

분리수거 척척박사

Smart Recycle

2021.12.15

건국대학교 컴퓨터공학부 졸업프로젝트1

컴퓨터공학부 201814116 권태윤

컴퓨터공학부 201613187 송용수

컴퓨터공학부 201613188 송태인

지도교수 김 기 천 교수님

목차

1. 영문 Summary.....	2
2. 개요.....	3
3. 요구사항 정의.....	5
4. 시스템 설계.....	10
5. 시스템 구현.....	20
6. 시스템 실행 결과.....	29
7. 시스템 시험 결과.....	33
8. 팀원 및 역할 분담.....	47
9. 맺음말.....	48

1. 영문 Summary

Recently, non-face-to-face consumption has increased due to COVID-19. Instead of going to a restaurant, order delivery food, shop by courier, and take-out cafe drinks, increasing the rate of packaging and disposable items. The garbage crisis has begun.

We may be somewhat curious about this situation. This is because recycling is practiced every day. Korea has an environment where not only apartments and companies, but also public trash cans can be separated, and is known as the second-best recycling country among OECD countries (as of 2013).

But the problem is that this is not the case. The recycling figure in Korea is known to be about 86% as of 2019, which is not the actual recycling rate.

This is because there are many trashes that has not been discharged in the right way. It is true that many people are confused about how to handle ramen containers and where to discharge them. If you look inside garbage bags thrown away in Korea, about 70% of them are resources that can be used as recyclables. Even recycling resources are incinerated or reclaimed due to improper separation and discharge, leading to resource shortages and environmental pollution. Just by preventing recycling resources from being thrown away in pay-as-you-go bags, you can save about 300 billion won a year in the purchase of pay-as-you-go bags.

Therefore, the "Doctor of ChuckChuck" project will inform you of the correct method of discharging separation collection.

There are two main functions.

1. When a user delivers an image to a model through a mobile phone, the model informs the classification result.
2. The user can know the separation collection method of objects that are difficult to separate through the separation collection category.

The camera uses the user's mobile phone, and the classification model used Tensorflow's EfficientNet-Lite V3 model, and accuracy was the most important.

The "Doctor of ChuckChuck" aims to help protect the environment through the proper classification of recyclables by informing every one of the methods of separation collection easily.

2. 개요

2.1 프로젝트 동기

최근 코로나 19로 인해 비 대면 소비가 늘어났다. 식당에 가는 대신 배달음식을 시켜 먹고, 택배로 장을 보고, 카페 음료는 테이크아웃 하면서 포장재와 일회용품 사용 비율이 급증했다. 쓰레기 대란이 시작된 것이다.

우리는 이 상황이 다소 의아할 수 있다. 매일같이 재활용을 위해 분리수거를 실천하기 때문이다. 우리나라는 아파트나 회사뿐만 아니라 공공 쓰레기통까지 어디든 분리수거 할 수 있는 환경이 꾸려져 있고, OECD 국가 중 재활용을 잘하는 국가 2위로 알려져 있기도 하다 (2013년 기준).

하지만 문제는 실상은 그렇지 않다는 것이다. 우리나라 재활용 수치는 2019년 기준 약 86%라고 알려져 있는데, 이는 실질적인 재활용률이 아니다.

올바른 방법으로 배출하지 않은 쓰레기가 많기 때문이다. 먹고 난 라면 용기를 어떻게 처리하여, 어디에 배출해야 하는지 헷갈리는 사람이 많은 것은 사실이다. 국내에서 버려지는 쓰레기 종량제 봉투 속을 살펴보면 약 70%는 재활용품으로 활용할 수 있는 자원이다. 분리배출을 올바르게 하지 않아 재활용될 수 있는 자원조차 소각되거나 매립되어 자원 부족과 환경오염으로 이어지고 있다. 재활용 가능한 자원을 종량제 봉투에 버리는 일만 막아도 연간 약 3천억 원 상당의 종량제 봉투 구매 비용을 절약할 수 있을 정도이다.

따라서 “분리수거 척척박사” 프로젝트를 통해 올바른 분리수거 배출 방법을 알려줄 것이다.

주요 기능으로는 크게 두 가지가 있다.

1. 사용자가 휴대폰을 통해 이미지를 모델에 전달하면 모델은 분류 결과를 알려준다.
2. 사용자는 분리수거 카테고리를 통해 분리수거 하기 까다로운 물체의 분리수거 방식을 알 수 있다.

카메라는 사용자의 휴대폰을 사용하며 분류 모델은 Tensorflow의 EfficientNet-Lite V3 모델을 활용하였으며 정확도를 가장 중요시했다.

“분리수거 척척박사”는 누구에게나 쉽게 분리수거 방식을 알려줌으로써 사람들의 올바른 재활용품 분류를 통해 환경보호에 일조하고자 한다.

(기사 출처: 장현지, ““열심히 한 분리수거, 재활용 안돼..” 헛수고?,” 「동아닷컴」, 2020년 12월 31일.)

2.2 핵심 시나리오

두 가지 방법으로 “분리수거 척척박사” 앱을 사용할 수 있다.

[방법 1] - 10가지의 재활용 쓰레기들을 직접 준비

1. 앱 실행 후 "사진 선택" 버튼을 클릭한다.
2. 재활용품 이미지를 업로드할 업로드 방식을 선택한다.
(새로운 사진 촬영 or 앨범에서 불러오기)
3. 쓰레기 사진을 촬영하거나 앨범에서 쓰레기 사진을 업로드한다.
4. 업로드한 이미지의 분류 결과를 확인한다.
5. "상세한" "상세한 분리수거 Tip" 버튼을 클릭해 해당 항목에 대한 분리수거 방법을 확인한다.

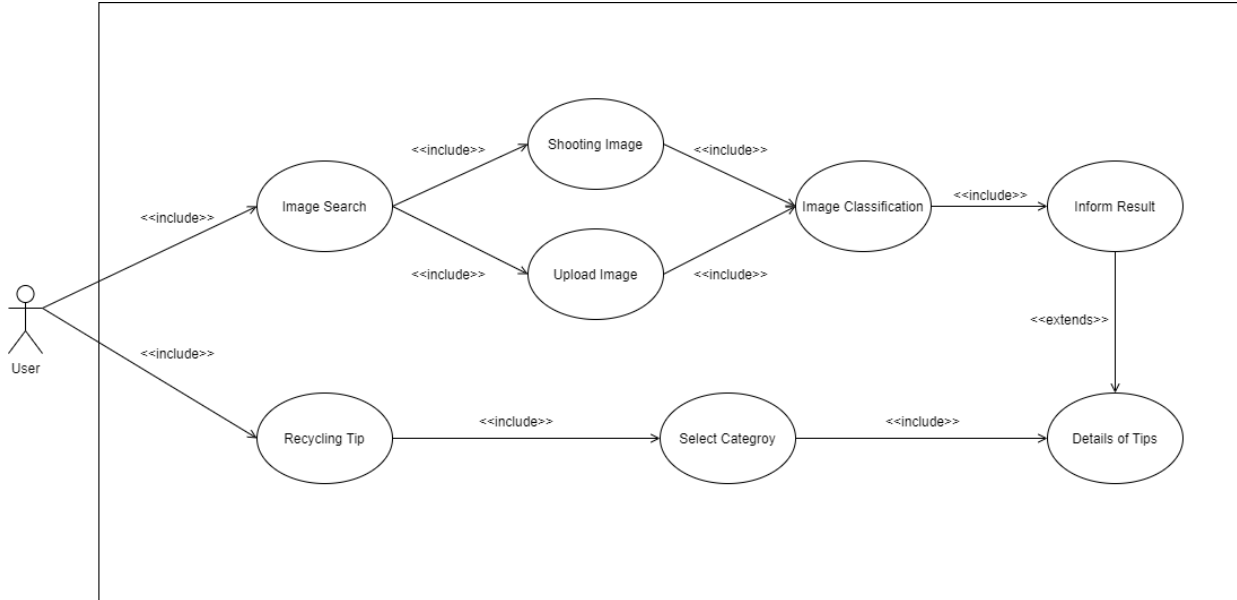
[방법 2]

1. 앱 실행 후 “분리수거 Tip” 버튼을 클릭한다.
2. 정보를 원하는 분리수거 카테고리를 선택한다.
3. 해당 항목에 대한 분리수거 방법을 확인한다.

3. 요구사항 정의

3.1 기능적 요구사항

3.1.1 Top Level Use Case Diagram



Use Case	1.1 Image Search
Actor	사용자
Description	사용자는 이미지 검색을 위해 이미지 검색 버튼을 누른다.

Use Case	1.2 Shooting Image
Actor	사용자
Description	카메라 앱으로 직접 접근하여 물체의 사진을 찍는다.

Use Case	1.3 Upload Image
Actor	사용자
Description	앨범 앱으로 직접 접근하여 물체의 사진을 불러온다.

Use Case	1.4 Image Classification
Actor	사용자
Description	사용자가 업로드한 이미지를 이미지 분류를 한다.

Use Case	1.5 Inform Result
Actor	사용자
Description	이미지 분류 결과를 얻는다.

Use Case	2.1 Recycling Tip
Actor	사용자
Description	분리수거 요령 정보를 얻기 위해 분리수거 팁 버튼을 누른다.

Use Case	2.2 Select Category
Actor	사용자
Description	여러 카테고리들 중 사용자가 정보를 원하는 품목을 선택한다.

Use Case	2.3 Details of Tips
Actor	사용자
Description	사용자가 선택한 품목의 재활용 방법 상세 요령을 알려준다.

3.1.2 각 기능별 동작 시나리오(Use Case Document)

Use case name	1.1 Image Search
Participating actors	User
Flow of events	
1. 사용자가 앱을 킨다.	
2. 이미지 검색을 위해 이미지 검색 버튼을 누른다.	
	3. 카메라로 촬영할 것인지, 앨범에서 불러올 것인지 사용자에게 요청한다.

