學 士 學 位 論 文

시각장애인의 상품 구매 보조를 위한 어플리케이션 개발

Development of a product purchase assistance application for the visually impaired.

이름, 이름, 이름

安 養 大 學 校

2021年 12月 20日

시각장애인의 상품 구매 보조를 위한 어플리케이션 개발

Development of a product purchase assistance application for the visually impaired.

이름*, 이름*, 이름*
영문이름*, 영문이름*, 영문이름*

본 개발의 목적은 시각장애인을 대상으로 편의점에서 상품 구매 시 소비자로서 상품에 대한 정확한 정보를 제공 받기 위하여 개발하였다. 사용자의 위치를 기준으로 최단거리 편의점의 경로 안내 및 판매중인 다양한 상품들 중 모호하게 표기되어 있는 음료수 캔상단에 위치한 점자대신 이미지 인식 기술을 이용하여 사용자가 선택한 상품의 전면을 인식 후 기본적인 정보를 음성으로 안내한다.

Abstract The purpose of this development was developed for the visually impaired to receive accurate information about the product as a consumer when purchasing a product at a convenience store. Based on the location of the user, we use image recognition technology instead of braille located at the top of the can of beverage, which is vaguely marked among various products on sale, and to guide basic information by voice.

Key Words : visually impaired, image recognition, convenience store

Ⅰ. 서 론

구글의 알파고를 기점으로 대중들에게 인공지능이라는 기술이 대중에게 다가오면서 딥러닝을 활용한 많은 서비스들이 출시 하였다. 인공지능 비서 부터 시작하여 코로나 바이러스로 인하여 얼굴 인식을 활용한 체온 감지 등 다양한 서비스들이 대중들에게 선보였다. 하지만 아직까지도 장애인을 위한 서비스가 많이 발전하지 못했다는 것을 알 수 있다.

기술이 발전함에 있어서도 시각장애인은 원시적인 방법인 점자표기를 손으로 만지며 글을 읽는다. 우리는 물건을 구매 할 때 상표와 판매점의 가격표를 확인한다. 하지만 시각장애인을 위한 점자표기가 된 상품은 없는 것이 대부분이며 그나마 점자표기가 되어있는 상품은 모호한 내용을 담고 있는 경우가 대부분이다. 특히 음료수 캔 상단에 위치하고 있는 점자의 경우 제품명이 아닌 제품의 분류명인 '탄산', '음료' 등 으로 표기 되어 있으며 대표적인 탄산음료 중 하나인 코카콜라는 제품 측면에는 탄산음료로 분류되어 있지만 점자표기로는 '음료'로 오표기 되어 있다.

언론 보도에서 제조사는 점자표기가 캔의 상단 부분에 위치해 있기 때문에 최대 4음절까지 적을 수 밖에 없으며 위치 변경 혹은 점자 내용 변경은 추가적인 비용이 소모되어 변경하지 않는다는 입장을 밝혔다. 이에 유일하게 표기가 정확히 되어 있는 음료는 진로 하이트사의 '테라' 제품만 표기 되어있다.

이에 시각장애인도 소비자로서 당연히 누려야 할 권리로 정확한 제품명과 가격, 영양정보 등을 안내 받을 수 있도록 하기 위해 이미지 인식을 활용한 개발을 하였다.

II. 관련 연구

1. 이미지 인식

인공지능 이미지 인식은 기계가 마치 사람처럼 사진이나 동영상으로부터 사물을 인식하거나 장면을 이해하는 것으로 정의 할 수 있다. 이러한 이미지 인식은 컴퓨터 비전(computer vision) 기술 중 하나에 해당한다. 이미지 인식에는 대표적으로 세 가지 태스크(task)가 존재하는데 이미지 내 특정 사물을 분류하는 태스크, 여러 사물을 동시에 검출하는 태스크, 사물들을 픽셀 단위로 식별하여 분할하는 태스크 등이 있다. 이러한 이미지 인식 기술에 있어 2012년 혁신적인 연구 결과가 나온다. 대규모 이미지 인식 경진대회인 ILSVRC에서 토론토 대학 연구진이 딥러닝이라 불리는 새로운 기법을 활용해 기존의 방법론에 대비해 압도적인 성능으로 우승한 것이다. 이 연구는 신경정보처리시스템학회에 발표되어 지금까지도 인공지능 분야에서 가장 많이 인용되고 있는 논문 중 하나가 되었다. 이를 계기로 딥 러닝이 학계, 산업계에 널리 받아들여지게 됨에 따라 딥 러닝 또한 폭발적으로 발전하여, 2015년 ILSVRC에서 사람의 인식률(94.90%)를 추월(96.43%)하고, 2020년에는 사람을 한참 뛰어넘은(98.7%)으로 진화했다.

2. 딥 러닝

심층 학습(深層學習) 또는 딥 러닝(영어: deep structured learning, deep learning 또는 hierarchical learning)은 여러 비선형 변환기법의 조합을 통해 높은 수준의 추상화(abstractions, 다량의 데이터나 복잡한 자료들 속에서 핵심적인 내용 또는 기능을 요약하는 작업)를 시도하는 기계 학습 알고리즘의 집합으로 정의되며, 큰 틀에서 사람의 사고방식을 컴퓨터에게 가르치는 기계 학습의 한 분야라고 이야기할 수 있다.

어떠한 데이터가 있을 때 이를 컴퓨터가 알아들을 수 있는 형태(예를 들어 이미지의 경우는 픽셀정보를 열벡터로 표현하는 등)로 표현(representation)하고 이를 학습에 적용하기 위해 많은 연구(어떻게 하면 더 좋은 표현기법을 만들고 또 어떻게 이것들을 학습할 모델을 만들지에 대한)가 진행되고 있으며, 이러한 노력의 결과로 deep neural networks, convolutional deep neural networks, deep belief networks와 같은 다양한 딥 러닝 기법들이 컴퓨터 비전, 음성인식, 자연어 처리, 음성/신호처리 등의 분야에 적용되어 최첨단의 결과들을 보여주고 있다.

2012년 스탠포드대학의 앤드류 응과 구글이 함께한 딥 러닝 프로젝트에서는 16,000개의 컴퓨터 프로세서와 10억 개 이상의 neural networks 그리고 DNN(deep neural networks)을 이용하여 유튜브에 업로드 되어 있는 천만 개 넘는 비디오 중 고양이 인식에 성공하였다. 이 소프트웨어 프레임워크를 논문에서는 DistBelief로 언급하고 있다. 이뿐만 아니라 마이크로소프트, 페이스 등도 연구팀을 인수하거나 자체 개발팀을 운영하면서 인상적인 업적들을 만들어 내고 있다.

3. 공공데이터

공공 데이터(open data)는 데이터베이스, 전자화된 파일 등 공공기관이 법령 등에서 정하는 목적을 위하여 생성 또는 취득하여 관리하고 있는 광(光) 또는 전자적 방식으로 처리된 자료 또는 정보로서 다음과 같은 것을 지칭한다. 첫째, 「전자정부법」에 따른 행정정보, 둘째, 「국가정보화 기본법」에 따른 정보 중 공공기관이 생산한 정보, 셋째, 「공공기록물 관리에 관한 법률」에 따른 전자기록물 중 대통령령으로 정하는 전자기록물, 그 밖에 대통령령으로 정하는 자료 또는 정보가 있다. 따라서, 공공데이터는 개별 공공기관이 일상적 업무수행의 결과물로 생성 또는 수집·취득한 다양한 형태(텍스트, 수치, 이미지, 동영상, 오디오 등)의 모든 자료 또는 정보를 대상으로 하며, "기계 판독이 가능한 형태"로 "제공"이되어야 한다. 여기서 "기계 판독이 가능한 형태"란 소프트웨어로 데이터의 개별 내용 또는 내부 구조를 확인하거나 수정, 변환, 추출 등 가공할 수 있는 상태를 말하며, "제공" 이란 공공기관이 이용자로 하여금 기계 판독이 가능한 형태의 공공데이터에 접근할 수 있게

하거나 이를 다양한 방식으로 전달하는 것을 말한다.

정부는 보유한 공공 정보를 적극적으로 개방하여 국민과 공유함으로써 소통과 협력을 확대하기 위해 공공데이터 정책을 추진하게 되었고, 2013년 7월 공공데이터의 제공 및 이용활성화에 관한 법률(공공 데이터 법)을 제정하고 10월부터 시행 되었다.

공공데이터 제공 및 이용 활성화에 관한 정책은 '행정안전부'가 총괄하며, 이 법률에 근거하여 2013년 11월, 정부나 공공기관이 보유한 공공데이터를 민간에서 쉽게 활용할 수 있도록 지원하기 위하여 한국정보화진흥원에 공공데이터활용지원센터'를 설치 하였다. 각 공공기관은 공공데이터제공책임관 및 실무자를 지정하고 관련 업무를 지원한다. 또한 정부는 공공데이터 관련 정책을 심의, 조정 및 그 추진사항을 점검하기 위하여 국무총리실 산하에 공공데이터전략위원회를 설립하였고, 공공기관의 공공데이터 제공거부 또는 중단문제 발생시 해결을 위한 공공데이터제공분쟁조정위원회를 설립하여 공공데이터 활성화 정책을 추진하고 있다.

4. Tensorflow & Keras

텐서플로 또는 텐서플로우는 다양한 작업에 대해 데이터 흐름 프로그래밍을 위한 오픈소스 소프트웨어 라이브러리이다. 심볼릭 수학 라이브러리이자, 인공 신경망 같은 기계 학습 응용프로그램 및 딥러닝에 사용된다. 이것은 구글 브레인팀의 연구와 제품개발을 위한 목적으로 2015년 아파치 2.0 오픈 소스 라이선스로 공개되었다.

케라스(Keras)는 파이썬으로 작성된 오픈 소스 신경망 라이브러리이다. MXNet, Deeplearning4j, 텐서플로, Microsoft Cognitive Toolkit 또는 Theano 위에서 수행할 수 있다. 딥 신경망과의 빠른 실험을 가능케 하도록 설계되었으며 최소한의 모듈 방식의 확장 가능성에 초점을 둔다. ONEIROS(Open-ended Neuro-Electronic Intelligent Robot Operating System) 프로젝트의 연구적 노력의 일환으로 개발되었으며[4] 주 개발자이자 유지보수자는 구글의 엔지니어 프랑소아 숄레(Francois Chollet)이다.

2017년, 구글의 텐서플로(Tensorflow) 팀은 텐서플로의 코어 라이브러리에 케라스를 지원하기로 결정하였다. Chollet는 케라스가 단대단(end-to-end) 기계 학습 프레임워크가 아닌 인터페이스의 역할을 염두에 두었다. 더 높은 수준의 더 직관적인 추상화 집합을 표현함으로써 백엔드 과학 컴퓨팅 라이브러리임에도 불구하고 신경망을 구성하기 쉽게 만들어준다. 마이크로소프트 또한 CNTK 백엔드를 케라스에 추가하는 작업을 수행하고 있으며 기능은 현재 CNTK 2.0과 더불어 베타 릴리스 단계에 있다.

