딥러닝 기반 안드로이드 GUI 테스팅 자동화 프레임워크

김종우[01] 허윤제[1] 박예지[1] 이지민[1] 손준익[1] 유준범[1]
[1]건국대학교 컴퓨터공학부
whddn112@konkuk.ac.kr, dbswp93@konkuk.ac.kr, tuilte38@gmail.com, fuzetea@konkuk.ac.kr,
sji6227@konkuk.ac.kr, jbyoo@konkuk.ac.kr

Deep-learning Based Android GUI Testing Automation Framework

JongWoo Kim[01] YoonJe Heo[1] YeJi Park[1] JiMin Lee[1] JunIk Son[1] JunBeom Yoo[1]
[1]Department of Computer Science and Engineering, Konkuk University

요 약

GUI 테스팅(GUI Testing)은 애플리케이션(앱) 화면의 그래픽 인터페이스를 조작했을 때 프로그램이 원하는 방식으로 동작하는지 검증하는 기법이다. 안드로이드는 운영체제 특성상 다양한 플랫폼 및 버전이 존재하므로 프로그램의 코드가 필요하지 않은 GUI 테스팅 기법이 활발히 연구되고 있다. 이에 본 연구에서는 딥러닝을 기반으로 안드로이드 애플리케이션에 대한 GUI 테스팅 프레임워크를 제시하였다. Object Detection을 통하여 화면의 오브젝트를 자동으로 인식하여 오브젝트에 대한 Action을 취하고 Test Model을 생성하였다. 그를 바탕으로 한 안드로이드 디바이스에서 생성한 Test Set을 통해 화면 해상도, 안드로이드 버전이 다른 세 디바이스에서 테스팅을 수행하였다. 실험 결과, Test Model의 기대되는 예측 Model과의 유사성은 최소 98%를 기록하였고, 테스트 수행 디바이스 에서 각 11개의 테스트 케이스 중 11개, 11개의 테스트 케이스 중 10개, 11개의 테스트 케이스 중 11개를 통과하였다. 따라서 멀티 플랫폼(Multi-Platform), 멀티 디바이스(Multi-Device)에서의 자동화된 GUI 테스팅이 구현되었다고 볼 수 있다.

1. 서 론

GUI 테스팅이란 화면을 구성하는 요소들에 클릭 등 액션을 취해 프로그램이 올바르게 행동하는지 검증하는 기법이다[1].

안드로이드라는 운영체제의 특성상, 안드로이드 애플리케이션은 다양한 기기와 플랫폼에서 사용되고 있다. 이를 검증하기 위해서는 여러 기기 및 플랫폼에서 해당 애플리케이션의 모든 기능이 수행된다는 것을 보장할 수 있어야 한다. 또한 테스팅 과정에서 디바이스의 해상도와 플랫폼 화면 구성에 따라 달라지는 오브젝트에 대응해야 한다. GUI 테스팅 기법은 해당 애플리케이션의 코드가 필요하지 않기 때문에 플랫폼이 달라지거나 애플리케이션 자체의 버전이 달라질 때도 테스팅이 가능하다. 현재 대중적으로 사용되는 테스팅 기법은 대표적으로 세 가지가 있다.

수동 테스트(Manual Testing)는 소프트웨어 기능의 정상 작동 여부를 수동으로 테스트하는 기법이다. 모든 테스트를 수동으로 진행하기 때문에 시간과 비용의 소모가 크다. Record-Playback 기법은 소프트웨어 작동 과정을 스크립트 형태로 기록 후 재현하는 기법이다. 대상 프로그램 동작에 따라 테스트 케이[1]스를 각각 생성해야 한다는 단점이 있다.[2]. 정적 분석 (Static Analysis) 기법은 소스코드와 메타 데이터를 정적 분석하여 Test Model을 생성하는 기법이다. 안드로이드 플랫폼은 버전 간 호환성을 지원하지 않으므로, 플랫폼이나 앱의 버전이 달라질 경우 코드와 메타 데이터를 다시 분석해야만 Test Model을 생성할 수 있다.[3].