Use case name	1.2 Shooting Image
Participating actors	User
Flow of events	
1. 카메라로 촬영 버튼을 선택한다.	
2. 가이드라인에 맞춰 물체를 촬영한다.	

Use case name	1.3 Upload Image
Participating actors	User
Flow of events	
1. 앨범에서 업로드 버튼을 선택한다.	
2. 가이드라인에 맞춰 미리 촬영한 물체의 사진을 업로드한다.	

Use case name	1.4 Image Classification
Participating actors	User
Flow of events	
1. 사진 업로드를 완료한다.	
	2. 업로드 된 이미지를 모델을 통해 분류한다.

Use case name	1.5 Inform Result
Participating actors	User
Flow of events	
	1. 이미지 분류 결과를 사용자에게 알려준다.
2. 사용자가 추가로 재활용 팁을 원하는 경우 2.3 Details of Tips로 이동한다.	

Use case name	2.1 Recycling Tip
Participating actors	User
Flow of events	
1. 사용자가 앱을 킨다.	
2. 재활용 방법 팁을 위해 재활용 팁 버튼을 누른다.	

Use case name	2.2 Select Category
Participating actors	User
Flow of events	
1. 미리 설정된 여러 재활용 카테고리들 중 정보를 원하는 카테고리를 선택한다. (ex. 플라스틱류, 종이류 등)	

Use case name	2.3 Details of Tips
Participating actors	User
Flow of events	
	1. 사용자가 선택한 카테고리의 재활용 팁들을 제공한다.

3.2 비기능적 요구사항

3.2.1 사용 편리성

- 목표 사용자: 분리수거 방법을 자세히 알지 못하는 사람
- 사용자가 사진 촬영 또는 업로드를 통해 간단하고 편리하게 결과를 향해 갈 수 있도록 해야 한다.
- 사용자가 쉽게 어플리케이션을 사용할 수 있도록 사용자의 촬영 화면에 어떤 방식으로 촬영해야 하는지 가이드 라인을 띄워준다.
- 사용자가 분리수거 규칙 정보에 쉽게 접근할 수 있도록 관련 정보를 카테고리화 하여 제공한다.

3.2.2 신뢰성

- 카메라 권한 및 저장소 권한이 필요한 어플리케이션이므로 비허용시 해당 어플리케이션 서비스를 제공하지 않는다.

3.2.3 성능

- 사용자가 올바르게 않은 방법으로 재활용하지 않도록 모델의 인식율이 높아야 한다.
- 사진 촬영 또는 업로드 후 결과를 알려주는 과정에서 최소한의 시간을 목표로 한다.
- 어플리케이션의 특성 상 높은 인식율과 빠른 속도 중 높은 인식율이 더욱 중요하기 때문에 Mask R-CNN, YOLO, MoblieNet, EffieientNet 등 가장 적합한 알고리즘을 이용해 모델을 설계한다.

3.2.4 이식성

- 개발 시 설정한 최소 안드로이드 버전에 맞춰 그 이상에서는 어플리케이션이 문제 없이 동작해야 한다.
- 안드로이드의 다양한 플랫폼에 맞춰 사용할 수 있도록 UI 구성 요소의 위치를 상대적으로 지정한다.
- iOS 환경에서는 사용할 수 없다.

3.2.5 유지관리

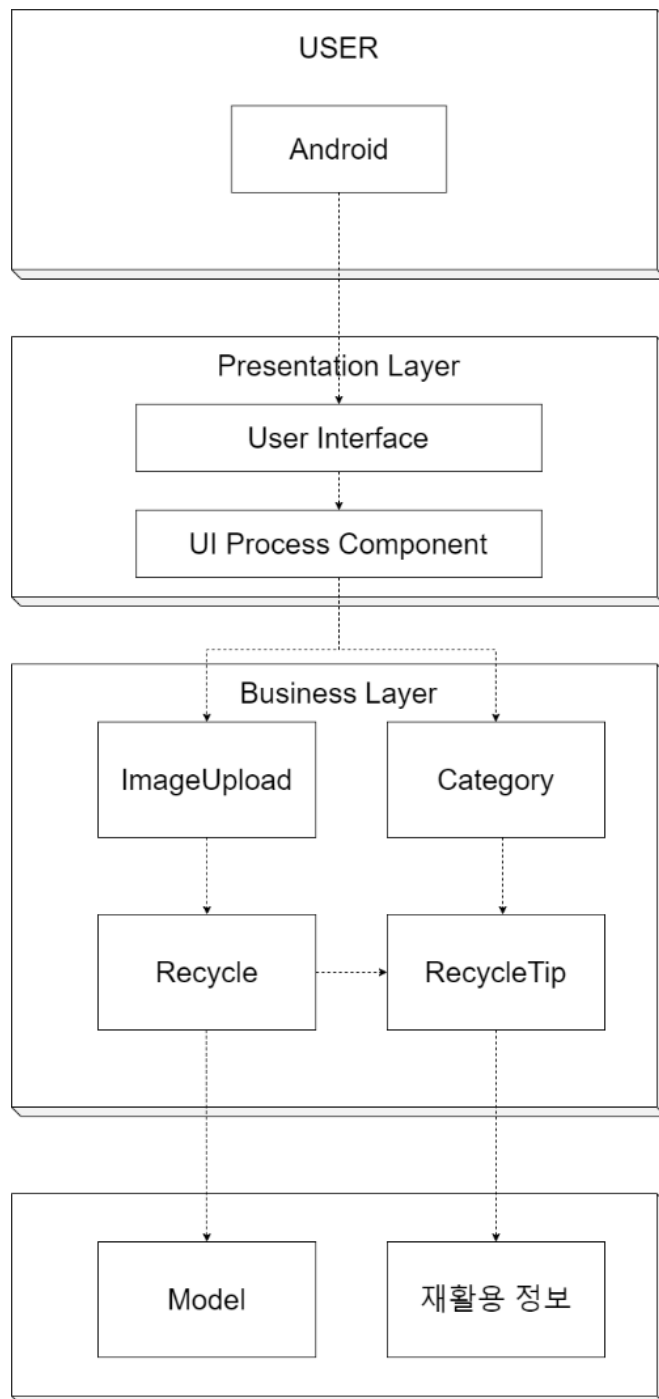
- 공공기관에서 제공하는 분리수거 관련 정보의 변동이 있을 시 어플리케이션에 변경된 정보를 제공한다.
- 도심지에서 주로 배출되는 종류의 재활용 쓰레기들을 목표로 모듈을 만들지만 차후에 인식 대상 범위를 놓여촌 등 교외에서 주로 발생하는 재활용 쓰레기로 확장한다.

3.2.6 인터페이스

- 학습된 모델을 어플리케이션과 연동시키는 과정에서 문제가 없도록 한다.

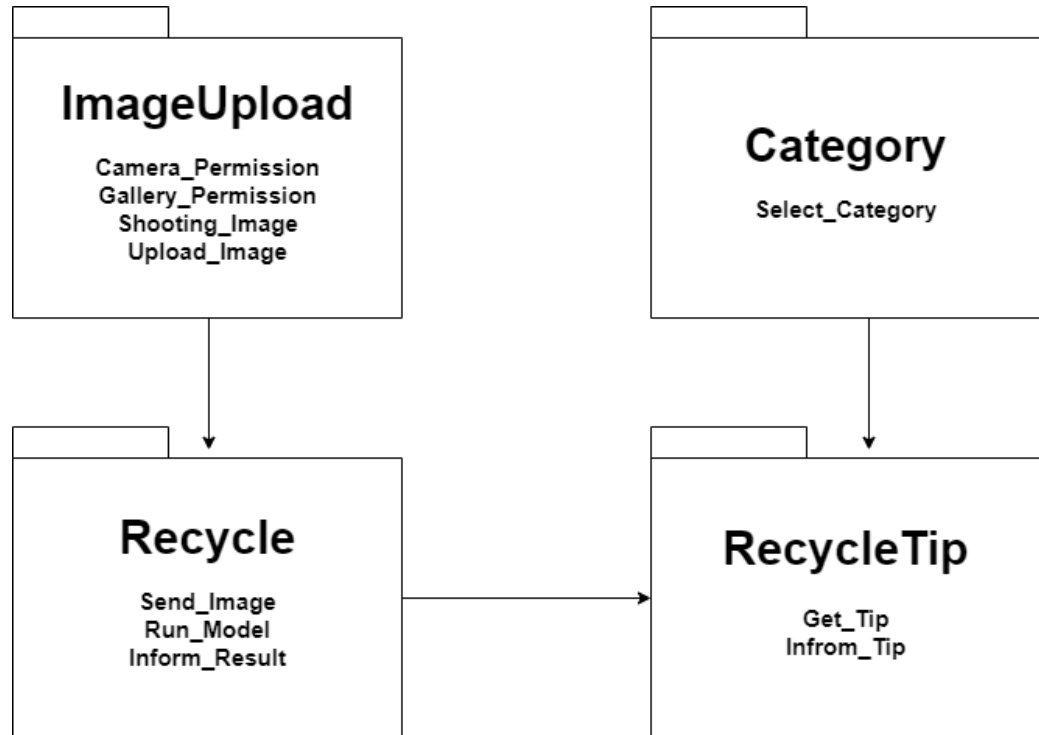
4. 시스템 설계

4.1 시스템 구조도



4.2 컴포넌트 다이어그램

4.2.1 Component 전체 구조도



4.2.2 Component 목록

Component ID	Component Name	개요	관련 use-case Diagram
C1	ImageUpload	사용자가 이미지 검색을 위해 이미지를 업로드한다.	Image Search Shooting Image Upload Image
C2	Recycle	학습된 모델을 통해 전달받은 이미지 분류	Image Classification Inform Result
C3	RecycleTip	원하는 품목에 대한 분리수거 상세정보 제공	Details of Tips
C4	Category	원하는 분리수거 카테고리 선택	Recycling Tip Select Category

