KU KONKUK UNIVERSITY

SW중심대학

이미지 인식을 이용한 음식 조리법 추천 앱 (찍 먹)

건국대학교

팀명 : 찍먹

팀원 : 강관주, 류진이, 허찬호

소속학과 : 컴퓨터공학부

지도교수 : 박능수 교수님



팀 소개

- 팀 소개

  저희 팀의 이름은 “찍먹”입니다.
  “찍고 먹자”의 줄임말인 팀 이름처럼 음식 사진만 찍으면 간편하게 레시피를 확인할 수 있는 앱을 개발하여 사용자들에게 편의를 제공하고자 합니다.

- 팀원

이름(애칭)	소속(학과)	역할	이메일	연락처
강관주	건국대(컴퓨터공학부)	팀장	xprtktm1@gmail.com	010-4110-7141
류진이	건국대(컴퓨터공학부)	팀원	wlsdl5685@naver.com	010-6417-5685
허찬호	건국대(컴퓨터공학부)	팀원	gjcksgh430@naver.com	010-7105-9493

프로젝트 개요

- 작품 개요

· 개발배경 : 최근 외식, 배달 물가 부담과 더불어 간편식의 판매량이 증가하고 있습니다. 간편식은 전자레인지나 에어프라이어를 이용하여 조리해 먹는 경우가 대부분인데, 조리 시간 등의 조리법은 포장지에서 눈에 띄지 않는 작은 글씨로 적혀 있는 경우가 많습니다.
또한, 전자레인지는 가정용/상업용에 따라 이용 시간이 상이하며, 에어프라이어는 예열 시간, 온도, 시간까지 무려 3가지 사항을 고려해야 합니다.
이에 촬영된 이미지나 카메라 촬영을 통해 조리법을 알려주는 앱을 개발하여 불편함을 해소해 보고자 합니다.

· 목표 시스템 : 저희 앱의 주요 목표는 딥러닝을 활용한 한국 음식 이미지 인식과 정보의 전달입니다. 이를 통해 음식 조리법을 궁금해하는 사용자들의 편의를 증진시키고자 합니다.

프로젝트 개요

- 작품 구성 및 상세내용

· 작품의 구성

먼저 Anaconda, Python language를 이용해 딥러닝 모델을 학습, 생성하고, 모델을 tflite 파일로 변환했습니다. 이후에 변환된 tflite 파일을 Android Studio에 옮기고, Kotlin language를 통해 음식 조리법 추천 앱을 제작하였습니다.

· 상세 내용

딥러닝 모델을 학습, 생성, 변환하기 위해 TensorFlow Lite, 케라스(Keras) 오픈 소스 신경망 라이브러리 등을 사용하였고, 학습을 위한 데이터셋은 AI Hub - 한국 이미지(음식), 구글 크롤링을 통해 획득하였습니다. 그리고, Android Studio에서는 UX/UI를 디자인하였습니다.

프로젝트 개요

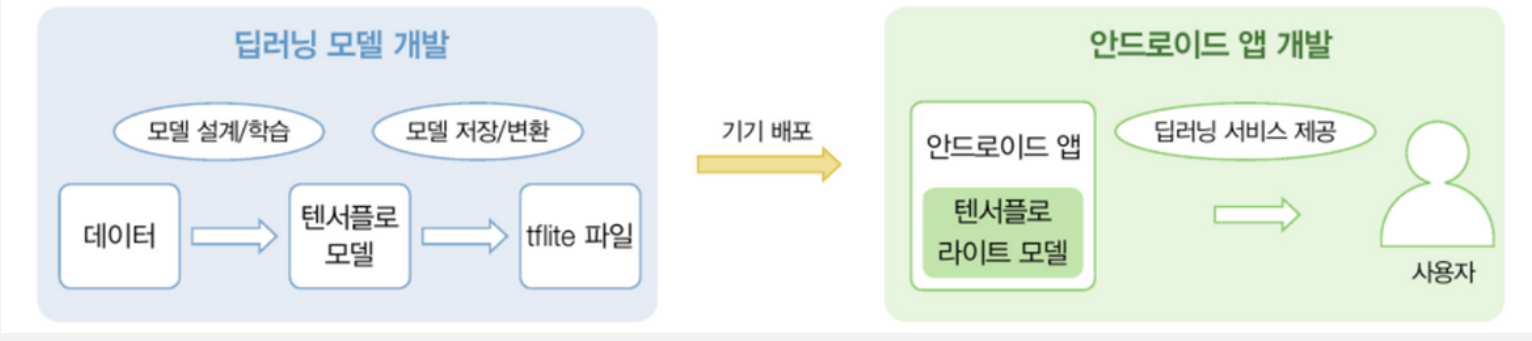
- 작품 구성 및 상세내용

· 상세 내용

모델 선택 방법

모델 선택 방법		설계	학습	변환
모델 직접 개발	모델 전체 직접 개발	필요	필요	필요
사전 학습 모델 이용	텐서플로 모델	불필요	필요	필요
		불필요	불필요	필요
	TFLite 모델	불필요	불필요	불필요
전이 학습		불필요	일부 필요	필요