기존 테스팅 기법들의 인력, 비용 문제를 보완하기 위해 본 연구에서는 딥러닝을 기반으로 한 GUI 테스팅 자동화 프레임워크를 제시한다. 해당 프레임워크는 딥러닝을 통해 화면의 오브젝트를 자동으로 인식 후 동작하도록 하여 여러 기기 및 플랫폼에 적용되는 Test Model을 생성한다. 해당 Test Model의 루트 노드로부터, 리프 노드까지의 path를 하나의 Test Case로 하는 Test Set을 생성한다. 이 Test Set들을 Test Script의 형태로 가공하고, 다른 기기에서 테스트를 수행하고 결과를 기록할 수 있도록 했다. 본 프레임워크의 효율성 검증을 위해 3개의 안드로이드 디바이스에서 사례연구를 진행하였고, 각 100%, 90.9%, 100%의 테스트 수행 정확도를 보였다.

2. 배경 지식

2.1 Faster R-CNN

딥러닝 기반 Object Detection에는 주로 R-CNN 신경망 구조가 사용된다. 본 연구에서 사용한 Faster R-CNN은 R-CNN의 성능을 개선한 모델이다. 우수한 정확성과 속도를 가지고 있으므로 실시간 Object Detection을 수행하기에 적합하다고 평가받고 있다[4].

2.2 Transfer Learning

딥러닝에 필요한 모델을 새로 구축하는 것은 비용이 많이 들거나 불가능한 경우가 많다[5]. 따라서 본 연구에서는 기존에 학습된 오픈소스 Object Detection API를 활용해 추가로 학습시키는 전이학습(Transfer Learning) 기법을 사용하였다.

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학지원사업의 연구결과로 수행되었음(2018-0-00213)

2.3 OCR(Optical Character Recognition)

광학 문자 인식이란 스캐너를 통해 입력된 영상에서 문자에 해당하는 부분을 인식하여 기계가 이해할 수 있는 문자로 변환하는 기술이다[6]. 본 연구에서는 Tesseract-ocr을 사용하였다.

3. 안드로이드 GUI 테스팅 자동화 프레임워크

본 프레임워크의 동작은 크게 3가지로 구분된다. 첫째, 타겟 앱의 Menu Tree를 생성한다. 각 오브젝트들에 대한 액션을 수행하며 화면 및 오브젝트들의 정보를 노드에 추가한다. 둘째, 생성된 Menu Tree를 기반으로 Test Script를 생성한다. Menu Tree의 루트 노드로부터 리프 노드까지가 한 개의 path가 되고 Menu Tree에서 가능한 모든 path를 커버하는 Test Case들을 생성한다. 해당 Test Case들을 다른 디바이스에서 테스트할 수 있도록 Test Script로 만들어준다. 이를 바탕으로 테스트를 수행할 디바이스의 화면과 생성된 Menu Tree의 노드를 비교하여 각 테스트 케이스들에 대한 결과를 확인한다.

