Journal of Korea Multimedia Society Vol. 21, No. 11, November 2018(pp. 1305-1316)
https://doi.org/10.9717/kmms.2018.21.11.1305

심층신경망 기반의 객체 검출 방식을 활용한 모바일 화면의 자동 프로그래밍에 관한 연구

윤영선[†], 박지수[††], 정진만[†††], 은성배[††††], 차 신[†††††], 소선섭[††††††]

Automatic Mobile Screen Translation Using Object Detection Approach Based on Deep Neural Networks

Young-Sun Yun[†], Jisu Park[††], Jinman Jung[†††], Seongbae Eun[††††], Shin Cha[†††††], Sun Sup So[††††††]

ABSTRACT

Graphical user interface(GUI) has a very important role to interact with software users. However, designing and coding of GUI are tedious and pain taking processes. In many studies, the researchers are trying to convert GUI elements or widgets to code or describe formally their structures by help of domain knowledge of stochastic methods. In this paper, we propose the GUI elements detection approach based on object detection strategy using deep neural networks(DNN). Object detection with DNN is the approach that integrates localization and classification techniques. From the experimental result, if we selected the appropriate object detection model, the results can be used for automatic code generation from the sketch or capture images. The successful GUI elements detection can describe the objects as hierarchical structures of elements and transform their information to appropriate code by object description translator that will be studied at future.

Key words: Automatic Programming, Gui Elements Detection, Sketch Image to Code, Deep Neural Networks, Object Detection

1. 서 론

그래픽 사용자 인터페이스(Graphical User Interface; GUI)는 컴퓨터 사용 시 기능이나 정보를 화면의 그림표시나 틀, 색상과 같은 그래픽 요소들을 이용하여 표시하거나 전달하는 인터페이스를 말한다[1-3]. GUI 없는 컴퓨팅 환경은 생각하기 어려울 정도로, GUI는 컴퓨터 및 모바일 사용 환경과 밀접한 관계를 가지고 있으며, GUI 요소의 설계 및 관련 작업은 프로그램 개발에서 매우 중요한 역할을 하며,

※ Corresponding Author : Young-Sun Yun,
Address: (34430) 70 Hannam-ro, Daedeok-gu, Daejeon, Korea, TEL : +82-42-629-7569, FAX : +82-42-629-7843, E-mail : ysyun@hnu.kr
Receipt date : Aug. 30, 2018, Revision date : Oct. 2, 2018
Approval date : Oct. 16, 2018
[†] Dept. of Computer, Communications, and Unmanned Tech., Hannam University
[††] Dept. of Information and Communication Eng., Hannam University (E-mail : jisuve.6@gmail.com)
[†††] Dept. of Computer, Communications, and Unmanned Tech., Hannam University (E-mail : jmjung@hnu.kr)
[††††] Dept. of Computer, Communications, and Unmanned Tech., Hannam University
(E-mail : sbeun@hnu.kr)
[†††††] Dept. of Computer, Communications, and Unmanned Tech., Hannam University
(E-mail : scha@hnu.kr)
[††††††] School of Computer Engineering, Kongju National University
(E-mail : triples@kongju.ac.kr)
※ This work was supported by 2018 Hannam University Research Fund.

다른 플랫폼 상으로 이전할 경우에는 동일한 UI를 가지도록 반복적인 작업을 진행하게 된다.

초기 GUI 설계 작업의 경우, 디자이너가 생각하는 UI와 실제 플랫폼에서 제공하는 UI가 다르기 때문에 온라인이나 오프라인으로 설계한 GUI 요소들을 서로 다른 플랫폼 상의 화면에서 확인하거나, 이미 개발된 화면을 다른 시스템에서 제공하는 UI 등으로 변환하여 소프트웨어 개발을 지원할 필요성이 존재한다. 이런 목적으로 GUI 요소의 화면 배치나 위젯 등을 검출하여 각 플랫폼에 맞는 형식의 코드를 자동 생성하는 연구가 진행되고 있다. 대표적으로 많이 고려하는 환경은 스크립트에 의하여 배치 정보가 텍스트 정보로 생성될 수 있는 안드로이드, iOS, HTML 기반의 모바일 환경이다[4-6].

본 연구의 최종 목적은 GUI 요소를 초기 설계할 때나 이미 개발된 화면을 이전할 때에 도움을 줄 수 있도록 자동으로 GUI 기술 언어를 각 플랫폼에 맞는 코드로 자동 생성하는 것이다. 연구의 최종 목표를 달성하기 위하여 기존 캡처 화면이나 수작업에 의하여 그려진 화면으로부터 GUI 요소를 검출하는 기술이 필요하며, 본 논문은 GUI 요소를 검출하는 부분을 개선하였다. 기존의 연구 방식이 전통적인 OCR 방식과 이미지 분석 등의 방식을 이용하여 생성 규칙을 정형화하거나 경험적 방식에 의하여 GUI 요소를 검출하는데 반하여, 본 논문에서는 심층 신경 회로망 방식을 이용한 객체 검출 기법을 이용하여 GUI 요소를 검출하였다. 제안된 방식은 기존의 정형화된 규칙 구성이나 경험적 방식에 의하지 않고, 수집된 데이터를 이용하여 GUI 요소를 검출하기 때문에 보다 직관적이며 기존 방식에 비하여 높은 자유도를 보인다. 본 논문의 구성은 2장에서 GUI 요소 검출에 관련된 연구에 대하여 설명하며, 3장에서는 본 연구에서 제안한 시스템을, 4장에서는 실험 및 결과를 제시하고 5장에서 결론을 맺는다.

2. GUI 요소 검출 관련 연구 및 심층 신경 회로망 기반의 객체 검출 기법

본 장에서는 GUI 요소 검출을 위한 기존의 규칙 및 경험 기반의 관련 연구와 본 논문에서 적용한 심층 신경 회로망 기반의 객체 검출 기법에 대하여 기술한다.