안드로이드 딥러닝 개발 프로세스



이미지셋 수집 → 직접 모델 설계/학습 → 학습된 모델 변환 → TFLite 모델 배포 → 안드로이드 앱 개발

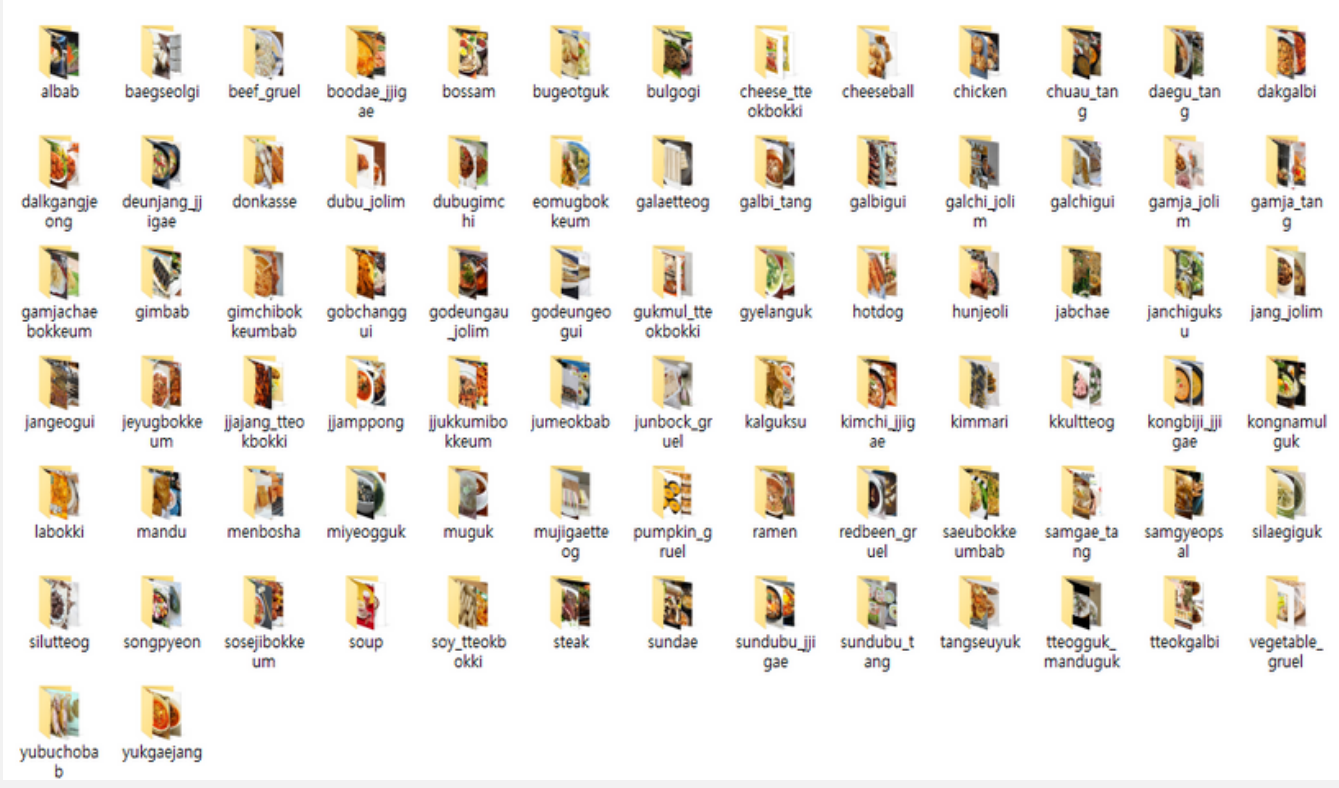
프로젝트 개요

- 작품 구성 및 상세내용

· 상세 내용

데이터 수집 - AI Hub - 한국 이미지(음식), 구글 크롤링

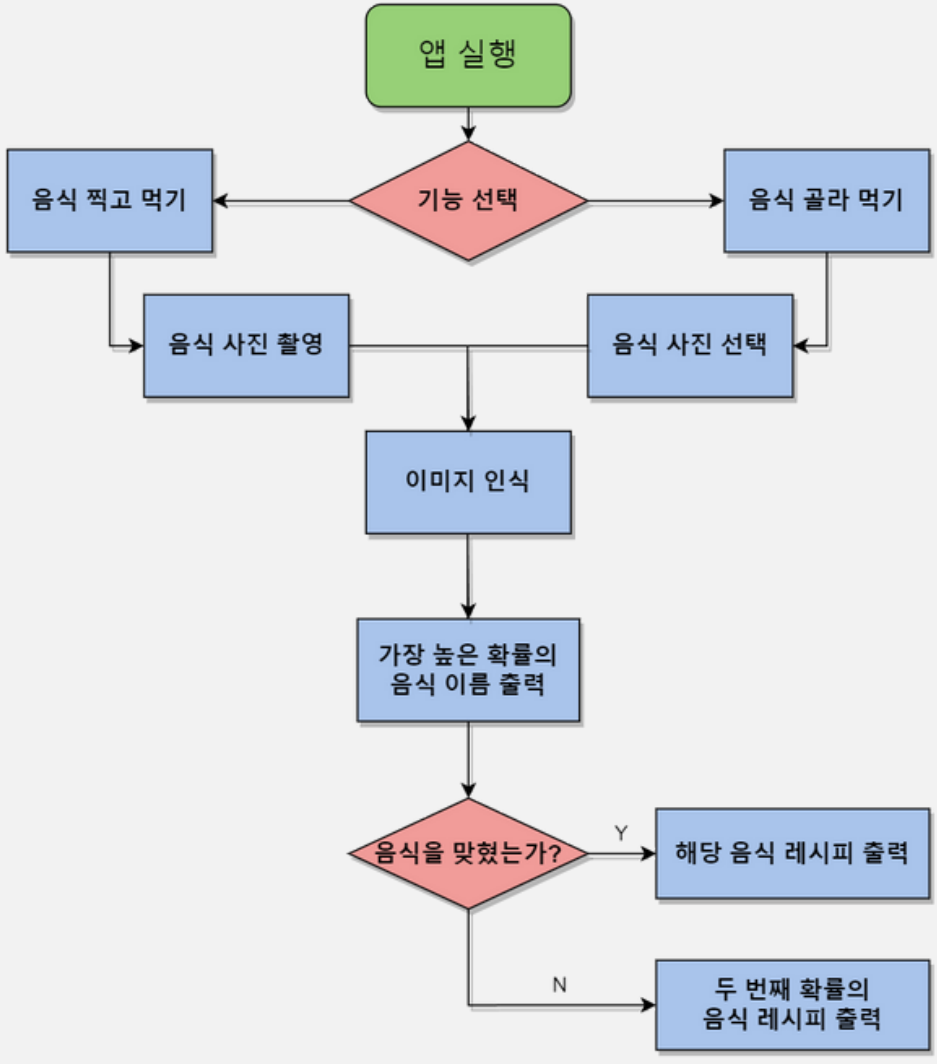
80개 클래스, 43258개 사진 파일 수집



작품 구성 및 상세내용

- 개발 세부 내용(설계서)

· 서비스 시나리오



프로젝트 개요

- 개발 세부 내용(설계서)

· 시스템 설계

이미지 전처리 과정 (80%는 훈련, 20%는 유효성 검사에 사용)

```
[ ]  1 batch_size = 32
     2 img_height = 180
     3 img_width = 180
```

```
[ ]  1 train_ds = tf.keras.utils.image_dataset_from_directory(
     2   data_dir,
     3   validation_split=0.2,
     4   subset="training",
     5   seed=123,
     6   image_size=(img_height, img_width),
     7   batch_size=batch_size)
```

```
Found 12716 files belonging to 19 classes.
Using 10173 files for training.
```

```
[ ]  1 val_ds = tf.keras.utils.image_dataset_from_directory(
     2   data_dir,
     3   validation_split=0.2,
     4   subset="validation",
     5   seed=123,
     6   image_size=(img_height, img_width),
     7   batch_size=batch_size)
```

```
Found 12716 files belonging to 19 classes.
Using 2543 files for validation.
```