5. TTS (Text to speech)

우리말로 '음성 합성 시스템'이라 부르며 보통 TTS라 부른다. 컴퓨터의 프로그램을 통해 사람의 목소리를 구현해내는 것으로 성우 없이도 거의 모든 단어와 문장의 음성을 쉽게 구할 수 있다. 하지만 사전 녹음된 목소리 자료를 기반으로 쓰는 만큼 억양이 자연스럽지 못하다는 단점이 있다. 본 개발에서 사용한 시스템은 Google Text-to-speech이며 안드로이드 운영 체제용으로 구글이 개발한 스크린 리더 기능이다. 화면상의 텍스트를 읽는 기능을 제공하며 수많은 언어를 지원한다. 텍스트 음성 변환은 책을 읽거나 사용자가 입력한 글 혹은 선택한 텍스트, 버튼을 안내한다.

5. 크롤링 (Crawling)

크롤링이란 데이터를 수집하고 분류하는 것을 의미한다. 주로 인터넷상의 웹페이지를 수집해서 분류하고 저장하는 것을 뜻하며 데이터가 어디에 저장되어 있는지 위치에 대한 분류 작업이 크롤링의 주요 목적이다. 크롤링의 주요 대상은 다양한 형태로 존재하는 데이터로, 데이터 생성 스타일에 따라 정형, 반정형 그리고 비정형 데이터로 구분되기도 한다.

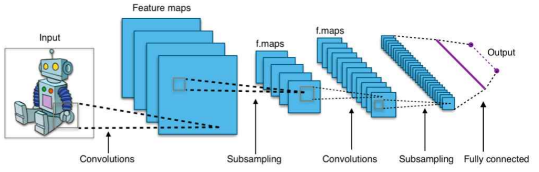
6. DarkNet

Darknet은 Joseph Redmon이 독자 개발한 신경망 프레임워크(neural network framework)로서 dnn(deep neural network)들을 학습시키고 실행시킬 수 있는 프레임워크다. Darknet을 이용하면 기존 정통 주류의 dnn 모

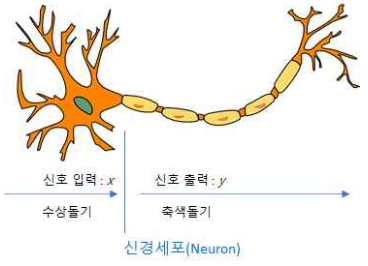
델 AlexNet, VGG-16, Resnet, Densenet 등 을 실행 할 수 있으며 YOLO 역시 Darknet을 통해 학습된 신경망 중 하나이다.

7. CNN

합성곱 신경망(Convolutional neural network, CNN)은 시각적 영상을 분석하는 데 사용되는 다층의 피드-포워드적인 인공신경망의 한 종류이다. 딥 러닝에서 심층 신경망으로 분류되며, 시각적 영상 분석에 주로 적용된다. 또한 공유 가중치 구조와 변환 불변성 특성에 기초하여 변이 불변 또는 공간 불변 인공 신경망 (SIANN)으로도 알려져 있다. 영상 및 동영상 인식, 추천 시스템, 영상 분류, 의료 영상 분석 및 자연어 처리 등에 응용된다.



합성곱 신경망은 정규화 된 버전의 다층 퍼셉트론이다. 다층 퍼셉트론은 일반적으로 완전히 연결된 네트워크, 즉 한 계층의 각 뉴런이 다음 계층의 모든 뉴런에 연결되는 신경망 구조이다. 이와 같이 네트워크가 완전 연결된 경우 주어진 데이터에 과적합 되는 경향이 있다. 일반적인 정규화를 위해 최적화 함수에 특정 척도를 추가하는 방법이 흔이 쓰이지만, CNN은 정규화를 위한 다른 접근 방식을 취한다. 데이터에서 계층적 패턴을 활용하고 더 작고 간단한 패턴을 사용하여 더 복잡한 패턴을 표현함으로써 정규화와 같은 효과를 내는 것이다. 따라서 합성곱 신경망의 연결 구조의 복잡성은 유사한 기능의 다층 퍼셉트론에 비해 극단적으로 낮다.



합성곱 신경망은 뉴런 사이의 연결 패턴이 동물 시각 피질의 조직과 유사하다는 점에 영감을 받았다. 개별 피질 뉴런은 수용장(receptive field)으로 알려진 시야의 제한된 영역에서만 자극에 반응한다. 상이한 뉴런의 수용 필드는 전체 시야를 볼 수 있도록 부분적으로 중첩된다.

합성곱 신경망을 이용한 영상 분류는 다른 영상 분류 알고리즘에 비해 상대적으로 전처리를 거의 사용하지 않는다. 이는 신경망이 기존 알고리즘에서 수작업으로 제작된 필터를 학습한다는 것을 의미한다. 기존 영상 분류 알고리듬에서 설계자가 영상의 특징들을 미리 이해해 알고리듬을 만드는 과정이 없는 것이 합성곱 신경망의 주요한 장점이다.

합성곱 신경망은 크게 합성곱층(Convolution layer)과 풀링층(Pooling layer)으로 구성된다.

III. 시스템 설계 및 구현

본 시스템을 구현하기 위해 파이썬(본 개발에서는 파이썬 버전 3.9)을 사용하고, IDE는 Pycharm을 활용했으며 이미지 인식을 위한 주요 라이브러리 Tensorflows 2.0을 기반으로 개발 하였으며 어플리케이션 코틀린(Kotlin 203-1.6.0 release) 및 IDE는 AndroidStudio archtic fox 2020.3.1.을 사용하였다.

딥러닝 학습에 사용된 컴퓨터 시스템은 Ryzen5 3600 CPU, 32GB RAM, B450 AORUS ELITE MainBoard, Nvidia GTX1060 6GB GPU를 사용하였다.

본 개발 시스템 구성도는 그림1과 같다.

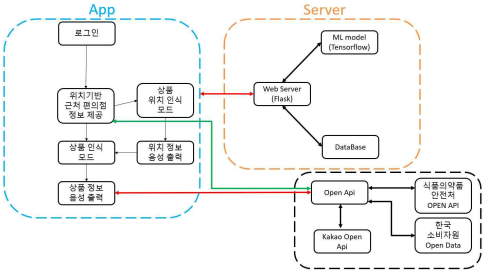


그림 1. 시각장애인의 상품 구매 보조 어플리케이션 시스템 구성도

그림1에서 보는 바와 같이 본 개발의 시스템은 백엔드 서버와 프론트엔드 어플리케이션, API 파트로 나뉜다.

1. 어플리케이션

1-1. 어플리케이션 기능 권한 부여

어플리케이션에서 카메라나 위치기능 등 휴대전화에 있

는 다양한 기능을 사용할 수 있도록 허용할 수 있다. 앱에서 휴대전화의 기능을 사용하기 위한 권한을 요청하는 알림을 전송하면 이를 허용 또는 거부하는 기능이다.

```
import android.content.pm.PackageManager
import android.content.pm.PackageManager
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat

1   // 카메라 권한 요청
    fun checkPermission() {
        val cameraPermission =
    ContextCompat.checkSelfPermission(
            this@MenuActivity,
            Manifest.permission.CAMERA
        )
        if(cameraPermission !=
    PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
    arrayOf(Manifest.permission.CAMERA), 1000)
        }
    }

2   // 위치 권한 사용
    requestPermissions(arrayOf(Manifest.permission.ACCES
    S_FINE_LOCATION), 1)
```

그림 1-1. 카메라 및 위치 기능 권한 부여 코드
Fig 1-1. Authorize camera and location functions

그림 1-1은 휴대전화의 카메라와 GPS의 정보를 사용하기 위해 엑세스를 허용할지 사용자가 결정할 수 있도록 요청하는 코드이다. 1) 휴대전화의 카메라 기능의 권한을 요청하는 역할을 한다. 2) 휴대전화의 GPS 기능의 권한을 요청하는 역할을 한다.

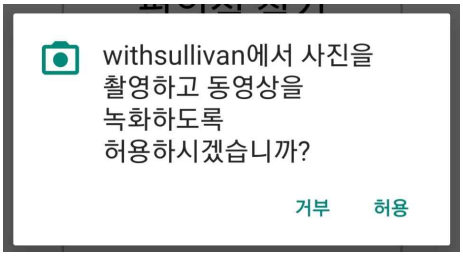


그림 1-2. 사용자에게 권한 요청 화면
Fig 1-2. Ask user for permission screen

1-2. Tmap Api를 사용한 도보 길찾기

사용자의 위치를 기준으로 하여 최단거리 편의점의 경로를 음성으로 안내해주는 보조기능이다. 주변의 편의점 4개 중 1개를 선택하여 원하는 편의점까지 안내를 해준다.

```
    import com.skt.Tmap.*
    import com.skt.Tmap.TMapData.*

    //Tmap 각종 객체 선언
    tmapdata = TMapData() //POI검색, 경로검색 등의
    지도데이터를 관리하는 클래스
    tmapview = TMapView(this)
    tmapview!!.setSKTMapApiKey(mApiKey)  // api key

    tmapview!!.setLanguage(TMapView.LANGUAGE_KOR
    EAN) // 언어 설정
    tmapgps = TMapGpsManager(this@SearchMap)
    //단말의 위치탐색을 위한 클래스
    tmapgps!!.setMinTime(1000) //위치변경 인식
    최소시간설정
    tmapgps!!.setMinDistance(5f) //위치변경 인식
    최소거리설정
    tmapgps!!.setProvider(TMapGpsManager.NETWORK_P
    ROVIDER) //네트워크 기반의 위치탐색
    tmapgps!!.OpenGps()

    tmapview.setIconVisibility(true)
    tmapview.zoomLevel = 18
    tmapview.mapType =
    tmapview.MAPTYPE_STANDARD
    tmapview.setLanguage(tmapview.LANGUAGE_KOREA
    N)
    tmapview.setTrackingMode(true)
    tmapview.setSightVisible(true)
    binding.linearLayoutTmap.addView(tmapview)
```

그림 1-3. Tmap 지도 불러오기 및 설정 코드
Fig 1-3. Tmap load and setup code

그림 1-3은 지도를 사용하기 위해 key값을 사용하여 화면에 지도를 띄우는 과정이다. 객체 함수를 사용하여 지도의 언어, 줌, 위치 등등 최적한 환경으로 설정하는 과정이다.

