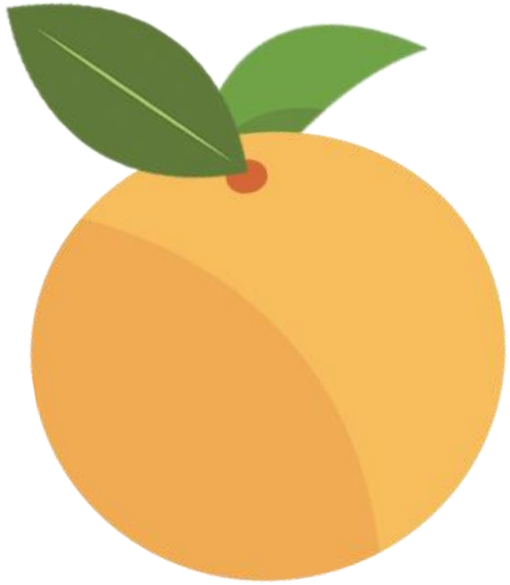


종합 설계

- 딥 러닝을 활용한 선과 시스템 개발 -

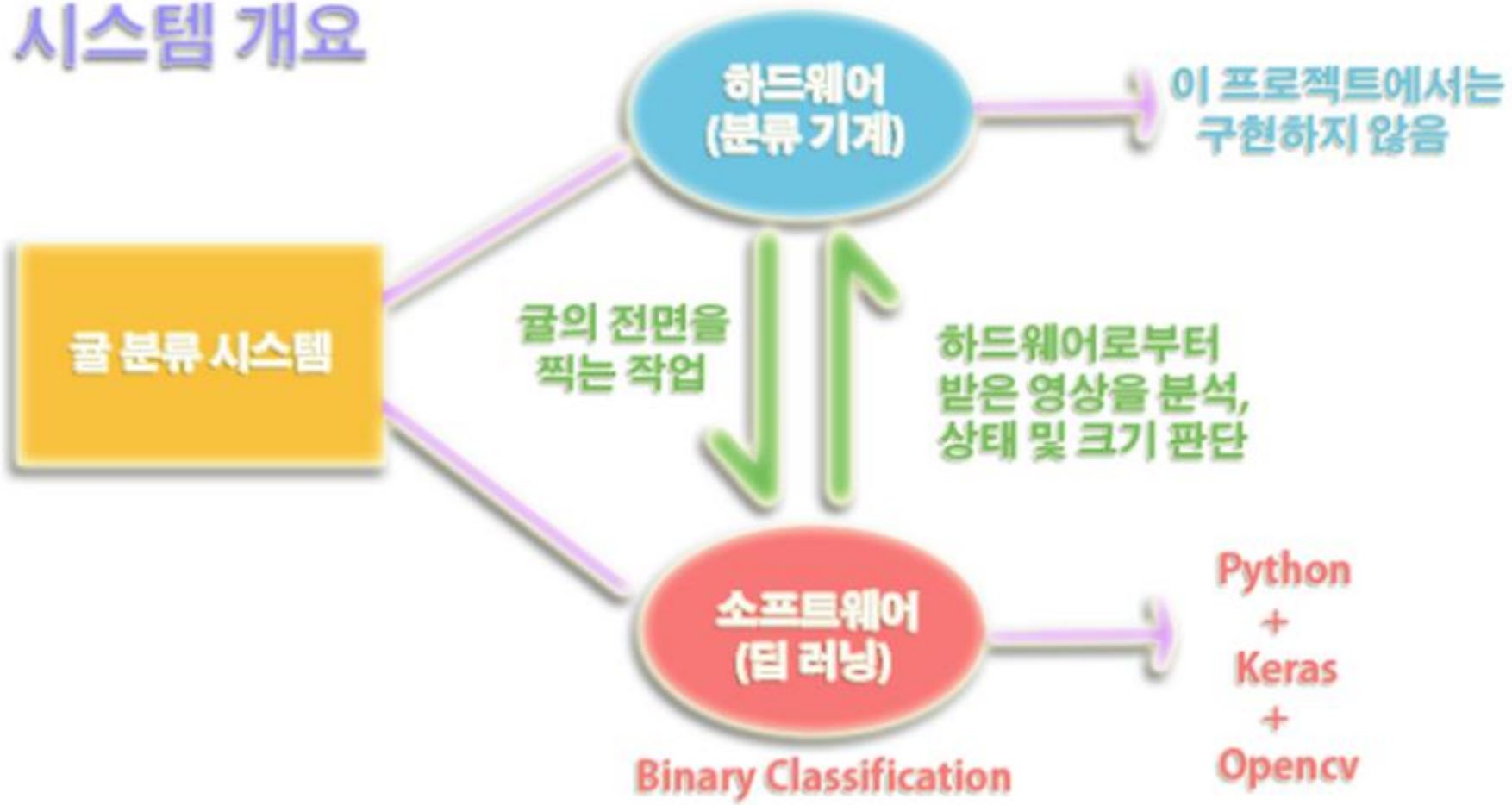
학 과: 정보통신공학과
지도교수: 박 장 우 교수님
팀 명: New-Learn
팀 원: 강동우, 이승기, 김윤태, 조재경