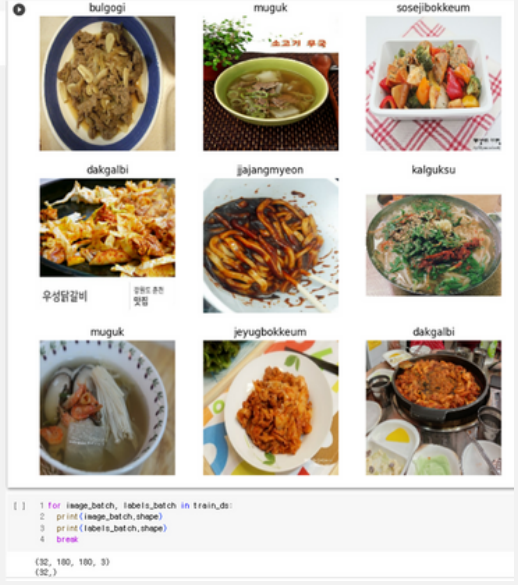
```
[ ]  1 class_names = train_ds.class_names
     2 print(class_names)
```

```
['baegseolgi', 'bulgogi', 'chicken', 'dakgalbi', 'donkasse', 'gimbab', 'gimchibokkeumbab', 'gukmul_tt
```

```
[ ]  1 normalization_layer = layers.Rescaling(1./255)
```

```
[ ]  1 normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
     2 image_batch, labels_batch = next(iter(normalized_ds))
     3 first_image = image_batch[0]
     4 # 픽셀값 범위 : `[0,1]`
     5 print(np.min(first_image), np.max(first_image))
```

```
0.0 1.0
```




```
[ ]  1 for image_batch, labels_batch in train_ds:
     2   print(image_batch.shape)
     3   print(labels_batch.shape)
     4   break
```

```
(32, 180, 180, 3)
(32,)
```

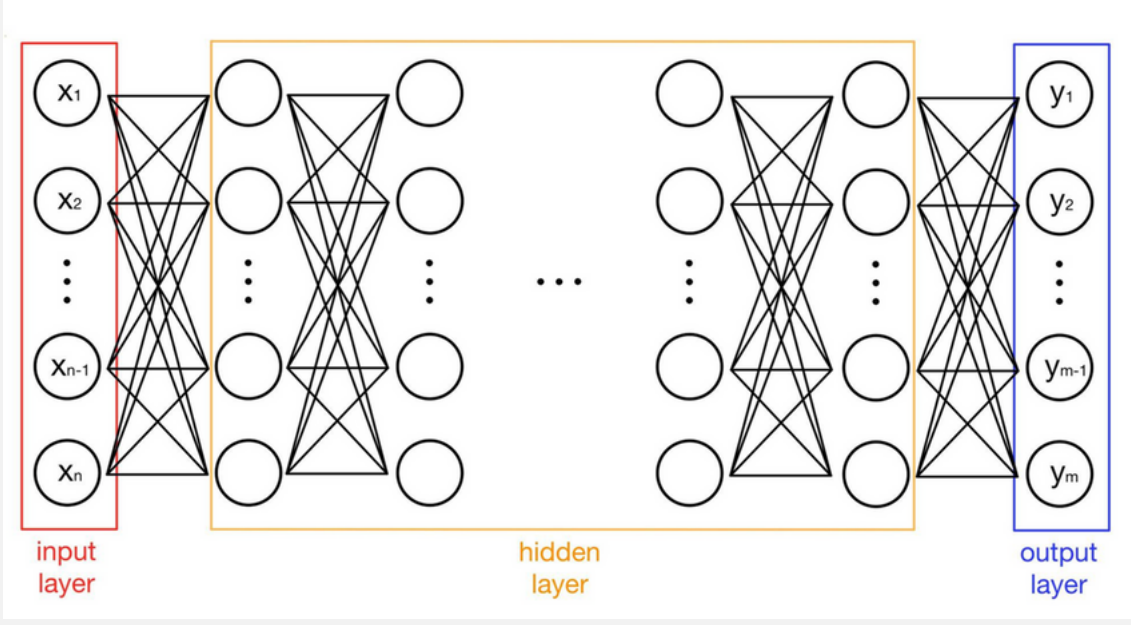
모델의 입력 크기 확인 → 입력 이미지 크기 변환 → 입력 이미지 채널 변환 → 입력 이미지 포맷 변환

프로젝트 개요

- 개발 세부 내용(설계서)

· 시스템 설계

모델 설계



- 다층 퍼셉트론 모델(완전 연결 신명망)로 설계
- 입력층, 은닉층, 출력층으로 구성
- 각 층 간의 모든 노드가 서로 연결

프로젝트 개요

SW중심대학

- 개발 세부 내용(설계서)