자동 코드 생성 연구를 위한 대표적인 GUI 요소 검출 연구는 형식 문법(formal grammar)을 이용하여 객체를 모델링하거나[7,8], 사용자와 GUI 객체들 간의 상호 작용을 모델링하기 위하여 수행되었다[10]. 최근에는 심층 신경 회로망 기법을 이용하여 GUI 요소가 위치된 이미지를 텍스트로 변환하는 기법을 이용하여 GUI 요소를 기술하는 방법도 제안되었다[11].

Parag[7] 등은 그래픽 객체를 추출하기 위하여 직선, 텍스트, 원 등의 기본 특징들을 이용하였으며, 검출된 결과를 1차 논리형식으로 기술하였다. 기본 특징들을 결합하고 기하학적 정보를 추가하여 사각형, 단어 등과 같은 비 단말 기호(non-terminal symbol)로 변환하였다. 이들 비 단말기호가 모여 그룹박스, 누름버튼(push button), 라디오버튼(radiobutton)과 같은 그래픽 요소들로 생성된다. 이와 달리 Nguyen과 Pham 등은 그래픽 객체의 분포 등으로 고려하여 OCR(Optical Character Recognition) 기법을 적용하여 텍스트 정보를 먼저 검출하였다. 윤곽선 처리를 통하여 객체의 대략적인 위치를 반영한 후 윤곽선 확장 및 검출 과정을 통하여 GUI 요소를 검출하는 방법을 사용하였다[8, 9]. 이들 연구에서는 GUI 요소의 분포, 아이콘의 형태와 방향, 복합 요소의 형태와 방향, GUI요소의 상대적인 크기를 반영한 비율 등을 반영하여 그래픽 요소를 검출하였다. Nguyen은 GUI 요소들을 계층적으로 분류하여 컨테이너, 리스트, 텍스트 등 순으로 기술하였으며, Pham 연구는 GUI 요소를 검출하는 것뿐만 아니라 그 변화를 추적하여 대응되는 GUI 요소들 간의 상호작용 순서에 따라 그 의미를 파악하고자 하였다.

윤곽선 추출이나 기본 그래픽 요소를 추출한 후 기본 특징들을 합성하는 구문론적 방법이외에 Beltrameli는 신경회로망을 이용한 캡셔닝(image captioning) 방식을 적용하여 GUI 요소를 모바일 화면을 구성하는 코드를 생성하였다[10]. 이미지 캡셔닝은 이미지로부터 텍스트 설명을 자동으로 생성하는 방법으로, 컴퓨터 비전과 자연어 처리 기법을 동시에 이용한다. 일반적인 방식은 컨볼루션 신경회로망(Convolutional Neural Network; CNN)을 암호화기(Encoder)로 모델링하고, 이미지를 CNN에 입력하여 특징을 추출하며, 추출된 특징은 재귀 신경 망(Recurrent Neural Network)으로 구성된 복호화기

(Decoder)로 전달된다. 이미지를 설명하는 문장들의 각 단어를 시간적 순서에 따라 재귀 신경망에서 학습시켜 최종적으로 학습되지 않는 이미지가 입력될 경우 해당되는 문장을 출력하는 방식이다. Beltrameli는 언어 모델을 위하여 재귀 신경망으로 1997년에 제안되어 널리 사용되는 LSTM(Long short-term Memory)[11]을 적용하였다.

이들 연구들은 기본적인 특징에 의하여 GUI 요소를 계층적 정보에 이용하여 수작업에 의하여 객체 구조를 정형화하거나, 텍스트와 이미지의 조합과 윤곽선 정보에 의하여 사용자 패턴의 분석과 같이 목적이 다르거나, 많은 경험적 정보의 사용으로 인한 일반화 모델링의 미흡, 전체 이미지로부터 문장을 생성하기 때문에 복잡한 GUI 요소들의 배치를 처리하지 못한다는 단점이 존재하여(Table 1. 참조), 새로운 방식을 이용한 GUI 요소의 검출 필요성이 대두되었다.

제안하는 방식은 심층 신경회로망을 이용하되, 이미지 캡셔닝 방식 대신 객체 검출(object detection) 방식을 적용하여 GUI 요소를 검출하고자 한다. 객체 검출 방식은 이미지나 비디오에서 특정 클래스의 객체를 검출하는 방식으로 객체의 위치(localization)와 분류(classification)를 동시에 진행하는 방식이다. 일반적으로 이미지 분류는 단일 객체나 동일 유형이 이미지 전체를 나타내고 있기 때문에, 이미지를 구성하는 객체보다는 전체 이미지 속성을 파악하는 것이며, 객체 검출은 한 이미지에 동일 객체가 몇 개 있는지를 파악하는 것과 같이 객체의 유무 판별과 객체의 이미지 분류를 동시에 진행하는 방식이다. 이미지 분류에 사용하기 위해서는 각 이미지 당 해당하는 속성 또는 태깅 정보를 필요로 하지만, 객체 검출에서는 해당하는 객체의 위치와 클래스 정보(속성, 또는 태깅 정보)를 요구하며, 단일 이미지에 여러 종류의 객체를 표현하기도 한다.

초기의 객체 검출 방식은 별도의 전처리 과정에서 검출 영역을 추출한 후 컨볼루션 신경회로망을 이용하여 객체를 검출하였다[12]. 이를 개선하기 위하여 제안된 R-CNN(Region-based Convolutional Neural Network) 방식은 선택 검색(selective search) 방식을 이용하여 입력 이미지에 대하여 영역을 제시하고, 각각에 대하여 CNN을 적용하여 SVM(Support Vector Machine)을 이용한 영역 분류와 객체의 크기를 조정하는 선형 회귀 방식을 적용하였다[13]. R-CNN 방식은 직관적이었지만 속도가 느리다는 단점이 있어 이를 개선하고자 각각의 영역에 대한 CNN 대신 전체 이미지에 대하여 하나의 CNN을 실행하고, SVM 대신 softmax 층으로 대체하는 Fast R-CNN 방식이 제안되었으며[14], 검출 속도를 더욱 향상시키기 위하여 선택 검색 알고리즘을 신경망으로 대체하는 Faster R-CNN이 제안되었다[15]. 이와 다른 방법으로 검출 속도를 향상시키기 위하여 전체 이미지를 균일한 격자로 분할하고 각각의 격자에 대하여 객체가 존재할 신뢰도를 계산하고, 경계 상자의 위치를 조정함으로써 하나의 네트워크가 특징 추출 및 경계 상자 생성 및 클래스를 분류하는 YOLO (You Only Look Once) 방식이 제안되어 획기적인 속도 향상이 이루어졌다[16]. YOLO와 비슷한 방식으로 영역 제안 및 지역 분류 방식을 단일 프로세스로 처리한 SSD(Single-Shot Detector) 방식도 제안