4.2.3 Component 명세

4.2.3.1 ImageUpload

Component ID	C1	Component Name	ImageUpload
Component 개요	사용자가 이미지 검색을 위해 이미지를 업로드한다.		
내부 클래스			
ID	Class Name	비고	
C1_01	Camera_Permission	카메라의 권한 요청	
C1_02	Gallery_Permission	갤러리 권한 요청	
C1_03	Shooting_Image	사진 촬영	
C1_04	Upload_Image	갤러리에서 이미지 선택	

4.2.3.2 Recycle

Component ID	C2	Component Name	Recycle
Component 개요	학습된 모델을 통해 전달받은 이미지 분류		
내부 클래스			
ID	Class Name	비고	
C2_01	Send_Image	모델에 이미지 전송	
C2_02	Run_Model	학습된 모델을 바탕으로 전달받은 이미지 분류	
C2_03	Inform_Result	분류된 결과 사용자에게 전송	

4.2.3.3 RecycleTip

Component ID	C3	Component Name	RecycleTip
Component 개요	원하는 품목에 대한 분리수거 상세정보 제공		
내부 클래스			
ID	Class Name	비고	
C3_01	Get_Tip	분류된 결과를 토대로 분리수거 상세 정보를 요청	
C3_02	Inform_Tip	분리수거 상세 정보를 사용자에게 제공	

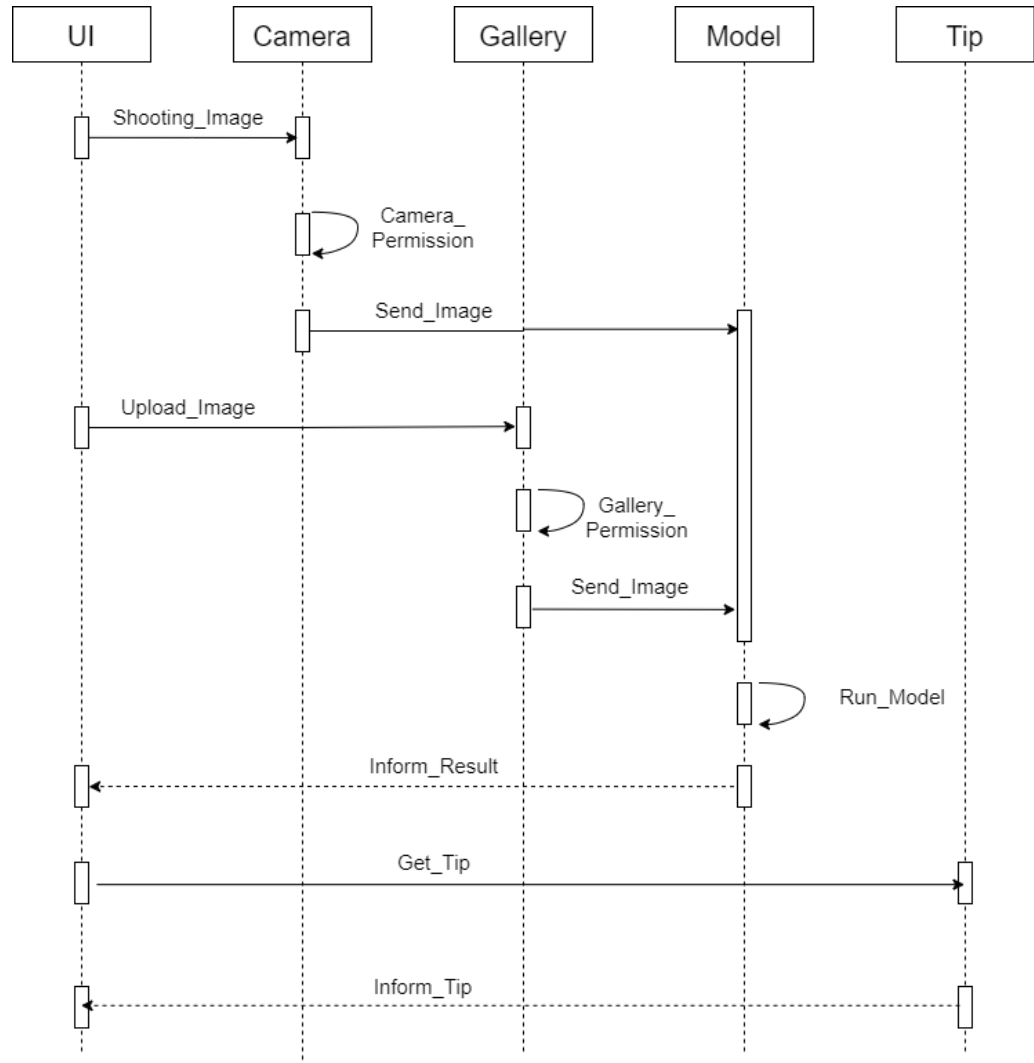
4.2.3.4 Category

Component ID	C4	Component Name	Category
Component 개요	원하는 분리수거 카테고리 선택		
내부 클래스			
ID	Class Name	비고	
C4_01	Select_Category	분리수거 카테고리 선택	