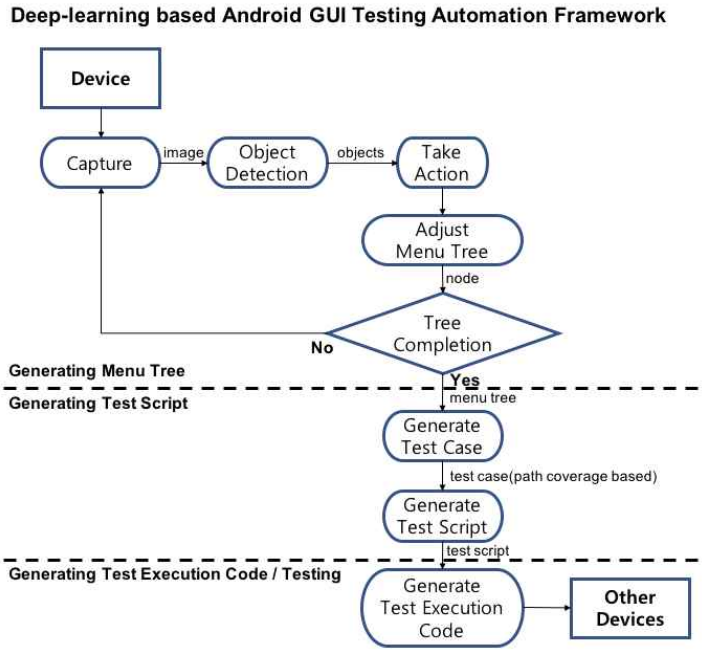

그림1. 프레임워크 작동절차 도식화

3.1 Generating Menu Tree

3.1.1 Object Detection

이미지 분류는 오브젝트를 눌렀을 때의 예상 액션을 바탕으로 버튼(Button), 입력창(TextField), 체크박스(CheckBox), 라디오버튼(RadioButton)의 네 가지로 분류하였다. 버튼 3513개, 입력창 227개, 체크박스 260개, 라디오 버튼 83개로 총 326장의 이미지를 사용하였다. 학습 모델은 MS COCO Resnet v101을 사용하였다. 학습은 Tensorflow 딥러닝 API를 사용해 30,000 스텝 진행했으며 전이학습 후 모델의 정확도는 IoU[1] 75% 이상 기준 89.4%이다.

3.1.2 Adjust Menu Tree

Pseudo code of Menu Tree Generating(mkTree)

```
Input: Node
Output: Generated Menu Tree
FOR i = 0 TO number of obj
    click(obj[i])
    IF screen transited
        IF equal to ancestor's screen
            back to previous screen
            CONTINUE
        ELSE
            add new node to the tree as child
            RETURN mkTree(newNode)
    ELSE CONTINUE
IF tempnode is root
    menu tree generating completed
ELSE
    RETURN mkTree(tempNode.getParent)
```

3.2 Generating Test Script

생성된 Menu Tree에 근거하여 Test Script를 도출한다. 앞선 과정을 통해 생성된 Test Case는 Menu Tree에서 가능한 모든 경로에 대해서 테스팅하는 패스 커버리지 기반이므로 GUI 기반의 앱 테스팅에 적합한 신뢰도를 충족할 수 있다. DFS 방식을 사용해 Menu Tree를 순회해 만들어진 Test Case를 다른 디바이스에서 테스팅 할 수 있도록 Test Script로 가공한다.

3.3 Generating Test Execution Code / Testing

Test Script를 기반으로 다른 디바이스에서 자동으로 테스트를 수행할 수 있는 Test Execution Code를 만든다. Execution Code는 디바이스를 제어하고 기존에 생성된 Menu Tree와 테스트 수행 디바이스의 화면을 비교하는 작업을 포함한다. 테스트는 디바이스 해상도와 플랫폼 화면 구성에 따라 달라지는 각 오브젝트의 크기, 비율, 모양, 폰트를 커버해야 하므로 Execution code 생성 단계에서 오브젝트 리사이징, OCR(Optical Character Recognition)을 적용하여 Pass 또는 Fail을 판단한다.

먼저 각각 오브젝트 이미지의 원본을 사용해 비교를 시도하고, 매칭이 되지 않을 경우 화면 비율에 맞게 오브젝트 이미지를 리사이징 하여 비교, 판단한다. 나인패치 버튼(9-Patch Button) 등 가변 비율을 가진 오브젝트는 위의 두 경우에서 판단할 수 없으므로 OCR을 이용한다. OCR을 이용해 오브젝트에서 텍스트를 검출해 비교, 판단한다. 마지막으로, 세 경우 모두 해당하지 않을 때는 다른 화면으로 판단한다.

4. 사례 연구

본 연구에서는 제안한 자동화 프레임워크를 적용하기 위해 간단한 상용 앱을 대상으로 실험을 수행하였다. 실험 환경은

1) Intersection over Union: 모델이 예측한 결과와 실제 결과 간에 겹치는 영역

다음과 같다.