Table 1. Comparison of related works

Works	Key ideas	Weakness
Grammar based object modelling[7]	- hierarchical object description of its parts by semantic relationships - represent the object with primitive features	- manually describe object structure with hierarchical information - primitive features are varied along to image qualities
Mining Visual log of SW[8]	- use OCR & vision system - main object is finding the user interactions with GUI elements	- focused on visual log of user interactions - texts, icons, and mixed forms are considered
REMAUI[9]	- use OCR & vision system - merge and identify images with heuristics	- too many rule-based heuristics
Pix2code[10]	- use image captioning based on deep neural networks	- cannot handle the complex structures - lack of detail information

되어 정확도와 객체 검출 속도에 따라 조절할 수 있게 되었다. SSD 방식은 다층의 컨볼루션 층에서 경계 상자 정보를 분산시켜 서로 다른 크기의 객체를 검출할 수 있도록 하면서 성능과 정확도를 향상시켰다[17]. 이후 작은 객체에 대하여 미흡한 성능을 보이던 YOLO의 정확도를 향상시키면서 속도를 그대로 유지하는 YOLOv2가 제안되었으며[18], YOLO와 YOLOv2의 최종 층에서만 경계 상자 정보와 클래스 정보를 저장하는 방식을 개선하여 3개의 검출 층으로 검출 정보를 분산한 YOLOv3가 제안되었다[19]. 본 연구에서는 직관적인 구조를 이용하여 빠른 검출 속도를 보이는 YOLOv2와 YOLOv3를 이용하여 GUI 요소를 검출하는데 사용하였다.

3. 제안 시스템

본 연구의 최종 목표는 모바일 환경의 화면이나 웹 화면을 스크린 캡처 또는 손으로 그린 간략한(스케치) 이미지로부터 각 플랫폼에 맞는 UI 코드를 자동 생성하는 시스템을 구축하는 것이다. 그 사전연구로서 화면 캡처나 스케치 이미지에서 GUI 요소를 검출한다. 본 연구의 최종 목표 시스템을 Fig. 1에 제시하였으며, 본 논문 연구에 해당하는 부분은 굵은 선으로 표시하였다.

3.1 시스템 구성

전체 시스템은 크게 GUI 요소를 검출하여 중간 단계의 객체 기술 언어(ODL; Object Description Language)로 표현하는 부분과 각 UI 정보를 이용하여 최종 플랫폼의 UI 표현(XML)으로 변환하는 부분으로 구성된다. GUI 요소는 화면 구성에 따라 다른 GUI 요소를 포함할 수 있는 컨테이너와 다른 요소를 포함하지 못하는 컴포넌트로 구성되며, 컨테이너는 컨테이너를 포함할 수 있다. 따라서 GUI 요소를 단순히 검출하는 단계로 최종적인 GUI 레이아웃을 생성할 수 없기 때문에 객체의 계층 구조를 형성하여 표현하여야 한다. 생성된 기본 계층 구조는 적용하고자 하는 플랫폼에 따라 분류, 모델 정보를 이용하여 최종적으로 목적 플랫폼의 언어로 변환되어야 한다. 본 연구에서는 스케치 이미지나 화면 캡처된 이미지로부터 GUI 요소를 검출하는 부분에 초점이 맞춰져 있다.

3.2 GUI 요소 결정

GUI 요소 검출을 위하여 객체 정의와 데이터를 수집하여야 한다. GUI 요소 검출을 위하여 안드로이드 환경에서 널리 사용되는 앱을 다운로드하여 각각의 GUI 요소들의 분포를 파악한 결과 총 577개의 위젯(GUI 요소)로부터 텍스트, 버튼, 이미지 뷰로 구성되는 기본(Basic) 컴포넌트가 63%, 복합(Composite) 컴포넌트는 약 12%, 컨테이너(Container) 컴포넌트는 약 25%의 분포를 보였다[20]. 이 분포 정보를 이용하여 최종적으로 상위 7개의 컴포넌트를 검출하도록 하였다. 컨테이너 정보는 다른 컴포넌트가 포함

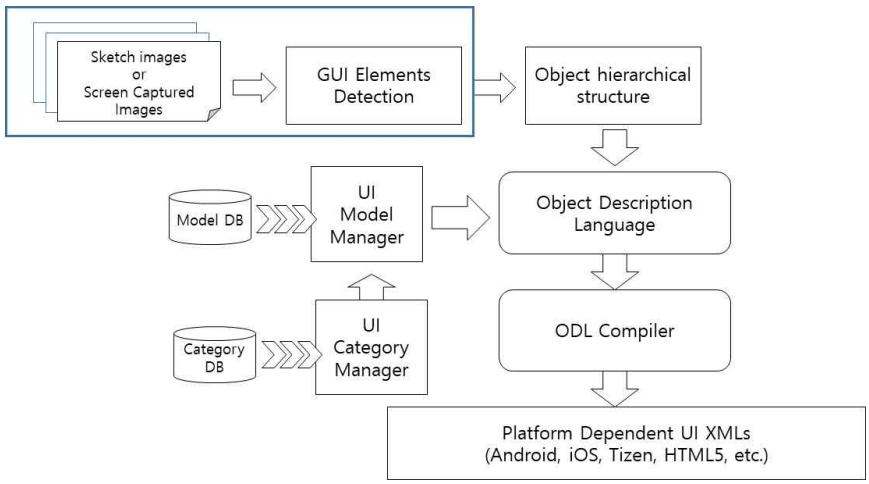


Fig. 1. Overall system and focused research module (boxed area) on this paper.