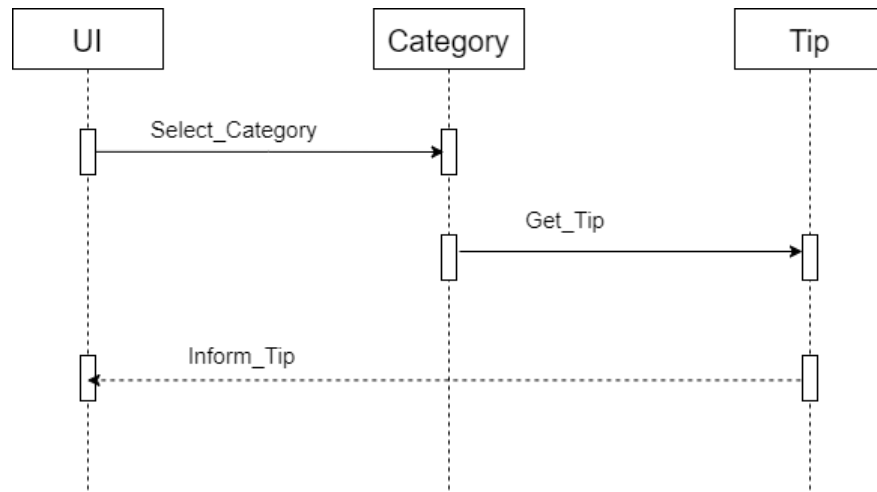
4.3 시퀀스 다이어그램

4.3.1 Component 내부 시퀀스 다이어그램

4.3.1.1 ImageUpload, Recycle, RecycleTip

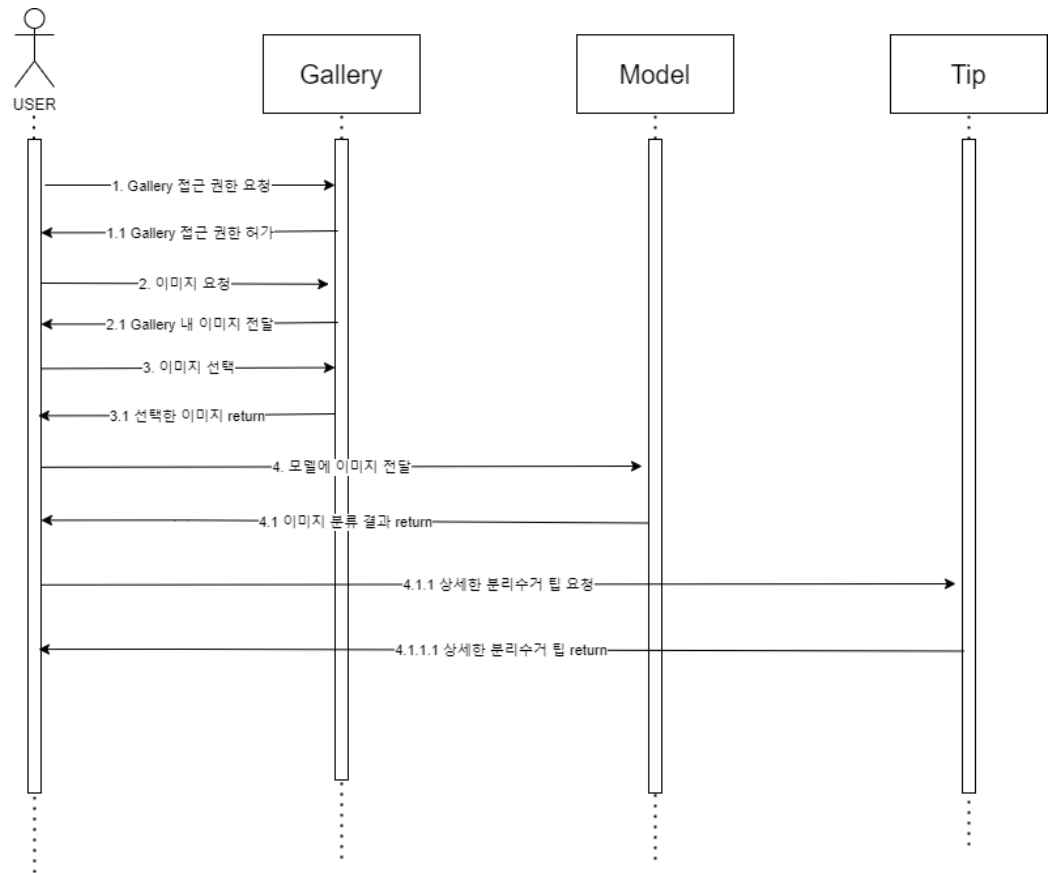


4.3.1.2 Category, RecycleTip

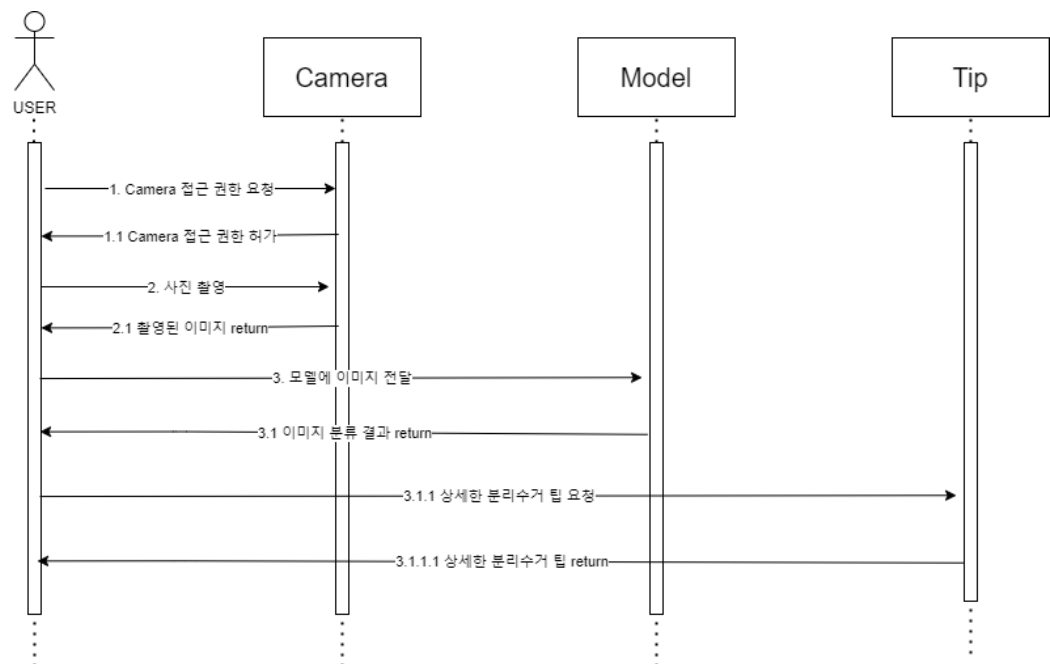


4.3.2 Use-case 별 시퀀스 다이어그램

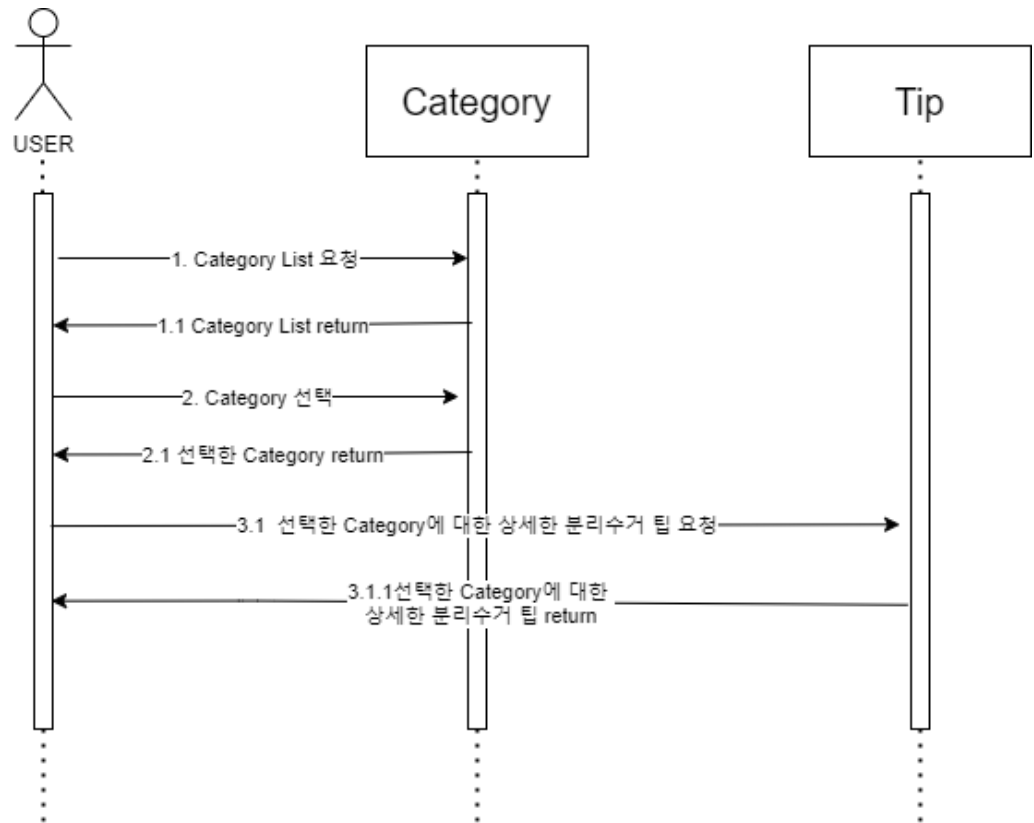
4.3.2.1 이미지 업로드 후 이미지 분류, 재활용 상세 팁



4.3.2.2 카메라 촬영 후 이미지 분류, 재활용 상세 팁

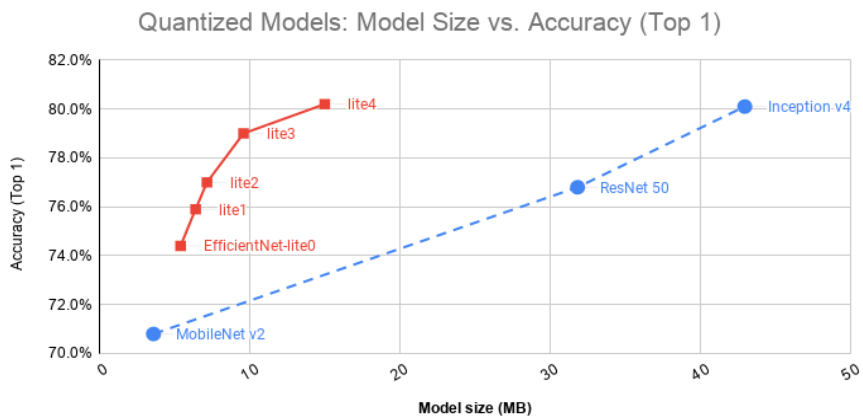
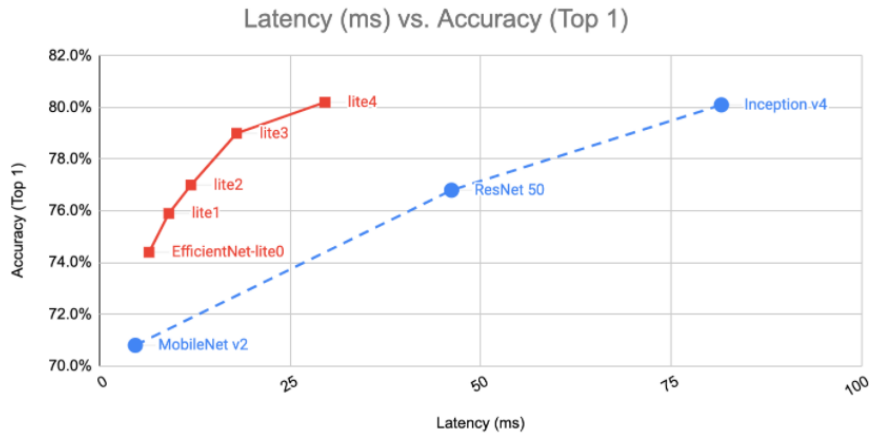


4.3.2.3 카테고리 선택 후 재활용 상세 팁



4.4 핵심 알고리즘

EfficientNet-Lite는 계산량과 매개변수 수를 현격하게 줄이면서도 최고의 정확도를 달성하는 이미지 분류 모델 모음이다. EfficientNet-Lite 모델은 양자화를 포함한 TensorFlow Lite에 맞게 최적화되어 있어 정확도 손실을 무시할 수 있을 정도의 수준에서 더 빠르게 추론할 수 있고 CPU, GPU 또는 Edge TPU에서 실행 가능하다.



Model	params	MAdds	FP32 accuracy	FP32 CPU latency	FP32 GPU latency	FP16 GPU latency	INT8 accuracy	INT8 CPU latency	INT8 TPU latency
efficientnet-lite0 ckpt	4.7M	407M	75.1%	12ms	9.0ms	6.0ms	74.4%	6.5ms	3.8ms
efficientnet-lite1 ckpt	5.4M	631M	76.7%	18ms	12ms	8.0ms	75.9%	9.1ms	5.4ms
efficientnet-lite2 ckpt	6.1M	899M	77.6%	26ms	16ms	10ms	77.0%	12ms	7.9ms
efficientnet-lite3 ckpt	8.2M	1.44B	79.8%	41ms	23ms	14ms	79.0%	18ms	9.7ms
efficientnet-lite4 ckpt	13.0M	2.64B	81.5%	76ms	36ms	21ms	80.2%	30ms	-

MobileNetV2, ResNet 50 등과 비교시 Efficientnet-lite3 모델이 용량 대비 정확도에서 훨씬 더 나은 결과를 가져와 Efficientnet-lite3 모델을 사용했다.

또한, 모든 EfficientNet-Lite모델들을 사용해본 결과 표와 달리 분리수거 데이터셋에서는 EfficientNet-Lite3가 다른 모델에 비해 학습속도 및 정확도, 용량이 가장 최적이라고 판단해 EfficientNet-Lite V3를 사용했다.

$$\begin{aligned}
&\text{depth: } d = \alpha^\phi \\
&\text{width: } w = \beta^\phi \\
&\text{resolution: } r = \gamma^\phi \\
&\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\
&\alpha \geq 1, \beta \geq 1, \gamma \geq 1
\end{aligned}$$

CNN의 성능을 높일 수 있는 요소로는 Depth (d), Width (w), Resolution (r) 3가지가 있다.