그림 1-4. Tmap api 사용 후 지도 화면 출력 결과
Fig 1-4. Map screen output result after using Tmap api

```
import com.skt.Tmap.*
import com.skt.Tmap.TMapData.*
import android.location.Location

override fun onLocationChange(location: Location) {
        tmapview!!.setLocationPoint(location.longitude,
location.latitude)
        tmapview!!.setCenterPoint(location.longitude,
location.latitude)

        latitude_start = location.latitude
        longitude_start = location.longitude

        Log.d(TAG, "$latitude_start, $longitude_start
현재위치")

        getAroundBizPoi()
    }
```

그림 1-5. GPS를 사용하여 지도에 현재 위치 표시

Fig 1-5. Show your current location on a map using GPS

그림 1-5는 휴대기기에 있는 GPS를 사용하여 실시간으로 현재 위도와 경도의 값을 가져오는 과정이다. 이를 통해서 지도 위에 현재위치를 표시할 수 있다.

```
import com.skt.Tmap.*
import com.skt.Tmap.TMapData.*

val tmapdata = TMapData()
val point = tmapview!!.centerPoint
tmapdata.findAroundNamePOI(
    point, "편의점", 1, 10
) { poiItem ->
    for (i in poiItem.indices) {
        val item = poiItem[i]
        Log.d(
            TAG, "POI Name: " +
item.poiName.toString() + ", " +
                "Address: " +
item.poiAddress.replace("null", "") + ", " +
                "Point: " + item.poiPoint.toString()
+ ", Poiid: " + item.poiid
        )
    }
```

그림 1-6. 현재위치를 기반으로 가까운 편의점 찾기

Fig 1-6. Find a convenience store near you based on your current location

그림 1-6은 명칭별 주변검색 POI데이터를 검색개수와 반경정보를 기준으로 요청한다. “편의점”을 포함한 데이터를 최대 10개를 가져오도록 요청한 코드이다.

```
POI Name: 세븐일레븐 시화테크노파크1호점, Address: 경기 시흥시 정왕동 ,
POI Name: 세븐일레븐 시화테크노파크2호점, Address: 경기 시흥시 정왕동 ,
POI Name: 세븐일레븐 시흥스틸랜드점, Address: 경기 시흥시 정왕동 , Poin
POI Name: CU 시흥스틸랜드점, Address: 경기 시흥시 정왕동 , Point: Lat
POI Name: CU 시화소망점, Address: 경기 시흥시 정왕동 , Point: Lat 37.
POI Name: CU 시화라성점, Address: 경기 시흥시 정왕동 , Point: Lat 37.
POI Name: GS25 시화메가점, Address: 경기 시흥시 정왕동 , Point: Lat 3
POI Name: CU 시흥메가점, Address: 경기 시흥시 정왕동 , Point: Lat 37.
POI Name: 이마트24 시화물류센터점, Address: 경기 시흥시 정왕동 , Point:
POI Name: GS25 정왕스틸랜드점, Address: 경기 시흥시 정왕동 , Point: La
```

그림 1-7. POI데이터 검색을 통한 결과 값

Fig 1-7. Result value through POIdata search

```
while (true) {
        if
(poiItem[count].poiName.toString().endsWith("주차장"))
{
            count += 1
        } else {

binding.Text1.setText(poiItem[count].poiName.toString()
)
            latitude_list[0] =
poiItem[count].poiPoint.latitude
            longitude_list[0] =
poiItem[count].poiPoint.longitude
            convenience_list[0] =
poiItem[count].poiName
            break
        }
    }
```

그림 1-8. "주차장"이 포함된 검색결과 제외

Fig 1-8. Exclude search results containing "parking lot"

그림 1-8은 “편의점” 검색 결과 중 같은 이름을 포함한 “주차장” 결과 값도 표기되는 것을 줄이기 위한 코드이다.

```
import com.skt.Tmap.TMapData
import android.location.Location

//경로를 받아와서 지도에 line으로 그려주는 함수
tmapdata.findPathDataWithType(
    TMapData.TMapPathType.PEDESTRIAN_PATH,
tMapPointStart, tMapPointEnd
) { polyLine -> tMapView.addTMapPath(polyLine) }
```

그림 1-9. 도보 길찾기 기능 구현 후 지도에 표시

Fig 1-9. Display on the map after implementing the walking directions function

그림 1-9는 Tmap에서 제공한 도보 길찾기 기능을 사용하여 시작점과 도착점을 기준으로 하여 지도 위에 선을 그려주는 기능이다.

그림 1-10. 주변 편의점까지 도보 길찾기 이용 결과

Fig 1-10. Results of walking to nearby convenience stores

```
// 길 안내
val root = document.documentElement
val nodeListPlacemark =
root.getElementsByTagName("Placemark")
for (i in 0 until nodeListPlacemark.length) {
      val nodeListPlacemarkItem =
nodeListPlacemark.item(i).childNodes

      for (j in 0 until nodeListPlacemarkItem.length) {
            if (nodeListPlacemarkItem.item(j).nodeName ==
"description") {
                  var insert_check = false

                  if
(!nodeListPlacemarkItem.item(j).textContent.contains(",")
) {
                        insert_check = true
                        Log.d(TAG,
nodeListPlacemarkItem.item(j).textContent)

path.add(nodeListPlacemarkItem.item(j).textContent)
                  }
            }
            if (nodeListPlacemarkItem.item(j).nodeName ==
"Linestring" || nodeListPlacemarkItem.item(j).nodeName
== "Point") {
                  val coordi_node =
nodeListPlacemarkItem.item(j).childNodes
                  for (k in 0 until coordi_node.length) {
                        if (coordi_node.item(k).nodeName ==
"coordinates") {
                              val coor_array =
coordi_node.item(k).textContent.split(",").toTypedArray()

path_coor.add(coor_array[0].toDouble())

path_coor.add(coor_array[1].toDouble())
                        }
                  }
            }
      }
}
```

그림 1-11. 도보 길찾기 정보 처리 코드

Fig 1-11. Code for processing walking directions information

그림 1-11은 도보 길찾기의 세부적인 안내를 하기 위한 과정이다. 횡단보도를 건너가거나, 골목의 방향 등등 정보를 가지고 올 수 있다. 이를 통해 목표지점까지 무사히 안내받을 수 있다.

```
보행자도로 을 따라 67m 이동
옥구마을계룡2차아파트 에서 우회전 후 보행자도로 을 따라 144m 이동
우회전 후 정왕대로28번길 을 따라 236m 이동
도착
```

그림 1-12. 목적지까지 길안내 출력 결과

```
override fun onLocationChanged(location: Location) {
      val longitude = location.longitude //경도
    val latitude = location.latitude //위도

      tMapView.setLocationPoint(longitude, latitude)
      tMapView.setCenterPoint(longitude, latitude, true)
      Log.d(TAG, "path size is : " + path.size + " path_coor is : "
+ path_coor.size + " nodecurrent: " + nodeCurrent)
      var intent = Intent(applicationContext,
MenuActivity::class.java)
      if (nodeCurrent == 0) {
            binding.Drawing.visibility = View.INVISIBLE
            binding.linearLayoutTmap.visibility = View.VISIBLE
            if (path.size > 0 && path_coor.size > 0) {
                  onRead("지금부터 안내를 시작할게요.
${path[nodeCurrent]} 하세요. 직진이에요.")
                  Log.d(TAG, "nodeCurrent: $nodeCurrent")
                  Log.d(TAG, Integer.toString(path_coor.size))
                  nodeCurrent++
            } else {
                  Log.d("back step required: ", "path is 0 & path_coor
is 0")
            }

      } else if (2 * nodeCurrent + 2 < path_coor.size) {
            if (path_coor[2 * nodeCurrent] > longitude - 0.0001 &&
path_coor[2 * nodeCurrent] < longitude + 0.0001 && path_coor[2
* nodeCurrent + 1] > latitude - 0.0001 && path_coor[2 *
nodeCurrent + 1] < latitude + 0.0001) {
                  onRead(path[nodeCurrent] + "하세요.")

                  Log.d(TAG, "nodeCurrent: $nodeCurrent")
                  nodeCurrent++
            }

      } else if (2 * nodeCurrent + 2 == path_coor.size) {
            if (path_coor[2 * nodeCurrent] > longitude - 0.0001 &&
path_coor[2 * nodeCurrent] < longitude + 0.0001 && path_coor[2
* nodeCurrent + 1] > latitude - 0.0001 && path_coor[2 *
nodeCurrent + 1] < latitude + 0.0001) {
                  onRead(path[nodeCurrent] + "하셨어요. 안내를
종료할게요.")

                  Log.d(TAG, "Got it!")
                  Log.d(TAG, "nodeCurrent: $nodeCurrent")
                  nodeCurrent++
            }

      } else if (2 * nodeCurrent + 2 > path_coor.size) {
            toast("메인화면으로 돌아갑니다.")
            startActivity(intent)
            lm!!.removeUpdates(this)
            finish()
      }
}
```

그림 1-13. 길찾기 음성 안내 출력 코드

Fig 1-13. Voice guidance output code

그림 1-13은 길안내 중에 목적지까지 방향을 알려주는 코드이다. 방향을 바꿔야 하거나 횡단보도를 건너야 할 경우 TTS로 방향을 안내해 주는 기능이다. 자신의 위치를 기준으로 하여 다음 안내의 위치 근처로 오면 다음 안내를 하도록 구현 하였다.

```
fun onRead(text: String?): String? {
    tts!!.setPitch(1.0f) //1.0톤 올려서
    tts!!.setSpeechRate(0.9f) //0.9배속으로 읽기
    tts!!.language = Locale.KOREAN
    tts!!.speak(text, TextToSpeech.QUEUE_FLUSH,
null, "")
    return text
}

// tts 설정
override fun onInit(status: Int) {
    if (status == TextToSpeech.SUCCESS) {
        val result = tts!!.setLanguage(Locale.KOREAN)

        if(result ==
TextToSpeech.LANG_MISSING_DATA || result ==
TextToSpeech.LANG_NOT_SUPPORTED) {
            Toast.makeText(this, "The Language
specified is not supported!",
Toast.LENGTH_SHORT).show()
        }
    } else {
        Toast.makeText(this, "Initilization Failed!",
Toast.LENGTH_SHORT).show()
    }
}
```

그림 1-14 TTS기능 설정 및 활성화

Fig 1-14 Setting up and activating the TTS function

그림 1-14는 텍스트 파일을 음성 파일로 읽어주는 기능으로, 길 안내를 음성으로 변환하기 위해 사용되었다. 설정을 통해 목소리의 높이와, 속도, 언어 등을 설정할 수 있다.