앱 화면	약 40개
오브젝트	화면당 2~20개

표1. 실험용 어플리케이션

실험에 사용한 대상 어플리케이션은 평균 2번의 동작으로 어플리케이션의 종단 화면으로 이동할 수 있고, 실험을 통해 Menu Tree를 생성하였을 때 3의 Depth를 가지고 있는 결과를 나타냈다. Object를 클릭하여 Node를 추가할 때, 분류한 Class에 따른 예측 Model과의 유사성을 최소 98% 확보할 수 있었다. 이를 통해 Test Script를 생성했을 때, 전체 메뉴트리의 리프 노드 수 만큼의 테스트케이스가 생기고, 각 테스트 케이스는 루트 노드로부터 리프 노드까지의 n개의 Node와 n-1개의 Edge를 가지고 있는 형태로 생성된다.

Android Device for Generating Test Model		
디바이스 1	안드로이드 버전	7.1.1
	모델 번호	SM-N950N
Android Device for Testing		
디바이스 2	안드로이드 버전	7.0
	모델 번호	SM-A510K
디바이스 3	안드로이드 버전	8.0.0
	모델 번호	SM-G930K
디바이스 4	안드로이드 버전	4.4.2
	모델 번호	SHV-E330K

표2. 실험에 사용한 안드로이드 기기

디바이스 1에서 Test Model을 생성한 후 다른 디바이스에서 Execution Code를 사용해 테스팅하였다.

	정확도
디바이스 2	100% (11/11)
디바이스 3	90.9% (10/11)
디바이스 4	100% (11/11)

표3. 실험 결과

앞선 사례 연구를 통해 제안한 프레임워크가 멀티 플랫폼 및 멀티 디바이스 환경에서 유의미한 신뢰도를 보이고 있음을 알 수 있다. 그러나 오브젝트가 애니메이션인 경우 화면을 캡쳐하는 시점에 따라 크기와 모양이 달라져 다른 오브젝트로 인식해 결과에 영향을 미칠 수 있다는 문제를 가지고 있다. 이는 향후 애니메이션 오브젝트에 대한 추가적인 고려를 통해 보완될 것으로 기대한다.

5. 결론 및 향후 연구

본 연구에서는 딥러닝을 기반으로 안드로이드 애플리케이션에 대한 GUI 테스팅 프레임워크를 제시하였다. Object Detection을 통하여 화면의 오브젝트를 자동으로 인식하여 동작하게 하였고 그를 바탕으로 한 디바이스에서 생성한 Test Set을 화면 해상도가 다른 세 디바이스에서 실행하였을 때 Pass라는 결과를 나타내었다. 따라서 멀티 플랫폼(Multi-Platform), 멀티 디바이스(Multi Device)에서의 자동화된 GUI 테스팅이 구현되었다고 볼 수 있다. 또한 테스팅을 함에 있어 사용자가 직접 수행해야 하는 부분이 최소화되었으며 테스팅에 소요되는 비용을 감소시킴으로써 기존 테스팅 기법의 단점을 보완하였다.

향후 연구에서는 오브젝트 간 의존성이 있는 경우(RadioButton, CheckBox), 예상치 못한 이벤트(Incoming Call, SMS 수신 등)에 대하여 대응할 수 있는 알고리즘을 개발하는 것이 과제이다.

참 고 문 헌

[1] G. Bae, G. Rothermel, and D. H. Bae, “Comparing Model-based and Dynamic Event-Extraction Based GUI Testing Techniques: An Empirical Study,“ The Journal of Systems and Software, 2014.

[2] 박두호 외 3인, “Record-PlayBack 기반의 안드로이드 소프트웨어 테스트 케이스 생성방안”, 한국컴퓨터종합학술대회 제38권 제1호(B), 2011

[3] 신원, “안드로이드 애플리케이션의 GUI 테스트를 위한 정적분석 기반 테스트 모델 및 테스트케이스 생성”,건국대학교, 2013

[4] Shaoqing Ren 외 3인, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS, 2015

[5] Sinno Jialin Pan and Qiang Yang Fellow, “A Survey on Transfer Learning”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 2009.

[6] Won Gyo Jung, “Optical character recognition system using the document form identification”, 고려대학교, 2008.