Table 2. Target GUI elements and corresponding distribution and rankings in [20]

GUI elements	Category	Distribution (%)	Ranking
TextView	Basic	33.69	1
Button	Basic	26.69	2
ListView†	Container	23.05	3
CheckButton	Composite	4.51	4
EditText	Composite	3.12	5
ImageView	Basic	2.43	6
Spinner	Composite	2.25	7
RadioButton	Composite	1.91	8

† The ListView is not chosen as target because of its category.)

되어있기 때문에 현 단계에서는 컨테이너를 검출하지 않고 추후 연구에서 진행하기로 하였다. Table 2에 최종 선정된 검출 대상 GUI 요소와[20]에서 파악된 분포를 표시하였다. 분석된 GUI 요소에서 List View는 이미지와 텍스트가 하나의 리스트로 복합적으로 구성되며, 여러 개의 리스트를 포함하고 있기 때문에 컨테이너에 해당되어 최종 GUI 요소에 포함시키지 않았다.

3.3 자동 코드 생성

계층적 구조를 갖는 GUI 요소들의 관계가 기술된다면 각 플랫폼에 해당하는 GUI 표현 방식으로 코드가 생성되어야 한다. 현재 널리 사용되는 iOS나 안드로이드의 경우 XML 형식으로 GUI 정보가 저장되며, 타이젠의 경우 웹 저장 형식의 경우 HTML5 형식으로 저장된다. 동일한 XML 형식으로 코드가 저장된다고 하더라도 각 플랫폼의 특성을 반영하여 표현되기 때문에, 검출된 GUI 요소는 계층적 정보를 갖는 객체 표현 언어의 형식으로 저장되고, 각 플랫폼과 GUI 모델 정보를 반영하여 최종 플랫폼에 맞는 형식으로 생성되어야 한다. 예를 들어 안드로이드 환경의 경우 각 화면은 배치도(layout)가 분리되어 각 화면마다 독립된 배치 정보를 가지고 있으나, iOS 환경의 경우에는 하나의 스토리보드에 관련된 모든 배치도가 포함된다. Fig. 2에 안드로이드와 iOS 환경의 GUI 기술 표현방식을 비교하였다.

Fig. 2에서 살펴본바와 같이 안드로이드의 경우 개별 배치도에 대한 XML 파일만 기술되기 때문에 여러 배치도가 조건에 따라 기술되는 iOS에 비하여 상대적으로 단순한 형태로 표시된다. 즉, GUI 요소를 계층적 정보에 의하여 기술되더라도 배치되는 형태에 따라 그 복잡도는 증가하게 된다. 따라서 복잡도를 고려하여 본 연구에서는 안드로이드와 iOS의 경우 단일 배치도에 의한 GUI 요소만을 대상으로 각 플랫폼에 대응되는 UI XML 표현의 자동 생성을 목적으로 하였으며, 추후 연구에서 이를 구현할 계획이다.

4. 실험 결과 및 분석

본 장에서는 GUI 요소 검출을 위하여 신경회로망에 기초한 객체 검출 기법을 스케치 이미지에 적용한 후, 그 결과를 비교 분석한다.

4.1 데이터 수집

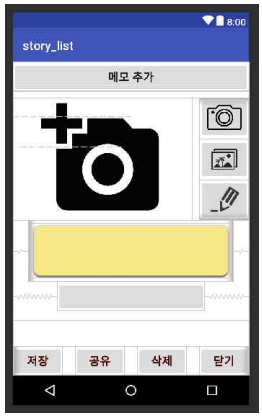
GUI 요소 검출을 위하여 안드로이드 환경에 익숙한 대학생들을 중심으로 10개의 정해진 앱의 형식을 지정하고 Table 2에서 정의한 7개의 GUI 요소를 이용하여 스케치하도록 하였다. 각 GUI 요소의 분포를 고려하여 총 280개의 화면을 그렸으며, 그 중에서 224개의 화면은 학습용으로 56개의 화면은 평가용으로 사용하였다. Table 3에 수집된 스케치 이미지의 분포를 요약하였다. 수집된 GUI 요소의 분포와 구글 스토어에서 다운로드 받아 분석한 컴포넌트의 분포를 비교하면, TextView와 Button을 포함하여 매우 유사함을 알 수 있다.

4.2 데이터 기술

수집된 스케치 이미지를 심층 신경 회로망을 이용하여 GUI 요소를 검출하기 위한 사전 작업으로 각 화면에서 컴포넌트(GUI 요소)의 이름과 위치를 저장하여야 한다. 본 연구에서는 github에 공개된 Label Img[21] 를 사용하여 GUI 요소의 이름과 위치를 지정하였으며, 일관성을 위하여 한명에 의하여 작업을 진행하였다. Fig. 3은 LabelImg를 이용하여 GUI 요소의 이름과 위치를 저장하는 과정을 보인다.

4.3 GUI 요소 검출 및 비교

GUI 요소를 검출하기 위하여 수집된 데이터를 이용하여 darknet 기반의 YOLO 시스템[22]을 python 기반으로 재구축하여 실험에 사용하였다. Darknet



(a) design layout on Android

(b) XML description on Android

(c) storyboard layout on iOS

(d) XML description on iOS

Fig. 2. Comparison of typical UI design screen on Android and iOS mobile environments.