1-3. 카메라를 사용한 음료 인식

카메라를 사용하여 원하는 음료를 촬영한 데이터를 서버로 전달해 주는 보조기능이다. 서버와 통신을 성공하였으면 통신 받은 결과 값을 출력해주는 기능이다.

```
private fun openCamera() {
    try {
        val mCameraManager =
this.getSystemService(CAMERA_SERVICE) as
CameraManager
        val characteristics =
mCameraManager.getCameraCharacteristics(mCameraId)
        val map =
characteristics.get(CameraCharacteristics.SCALER_STREA
M_CONFIGURATION_MAP)

        val largestPreviewSize =
map!!.getOutputSizes(ImageFormat.JPEG)[0]

setAspectRatioTextureView(largestPreviewSize.height,
largestPreviewSize.width)

        mImageReader = ImageReader.newInstance(
            largestPreviewSize.width,
            largestPreviewSize.height,
            ImageFormat.JPEG,
            7
        )
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.CAMERA)
            != PackageManager.PERMISSION_GRANTED
        ) return

        mCameraManager.openCamera(mCameraId,
deviceStateCallback, mHandler)
    } catch (e: CameraAccessException) {
        Handler(Looper.getMainLooper()).post {
            Toast.makeText(this@DetectActivity,
"카메라를 열지 못했습니다.",
Toast.LENGTH_SHORT).show()
        }
    }
}
```

그림 1-15. 카메라 기능 구현

Fig 1-15. Implementing camera functions

그림 1-15는 이전에 기기에 카메라 기능에 대한 권한을 받아 camera api를 사용하여 CamaeraManager의 메소드인 openCamera()를 이용하여 전면 카메라 기능 구현.

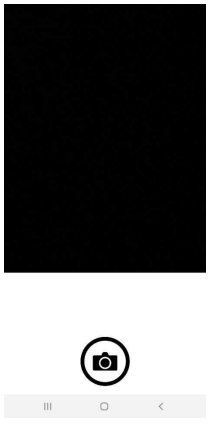


그림 1-16. 카메라 기능 xml 화면

Fig 1-16. camera function xml screen

```
val dir = externalCacheDir
var file = File.createTempFile("photo_", ".jpg", dir)
val readerListener = object :
ImageReader.OnImageAvailableListener {
    override fun onImageAvailable(reader:
ImageReader?) {
        var image: Image? = null

        try {
            image = imageReader.acquireLatestImage()

            val buffer = image!!.planes[0].buffer
            val bytes = ByteArray(buffer.capacity())
            buffer.get(bytes)

            var output: OutputStream? = null
            try {
                output = FileOutputStream(file)
                output.write(bytes)
            } finally {
                output?.close()
            }
            var uri = Uri.fromFile(file)
            var bitmap: Bitmap =
BitmapFactory.decodeFile(file.path)


        } catch (e: FileNotFoundException) {
            e.printStackTrace()
        } catch (e: IOException) {
            e.printStackTrace()
        } finally {
            image?.close()
        }
    }
}
```

그림 1-17. 카메라로 촬영 후 캐시 파일로 저장

Fig 1-17. Save as cache file after shooting with camera

그림 1-17은 File object 내의 createTempFile() function을 이용하여 임시파일을 생성하도록 구현하였다. 파일 이름은 photo_(랜덤숫자).jpg 임시파일로 저장된다.

```
// imageReader 객체에 위에서 만든 readerListener 를
달아서, 이미지가 사용가능하면 사진을 저장한다
imageReader.setOnImageAvailableListener(readerListener,
null)
photoFile = file
val captureListener = object :
CameraCaptureSession.CaptureCallback() {
    override fun onCaptureCompleted(
        session: CameraCaptureSession,
        request: CaptureRequest,
        result: TotalCaptureResult
    ) {
        super.onCaptureCompleted(session, request, result)
        Handler(Looper.getMainLooper()).post {
            Toast.makeText(this@DetectActivity, "사진이
촬영되었습니다", Toast.LENGTH_SHORT)
                .show()
        }
        when (cameracheck) {
            1 ->
1   img_networking("http:/121.162.15.236:80/imgInformation",
photoFile)
            2 ->
2   img_search_networking("http:/121.162.15.236:80/imgSearch",
photoFile)
        }
        initCameraAndPreview()
    }
}
```

그림 1-18 사진 촬영 후 서버로 전송하는 코드

Fig 1-18. Code to send to server after taking photo

그림 1-18은 촬영한 사진 파일을 서버로 전송하는 과정이다. 1)은 사진 촬영 후 음료에 대한 이름, 가격, 영양 정보를 가져오는 코드이다. 2)은 찾고자 하는 음료와 촬영한 음료와 일치하는지 검사하는 코드이다.

```
override fun onResponse(call: Call, response: Response) {
    Log.d(TAG, "요청 완료")
    Log.d(TAG, "파일 이름: ${photoFile.name}")

    val resStr = response.body!!.string()
    val json = JSONObject(resStr)

    val obj = json.getString("name")
    Handler(Looper.getMainLooper()).post {
        Toast.makeText(applicationContext, obj,
Toast.LENGTH_SHORT).show()
    }
}

override fun onFailure(call: Call, e: IOException) {
    Log.d(TAG, "서버 요청 실패 ")
}
```

그림 1-19. 이미지 파일 http통신 유무

Fig 1-19. Request code to image file server

1-19는 서버 http응답을 받았는지에 대해 유무를 판단하여 사용자에게 전달하는 과정이다. 성공할 경우 파일을 서버로 전달하고, 실패할 경우 사용자에게 실패했다고 메시지로 전달한다.

1-4. 음성인식을 사용한 음료 판별

사용자의 음성을 인식하여 원하는 음료를 찾는 보조 기능이다. 원하는 음료를 말해서 카메라로 찍으면 음성으로 말한 음료와 카메라로 찍은 음료와 비교해서 같은지 다른지 판단하여 사용자에게 전달한다.

```
import android.speech.RecognizerIntent
import android.speech.SpeechRecognizer

// 음성 인식
private fun askSpeechInput() {
    if (!SpeechRecognizer.isRecognitionAvailable(this))
{
        Toast.makeText(this, "음성 인식을 사용할 수
없습니다.", Toast.LENGTH_SHORT).show()
    } else {
        val i =
Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEEC
H)
        i.putExtra(

RecognizerIntent.EXTRA_LANGUAGE_MODEL,

RecognizerIntent.LANGUAGE_MODEL_FREE_FORM
        )

i.putExtra(RecognizerIntent.EXTRA_LANGUAGE,
Locale.getDefault())
        i.putExtra(RecognizerIntent.EXTRA_PROMPT,
"원하는 음료를 말씀하세요.")
        startActivityForResult(i, RQ_SOEECH_REC)
    }
}
```

그림 1-20. Google 보이스 입력 활성화 코드

Fig 1-20. Google Voice Input Activation Code

1-20은 google speech api를 사용하여 사용자의 음성을 텍스트로 변화해주는 기능이다.

RecognizerIntent.ACTION_RECOGNIZE_SPEECH 함수에 의해 인식이 된다.

```
import android.speech.RecognizerIntent
import android.speech.SpeechRecognizer

override fun onActivityResult(requestCode: Int,
resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode,
data)

    // 음성 인식 결과값 변환
    if (requestCode == RQ_SOEECH_REC &&
resultCode == Activity.RESULT_OK) {
        val result =
data?.getStringArrayListExtra(RecognizerIntent.EXTRA
_RESULTS)
        VoiceText = result?.get(0).toString()
        if (resultCode == RESULT_OK) {

text_networking("http:/121.162.15.236:80/text",
VoiceText)
        }
    }
}
```

그림 1-21. 음성 인식 결과값 서버로 전송

Fig 1-21. Speech recognition result sent to server

1-21은 음성인식 후 문자로 변환된 데이터 값을 서버로 전송하는 코드이다. 이 방식도 카메라와 같이 http로 통신을 하여 서버로 전송하는 방식이다.

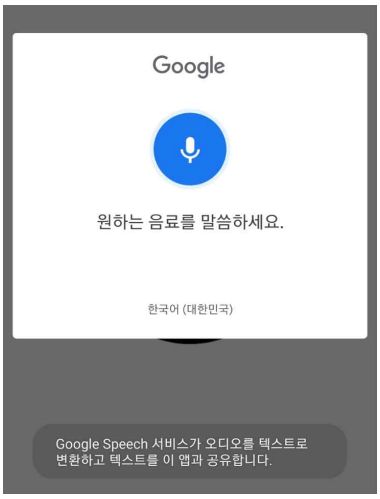


그림 1-22. google speech api 화면

Fig 1-22. Google Voice API screen

1-22화면에서 원하는 음료를 입력 받고 통신에 성공 하였으면 카메라 기능으로 이동되며, 촬영한 음료와 입력받은 음료와 비교하여 결과가 같은지 사용자에게 전달한다.

1-5. AsyncTask를 이용한 HttpUrlConnection

안드로이드는 서버와 통신하기 위한 방법으로는 HTTP통신이 있다. HTTP통신으로 URL 접속을 통해 데이터를 읽어오는 방법을 사용했다. DB에 존재하는 데이터를 가져오기 위해 서버 통신을 한다. 안드로이드의 특성상 외부 DB에 직접 접근할 수 가 없도록 되어있어 중간 매체인 WEB을 활용 했다. 그리고 네트웍 뿐만 아니라 앱의 동작을 매끄럽게 하기 위해서는 비동기 방식으로 동작할 수 있도록 AsyncTask 를 활용 하였다.

```
// 보낼 데이터가 없으면 파라미터를 비운다.
if (_params == null) sbParams.append("") else {
    // 파라미터가 2개 이상이면 파라미터 연결에 &가 필요하므로 스위칭할 변수 생성.
    var isAnd = false
    // 파라미터 키와 값.
    var key: String
    var value: String
    for ((key1, value1) in _params.valueSet()) {
        key = key1
        value = value1.toString()

        // 파라미터가 두개 이상일때, 파라미터 사이에 &를 붙인다.
        if (isAnd) sbParams.append("&")

sbParams.append(key).append("=").append(value)

        // 파라미터가 2개 이상이면 isAnd를 true로 바꾸고 다음 루프부터 &를 붙인다.
        if (!isAnd) if (_params.size() >= 2) isAnd = true
    }
}
```

그림 1-23. StringBuffer에 파라미터 연결

Fig 1-23. Connect parameters to StringBuffer

1-23은 StringBuffer에 파라미터 연결 하는 과정이다. StringBuffer는 mutable 객체이다. 즉 StringBuffer 객체는 생성이후에 해당 객체의 데이터를 변경할 수 있다. 대개는 append 메소드 정도만 사용하지만 그 외에도 데이터 변경이 이루어지는 메소드를 다수 제공한다.

```
try {
    val url = URL(_url)
    urlConn = url.openConnection() as HttpURLConnection

    // [2-1]. urlConn 설정.
    urlConn.setRequestMethod("POST") // URL 요청에 대한 메소드 설정 : POST.
    urlConn.setRequestProperty("Accept-Charset", "UTF-8") // Accept-Charset 설정.
    urlConn.setRequestProperty(
  "Context_Type",
  "application/x-www-form-urlencoded;cahrset=UTF-8"
    )

    // [2-2]. parameter 전달 및 데이터 읽어오기.
    val strParams = sbParams.toString() //sbParams에 정리한 파라미터들을 스트링으로 저장. 예)id=id1&pw=123;
    val os: OutputStream = urlConn.getOutputStream()
    os.write(strParams.toByteArray(charset("UTF-8"))) // 출력 스트림에 출력.
    os.flush() // 출력 스트림을 플러시(비운다)하고 버퍼링 된 모든 출력 바이트를 강제 실행.
    os.close() // 출력 스트림을 닫고 모든 시스템 자원을 해제.

    // [2-3]. 연결 요청 확인.
    // 실패 시 null을 리턴하고 메서드를 종료.
    if (urlConn.getResponseCode() !== HttpURLConnection.HTTP_OK) return null

    // [2-4]. 읽어온 결과물 리턴.
    // 요청한 URL의 출력물을 BufferedReader로 받는다.
    val reader = BufferedReader(InputStreamReader(urlConn.getInputStream(), "UTF-8"))

    // 출력물의 라인과 그 합에 대한 변수.
    var line: String?
    var page: String? = ""

    // 라인을 받아와 합친다.
    while (reader.readLine().also { line = it } != null) {
  page += line
    }
    return page
} catch (e: MalformedURLException) { // for URL.
    e.printStackTrace()
} catch (e: IOException) { // for openConnection().
    e.printStackTrace()
} finally {
    if (urlConn != null) urlConn.disconnect()
}
```

그림 1-24. HttpURLConnection을 통해 web의 데이터를 가져온다.

