靠性验证**

很多智能软件系统通过各种传感器获取外界环境信息，将环境信息作为输入传递给决策模块，然后决策模块根据内部逻辑控制智能软件系统的行为。准确地辨别、分析外界事物是智能软件系统做出正确决策的前提。如果不能准确识别可能带来安全隐患的事物，智能软件系统行为的正确性就得不到保证。例如，2016年一辆特斯拉公司的无人车与一辆拖车在行驶中相撞，造成一人死亡。原因是拖车的颜色是白色并且底盘较高，导致无人车将拖车视为天空。因此，验证无人车具有可靠的辨别外界事物能力对提高无人车的安全性具有重要作用。研究如下问题：

- **智能软件系统识别外界事物的可靠性：**在实际环境中，很多事物易于和周围环境混淆。例如，无人车主要通过摄像机获取路况和周围车辆等关键信息的二维图像，这种信息呈现方式使得易于环境混淆的事物更加难以区别。如果无人车不能区分这些事物与周围环境的差别，就可能出现事故。
- **智能软件系统将外界信息转化为决策模型输入的准确性：**智能软件系统将获取的外界信息进一步加工得到决策模块可以直接使用的数据。在这个过程中，外界信息与处理后的信息的等价性难以保证。

批注 [付15]：从我这儿现在的理解，我觉得这部分内容合并到上一部分更好

批注 [付16]：我对这个术语的理解和你的有些不一样

我看到这个题目，想到的研究问题是：智能软件系统与环境交互的实时性应当得到可靠的保障。为此，我们需要测试智能软件系统与环境交互是不是满足某种时效，即在一定的时间内系统是否能够分析决策并做出一定的响应。这个分两个层面，一是能不能在确定的时间做出响应，这在某种程度上反映系统实时交互的可靠性；二是在做出响应的前提下，能否做出正确的响应，这也在反映系统交互的可靠性。

我看完下面的论述后，我认为你想说的是智能系统是否能够准确可靠地识别外界事物，即第二个层面的事情，而没描述第一个层面的事情。

我个人认为第一个层面比较重要，当然，第二个层面也很有意义。第二个层面主要与系统内部人工智能（或其他机制）的业务逻辑是否正确（或可靠）有密切关系，相当于是“系统核心”是否靠谱，而第一个层面，着重于“系统核心”与“外界环境”之间的那一部分，即数据输入输出的部分，这部分应当也是测试人员需要关注的重点（这个点我之前是没有想到的，应该作为研究内容的一个重要部分。）