Table 3. Distribution of GUI elements

		Occurrence			
		for training	for test	Sub total	Percentage(%)
Layouts (Screen)		224	56	280	-
GUI elements (Components)	TextView	816	232	1,048	667.5
	Button	700	162	862	549.0
	CheckButton	145	41	186	118.5
	EditText	184	66	250	159.2
	ImageView	165	27	192	122.3
	Spinner	106	51	157	100.0
	RadioButton	89	18	107	68.2
	Total	2,234	584	2,818	1,794.9

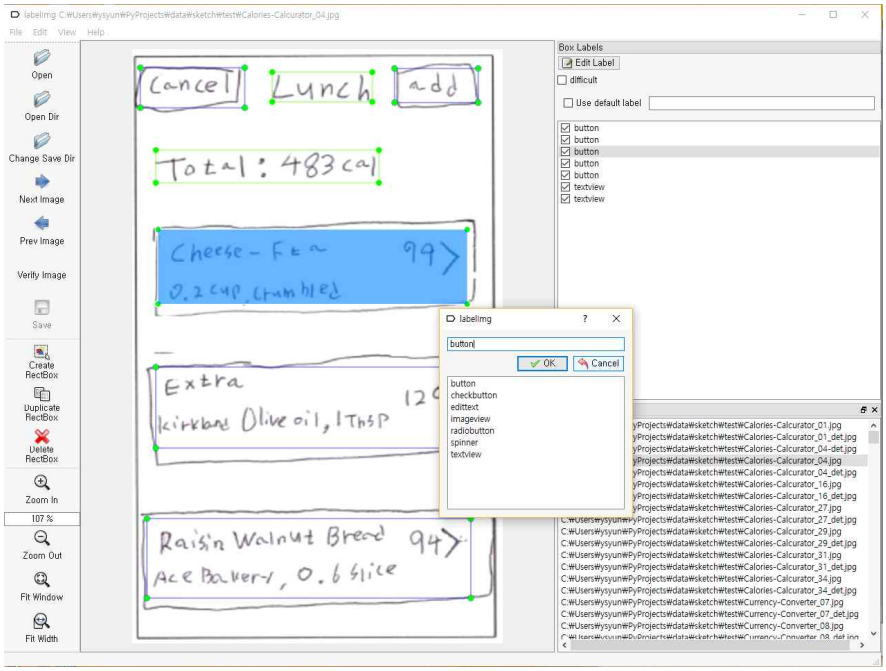


Fig. 3. Annotation process for sketch image using LabelImg [21].

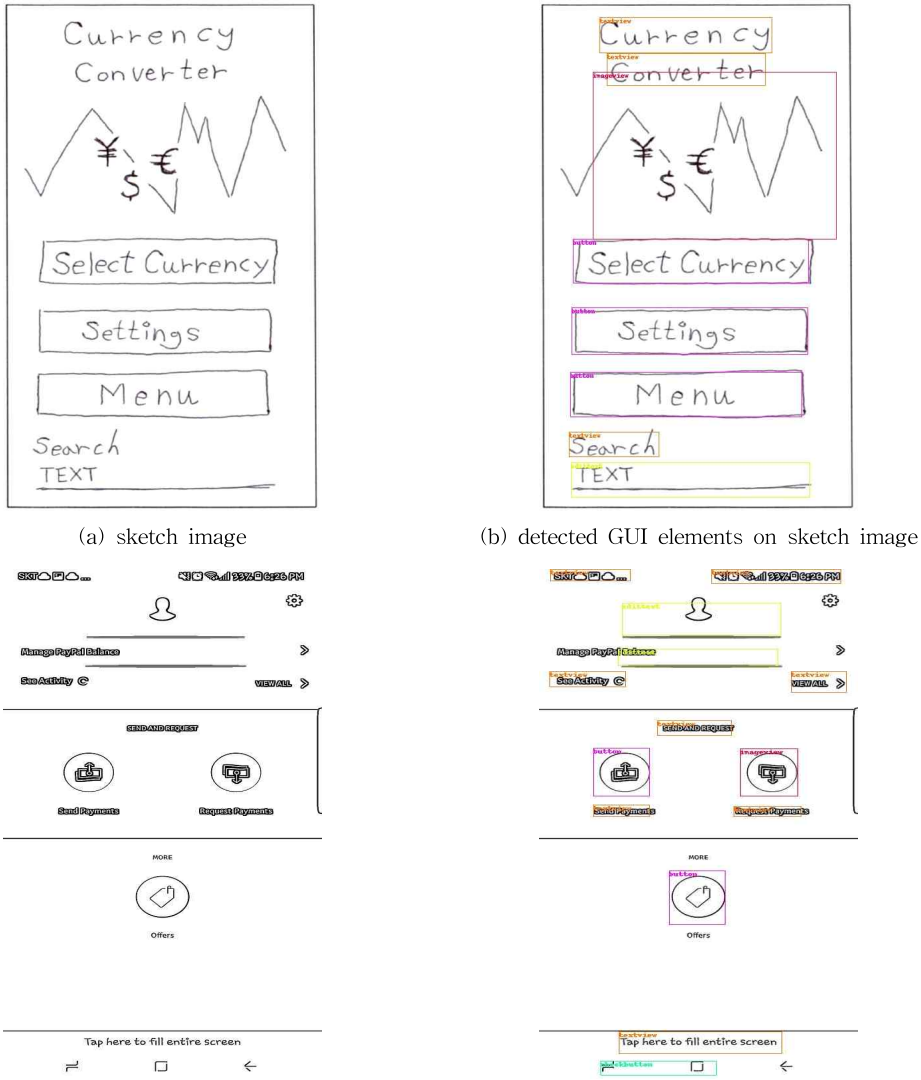
시스템은 C 언어와 NVidia의 그래픽 가속 라이브러리인 CUDA 기반의 신경 회로망 프레임워크로서 2013년에 J. Redmon이 공개한 이후, 꾸준하게 응용 분야를 넓혀왔으며 YOLO, YOLOv2, YOLOv3 등이 추가로 공개되었다. C 언어 기반의 darknet 시스템을 심층 신경망 연구에 널리 사용되는 python 2.7과 pytorch 0.3환경으로 구현한 YOLOv2[23]를 참조하여, python 3와 pytorch 0.4 환경에서 동작할 수 있도록 수정 및 보완하고, 최근에 공개된 YOLOv3를 학습과 평가가 가능하도록 구현하였다[24]. YOLOv2와 YOLOv3 버전 모두 기존 COCO dataset[25]에서 미리 학습된 값을 다운로드 받아 초기 가중치로 설정한 후 Windows 10의 NVIDIA GTX 1080 GPU 환경에서 학습을 시켰다. YOLOv2의 경우 YOLO에서 기본적으로 제시한 총 31개의 층으로 구성하였으며, 중요 사항으로는 23개의 컨볼루션 층과 5개의 max 풀링 층, 한 개의 검출 층으로 구성하였다. YOLOv3의 경우, 총 106개의 층으로 구축하였으며, 75개의 컨볼루션 층과 max 풀링층 없이 23개의 건너뛰기(shortcut) 층, 3개의 검출 층으로 구성하였다. Table 4에 YOLOv2와 YOLOv3의 중요 네트워크 구성을 비교하였다.