Fig 1-24. Get data from web through HttpURLConnection.

1-24는 HttpURLConnection 인스턴스는 단일 요청을 만드는 데 사용되지만 HTTP 서버에 대한 기본 네트워크 연결은 다른 인스턴스에서 투명하게 공유될 수 있다. 요청 후 HttpURLConnection의 InputStream 또는 OutputStream에서 close() 메서드를 호출하면 이 인스턴스와 연결된 네트워크 리소스가 해제될 수 있지만 공유 영구 연결에는 영향을 주지 않는다. 연결 해제 메서드를 호출하면 해당 시점에 영구 연결이 유휴 상태인 경우 기본 소켓이 닫힐 수 있다.

1-6. 어플리케이션 프로젝트 UI

그림 1-25. 시작_화면
Fig 1-25. start_screen

그림 1-26. 메인메뉴_화면
Fig 1-26. main menu_screen

그림 1-27. 음성검색_화면
Fig 1-27. voice search_screen

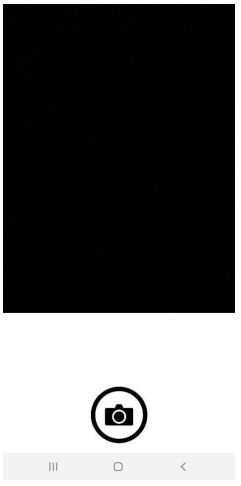


그림 1-28. 사진촬영_화면
Fig 1-28. Photo shoot_screen

그림(좌) 1-29. 편의점 검색_화면
Fig 1-29. Convenience store search_screen

그림(우) 1-30. 도보 길안내_화면
Fig 1-30. Walking Directions_screen

2. 서버

2-1. 데이터베이스 처리 프로세스

처리 프로세스는 서버 프로세스에서 활용하기 위해 데이터를 전처리하는 과정이다. 공공 데이터의 포맷을 통일하고 분석 프로세스에 활용할 컬럼을 추출 및 필터링하는 과정을 거친다. 그리고 데이터베이스를 사용하기 위한 준비를 한다.

```
1   import csv
    import requests
    from bs4 import BeautifulSoup
    import pymysql
    import openpyxl
    import pandas as pd
    import urllib.request
    import json

    mydb = pymysql.connect(
            user='root',
            passwd='smtown05',
            host='localhost',
            db='sulivan',
            charset='utf8'
2       )
    cur = mydb.cursor()

    df1 = pd.read_csv("한국소비자원 생필품 및 서비스 가격
    정보_20210827.csv", encoding='euc-kr', index_col=0)
    df1.drop(['Unnamed: 1'], axis = 1, inplace=True)
    cu = df1.loc[df1['판매업소'] == 'CU(본사)']
    gs25 = df1.loc[df1['판매업소'] == 'GS25(본사)']
    seveneleven = df1.loc[df1['판매업소'] ==
    '세븐일레븐(본사)']
```

그림 2-1. 모듈 호출 및 디렉토리 불러오기
Fig 2-1. Call Modules and Assigned Directories

그림 2-1는 필요한 모듈을 모두 호출하고 공공데이터가 저장된 디렉토리의 경로를 변수에 저장하는 과정이다. 1) drop 함수는 해당 경로의 필요없는 부분을 삭제해주는 역할을 한다. 2) loc 속성은 인덱스를 통해 행 데이터를 가져옵니다.

```
for index, row in cu.iterrows():
    price = row[0]
    convenience = row[1]
    beveragename = index
    cur.execute(insert_query3, (beveragename, price, convenience))
    mydb.commit()
for index, row in gs25.iterrows():
    price = row[0]
    convenience = row[1]
    beveragename = index
    cur.execute(insert_query3, (beveragename, price, convenience))
    mydb.commit()
for index, row in seveneleven.iterrows():
    price = row[0]
    convenience = row[1]
    beveragename = index
    cur.execute(insert_query3, (beveragename, price, convenience))
    mydb.commit()
```

그림 2-2. 가격 정보 전처리 및 데이터베이스 저장
Fig 2-2. Price Information Preprocessing and DB Save

그림 2-2는 생필품의 가격 정보를 전처리하고 해당 정보를 데이터베이스에 저장해주는 과정이다. 우선 해당 정보의 생필품 이름과 가격, 편의점의 정보를 행단위로 끊어서 데이터베이스에 저장해준다. 각각의 편의점 별로 다르게 저장해주는 과정이다.

beveragename	price	convenience
카스 프레시 6캔	12000	CU(본사)
카스 프레시(단품 500ml)	2700	CU(본사)
하이트 엑스트라콜드(6캔)	12000	CU(본사)
하이트 엑스트라콜드(단품)	2000	CU(본사)
비타500(묶음)	8000	CU(본사)
비타파워	6000	CU(본사)
컨피던스	1500	CU(본사)
스프라이트	2900	CU(본사)
삼다수(2L)	1700	CU(본사)
삼다수(500ml)	950	CU(본사)
아이시스(2L)	1700	CU(본사)
아이시스(500ml)	950	CU(본사)

그림 2-3. 가격 정보 전처리 및 DB 저장 결과
Fig 2-3. Price Information Preprocessing and DB Save Output

```
url = 'https://pyony.com/search/?event_type=&category=1&item=&sort=&q='
url2 = "https://www.fatsecret.kr/%EC%B9%BC%EB%A1%9C%EB%A6%AC-%EC%98%81%EC%96%91%EC%86%8C/search?q=%EC%9D%8C%EB%A3%8C&pg="
rurl = "https://www.fatsecret.kr/%EC%B9%BC%EB%A1%9C%EB%A6%AC-%EC%98%81%EC%96%91%EC%86%8C/search?q=%EC%97%90%EB%84%88%EC%A7%80+%EC%9D%8C%EB%A3%8C&pg="
insert_query = "insert into event (shopname, beveragename, eventname, price) values (%s, %s, %s, %s);"
insert_query2 = "insert into nutritionfacts (foodname, kcal, carbohydrate, protein, Fat) values (%s, %s, %s, %s, %s);"
insert_query3 = "insert into beverage (beveragename, price, convenience) values (%s, %s, %s);"
```

그림 2-4. 편의점 행사정보 및 음료수 영양정보 전처리
Fig 2-4. Convenience store event information and Beverage Nutrition Facts Preprocessing

편의점 행사정보 및 음료수 영양정보 데이터를 추출하기 위한 전처리 과정이다. 크롤링을 위한 주소를 변수에 저장해주는 과정이다.

```
for page in range(0, 100):
    response = requests.get(url2+str(page))
    for i in range(0, 10):
        if response.status_code == 200:
            html = response.text
            soup = BeautifulSoup(html, 'html.parser')
            foodname = soup.select('.prominent')[i].get_text()
            url3 = soup.select(".prominent")[i]['href']
            response2 = requests.get("https://www.fatsecret.kr"+url3)
            html = response2.text
            soup2 = BeautifulSoup(html, 'html.parser')
            kcal = soup2.select('.nutrient.black.right.tRight')[0].get_text()
            carbohydrate = soup2.select('.nutrient.black.right.tRight')[1].get_text()
            protein = soup2.select('.nutrient.black.right.tRight')[2].get_text()
            Fat = soup2.select('.nutrient.black.right.tRight')[3].get_text()
            cur.execute(insert_query2, (foodname, kcal, carbohydrate, protein, Fat))
            mydb.commit()
```

그림 2-5 음료수 영양정보 전처리 코드
Fig 2-5 Beverage Nutrition Facts Preprocessing Code

음료수 영양정보를 정리해논 사이트를 크롤링해서 해당하는 음료수의 이름, 칼로리, 탄수화물, 단백질, 지방의 정보를 읽어들여서 해당하는 정보에 맞게 끊어서 행단위로 해당하는 정보를 데이터베이스에 저장 한다.

foodname	kcal	carbohydrate	protein	Fat
1 % 지방 우유	176 kJ	4.99g	3.37g	0.97g
170일 배	397 kJ	23.00g	0.00g	0.00g
2 % 지방 우유	510 kJ	11.42g	8.05g	4.81g
790일 알로에	335 kJ	18.00g	0.00g	0.50g
Assam Black Tea	502 kJ	30.00g	0.00g	0.00g
At home 오렌지주스	188 kJ	11.00g	0.50g	0.00g
Be 코코넛 워터	84 kJ	5.00g	0.00g	0.00g
G2	84 kJ	8.00g	0.00g	0.00g
GET 카페라떼	778 kJ	27.00g	6.00g	6.00g
I'm Real Greenkiwi	523 kJ	31.00g	1.00g	0.00g
Liftoff	42 kJ	3.00g	0.00g	0.00g