· 시스템 설계

모델 생성 및 훈련

```
[ ]  1 model = Sequential([
     2   data_augmentation,
     3   layers.Rescaling(1./255),
     4   layers.Conv2D(16, 3, padding='same', activation='relu'),
     5   layers.MaxPooling2D(),
     6   layers.Conv2D(32, 3, padding='same', activation='relu'),
     7   layers.MaxPooling2D(),
     8   layers.Conv2D(64, 3, padding='same', activation='relu'),
     9   layers.MaxPooling2D(),
    10   layers.Dropout(0.2),
    11   layers.Flatten(),
    12   layers.Dense(128, activation='relu'),
    13   layers.Dense(num_classes, name="outputs")
    14 ])
```

```
[ ]  1 model.compile(optimizer='adam',
     2               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
     3               metrics=['accuracy'])
```

```
[ ]  1 model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_1 (Sequential)   (None, 180, 180, 3)       0

 rescaling_2 (Rescaling)     (None, 180, 180, 3)       0
```

```
1 epochs = 15
2 history = model.fit(
3   train_ds,
4   validation_data=val_ds,
5   epochs=epochs
6 )
```

```
Epoch 1/15
318/318 [==============================] - 12s 28ms/step - loss: 2.2255 - accuracy: 0.292
Epoch 2/15
318/318 [==============================] - 9s 28ms/step - loss: 1.8560 - accuracy: 0.3926
Epoch 3/15
318/318 [==============================] - 9s 29ms/step - loss: 1.6193 - accuracy: 0.4833
Epoch 4/15
318/318 [==============================] - 9s 28ms/step - loss: 1.4619 - accuracy: 0.5361
Epoch 5/15
318/318 [==============================] - 9s 29ms/step - loss: 1.3513 - accuracy: 0.5742
Epoch 6/15
318/318 [==============================] - 9s 29ms/step - loss: 1.2652 - accuracy: 0.6058
Epoch 7/15
318/318 [==============================] - 9s 28ms/step - loss: 1.2014 - accuracy: 0.6216
Epoch 8/15
318/318 [==============================] - 9s 28ms/step - loss: 1.1459 - accuracy: 0.6372
Epoch 9/15
318/318 [==============================] - 9s 29ms/step - loss: 1.0927 - accuracy: 0.6583
Epoch 10/15
318/318 [==============================] - 9s 28ms/step - loss: 1.0637 - accuracy: 0.6625
Epoch 11/15
318/318 [==============================] - 9s 29ms/step - loss: 1.0084 - accuracy: 0.6811
Epoch 12/15
```

프로젝트 개요

- 개발 세부 내용(설계서)

· 시스템 설계

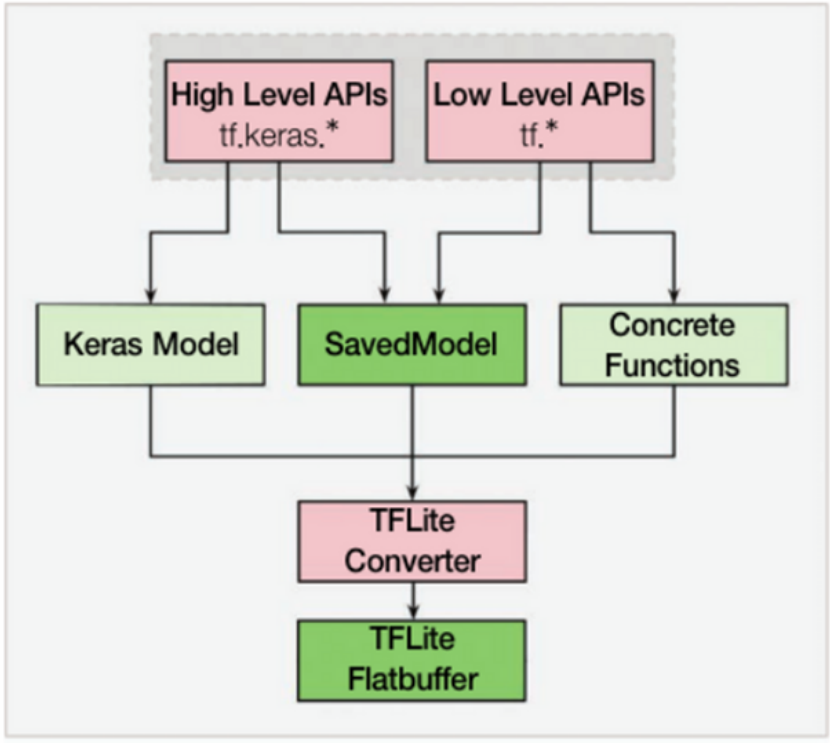
모델 훈련 결과 시각화 (우측은 Overfitting을 막기 위해 Data augmentation 추가, 모델에 Dropout 도입)

프로젝트 개요

- 개발 세부 내용(설계서)