EfficientNet 모델은 기본적으로 width scaling과 depth scaling, resolution scaling 3가지의 방식을 통해 모델을 학습시키며 EfficientNet의 알파, 베타, 감마 값은 간단한 grid search를 통해 구하는 방식을 제안하고 있으며, 처음 단계에서는 파이를 1로 고정한 뒤, 타겟 데이터셋에서 좋은 성능을 보이는 알파, 베타, 감마 값을 찾아낸다.

Table 2. EfficientNet Performance Results on ImageNet (Russakovsky et al., 2015). All EfficientNet models are scaled from our baseline EfficientNet-B0 using different compound coefficient ϕ in Equation 3. ConvNets with similar top-1/top-5 accuracy are grouped together for efficiency comparison. Our scaled EfficientNet models consistently reduce parameters and FLOPS by an order of magnitude (up to 8.4x parameter reduction and up to 16x FLOPS reduction) than existing ConvNets.

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPS	Ratio-to-EfficientNet
EfficientNet-B0	76.3%	93.2%	5.3M	1x	0.39B	1x
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	78.8%	94.4%	7.8M	1x	0.70B	1x
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	79.8%	94.9%	9.2M	1x	1.0B	1x
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	81.1%	95.5%	12M	1x	1.8B	1x
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	82.6%	96.3%	19M	1x	4.2B	1x
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	83.3%	96.7%	30M	1x	9.9B	1x
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	84.0%	96.9%	43M	1x	19B	1x
EfficientNet-B7	84.4%	97.1%	66M	1x	37B	1x
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

We omit ensemble and multi-crop models (Hu et al., 2018), or models pretrained on 3.5B Instagram images (Mahajan et al., 2018).

위 표는 ImageNet에 대한 실험 결과이다. 기존 ConvNet들에 비해 EfficientNet 모델은 비슷한 정확도를 보이면서 parameter수와 FLOPS 수를 굉장히 많이 절약할 수 있는 것을 알 수 있다. 또한, 기존에 ImageNet 데이터셋에서 가장 높은 정확도를 달성했던 GPipe 보다 EfficientNet 모델은 더 높은 정확도를 달성하는 것을 확인할 수 있다.

4.5 타 모듈(오픈소스, 오픈, Lib등)연계

4.5.1 활용도구

학습 시스템 : Google Colab / TensorFlow Lite 2.5.0 version / Android Studio

인식 시스템 : Android phone (minSdkVersion 28 / targetSdkVersion 30)

4.5.2 오픈소스 : Tensorflow Lite

TensorFlow Lite는 개발자가 모바일, 내장형 기기, IoT 기기에서 모델을 실행할 수 있도록 지원하여 기기 내 머신러닝을 사용할 수 있도록 하는 도구 모음이다.

주요 특징

- 기기 내 머신러닝에 최적화됨, 5가지 핵심 제약사항 해결: 지연 시간(서버까지의 왕복 없음), 개인 정보 보호(기기에 개인 정보를 남기지 않음), 연결성(인터넷 연결이 필요하지 않음), 크기(모델 및 바이너리 크기 축소), 전력 소비(효율적인 추론 및 네트워크 연결 불필요)
- 여러 플랫폼 지원: Android 및 iOS 기기, 내장형 Linux 및 마이크로 컨트롤러 등
- 다양한 언어 지원: 자바, Swift, Objective-C, C++ , Python 등
- 고성능: 하드웨어 가속 및 모델 최적화 사용
- 포괄적인 예: 다양한 플랫폼에서의 일반적인 머신러닝 작업(예: 이미지 분류, 객체 감지, 자세 추정, 질문 답변, 텍스트 분류 등)

5. 시스템 구현

5.1 핵심코드 상세 설명

5.1.1 Shooting_Image : 카메라를 통해 사진을 촬영하는 부분이다.

```
private fun Shooting_Image() { //카메라 촬영
    Intent(MediaStore.ACTION_IMAGE_CAPTURE).also { takePictureIntent ->
        takePictureIntent.resolveActivity(packageManager)?.also { it: ComponentName
            val photoFile: File? = try {
                createImageFile()
            } catch (e: IOException) {
                Log.e(TAG, e.message.toString())
                null
            }
            photoFile?.also { it: File
                val photoURI: Uri = FileProvider.getUriForFile(
                    context: this,
                    authority: "org.tensorflow.codelabs.objectdetection.fileprovider",
                    it
                )
                takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI)
                startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE)
            }
        }
    }
}
```

```
@Throws(IOException::class)
private fun createImageFile(): File { //이미지 파일 생성
    val timeStamp: String = SimpleDateFormat(pattern: "yyyyMMdd_HH:mm:ss").format(Date())
    val storageDir: File? = getExternalFilesDir(Environment.DIRECTORY_PICTURES)
    return File.createTempFile(
        prefix: "JPEG_${timeStamp}_",
        suffix: ".jpg",
        storageDir
    ).apply { this: File
        // 사진 저장 this: File
        currentPhotoPath = absolutePath
    }
}
```

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

    //결과 출력
    binding.name.visibility = View.VISIBLE

    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == Activity.RESULT_OK) {
        Send_Image(getCapturedImage())
    } else if (requestCode == OPEN_GALLERY && resultCode == Activity.RESULT_OK){
        var currentImageUrl : Uri? = data?.data
        val bitmap = MediaStore.Images.Media.getBitmap(contentResolver,currentImageUrl)
        Send_Image(bitmap)
    }
}
```

- 카메라를 통해 사진을 촬영하기 위해 createImageFile을 통해 이미지 파일을 생성한 뒤 생성된 파일을 가져온다.
- Shooting_Image 함수가 호출되는 경우 requestCode REQUEST_IMAGE_CAPTURE가 전달되며 onActivityResult에서 if구문이 실행된다.

5.1.2 Upload_Image : 갤러리를 통해 디바이스에 존재하는 사진을 가져오는 부분이다.

```
private fun Upload_Image(){ //갤러리에서 사진 가져오기
    val intent:Intent = Intent(Intent.ACTION_GET_CONTENT)
    intent.setType("image/*")
    startActivityForResult(intent,OPEN_GALLERY)
}
```

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

    //결과 출력
    binding.name.visibility = View.VISIBLE

    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == Activity.RESULT_OK) {
        Send_Image(getCapturedImage())
    } else if(requestCode == OPEN_GALLERY && resultCode == Activity.RESULT_OK){
        var currentImageUrl : Uri? = data?.data
        val bitmap = MediaStore.Images.Media.getBitmap(contentResolver,currentImageUrl)
        Send_Image(bitmap)
    }
}
```

- Upload_Image 함수가 호출되는 경우 requestCode로 OPEN_GALLERY가 전달되며 onActivityResult에서 else if구문이 실행된다.
- 갤러리에서 선택된 사진을 Uri를 통해 가져온다.

5.1.3 Send_Image : 전달 받은 사진을 모델에 전달하는 부분이다.

```
private fun Send_Image(bitmap: Bitmap) {  
    imageView.setImageBitmap(bitmap) //사진 화면에 띄우기  
    tvPlaceholder.visibility = View.INVISIBLE //분리수거 척척박사 문장 제거  
  
    lifecycleScope.launch(Dispatchers.Default) { Run_Model(bitmap) } //모델에 사진 전송  
    //계산 많이 함 ->백그라운드 스레드에서 OD 수행(app ui blocking예방)  
    //main thread에서 사용 x  
}
```

- imageView는 activity_main.xml의 ImageView를 뜻하며 전달받은 image인 bitmap으로 사진을 변경한다.
- tvPlaceholder는 activity_main.xml의 TextView를 뜻하며 ‘분리수거 척척박사’ 문장을 화면에 띄운다. 결과 사진을 보여줄 때는 해당 문장을 제거한다.
- Object Detection 수행 시, 계산이 많이 요구되기 때문에 백그라운드 Thread에서 수행하며 이때 app UI변경은 불가능하다.

5.1.4 Run_Model : Object Detection을 실행하는 부분이다.

```
//새로운 이미지를 선택할때마다 호출
private fun Run_Model(bitmap: Bitmap) {
    // Step 1: 텐서 이미지 만들기
    //텐서 이미지 = 텐서라이트 모델의 입력 유형 -> 기본적으로 입력 이미지를 가져온다.(비트맵 -> 텐서 이미지)
    val image = TensorImage.fromBitmap(bitmap)

    // Step 2: ObjectDetector 인스턴스 (객체를 만들기 위한 구성 포함)
    val options = ObjectDetector.ObjectDetectorOptions.builder()
        .setMaxResults(1)
        .setScoreThreshold(0.7f)
        .build()

    //물체 감지기 만들기
    val detector = ObjectDetector.createFromFileAndOptions(
        context: this,
        modelPath: "model011.tflite",
        options
    )

    // Step 3: 모델에 사진 전송
    val results = detector.detect(image)
    Log.i( tag: "results",results.toString())
}
```

```
// Step 4: 탐지 결과 출력
val resultToDisplay = results.map { it: Detection!
    val category = it.categories.first() //정확도가 제일 높은 카테고리 선택

    if(category.label == "FeedingBottle") {
        label = "젖병"
    } else if(category.label == "battery"){
        label = "건전지"
    } else if(category.label == "can"){
        label = "캔"
    } else if(category.label == "cupramen"){
        label = "컵라면"
    }else if(category.label == "icepack"){
        label = "아이스팩"
    }else if(category.label == "milk"){
        label = "우유팩"
    }else if(category.label == "papercup"){
        label = "종이컵"
    }else if(category.label == "petbottle"){
        label = "페트병"
    }else if(category.label == "rubberGlove"){
        label = "고무장갑"
    }else if(category.label == "spoon"){
        label = "일회용 숟가락"
    }else{
        label = ""
    }

    val text1 = "${category : "+ label+" ",} ${"정확도 : "+category.score.times( other: 100).toInt())}% " //%결과로 출력
    val text2 = "${label}, ${category.score.times( other: 100).toInt())}% " //%결과로 출력
}
```



```

val text1 = "${카테고리 : "+ label+"..."}, ${"정확도 : "+category.score.times( other: 100).toInt()}" //결과로 출력
val text2 = "${label}, ${category.score.times( other: 100).toInt()}" //결과로 출력

//값 저장해두기
//App.prefs1!!.myIndex1 = category.label.toString()
App.prefs1!!.myIndex1 = label

// 객체에 결과 값 전달
DetectionResult(it.boundingBox, text1, text2)
}

//탐지 결과 그리기
val imgWithResult = Inform_Result(bitmap, resultToDisplay)
runOnUiThread {
    binding.imageView.setImageBitmap(imgWithResult)
}