그림 2-6 음료수 영양정보 전처리 결과
Fig 2-6 Beverage Nutrition Facts Preprocessing Output

```
for page in range(0, 80):
    response = requests.get(url+str(page))
    if response.status_code == 200:
        html = response.text
        soup2 = BeautifulSoup(html, 'html.parser')
        keys = soup2.select(".col-md-6")
        for key in keys:
            shopname =
key.select_one('small').get_text()
            beveragename =
key.select_one("strong").get_text()
            price = key.select_one('a > div >
div.card-body.px-2.py-2 >
div:nth-child(2)').contents[6].get_text().strip()
            eventname = ''
            if key.select_one('small').get_text() ==
'MINISTOP(미니스톱)':
                eventname =
key.select_one(".badge.bg-ministop.text-white").get_text
()
            elif key.select_one('small').get_text() ==
'GS25(지에스25)':
                eventname =
key.select_one(".badge.bg-gs25.text-white").get_text()
            elif key.select_one('small').get_text() ==
'CU(씨유)':
                eventname =
key.select_one(".badge.bg-cu.text-white").get_text()
            elif key.select_one('small').get_text() ==
'EMART24(이마트24)':
                eventname =
key.select_one(".badge.bg-emart24.text-white").get_text
()
            elif key.select_one('small').get_text() ==
'7-ELEVEN(세븐일레븐)':
                eventname =
key.select_one(".badge.bg-seven.text-white").get_text()
            cur.execute(insert_query, (shopname,
beveragename, eventname, price))
            mydb.commit()
    else:
        print(response.status_code)
```

그림 2-7 편의점 행사 정보 전처리 코드
Fig 2-7 Convenience store event information Preprocessing Code

편의점 행사정보를 정리해논 사이트를 크롤링해서 편의점이름, 행사상품, 행사종류, 행사가격보를 읽어들여서 해당하는 정보에 맞게 끊어서 행단위로 해당하는 정보를 데이터베이스에 저장한다.

shopname	beveragename	eventname	price
MINISTOP(미니스톱)	국산검은콩21곡두유180ml	2+1	1,400 원
MINISTOP(미니스톱)	닥터유단백질초코240ml	2+1	2,000 원
MINISTOP(미니스톱)	깨수깡160ml캔	1+1	5,000 원
MINISTOP(미니스톱)	동서)티오피스무스라떼240	2+1	1,800 원
MINISTOP(미니스톱)	롯데)미린다파인355캔	1+1	1,200 원
MINISTOP(미니스톱)	G7커피믹스10T	2+1	2,500 원
MINISTOP(미니스톱)	몰몬트오렌지400펫	2+1	1,900 원
MINISTOP(미니스톱)	카페라떼마일드220	2+1	1,600 원
MINISTOP(미니스톱)	가나초코310	2+1	1,500 원
MINISTOP(미니스톱)	깨수깡160ml캔	1+1	5,000 원

그림 2-8 편의점 행사정보 전처리 결과
Fig 2-8 Convenience store event information Preprocessing Output

2-2. 서버 프로세스

```
@app.route('/imgInformation', methods=['GET',
'POST'])
def imageInformation():
    files_ids = list(flask.request.files)
    print(files_ids)
    image_num = 1
    for file_id in files_ids:
        imagefile = flask.request.files[file_id]
        filename =
werkzeug.utils.secure_filename(imagefile.filename)
        timestr =
time.strftime("%Y%m%d-%H%M%S")
        imagefile.save(timestr + '_' + filename)
        image_num = image_num + 1
        name = find(timestr + '_' + filename)
        sql = 'select * from nutritionfacts where
foodname like "%{}%"'.format(name)
        sql2 = 'select * from beverage where
beveragename like "%{}%"'.format(name)
        sql3 = 'select * from event where
beveragename like "%{}%"'.format(name)
        mycursor.execute(sql)
        nutrition = mycursor.fetchall()
        nutrition_facts = "칼로리"+nutrition[0][1] +
"지방"+nutrition[0][2] + "탄수화물" + nutrition[0][3] +
"단백질" + nutrition[0][4]
        mycursor.execute(sql2)
        price = mycursor.fetchall()
        mycursor.execute(sql3)
        event = mycursor.fetchall()
        if len(event) > 1:
            event = event[0][2] + "행사상품입니다."
        else:
            event = "행사상품이아닙니다."

        os.remove(timestr + '_' + filename)
    return jsonify({'object': name, 'price':
price[0][1]+"원", 'nutrition_facts': nutrition_facts,
'event':event})
```

그림 2-9 음료수 이미지 인식 및 가격, 영양 정보 통신 코드
Fig 2-9 Beverage Image recognition and Price, Nutritional Information communication Code

그림 2-9는 시각장애인이 어플리케이션으로 음료수사진을 찍어서 서버로 보내주면 서버에서 해당하는 이미지를 딥러닝을 이용하여 해당하는 이미지의 음료수를 정보를 분류해서 해당하는 음료의 이름이 변수에 저장해주고 해당하는 음료의 이름, 가격 및 영양정보를 데이터베이스에서 찾아줘서 해당 음료의 이름, 가격 및 칼로리, 탄수화물, 단백질 지방의 정보를 어플리케이션으로 보내주는 과정이다.

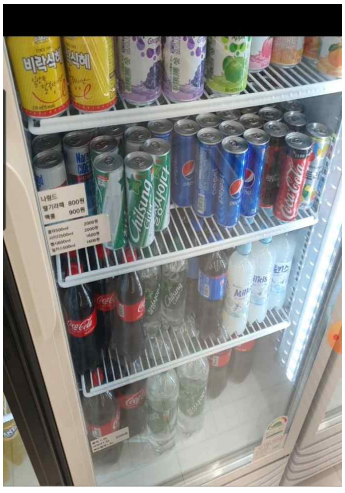


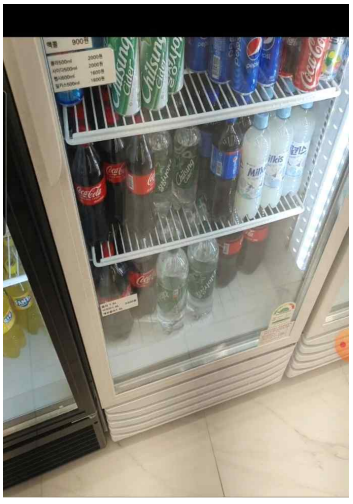
그림 2-10 음료수 이미지 인식 및 가격, 영양 정보 통신 결과
Fig 2-10 Beverage Image recognition and Price, Nutritional Information Communication Output

```
@app.route('/text', methods=['GET', 'POST'])
def text():
    texts = request.form
    select_name = "당신이 찾는 음료는" + texts['beveragename'] + "입니다."

    return jsonify({"name": select_name})
```

그림 2-11 음성 입력 통신 코드
Pig 2-11 Voice Input Communication Code

그림 2-11은 어플리케이션에서 시각장애인이 원하는 음료를 찾아주기위해서 시각장애인이 원하는 음료를 음성으로 말해주면 해당 음성을 서버에 보내주면 해당하는 음료가 시각장애인이 원하는 음료인지 확인하기 위한 통신의 과정이다.



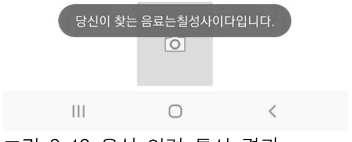


그림 2-12 음성 입력 통신 결과
Pig 2-12 Voice Input Communication Output

```
@app.route('/imgSearch', methods=['GET', 'POST'])
def imageSearch():
    files_ids = list(flask.request.files)
    image_num = 1

    for file_id in files_ids:
        imagefile = flask.request.files[file_id]
        filename = werkzeug.utils.secure_filename(imagefile.filename)
        timestr = time.strftime("%Y%m%d-%H%M%S")
        imagefile.save(timestr + '_' + filename)
        image_num = image_num + 1
        name = find(timestr + '_' + filename)
        name = "당신이 고른 음료는" + name + "입니다."

        os.remove(timestr + '_' + filename)
    return jsonify({'name': name})
```

그림 2-13 음료 찾기 통신 코드
Pig 2-13 Find Beverage Communication Code

그림 2-13은 시각장애인이 어플리케이션으로 음료사진을 찍었을 때 해당하는 음료의 이름만 알려주는 기능을 한다. 해당하는 사진을 딥러닝을 이용하여 해당하는 음료의 정보를 어플리케이션으로 보내주는 기능을 한다.

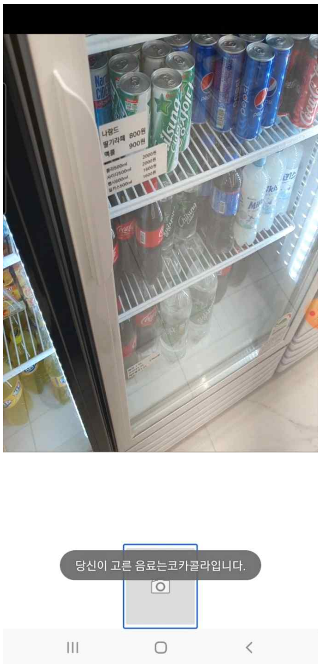


그림 2-14 음료 찾기 통신 결과

Fig 2-14 Find Beverage Communication Output

장동혁

3. 이미지 인식

3-1. 이미지 학습

이미지 인식은 사용자가 정보를 원하는 제품을 스마트폰 카메라로 촬영 후 촬영된 이미지를 서버로 전송. 전송된 이미지를 분석 후 학습 된 모델로 객체를 검출하여 결과 값을 리턴 하는 과정이다.

```
1   from PIL import Image
    import os, glob, numpy as np
    from sklearn.model_selection import train_test_split

2   caltech_dir = "DataSet"
    categories = ["코카콜라", "칠성사이다"]
    nb_classes = len(categories)
```

그림 3-1. 모듈 호출 및 디렉토리 불러오기

Fig 3-1. Call Modules and Assigned Directories

그림 3-1은 이미지 프로세싱을 위한 모듈을 호출하며 카테고리 별로 네이밍을 하는 과정이다. 1) 파이썬의 내장모듈이 아닌 라이브러리들은 명령 프롬프트에서 pip 명령, 또는 conda 명령을 이용한 별도의 설치가 필요하다. 2) 이미지가 저장되어있는 경로를 미리 명시해 놓으며 카테고리를 리스트화 하여 저장한다.

```
1   image_w = 64
    image_h = 64

    pixels = image_h * image_w * 3

    X = []
    y = []

    for idx, cat in enumerate(categories):
        label = [0 for i in range(nb_classes)]
        label[idx] = 1

        image_dir = caltech_dir + "/" + cat
        files = glob.glob(image_dir+"/*.jpg")
        print(cat, " 파일 길이 : ", len(files))

2       for i, f in enumerate(files):
            img = Image.open(f)
            img = img.convert("RGB")
            img = img.resize((image_w, image_h))
            data = np.asarray(img)

            X.append(data)
            y.append(label)

            if i % 700 == 0:
                print(cat, " : ", f)

3   X = np.array(X)
    y = np.array(y)

    X_train, X_test, y_train, y_test = train_test_split(X, y)
    xy = (X_train, X_test, y_train, y_test)
    np.save("multi_image_data.npy", xy)
```

그림 3-2. 디렉토리 불러오기 및 이미지 조절

Fig 3-2. Assigned Directories and image resize.