· 시스템 설계

훈련된 모델 → TensorFlow Lite 모델 형식으로 변환

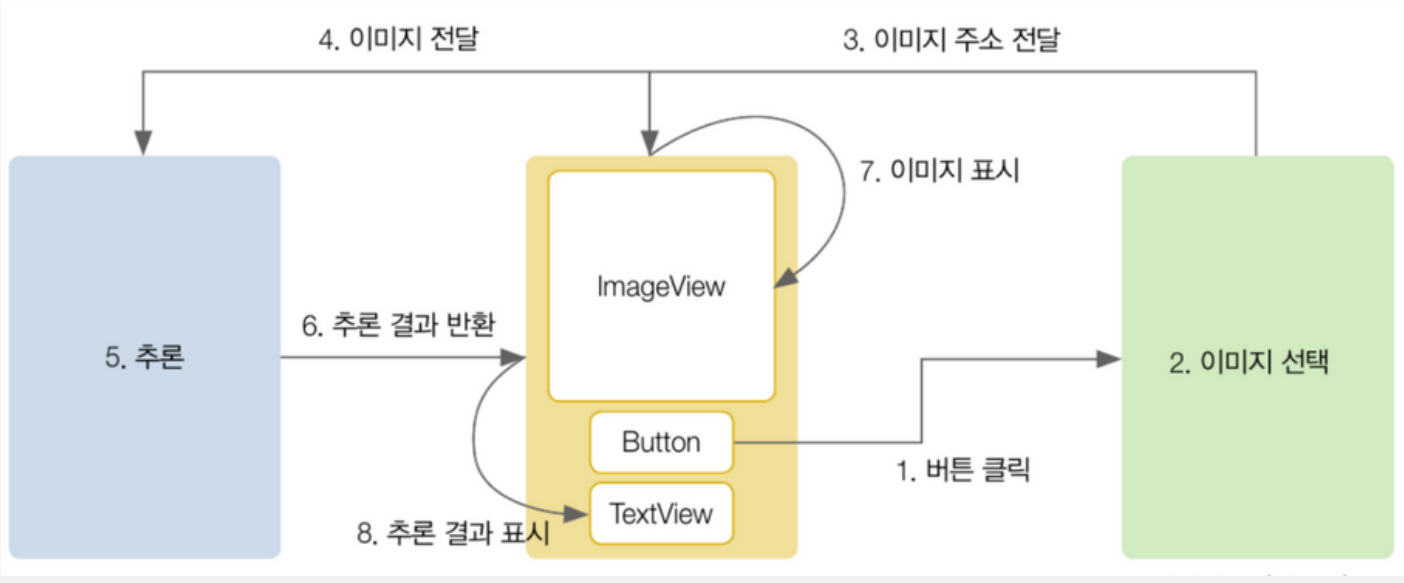



```
[ ]  1 # 모델 변환
     2 converter = tf.lite.TFLiteConverter.from_keras_model(model)
     3 tflite_model = converter.convert()
     4
     5 # 모델 저장
     6 with open('model.tflite', 'wb') as f:
     7   f.write(tflite_model)
```

- 개발 세부 내용(설계서)