```

- TensorImage.fromBitmap함수를 통해 tflite 모델의 입력 유형으로 이미지를 변경시켜준다.
- ObjectDetecotr.ObjectDetectorOptions.builder()에서는 탐지할 수 있는 물체의 수는 1개로 정하였으며 threshold 값을 0.7으로 지정함으로써 70%이상의 정확도가 나온 물체만 나오도록 하였다.
- 여러가지 모델을 사용해본 결과 EfficientNet-Lite3 모델을 사용한 결과 가장 높은 정확도와 빠른 속도를 보여줘 해당 모델을 선택했다.
- Detector.detect를 통해 tflite 모델의 입력 유형으로 변환한 이미지를 모델에 전송하였다.
- Categories.first()를 통해 정확도가 제일 높은 카테고리를 category 변수에 저장하였다. 이후 category의 변수에서 label값을 추출한 뒤 조건문을 통해 label변수에 저장하였다.
- DetectionResult()에서는 객체에 카테고리 and 정확도를 전달하였다.
- 이후 자세한 분리수거 액티비티에서 분류 결과를 가져오기 위해 Sharedpreference에 분류 결과를 저장하였다.

5.1.5 Inform_Result : 탐지된 결과를 바운딩 박스와 함께 결과를 알려준다.

```
//입력 이미지 위에 띄울 탐지 결과를 그리기
private fun Inform_Result(
    bitmap: Bitmap,
    detectionResults: List<DetectionResult>
): Bitmap {
    val outputBitmap = bitmap.copy(Bitmap.Config.ARGB_8888, isMutable: true)
    val canvas = Canvas(outputBitmap)
    val pen = Paint()
    pen.textAlign = Paint.Align.LEFT

    detectionResults.forEach { it: DetectionResult
        // 바운딩 박스 그리기 it: DetectionResult
        pen.color = Color.RED
        pen.strokeWidth = 8F
        pen.style = Paint.Style.STROKE
        val box = it.boundingBox
        canvas.drawRect(box, pen)

        //MainActivity에 결과 출력
        Thread {
            runOnUiThread {
                binding.name.setText(it.text)
            }
        }.start()

        val tagSize = Rect(left: 0, top: 0, right: 0, bottom: 0)
    }
```

- 전달받은 이미지 위에 탐지 결과와 바운딩 박스를 그려주는 함수이다.
- ObjectDetection작업을 수행하는 동안 Thread 작업으로 수행되기 때문에 탐지 결과를 메인 화면에 나타내는 부분은 따로 Thread를 수행하여 값을 변경시켜주었다.


5.2 시스템 구현 시 문제점과 해결방안

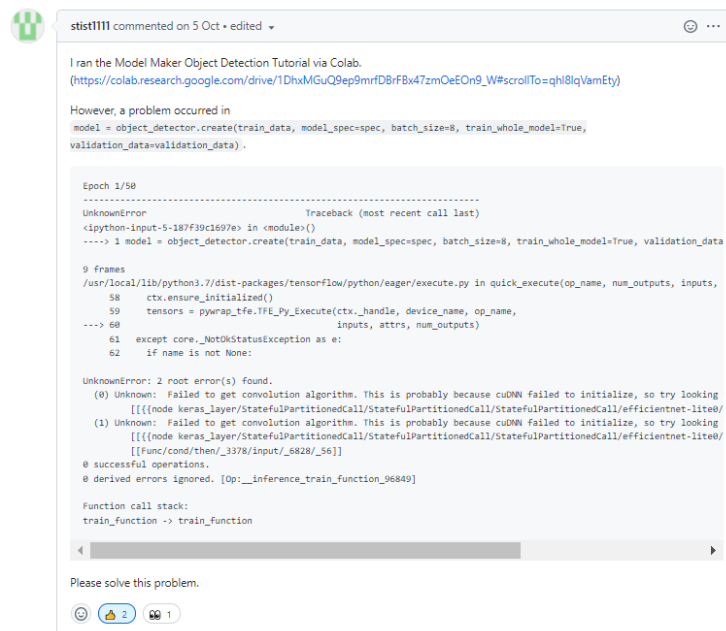
5.2.1 Model 학습 중 발생한 GPU 에러

모델 학습을 진행시키면서 GPU를 사용했을 때 알 수 없는 에러가 발생하였고, 2주 간 해당 문제를 여러 방면으로 해결하려 노력하였으나 해결하지 못했다.

마지막 방법으로 Tensorflow와 Colab 공식 Github에 이슈를 남겼고, 결국 해당 문제를 해결할 수 있었다.

Model Maker Object Detection Tutorial Bug #52255

 stist1111 opened this issue on 5 Oct · 16 comments



 Closed tykwon97 opened this issue on 7 Oct - 1 comment

 Closed tykwon97 opened this issue on 7 Oct - 1 comment

Assignees

Labels

Successfully merging a pull request may close this issue

None yet

Notifications [Customize](#)

 Unsubscribe

You're receiving notifications because you authored the thread.

분류 결과를 Android Studio에서 화면 전환 없이 분류된 결과 값만 text로 전달하고 싶었으나 intent의 경우 화면 전환을 하지 않는 경우 값이 전달되지 않았다.

<Sharedpreference 기본 틀>

```
class App: Application() { //prefs를 저장
    companion object {
        lateinit var prefs1: MyIndexSaveClass
    }

    override fun onCreate() {
        prefs1 = MyIndexSaveClass(applicationContext) //prefs 객체를 생성 및 보관(lateinit)
        super.onCreate()
    }
}
```

<SharedPreferences를 활용해 값을 저장하는 방법>

```
//값 저장해두기
//App.prefs1!!.myIndex1 = category.label.toString()
App.prefs1!!.myIndex1 = label
```