그림 3-2는 이미지 프로세싱을 위하여 사이즈 조절을 하며 변환된 이미지를 numpy배열에 저장하여 npy로 저장하는 기능을 수행한다. 1) 빠른 인식을 위하여 학습에 사용될 이미지를 64*64 크기로 조절을 한다. 2) 조절 된 이미지들을 RGB형태로 변환하며 해당 데이터를 numpy array로 저장한다. 3) 저장된 데이터를 np.save를 이용하여 npy파일 형태로 저장한다.

```
1  import matplotlib.pyplot as plt
   import os, glob, numpy as np
   from keras.models import Sequential
   from keras.layers import Conv2D, MaxPooling2D, Dense,
   Flatten, Dropout
   from keras.callbacks import EarlyStopping,
   ModelCheckpoint
   import matplotlib.pyplot as plt
   import tensorflow as tf
   from tensorflow.python.framework.convert_to_constants
   import convert_variables_to_constants_v2

2  config = tf.compat.v1.ConfigProto()
   config.gpu_options.allow_growth = True
   session = tf.compat.v1.Session(config=config)

3  X_train, X_test, y_train, y_test =
   np.load('multi_image_data.npy', allow_pickle=True)

4  X_train = X_train.astype(float) / 255
   X_test = X_test.astype(float) / 255
```

그림 3-3. 모듈 호출 및 npy 불러오기 및 일반화 , GPU 로드

Fig 3-3. Call Modules and Assigned npy file, npy file generalization, GPU load

그림 3-3은 이미지 프로세싱을 위한 모듈을 호출하며 카테고리 마다 네이밍을 하는 과정이다. 1) 파이썬의 내장모듈이 아닌 라이브러리들은 명령 프롬프트에서 pip 명령, 또는 conda 명령을 이용한 별도의 설치가 필요하다. 2) GPU 사용을 위하여 GPU 확인 및 Session확인을 위한 함수를 사용한다. 3) 그림3-2에서 저장한 npy를 불러온다. 4) 불러온 npy는 학습을 위하여 3차원데이터를 2차원 배열로 표기하기 위해 astype 함수를 통해 실수화(float) 한 후 255로 나누어 일반화를 한다.

```
1  model = Sequential()
   model.add(Conv2D(32, (3,3), padding="same",
   input_shape=X_train.shape[1:], activation='relu'))
   model.add(MaxPooling2D(pool_size=(2,2)))
   model.add(Dropout(0.25))

   model.add(Conv2D(64, (3,3), padding="same",
   activation='relu'))
   model.add(MaxPooling2D(pool_size=(2,2)))
   model.add(Dropout(0.25))

   model.add(Flatten())
2  model.add(Dense(256, activation='relu'))
   model.add(Dropout(0.5))
   model.add(Dense(nb_classes, activation='softmax'))
   model.compile(loss='categorical_crossentropy',
   optimizer='adam', metrics=['accuracy'])
```

그림 3-4. 학습 레이어 생성 및 학습 진행.

Fig 3-4. Call Modules and Assigned npy file, npy file generalization, GPU load

그림 3-4는 학습에 필요한 모델 레이어(Layer)를 생성하며 각 레이어별로 신경망을 구성한다. 1) 모델을 구성하기 위해 Sequential()을 이용하여 설계를 시작한다. 사용된 레이어는 컨볼루션(Convolution Layer)를 사용하였다. 이미지 혹은 영상처리에 주로 사용되는 Conv2D레이어를 사용하였다. 첫 Layer는 64개의 filter를 사용하였으며 커널은 (3, 3)으로 설정, Padding값은 same 정의하여 출력 이미지 사이즈와 입력 임지 사이즈를 동일하게 설정하였다. 또한 활성화 함수는 relu로 설정 하였다.

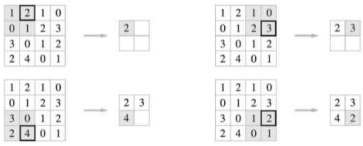


그림 3-5. MaxPooling 예시

Fig 3-5. Example MaxPooling

이후 이미지의 크기를 줄이면서 데이터의 손실을 막으며 합성곱 계층의 과적합(Overfitting)을 막기 위해 MaxPooling2D Layer를 추가하였다. Pooling의 크기는 (2, 2)로 설정하였다. 또한 Padding 값은 'same'으로 하여 입력 데이터와 출력 데이터의 크기를 동일하게 설정하였다. 'same'의 경우 데이터가 없는 부분에만 zero-padding이 생겨 데이터의 손실을 막아준다. 이후 추가적으로 과적합을 방지하기 위한 정규화 기법으로 dropout을 추가하였다. 계속해서 과적합을 방지하는 이유는 충분한 데이터가 부족하기 때문에 여러 학습을 통하여 과적합 방지를 진행하여 최적의 결과가 나오게 함을 위함이다.

이후 동일한 층을 생성 후 Flatten() 함수를 이용하여 입력 이미지를 1차원 배열로 만든다. 이후 Softmax 활성화 함수를 통하여 최종적인 분류를 한다.

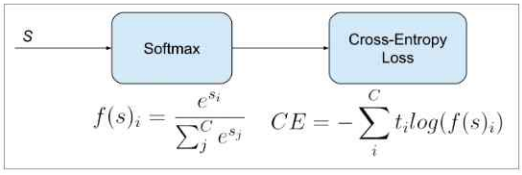


그림 3-6. Categorical Cross Entropy formula

Fig 3-6. Categorical Cross Entropy formula

설정한 합성곱 계층을 이용하여 compile()을 진행한다. 손실함수(loss function)은 Categorical Cross Entropy를 사용하였다. 주로 분류에 사용되며 특히 제품의 종류를 구분해야하는 본 개발과 적절하다고 생각되어 선택하였다.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\omega J(\omega_t)$$

$$v_t = \beta_2 m_{t-1} + (1-\beta_2)(\nabla_\omega J(\omega_t))^2$$

$$\omega_{t+1} = \omega_t - m_t \frac{\eta}{\sqrt{v_t + \epsilon}}$$

그림 3-6. Adaptive Moment Estimation formula

Fig 3-6. Adaptive Moment Estimation formula

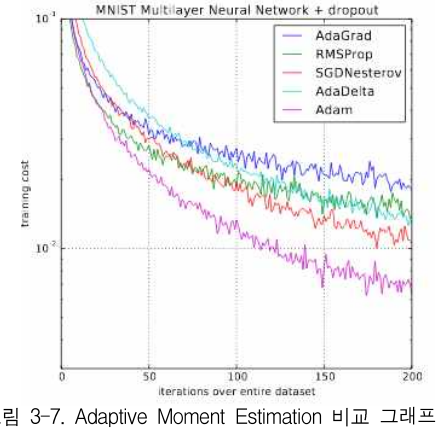


그림 3-7. Adaptive Moment Estimation 비교 그래프

Fig 3-7. Adaptive Moment Estimation graph

Optimizer는 그림 3-6의 Adam을 사용하였다. Adam은 RMSProp과 Momentum을 배분하여 방향과 스텝사이즈를 상황에 맞게 대응하여 조절하는 Optimizer이다.

Momentum에서 관성계수 m과 함께 계산된 Vt로 parameter를 업데이트하지만 Adam에서는 기울기 값과 기울기의 제곱 값의 지수이동평균을 활용하여 step변화량을 조절한다.

학습 결과 측정 방법은 'Accuracy'를 기준으로 측정하였다.

```
1  if not os.path.exists(model_dir):
       os.mkdir(model_dir)

   model_path = model_dir + '/asd.pb'
   checkpoint = ModelCheckpoint(filepath=model_path ,
   monitor='val_loss', verbose=1, save_best_only=True)
   early_stopping = EarlyStopping(monitor='val_loss',
   patience=6)


   model.summary()

2  history = model.fit(X_train, y_train, batch_size=64,
   epochs=30, validation_data=(X_test, y_test))

   print("정확도 : %.4f" % (model.evaluate(X_test,
   y_test)[1]))
```

그림 3-8. 저장된 레이어 불러오기 및 학습

Fig 3-8. Categorical Cross Entropy formula

그림 3-8은 저장한 pb 파일을 불러오며 체크 포인트 생성 및 모델 학습을 하는 코드이다. batch size는 64, epochs는 30회로 설정하였다. 위와 같이 설정한 기준은 여러 값을 대입 하여 학습해본 결과 가장 높은 인식률을 나타내었기 때문이다.

3-2. 손 기준 최근접 객체 우선 인식

```
1  def find(path):
       cv2.namedWindow('image')
       img = cv2.imread(path, cv2.IMREAD_COLOR)
       img = resize(img)
       img_hsv = cv2.cvtColor(img,
   cv2.COLOR_BGR2HSV)
       # 잡음 제거
       img_hsv =
   cv2.fastNlMeansDenoisingColored(img_hsv, None, 10,
   10, 7, 21)
       lower = np.array([0, 48, 80], dtype="uint8")
       upper = np.array([20, 255, 255], dtype="uint8")
       img_hand = cv2.inRange(img_hsv, lower, upper)

       contours, hierarchy = cv2.findContours(img_hand,
   cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
2      max = 0
       maxcnt = None
       for cnt in contours:
           area = cv2.contourArea(cnt)
           if max < area:
               max = area
               maxcnt = cnt
```

그림 3-9. 이미지 불러오기 및 잡음 제거

Fig 3-9. load image file and noise canceling

본 개발의 이미지 인식 기능 중 하나인 손을 기준으로 하는 근접 객체를 우선 인식하는 코드이다. 1)먼저 이미지를 불러온 후 이미지 사이즈 조정 및 전체 색을 변경한다. 변경하는 색은 cv2.COLOR_BGR2HSV 이다. 또한 2)잡음 제거를 위하여 fastNlMeansDenoisingColored를 활용한 잡음 제거를 진행한다.

이후 findContours를 이용하여 경계선을 찾은 뒤 가장 큰 영역을 찾는다.

```
hull = cv2.convexHull(maxcnt)
    xRangeList = []
    for i in range(len(hull)):
        xRangeList.append(hull[i][0][0])

    xRangeList = sorted(xRangeList)
    mask = np.zeros(img.shape).astype(img.dtype)
    color = [255, 255, 255]
    cv2.fillPoly(mask, [maxcnt], color)
    img_hand = cv2.bitwise_and(img, mask)
    cv2.drawContours(img_hand, [maxcnt], 0, (255, 0, 0),
3)
    cv2.drawContours(img_hand, [hull], 0, (0, 255, 0), 3)

    img_hand = img[:, xRangeList[0]:xRangeList[-1]]
    cv2.imwrite(path, img_hand)

    return Detection_img.predict(path)
```

그림 3-10. 이미지 파일 변환

Fig 3-10. Convert image file

본 3-9에서 잡음 제거 및 경계선 찾기 및 최대 크기의 영역의 각 꼭짓점의 다각선을 만든 이후 X축 기준의 최대 최소를 저장. 경계선 내부를 255로 채운다. 이후 나온 이미지를 Detection_img의 predict 함수로 전달한다.

```
from keras.models import load_model
from PIL import Image
import numpy as np
import tensorflow as tf
from tensorflow.keras import datasets, layers, models

gpus =
tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu,
True)
    except RuntimeError as e:
        print(e)
```

그림 3-11. 이미지 인식을 위한 GPU 로드

Fig 3-11. Detection image GPU load.