· 시스템 설계

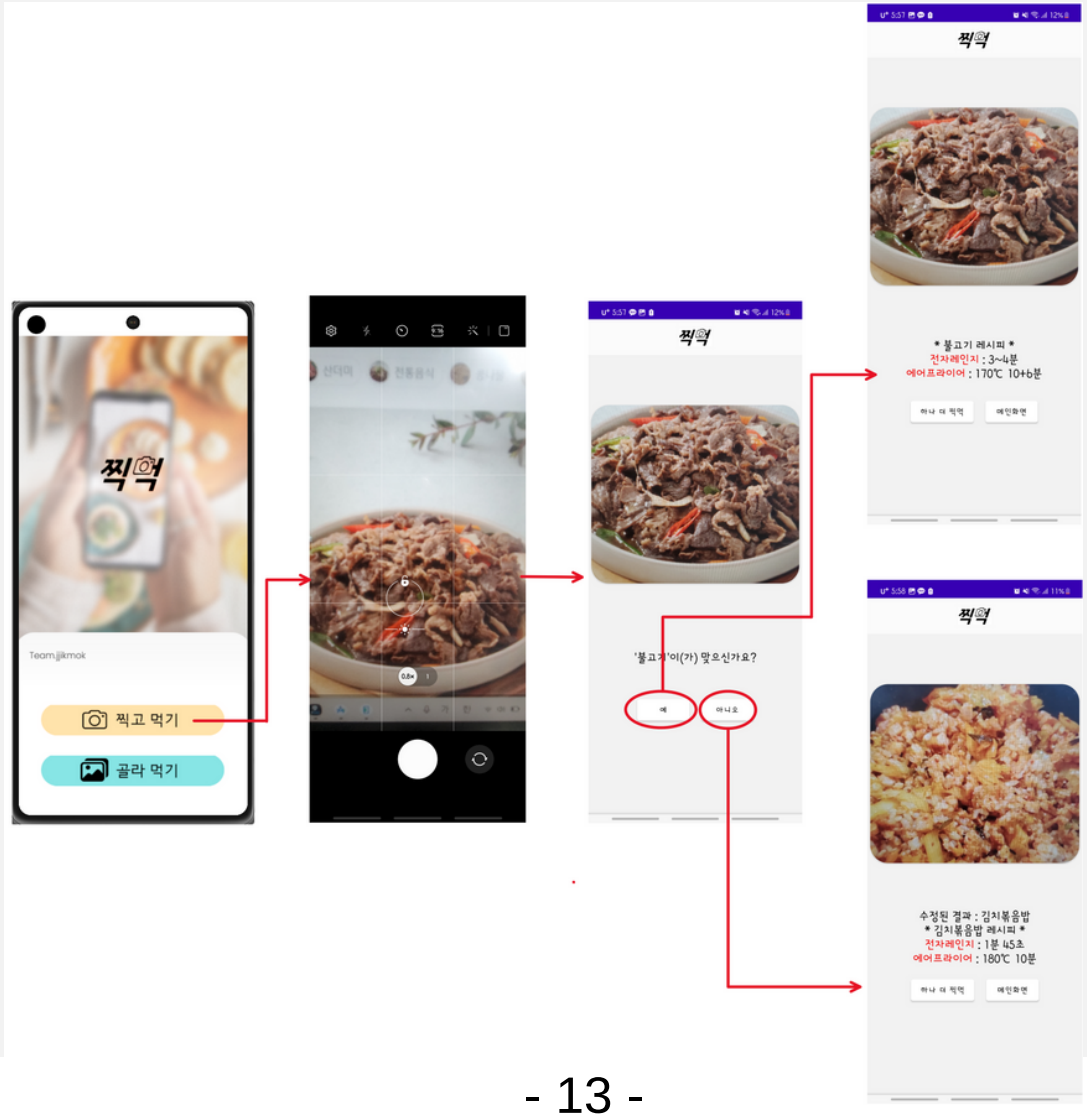
안드로이드 앱 내에서의 추론과정



작품 구성 및 상세내용

- **개발 세부 내용(설계서)**

· UI 설계 - “찍고 먹기 버튼”