검출 결과 입력된 스케치 이미지에 대하여 정확하게 GUI 요소와 위치를 검출하는 것을 알 수 있었으며, 학습에 포함되지 않는 모바일 화면을 윤곽선 처리한 이미지도 검출하여 GUI 요소 검출 모델을 개선한다면 모바일 화면의 코드 변환에 사용할 수 있을 것으로 기대한다. Fig. 4에 스케치 이미지와 윤곽선으로 표현된 캡처된 이미지의 검출 결과를 제시하였다.

Table 4. Comparison of key layers on YOLOv2 and YOLOv3 models

Layers	YOLOv2	YOLOv3
convolution	23	75
max pooling	5	-
shortcut	-	23
detection	1	3
Total	31	106

구축된 시스템은 학습된 데이터를 그대로 이용하여 화면에 직접 펜으로 그리는 경우에도 GUI 요소를 쉽게 검출하여, 사용자가 쉽게 UI를 구축할 수 있도록 하였다. Fig. 5는 마우스로 GUI 화면을 그린 후, 검출된 GUI 요소를 확인하는 과정을 표현한다. 학습된 모델은 펜으로 그린 그림을 스캔하여 사용하였으나, 마우스를 이용하여 화면에 그렸기 때문에 학습



(a) sketch image (b) detected GUI elements on sketch image

(c) captured image with edges (d) detected GUI elements on capture image

Fig. 4. Detected GUI elements for sketch image and captured edge image.

모델과 다른 패턴을 보이나 기본적인 특성을 잘 반영한 것을 확인할 수 있다. Fig. 5 결과에서 Check Button이나 RadioButton의 경우 텍스트가 포함되었고, 두 버튼을 차별화할 수 있는 기호(○, □)가 텍스트와 비슷하여 TextView로 인식되었음을 알 수 있다. 이는 TextView 모델이 CheckButton과 Radio Button의 특징을 흡수하였거나 CheckButton과 RadioButton의 변이가 TextView와 혼동되었기 때문으로 판단한다. 이런 부분을 고려하여 각 GUI 요소에 대한 태그 정보를 이용하여 개선한다면 좀 더 정확한 GUI 요소 검출기를 구축할 수 있을 것이다.

4.4 검출 성능 비교

기존의 YOLOv2와 YOLOv3 모델의 경우 동일한 폭과 너비를 가진 이미지를 처리하였으나, 본 연구에서는 모바일 화면의 경우 가로와 세로의 비율이 다르기 때문에 480x800 크기를 갖는 이미지를 대상으로 학습 및 평가를 진행하였다. 두 모델의 네트워크 층 수가 다르고 구성 매개 변수의 수가 다르기 때문에 YOLOv2의 경우에는 1,000번, YOLOv3의 경우에는

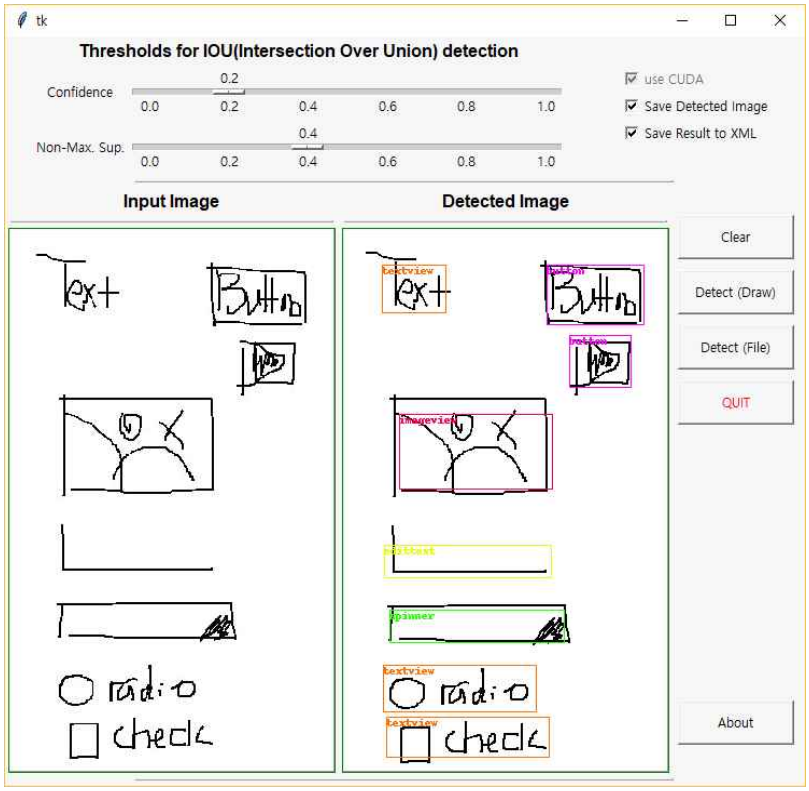


Fig. 5. Detected GUI elements for hand drawing (sketch) image.

2,000번의 반복(epoch) 학습을 진행하였다. YOLOv2나 YOLOv3 모두 1,000번이나 2,000번째 학습 결과가 가장 높은 성능을 보이지는 않았지만, 비교의 공정성을 위하여 모두 1,000번과 2,000번째 학습된 가중치를 이용하여 성능을 비교하였다. Table 5와 같이 성능 비교 결과 YOLOv2의 경우 87% YOLOv3의 경우 87.5%의 mAP(mean accuracy precision)를 보였다. 학습을 더 진행하는 경우 YOLOv2의 경우 성능이 약 3% 정도 향상되었으며, YOLOv3의 경우 성능은 향상되나 YOLOv2보다 최종 성능이 저하되는 경향을 보였다. 이는 학습에 사용되는 데이터의 양이 적은 영향을 받았을 수도 있기 때문에, 본 논문에서는 각 GUI 요소의 검출 경향을 상호 분석하였다.