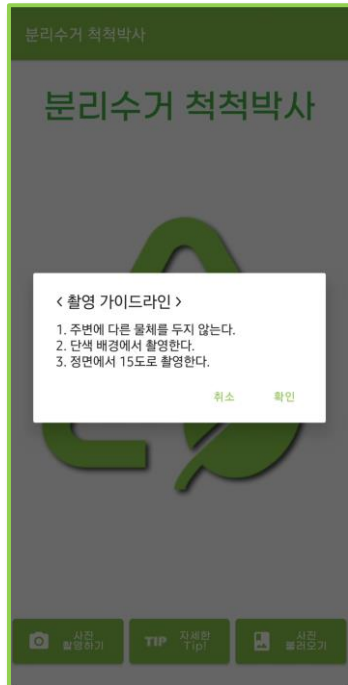
<디바이스의 저장공간에 저장된 text파일>

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="myIndex1">젓병</string>
</map>
```

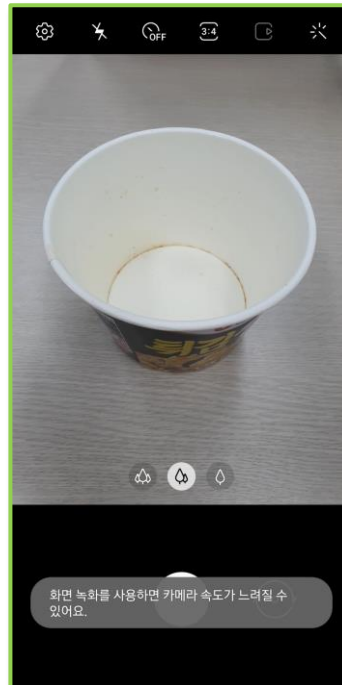
6. 시스템 실행 결과

6.1 주요 장면 스냅샷(1)

1) 가이드라인



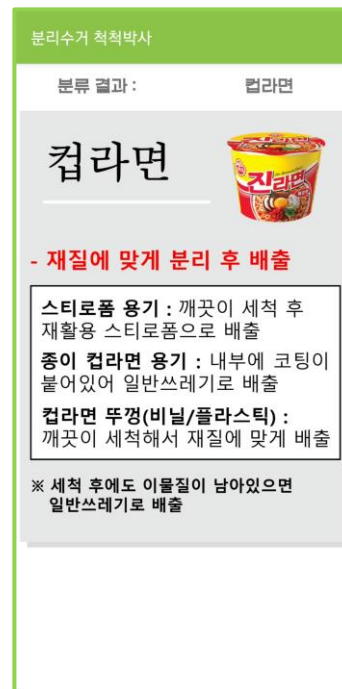
2) 사진 촬영



2) 결과 화면



4) 자세한 Tip



분리수거 척척박사 프로젝트의 핵심 기능으로 사용자는 분리수거 하고 싶은 물건을 카메라로 촬영하거나 앨범에서 불러온다.

사용자가 업로드한 사진의 물건이 어떤 분리수거 항목인지 정확도와 함께 알려준다.

또한 해당 항목의 분리수거 방법을 사용자가 원한다면 자세한 설명도 제공한다.

현재 페트병, 캔, 우유팩, 컵라면, 종이컵, 젓병, 건전지, 아이스팩, 일회용 손가락, 고무장갑 10개의 항목에 대해서 분류가 가능하다.

6.2 주요 장면 스냅샷(2)

1) Intro



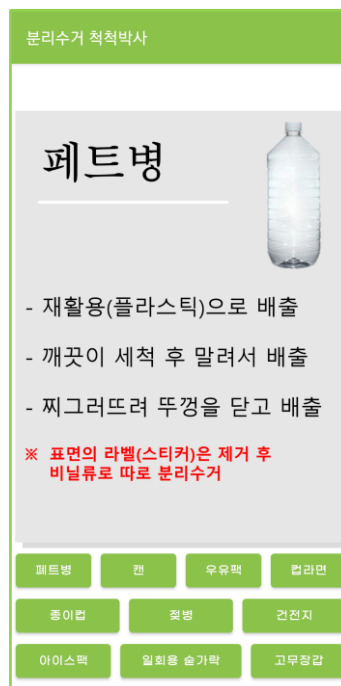
2) 분리수거 Tip



3) 자주 헛갈리는 항목



4) 자세한 분류 결과



모델에 의해 분류되는 10가지 카테고리 이외에도 부가 기능으로 가장 기본적인 분류 기준인 종이, 캔, 유리, 플라스틱, 비닐류, 스티로폼, 형광등, 가전제품, 의류 9가지 항목에 대해서 분리수거 방식을 알려준다.

또한 자주 헛갈리는 항목 버튼을 클릭하면 스냅샷(1)에서 보여줬던 자주 헛갈리는 항목 10가지에 대해서 이미지 업로드 없이 분리수거 방법을 알 수 있다.

7. 시스템 시험 결과

7.1 시험 항목과 시험 결과

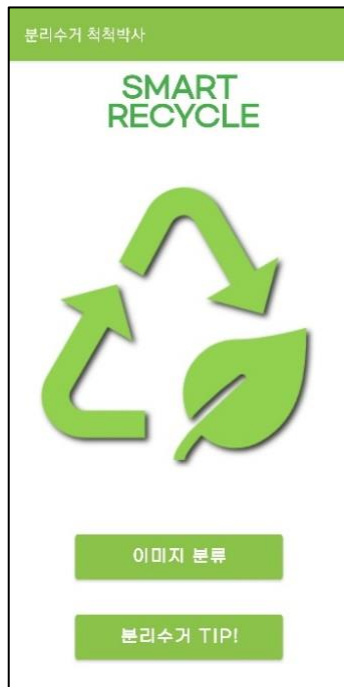
<기본 기능>

1. 처음 사용하는 사용자도 쉽게 사용할 수 있는가?

➔ 지인들에게 제작한 어플을 평가한 결과 평균적으로 4.3의 점수를 얻었다.

➔ 5점 Likert Scale 활용 (평가 항목: 사용성, 정확도, 사용자 편의성, 활용 방안)

2. 앱 실행 시 Intro 화면이 잘 실행되는가?



➔ 정상 작동

3. Intro 화면에서 ‘이미지 분류’ 버튼 클릭 시 사진 업로드 화면이 잘 실행되는가?



➔ 정상 작동

4. 사진 업로드 화면에서 ‘새로운 사진 촬영’ 버튼 클릭 시 카메라가 잘 실행되는가?



➔ 정상 작동

5. 사진 업로드 화면에서 ‘앨범에서 불러오기’ 버튼 클릭 시 갤러리가 잘 실행되는가?



➔ 정상 작동

6. 카메라를 통해 찍은 사진이 결과화면에 잘 나타나는가?



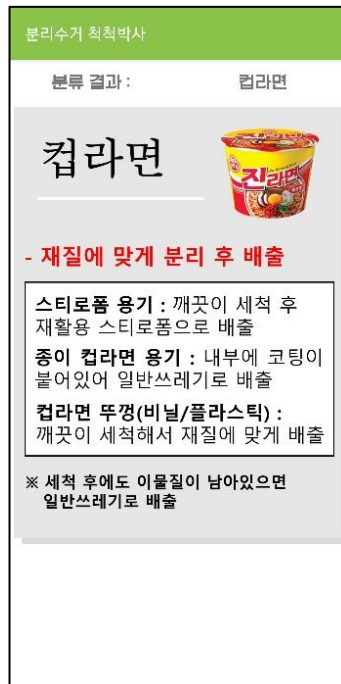
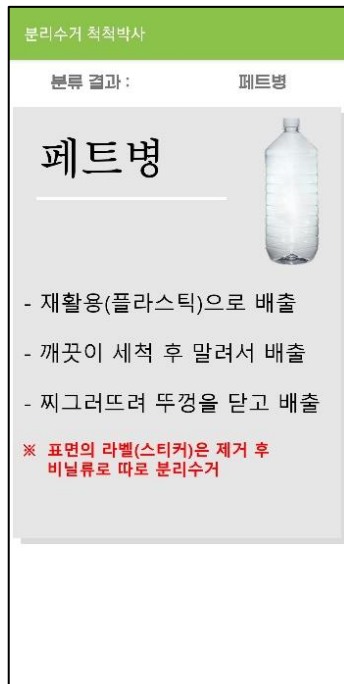
➔ 정상 작동

7. 갤러리에서 업로드한 사진이 결과화면에 잘 나타나는가?



➔ 정상 작동

8. 결과 화면에서 ‘상세한 분리수거 Tip’ 버튼 클릭 시 분류 결과와 일치하는 재활용 품목의 분리수거 Tip 화면이 잘 나타나는가?



➔ 정상 작동

9. 메인 화면에서 ‘분리수거 Tip’ 버튼 클릭 시 카테고리 화면이 잘 실행되는가?



➔ 정상 작동

10. 카테고리 화면에서 특정 카테고리 선택 시 해당 카테고리화 일치하는 재활용 품목의 분리수거 Tip 화면이 잘 나타나는가?



➔ 정상 작동

<모델 성능>

통과 조건은 모두 70%의 정확도로 정하였다.

1. 모든 카테고리별로 최소 요구 정확도를 만족하는가?

1)캔

2)건전지

3)젖병

분리수거 척척박사

카테고리 : 캔, 정확도 : 87%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

분리수거 척척박사

카테고리 : 건전지, 정확도 : 82%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

분리수거 척척박사

카테고리 : 젖병, 정확도 : 97%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

4)종이컵

5)우유팩

6)페트병

분리수거 척척박사

카테고리 : 종이컵, 정확도 : 95%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

분리수거 척척박사

카테고리 : 우유팩, 정확도 : 95%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

분리수거 척척박사

카테고리 : 페트병, 정확도 : 89%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

7) 고무장갑

분리수거 척척박사

카테고리 : 고무장갑, 정확도 : 95%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

8) 일회용 숟가락

분리수거 척척박사

카테고리 : 일회용 숟가락, 정확도 : 83%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

9) 컵라면

분리수거 척척박사

카테고리 : 컵라면, 정확도 : 81%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

10) 아이스팩

분리수거 척척박사

카테고리 : 아이스팩, 정확도 : 98%



사진 촬영하기

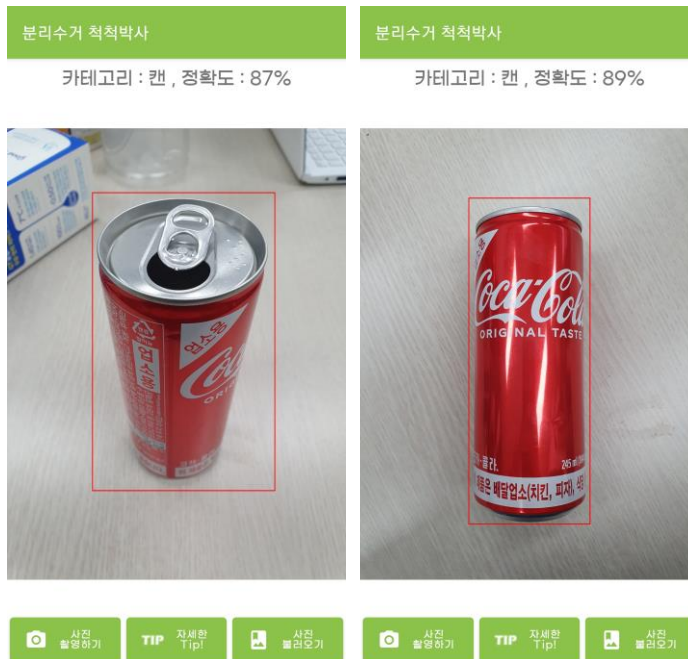
TIP 자세한 Tip!

사진 불러오기

➔ 모두 정상 작동

2. 같은 물체에 대해 다른 각도에서 동일한 결과를 출력하는가?

1)캔



➔ 정상 작동

2)건전지



➔ 정상 작동

3)젖병

분리수거 척척박사

카테고리 : 젖병, 정확도 : 97%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

분리수거 척척박사

카테고리 : 젖병, 정확도 : 81%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

➔ 정상 작동

4)종이컵

분리수거 척척박사

카테고리 : 종이컵, 정확도 : 95%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

분리수거 척척박사

카테고리 : 종이컵, 정확도 : 94%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

➔ 정상 작동

5)우유 팩

분리수거 척척박사

카테고리 : 우유팩, 정확도 : 95%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

분리수거 척척박사

카테고리 : 우유팩, 정확도 : 93%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

➔ 정상 작동

6)페트병

분리수거 척척박사

카테고리 : 페트병, 정확도 : 89%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

분리수거 척척박사

카테고리 : 페트병, 정확도 : 89%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

➔ 정상 작동

7) 고무장갑

분리수거 척척박사

카테고리 : 고무장갑, 정확도 : 94%



사진 촬영하기

TIP

자세한 Tip!



사진 불러오기

분리수거 척척박사

카테고리 : 고무장갑, 정확도 : 96%



사진 촬영하기

TIP

자세한 Tip!



사진 불러오기

➔ 정상 작동

8) 일회용 숟가락

분리수거 척척박사

카테고리 : 일회용 숟가락, 정확도 : 94%



사진 촬영하기

TIP

자세한 Tip!



사진 불러오기

분리수거 척척박사

카테고리 : 일회용 숟가락, 정확도 : 83%



사진 촬영하기

TIP

자세한 Tip!



사진 불러오기

➔ 정상 작동

9)컵라면

분리수거 척척박사

카테고리 : 컵라면, 정확도 : 81%



사진 촬영하기

TIP

자세한 Tip!



사진 불러오기

분리수거 척척박사

카테고리 : 컵라면, 정확도 : 98%



사진 촬영하기

TIP

자세한 Tip!



사진 불러오기