이미지 인식을 위한 GPU 로드 및 필요 라이브러리 선언. 본 논문에서는 NVIDIA GeForce GTX1060 6GB를 사용.

```
    image_w = 64
    image_h = 64
    pixels = image_h * image_w * 3
    X = []
    filenames = []

    img = Image.open(f)
    img = img.convert("RGB")
    img = img.resize((image_w, image_h))
    data = np.asarray(img)
    filenames.append(f)
    X.append(data)
    X = np.array(X)
    model = load_model('model/capstone_model.h5')

    prediction = model.predict(X)

    np.set_printoptions(formatter={'float': lambda x:
"{0:0.3f}".format(x)})
    cnt = 0
```

그림 3-12. 이미지 인식

Fig 3-12. Predict Image

전달 받은 이미지를 먼저 RGB로 convert 한 뒤 해당 이미지의 사이즈를 조정. 이미지를 numpy의 asarray로 3차원 변환.

변환된 이미지의 3차원 배열을 기존에 학습 해 놓은 모델을 불러 온 뒤 predict를 실행.

```
    for i in prediction:
        pre_ans = i.argmax()
        pre_ans_str = ''
        if pre_ans == 0:
            pre_ans_str = "코카콜라"
        elif pre_ans == 1:
            pre_ans_str = "칠성사이다"

        if i[0] >= 0.8:
            return pre_ans_str
        if i[1] >= 0.8:
            return pre_ans_str
```

그림 3-13. 최종 결과값 리턴

Fig 3-13. Return result

실행한 predict 따라 저장된 예측 레이블의 값에 따라 최종적으로 사용자에게 전송할 데이터를 분류. 본 논문에서는 예시를 위하여 레이블을 두 개만 진행.

Ⅳ. 실험

메인 화면

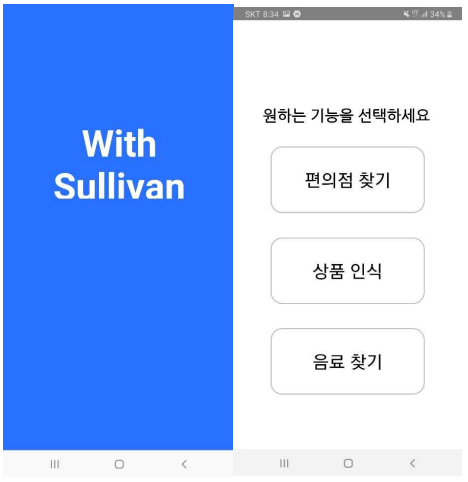


그림 1-1. 메인 화면

Fig 1-1. Main View

사용자에 특화된 메뉴 버튼으로 편의점 찾기를 실행

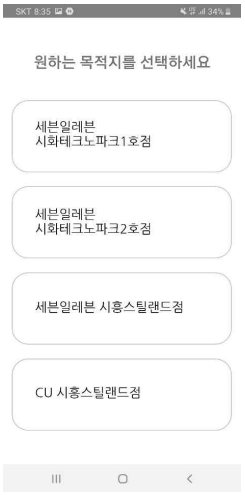


그림 1-2. 편의점 검색_화면

Fig 1-2. Convenience store search_screen

편의점 찾기 실행 시 사용자 도보기준 최단거리 편의점 안내

그림 1-3. 도보 길안내_화면

Fig 1-3. Walking Directions_screen

사용자가 선택한 편의점을 도보로 안내 실시간 이동에 따라 경로 안내. 횡단보도, 좌회전, 우회전 등 실시간 안내 제공.

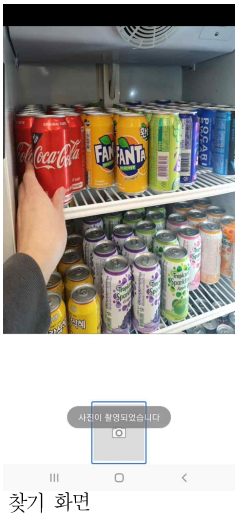


그림 1-4. 음료 찾기 화면

Fig 1-4. Find product mode screen

편의점에 도착 시 자동으로 상품 찾기 모드로 변경.

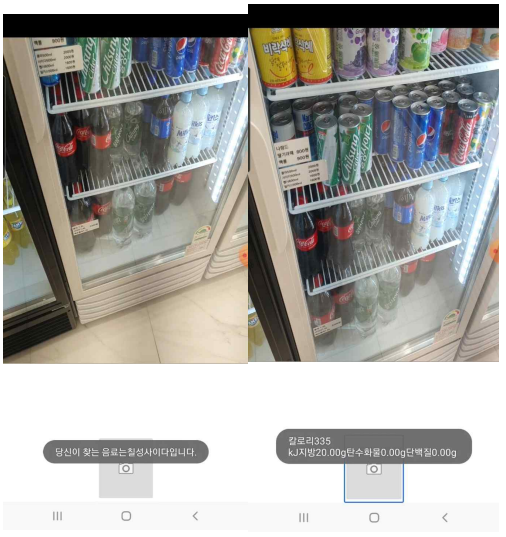


그림 1-6 사용자가 촬영한 이미지 결과 전송
Pig 1-6 image Detection and Result output

사용자가 촬영한 이미지의 결과 값을 서버와 통신한 후 인식된 결과 값을 공공데이터와 결합하여 가격데이터, 할인 정보, 영양 정보를 실시간 데이터로 반영하여 사용자에게 음성으로 전송.

Ⅴ. 결론

본 논문에서는 공공데이터 API 및 수집한 데이터 셋을 활용하여 만든 어플리케이션으로 시각장애인들이 당연히 누려야 할 소비자로서의 권리를 보장 받으며 기업의 사회적 책임을 무시하며 손해를 이유로 방관하는 태도를 대중들에게 알리는 역할을 할 수 있는 개발을 수행하였다. 본 개발을 통하여 시각장애인 뿐만 아니라 모든 장애인들의 불편함을 해소 할 수 있는 시발점이 될 것이다.

References

[1] Yoeng-Woon Kim, Jong-Ki Park, Jae-Hoon Yu, Young-Sup Hwang, Jeong Heo, “Energy Efficient Sleep Scheduling By utilizing Response Time Slowdown of Concurrent HTTP Connection”, Proceedings of the Korean Information Science Society Conference, pp. 238-242, Jun 2007.
DOI: http://koreascience.or.kr/article/CFKO200724737420258.page

[2] Chang Gi Kim, Jeong Min Seo, "An Design and Implementation of Navigation System for Visually Impaired Persons Based on Smart Mobile Devices", The Journal of the Korea Contents Association, pp. 24-30, Dec 2014
DOI: http://koreascience.or.kr/article/JAKO201506565684321.page

[3] Tae-Guon Kim, Bong-Wan Kim, Dae-Lim Choi, Yong-Ju Lee, "Implementation of Korean TTS Service on Android OS", The Journal of the Korea Contents Association, pp. 9-16, Dec 2011.
DOI: http://koreascience.or.kr/article/JAKO201208040008467.page

[4] Tae-Jin Yun, Hyo-Jong Seo, Do-Heon Kim, "Text/Voice Recognition & Translation Application Development Using Open-Source", Proceedings of the Korean Society of Computer Information Conference, pp. 425-426, Jul 2017
DOI: http://koreascience.or.kr/article/CFKO201731342442605.page

[5] Eun-bi Shin, Tae-Kyung Roh, "Inforamtion Application for The blind people", Proceedings of the Korean Institute of Information and Commucation Sciences Conference, pp. 358-359, May 2018.
DOI: http://koreascience.or.kr/article/CFKO201821464986654.page

[6] Hyeon Seung Cho, Gi Hoon Lee, Sin Woo Kim, Hee Chang Jeong, Dong Il Kim, "Place search service by network", Proceedings of the Korean Institute of Information and Communication Sciences Conference, pp. 655-656, May 2018
DOI: http://koreascience.or.kr/article/CFKO201821464987066.page

[7] Yoeng-Woon Kim, Jae-Hoon Yu, Jeong Heo, “Blind Helper program development by using Wireless Camera and Window phone”, Proceedings of the Korea Information Processing Society Conference, pp. 474-477, Nov 2012.
DOI: http://koreascience.or.kr/article/CFKO201221868476986.page

[8] Chang-Gi Kim, Jeong-Min Seo, “A Navigation System for Visually Impaired Persons Using Smart Phone”, Proceedings of the Korean Society of Computer

Information Conference, pp. 405-406, July 2014
DOI: http://koreascience.or.kr/article/CFKO201421553670167.page
[9] Sun-Young Bae, “Research Trends on Related to Artificial Intelligence for the Visually Impaired : Focused on Domestic and Foreign Research in 1993-2020”, The Journal of the Korea Contents Association, Vol. 20, No. 10, pp. 688-701, Sep 2020
DOI: http://koreascience.or.kr/article/JAKO202031458604056.page
[10] Young-Jun Kim, Jong-Geun Yun, Jae-Hyuk Hur, Jae-Ho Shin, Woo-Chul Kang, “An auxiliary mechanism for vision obstruction using DeepLearning”, Proceedings of the Korea Information Processing Society Conference, pp. 853-856, Nov 2071.
DOI: http://koreascience.or.kr/article/CFKO201735553776826.page
[11] Sang-Heon Park, Tae-Jae Jeon, Sang-Hyuk Kim, Sang-Youn Lee, Ju-Wan Kim, “Deep learning based symbol recognition for the visually impaired”, The Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol. 9, No. 3, pp. 249-256, Apr 2016.
DOI: http://koreascience.or.kr/article/JAKO201621650894964.page
[12] Yun-Jik Lee, Young-Joon Hwnag, Tae-Ho Lee, Han-Byoul Kang, Ki-Young Lee, “Wardrobe System for Blind Based On Image Processing and Deep Learning”, Proceedings of the Korea Information Processing Society Conference, pp. 962-964, Oct 2019
DOI: http://koreascience.or.kr/article/CFKO201924664108449.page
[13] Sang-Hyeok Han, Da-Soo Park, Chae-Min Lim,, Ji-Woon Jeong, “Convenience Store Product Recognition Application for the Blind”, Proceedings of the Korea Information Processing Society Conference, pp. 1298-1301 Nov 2021
DOI: http://koreascience.or.kr/article/CFKO202133648981958.page