Table 5. Comparison of Mean Average Precisions for YOLOv2 and YOLOv3

GUI Elements	YOLOv2 mAP (%)		YOLOv3 mAP (%)	
	training	test	training	test
TextView	100	78.5	100	79.5
Button	100	94.9	100	97.2
CheckButton	100	94.5	100	90.1
EditText	98.9	87.6	100	82.2
ImageView	100	80.2	99.4	81.9
Spinner	100	99.7	100	93.1
RadioButton	100	73.6	100	88.9
Total	99.8	87.0	99.9	87.5

성능 분석 결과 제안된 방식은 기존 연구와 직접적인 비교는 힘들지만 기존의 방식이 60~80%의 정합율을 보이는 것에 비하여 높은 성능을 보인다. YOLOv2는 TextView와 RadioButton 요소 등이 전체 평균 대비 낮은 성능을 보였으며, YOLOv3는 TextView와 ImageView가 전체 평균 대비 낮은 성능을 보였다. RadioButton의 경우 학습과 평가용으로 수집된 컴포넌트 수가 가장 적기 때문에 YOLOv2에서 가장 낮은 성능을 보인 것으로 파악되며, YOLOv3의 경우 검출 층이 분리되어 객체 정보가 분산 저장되었기 때문에 학습데이터가 적은 RadioButton에 대해서도 성능이 비슷하다고 판단할 수 있

다. 가장 많은 학습 데이터가 확보된 TextView의 경우 성능이 다른 GUI 요소에 비하여 낮은 이유는 TextView의 변이(variation)가 많고 CheckButton과 RadioButton과 같이 텍스트정보가 포함된 다른 요소와 혼동되었기 때문으로 판단한다. ImageView의 경우 이미지를 표현하는 방식이 사용자마다 달라 단순 사각형, 사각형 대각선 표시, 경계 없는 이미지 형태의 그림 등이 포함되어, 변이 폭이 넓은 것으로 판단하였다. 이 분석으로부터 수집된 데이터가 비슷한 양의 균일 분포를 보인다면 검출 시간이 빠른 YOLOv2 버전을, 객체들의 크기 변화가 많고 객체들의 성능이 균일한 분포를 보이기 원한다면 YOLOv3를 선택하는 것이 바람직할 것으로 판단한다.

5. 결 론

본 연구는 동일한 응용 프로그램을 다양한 플랫폼으로 이식하거나, 모바일 환경에 특화된 화면을 설계한 후 바로 UI 화면에서 코드를 확인할 수 있도록 자동으로 캡처 화면이나 스케치 이미지로부터 UI 코드를 자동 생성하는 것을 최종 목적으로 진행되었다. 연구의 목적을 달성하기 위하여 가장 중요한 부분은 GUI 요소를 검출하는 부분과 검출된 정보를 이용하여 자동으로 플랫폼에 적합한 코드를 생성하는 것이다. 본 논문에서는 자동으로 GUI 요소를 검출하는 부분에 관한 연구를 기술하였다. GUI 요소를 검출하기 위하여 기존의 연구에서는 직선이나 원 등과 같은 기본 요소를 이용하여 계층적 정보로 구성하여 GUI 요소를 모델링하거나, 텍스트 정보를 제외한 이미지로부터 윤곽선 정보를 확장하여 GUI 요소를 모델링하는 방법을 사용하였다. 기존 방법이 직관적이고 경험적 방법에 의존하기 때문에 최근에는 심층 신경망 기법을 이용하여 이미지로부터 코드 정보를 기술하는 방법을 사용하기도 하였으나, 복잡한 이미지의 경우에는 적용하기 어려운 단점이 존재하였다. 이를 극복하고자 본 논문에서는 객체 검출 방법을 적용하여 GUI 요소를 검출하는 방법을 제안하였다. 객체 검출 방법은 기존의 이미지 분류 기법에 객체의 위치 정보를 파악하는 연구 방법으로 객체 검출 과정과 분류 과정을 통합하여 좋은 성능을 보이고 있다. 본 논문에서는 객체 검출 기법으로 널리 사용되는 YOLOv2와 YOLOv3 방법을 이용하여 GUI 요소를 검출하였으며 두 방법 모두 약 87%의 mAP 성능을 보였다. YOLOv3는 YOLOv2의 검출층이 1개인 것을 여러 개의 층으로 분리하여 객체 검출 정보를 분산한 방법이다. 실험 분석 결과, 입력 이미지에서 학습에 필요한 특정 객체 집합 수가 적은 경우 YOLOv3가 유용한 방법으로 판단되었으며, 학습 데이터와 객체 클래스 정보에 따라 적절한 모델을 선택한다면 좋은 성능을 얻을 수 있을 것으로 판단한다.

기존의 연구 방식이 전통적인 OCR 방식과 이미지 분석 등의 방식을 이용하여 생성 규칙을 정형화하거나 경험적 방식에 의하여 GUI 요소를 검출하는데 반하여, 본 논문에서는 심층 신경 회로망 방식을 이용한 객체 검출 기법을 이용하여 GUI 요소를 검출하였다. 제안된 방식은 간접 비교를 통하여 기존의 전통적인 방식이나 캡셔닝 방식에 비하여 높은 검출 성능을 보였다. 또한 기존의 정형화된 규칙 생성이나 경험적 방식에 의하지 않고, 수집된 데이터를 이용하여 GUI 요소를 검출하기 때문에 보다 직관적이며 경계 상자와 같은 기하학적 정보와 문맥 정보를 동시에 추출하여 GUI 요소의 계층적 정보를 판단할 수 있다는 장점이 있다. 계층적 정보를 이용한다면 단순 GUI 요소들 간의 컨테이너 관계 정보 추출과 검출에 실패한 복합 GUI 요소를 재구성할 수 있을 것으로 기대된다.