➔ 정상 작동

10)아이스팩

분리수거 척척박사

카테고리 : 아이스팩, 정확도 : 98%



사진 촬영하기

TIP

자세한 Tip!



사진 불러오기

분리수거 척척박사

카테고리 : 아이스팩, 정확도 : 95%



사진 촬영하기

TIP

자세한 Tip!



사진 불러오기

➔ 정상 작동

3. 한 물체에 대해서 다양한 형태를 동일하게 인식하는가?

1) 우유팩

분리수거 척척박사

카테고리 : 우유팩, 정확도 : 95%



사진
촬영하기

TIP

자세한
Tip!



사진
물러오기

분리수거 척척박사

카테고리 : 우유팩, 정확도 : 74%



사진
촬영하기

TIP

자세한
Tip!



사진
물러오기

➔ 정상 작동

2) 캔

분리수거 척척박사

카테고리 : 캔, 정확도 : 87%



사진
촬영하기

TIP

자세한
Tip!



사진
물러오기

분리수거 척척박사

카테고리 : 캔, 정확도 : 89%



사진
촬영하기

TIP

자세한
Tip!



사진
물러오기

➔ 정상 작동

3)페트병

분리수거 척척박사

카테고리 : 페트병 , 정확도 : 90%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

분리수거 척척박사

카테고리 : 페트병 , 정확도 : 89%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

➔ 정상 작동

4)아이스팩

분리수거 척척박사

카테고리 : 아이스팩 , 정확도 : 98%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

분리수거 척척박사

카테고리 : 아이스팩 , 정확도 : 95%



사진 촬영하기

TIP 자세한 Tip!

사진 불러오기

➔ 정상 작동

8. 팀원 및 역할 분담

권태운

- 안드로이드 UI 제작 및 앱 개발 : 사진 촬영 및 불러오기를 통해 분리수거 방법을 알아보는 것과 카테고리 별로 정리된 분리수거 방법을 알아보는 2가지 시나리오를 바탕으로 UI를 구성하고 학습 모델을 연동해 어플리케이션을 개발했다.
- 모델과 안드로이드 연동 : 빠른 연산 속도와 안드로이드 환경에서의 연동성을 고려하여 학습 모델을 어플리케이션 바이너리 번들에 묶어 어플리케이션을 개발했다.
- 데이터 수집 및 라벨링 : 건전지, 캔, 아이스팩, 우유팩, 고무장갑, 일회용 숟가락, 컵라면 용기, 젓병, 종이컵, 페트병 총 10가지 항목에 대해 이미지 수집 및 촬영을 통해 재활용 쓰레기 이미지 데이터셋을 구축했다.

송용수

- 지속적인 모델 학습 및 업데이트 : EfficientNet-Lite V3 모델과 Model Maker를 이용해 데이터셋이 새로 갱신될 때마다 지속적으로 모델을 학습하고 업데이트했다.
- 데이터 수집 및 라벨링 : 건전지, 캔, 아이스팩, 우유팩, 고무장갑, 일회용 숟가락, 컵라면 용기, 젓병, 종이컵, 페트병 총 10가지 항목에 대해 이미지 수집 및 촬영을 통해 재활용 쓰레기 이미지 데이터셋을 구축했다.

송태인

- csv 파일 제작 및 업데이트 : Model Maker를 이용한 모델 학습을 위해 각 데이터셋의 라벨링 좌표값과 Training, Test, Validation 분류 태그를 추가해 csv 파일을 제작 및 업데이트했다.
- 분리수거 Tip 자료 수집 후 추가 : 환경부, 각 지자체 환경 관련 부서에서 제공하는 분리수거 배출방법 가이드를 수집해 어플리케이션에 추가했다.
- 데이터 수집 및 라벨링 : 건전지, 캔, 아이스팩, 우유팩, 고무장갑, 일회용 숟가락, 컵라면 용기, 젓병, 종이컵, 페트병 총 10가지 항목에 대해 이미지 수집 및 촬영을 통해 재활용 쓰레기 이미지 데이터셋을 구축했다.

9. 맺음말

9.1 졸업 프로젝트를 통해 배운 점

권태운 : 프로젝트를 진행하며 다양한 문제들이 있었고 그 과정에서 다양한 경험을 했다. 모델을 안드로이드에 탑재하는 과정에서도 문제가 있었다. 맨 처음에는 파이토치로 모델을 제작했는데 파이토치의 모델 추출 방식으로는 안드로이드에 바로 탑재가 되지 않았고 한번 변형을 한 뒤에 안드로이드에 탑재해야 했다. 하지만 안드로이드에 맞게 변형된 모델도 탑재가 되지 않는 문제가 발생했다. 다양한 참고 자료와 구글링을 통해 검색해보았지만, 중간에 업데이트가 있었는지 우리의 방식은 적용되지 않았다. 포기하지 않고 팀원들과 원인을 찾아보았고 우여곡절 끝에 안드로이드에 탑재되는 모델을 제작했으며 성능 또한 기대 이상이었다.

모델을 학습시키는 과정에서도 문제가 발생했었는데 전날까지만 해도 잘 돌아가던 모델 학습 코드가 갑자기 작동하지 않는 것이었다. 해당 코드는 CPU를 통해 활용할 경우 잘 돌아갔지만 GPU를 활용해 구동할 경우 에러가 발생해 초기에는 GPU 문제인가 하고 구글 colab 상위 요금제를 구동했지만 같은 문제가 발생하였다. 코드 문제로 인해 진행이 디터졌고 검색을 해보아도 찾을 수가 없었다. 이후 공식 Git hub Colab사이트에 issue를 남겨보았고 다양한 사람들과 소통하며 해결할 수 있었다.

팀원들도 함께 포기하지 않고 힘을 합치면 무엇이든 할 수 있다는 자신감이 생겼고 이를 바탕으로 앞으로 있을 어떠한 역경도 잘 해결해 나갈 수 있을 것 같다.

송용수 : 프로젝트의 특성 상 모델을 학습시키기 위해 기존에 없던 재활용 쓰레기 이미지 데이터셋을 조원들과 새로 만들어야 했다. Github에서 garythung이 배포하는 Trashnet이라는 쓰레기 관련 데이터셋이 있었지만 해당 데이터셋은 유리, 카드로드지, 종이, 플라스틱, 금속, 일반 쓰레기 총 6가지 항목으로 구성되어 있어 이번 프로젝트에 반영하기 어려웠다. 구글링을 통해서 많은 이미지를 모을 수 있었지만 라벨링을 할 수 있는 훼손 또는 변형된 쓰레기 이미지를 구하기는 어려워 직접 이미지를 촬영하고 각도, 명도 등을 변형하며 꽤나 원시적인 방법으로 데이터를 추가했다. 그동안 기계학습에 있어서 알고리즘과 모델의 속도와 크기의 최적화만 중요하다고 생각했지만 이번 프로젝트를 진행하면서 최종적으로 어떤 분야에 쓰이는지에 해당하는 데이터셋 자체의 중요성과 그 자체로 자산이 될 수 있다는 것을 알게 되었다.

송태인 : 첫 학기에는 요구사항분석부터 시작해 시스템 설계를 거쳐 프로토타입을 만들어본 뒤 다음 학기에는 부족한 점을 찾아 계속해서 보완해 프로젝트를 최종 완성하는 1년간의 긴 프로젝트였다. 처음에는 요구사항분석이나 시스템 설계의 필요성을 느끼지 못했는데 1학기에 잘 설계해 놓은 문서 덕분에 팀원 간의 소통이 잘 되면서 수월하게 프로젝트를 진행할 수 있었고, 문서의 중요성을 프로젝트를 진행해 나가면서 더 크게 느꼈다. 또한 팀원 모두가 인공지능 관련 지식이 부족한 상태에서 스테디를 진행하면서 다양한 알고리즘을 통해 여러 모델들을 만들어보면서 분리수거 데이터셋에 가장 적합한 모델을 찾을 수 있었다. 그리고 프로젝트를 진행하면서 모델과 안드로이드 연동 문제, 모델 학습 중 발생한 GPU 에러 등 어려움이 생겼을 때는 결국 해결하기 위해서는 공식 문서와 공식 사이트의 중요하다는 것을 알 수 있었다. 무엇보다 1년간의 긴 프로젝트를 진행하면서 팀원들과의 소통이 얼마나 중요한지 배울 수 있었던 시간이었다.

9.2 향후 졸업작품의 활용 방안

9.2.1 자동 무인 쓰레기통 분류기

기존 프로젝트를 확장해 학습시킨 모델과 쓰레기통, 카메라를 분류설비와 결합하는 방식으로 활용한다면 자동 무인 쓰레기통 분류기를 만들 수 있다. 이러한 쓰레기 분류기를 미국과 같은 분리수거가 잘 이루어지지 않는 지역에서 활용할 경우, 자원을 절약하고 환경오염을 줄이는 에코 사회로 한발짝 더 다가 갈수 있다.

9.2.2 유치원생 교육 도구로 활용

유치원이나 초등학교 저학년 상대로 환경 교육 도구로 사용할 수 있다. 유치원생이나 초등학교 저학년을 대상으로 하는 교구로서 가장 중요한 것은 얼마나 아이들의 흥미를 끌 수 있는 지이다.

분리수거 척척박사 어플리케이션은 아이들이 휴대폰을 이용해 촬영하며 직접 분류결과를 확인할 수 있기 때문에 교구로서 흥미를 느낄 것이다.

또한, 시간이 지날수록 환경 문제가 대두되고 있기 때문에 분리수거 교육에 대한 중요성은 점점 올라가고 있다. 분리수거 척척박사 어플리케이션은 분리수거 방식을 알려주기 때문에 아이들은 분리수거 방법을 쉽게 배울 수 있다.