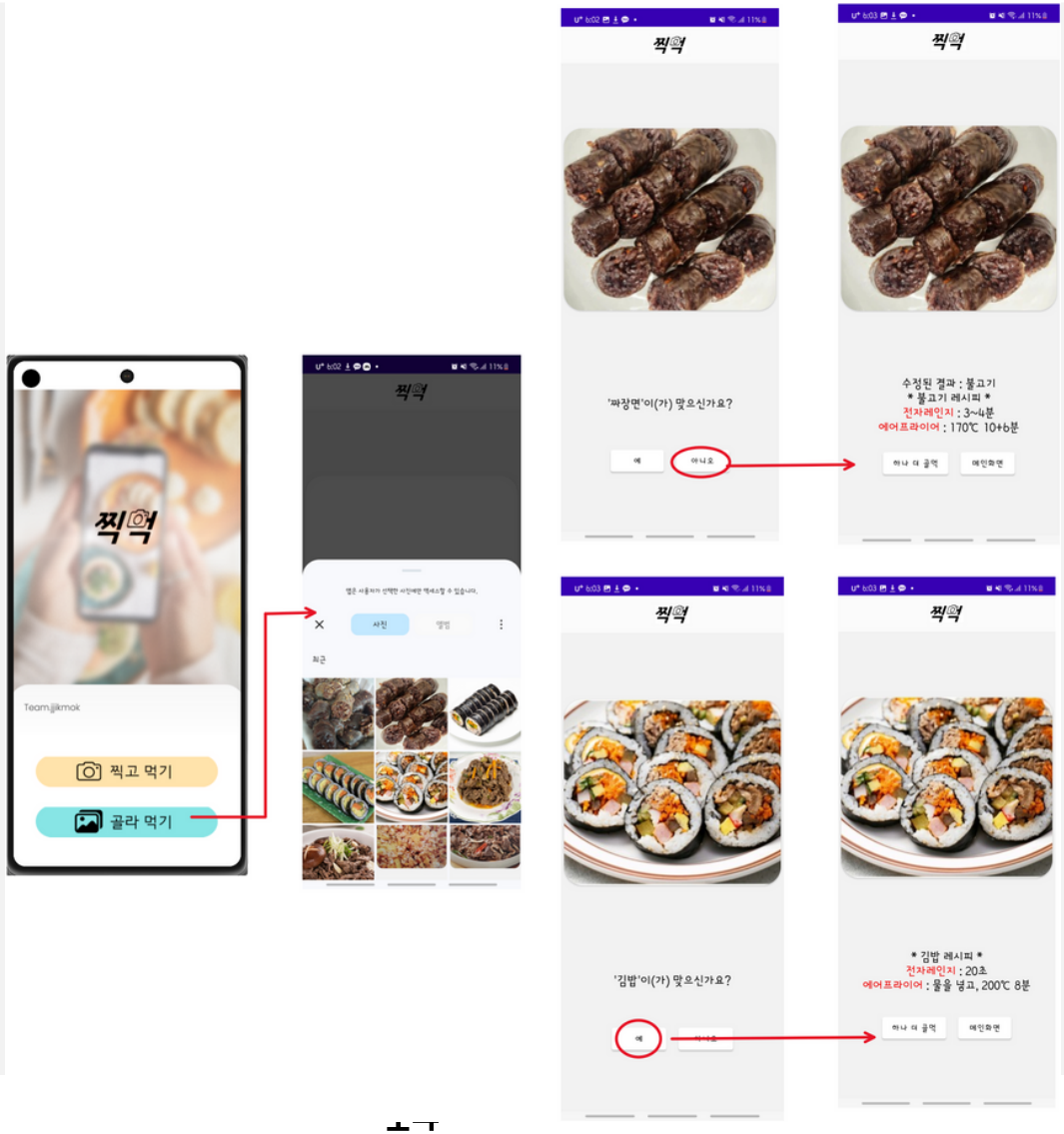


작품 구성 및 상세내용

SW중심대학

- **개발 세부 내용(설계서)**

· UI 설계 - “골라 먹기 버튼”



- **개발 세부 내용(설계서)**

· Interface 설계

- 홈 화면 : "찍고 먹기" 버튼과 "골라 먹기" 버튼 선택 가능
- 찍고 먹기 버튼 : 버튼 선택 시 카메라를 실행하여 음식 사진 촬영 가능
- 골라 먹기 버튼 : 버튼 선택 시 파일에서 사진을 불러오기 가능
- 사진 확인 화면 : 사용자가 촬영한 사진을 딥러닝 모델이 인식하여 어떤 음식인지 확인
- "예" 버튼 : 버튼 클릭 시 해당 음식에 대한 전자레인지, 에어프라이어 조리 시간 출력
- "아니오" 버튼 : 버튼 클릭 시 딥러닝 모델이 인식한 음식 중 2번째로 확률이 높은 음식 조리법 출력
- "하나 더 찍먹" 버튼 : 버튼 클릭 시 찍고 먹기 버튼과 동일하게 카메라 실행하여 음식 사진 촬영 가능
- "메인화면" 버튼 : 홈 화면으로 돌아감

구현결과

- 구현결과 요약

  직접 수집한 사진으로 모델을 성공적으로 학습시켰고 Overfitting등의 문제도 성공적으로 해결하였다.
  모델을 안드로이드 스튜디오로 배포 후 앱 개발을 성공적으로 마쳤다.

  카메라로 사진을 찍어 음식을 인식하는 ‘찍고 먹기’ 기능과,
  갤러리 내 사진으로 음식을 인식하는 ‘골라 먹기‘ 두 기능 모두 높은 정확도로 작동한다.

  현재는 인식한 음식에 대한 간단한 조리법 만을 사용자에게 제공하지만,
  만들어진 딥러닝 모델을 활용해 한식에 관심이 많은 외국인 등의 사용자를 위해 더 많은 정보를 제공하는 애플리케이션으로 발전시켜 보고자 한다.

기대 효과

- 프로젝트 성과에 대한 기대효과

  기존에 있던 요리 조리법 앱들은 재료를 등록하거나 음식을 검색하여야만 원하는 조리법을 얻을 수 있었다. 저희 찍먹 앱은 '이미지 인식' 기능을 이용하기 때문에 앱을 이용하는 사용자들이 편하게 조리법을 얻을 수 있을 것이며, 이런 차별성은 현재 기존에 있는 앱들 사이에서도 경쟁력이 있다고 생각한다.

- 활용가능성 및 확장 가능성

  더 정확한 레시피를 제공할 수 있을 것이다.
  데이터베이스를 구축 후 연결할 수 도 있고, 레시피를 제공하는 다른 사이트들과 협업을 맺어 레시피를 확장할 수 있다고 생각한다.

  또한 음식 조리는 음식의 양, 상태에 따라 조리법이 상이하기 때문에, 음식의 종류만을 판별하는 것이 아닌 음식의 양과 상태를 파악할 수 있게 이미지 인식기능을 확장할 수 있을 것이라고 생각한다.

최종 결과물

SW중심대학

- **최종 결과물에 대한 동영상 / 사진**

1. 동영상 링크 :

https://youtube.com/shorts/WEeEmGQSFmA?feature=share

2. 우수작품 영상에서 가장 ‘하이라이트’ 장면 시간 기재 :
   “00:05:00~00:10:00”
   “00:29:00~00:37:00”