추후 연구로는 심층 신경 회로망 모델에서 계층 정보를 적용하여 컨테이너를 검출하는 방법과 기존 검출 정보의 차집합(difference)에 의하여 신경회로망을 반복 적용하는 방법에 대한 연구를 진행할 예정이다. 성공적으로 GUI 요소 검출 방법에 의하여 컴포넌트와 컨테이너 정보가 획득된다면, UI 대분류와 모델 정보에 따라 객체 기술 언어로 표현된 메타 표현을 각 플랫폼의 해당 UI 코드로 생성하는 과정에 대한 연구 또한 필요할 것이다.

REFERENCE

[1] Graphical user interface, https://en.wikipedia.org/wiki/Graphical_user_interface (accessed Aug., 20, 2018).

[2] Macintosh, https://en.wikipedia.org/wiki/Macintosh (accessed Aug., 20, 2018).

[3] Microsoft Windows, https://en.wikipedia.org/wiki/Microsoft_Windows (accessed Aug., 20, 2018).

[4] Android developers, https://developer.android.com/ (accessed Aug., 22, 2018).

[5] iOS12 Apple developer, https://developer.apple.com/ios/ (accessed Aug., 22, 2018).

[6] Tizen delelopers, https://developer.tizen.org/ (accessed Aug., 22, 2018).

[7] T. Parag, C. Bahlmann, V. Shet, and M. Singh, "A Grammar for Hierarchical Object Description in Logic Programs," *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 33-38, 2012.

[8] T. Nguyen and C. Csallner, "Reverse Engineering Mobile Application User Interfaces with REMAUI (T)," *Proceeding of IEEE/ACM International Conference on Automated Software Engineering*, pp. 248-259. 2015.

[9] H. Pham, T. Nguyen, P. Vu, and T. Nguyen, "Toward Mining Visual Log of Software," *arXiv e-prints arXiv:1610.08911*, 2016.

[10] T. Beltramelli, "pix2code: Generating Code from a Graphical User Interface Screenshot," *arXiv e-prints arXiv:1705.07962*, 2017.

[11] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Computation*, Vol 9. No. 8. pp. 1735-1780, 1997.

[12] Y. Jeong, I. Ansari, J. Shim, and J. Lee, "A Car Plate Area Detection System Using Deep Convolution Neural Network," *Journal of Korea Multimedia Society*, Vol. 20, No. 8, pp. 1166- 1174, 2017.

[13] R. Cirshick, J. Donahue, T. Darrell, J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *arXiv e-prints arXiv:1311.2524v5*, 2014.

[14] R. Girshick, "Fast R-CNN," *arXiv e-prints arXiv:1504.08083v2*, 2015.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNNN: Towards Real-Time Object Detection with Region Proposal Networks," *arXiv e-prints arXiv:1506.01497v3*, 2016.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv e-prints arXiv:1506. 02640v4*, 2016.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, et al., "SSD: Single Shot Multibox Detector," *arXiv e-prints arXiv:1512.02325v5*, 2016.

[18] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *arXiv e-prints arXiv:1612.02842*, 2016.

[19] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv e-prints arXiv: 1804.02767*, 2018.

[20] S. Kim, J. Park, J. Jung, S. Eun, Y.-S. Yun, S. So, et al., "Identifying UI Widgets of Mobile Applications from Sketch Images," *Journal of Engineering and Applied Sciences*, Vol. 13, No. 6, pp. 1561-1566, 2018.

[21] Tzutalin, LabelImg, Git code (2015). https://github.com/tzutalin/labelImg (accessed Mar., 7, 2018).

[22] J. Redmon, Darknet: Open Source Neural Networks in C, http://pjreddie.com/darknet/, 2013. (accessed Mar., 7, 2017).

[23] Marvis, https://github.com/marvis/pytorch-yolo2, 2017. (accessed Mar., 7, 2018).

[24] Y.-S. Yun, https://github.com/andy-yun/pytorch-0.4-yolov3, 2018. (accessed Jun., 1, 2018).

[25] Common Objects in Context, http://cocodataset.org/#home (accessed Aug., 22, 2018)

윤 영 선

2001년 KAIST 전산학전공(박사)
2001년~현재 한남대학교 정보통신공학과 교수
2006년 한국전자통신연구원 초빙연구원
2012년 University of Washington 방문학자
2004년~현재 Interspeech Scientific Reviewer
관심분야 : 음성인식, 음성처리, 웹 접근성, 내장형시스템 등

박 지 수

2017년 건양대학교 의료IT공학과 졸업(학사)
2017년~한남대학교 정보통신공학과 재학(석사)
관심분야 : 영상인식, 패턴인식, 의료정보시스템, 임베디드 시스템 등

정 진 만

2008년 서울대학교 컴퓨터공학과 졸업(학사)
2014년 서울대학교 전기컴퓨터공학과 졸업(박사)
2014년~현재 한남대학교 정보통신공학과 교수
관심 분야 : 운영체제, 임베디드 시스템, IoT, 시스템 보안

은 성 배

1985년 서울대학교 컴퓨터공학과 학사
1987년 KAIST 전산학전공(석사)
1987년~1990년 한국전자통신연구원 TDX개발단 연구원
1995년 KAIST 전산학전공(박사)
1995년~현재 한남대학교 정보통신공학과 교수
관심분야 : 실시간 시스템, 임베디드 시스템 등

차 신

1995년 KAIST 전산학과 졸업(박사)
1986년~2000년 LG전자기술원 책임연구원
2000년~2013년 (주)IA 멀티미디어통신사업부 사업본부장
2013년~2015년 (주)슈어소프트테크 고신뢰검증센터 센터장
2016년~현재 한남대학교 컴퓨터통신무인기술학과 교수
관심분야 : 소프트웨어 신뢰성, 안전성공학, IoT 보안>

소 선 섭

1986년 이화여자대학교 전산학과 졸업(학사)
1988년 KAIST 전산학과 졸업(석사)
2001년 KAIST 전산학과 졸업(박사)
1995년~현재 공주대학교 컴퓨터공학부 교수
관심분야 : 소프트웨어 테스팅, 임베디드 소프트웨어, USN