Contents

1.시스템 소개

(1) 전체 시스템 구성

시스템 개요



(2) 개발 배경

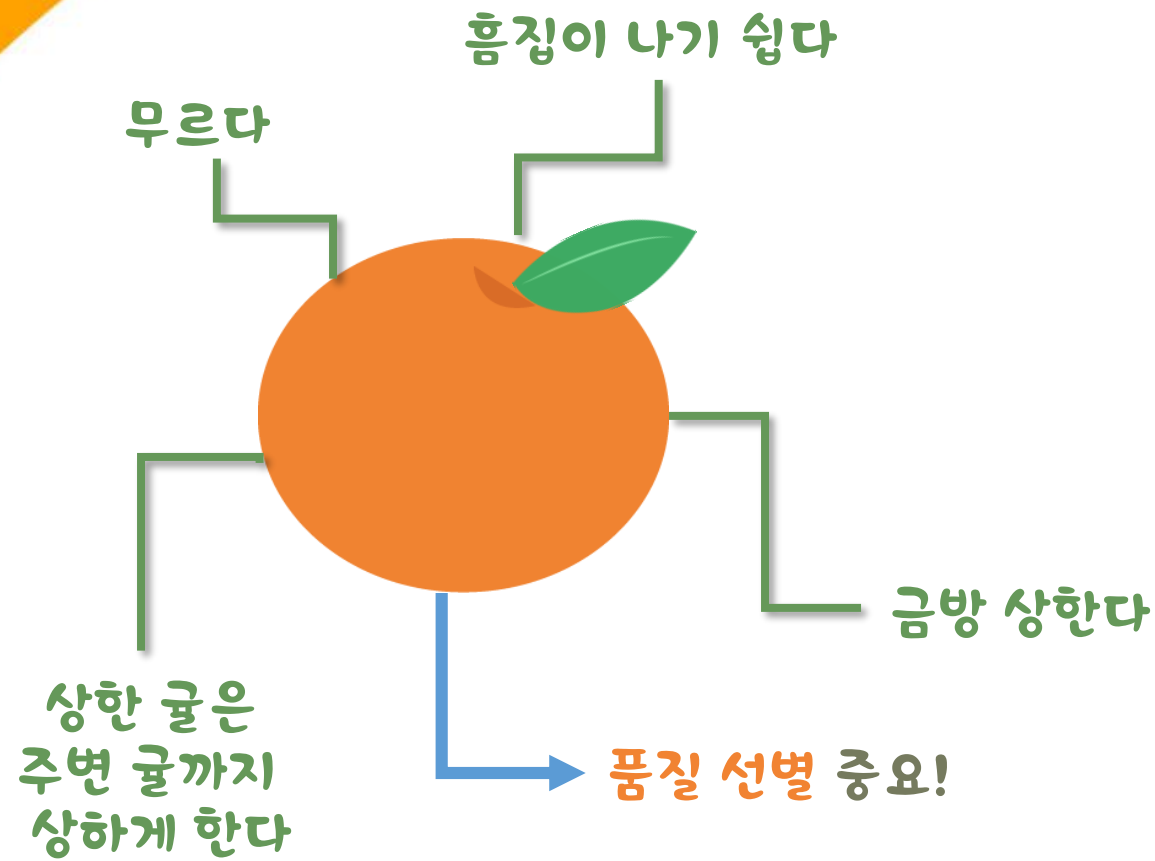
국내 과일 분류기1(↑영상)

국내 과일 분류기
중량이나 크기로만 등급 결정
-> 포장 시 육안으로 상태 구별

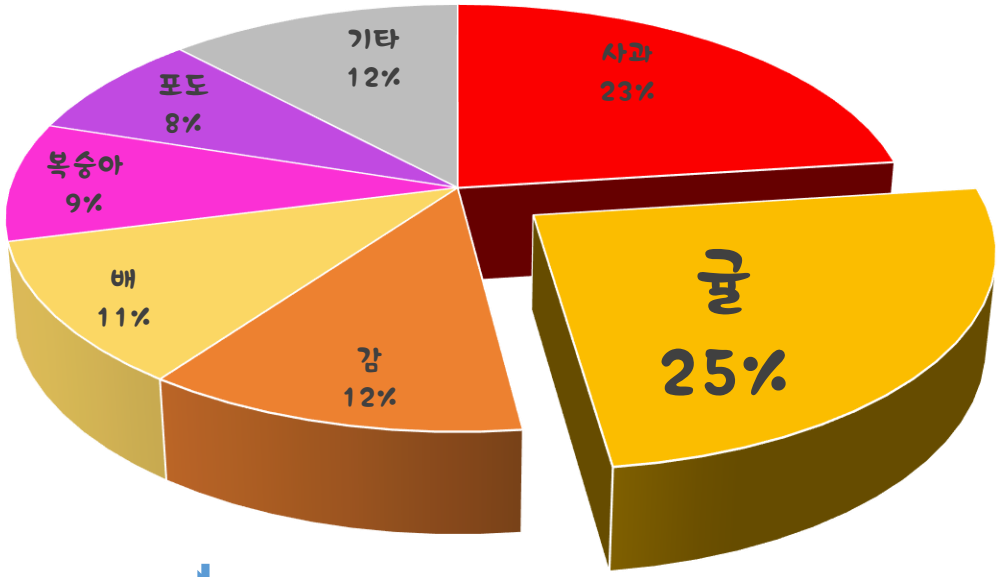
국내 과일 분류기2(↑영상)

- 추가적인 인건비 발생
- 투자한 시간에 비해 작업량이 떨어져 비효율적
- 귤의 상태를 다각도로 볼 수 없음

왜 귤을 선택했나?



[출처] 과실생산량(2017, 통계청)



다른 과일보다 생산량이 많다!

→ 효율적인 귤 선별기는 수요가 있을 것이다!

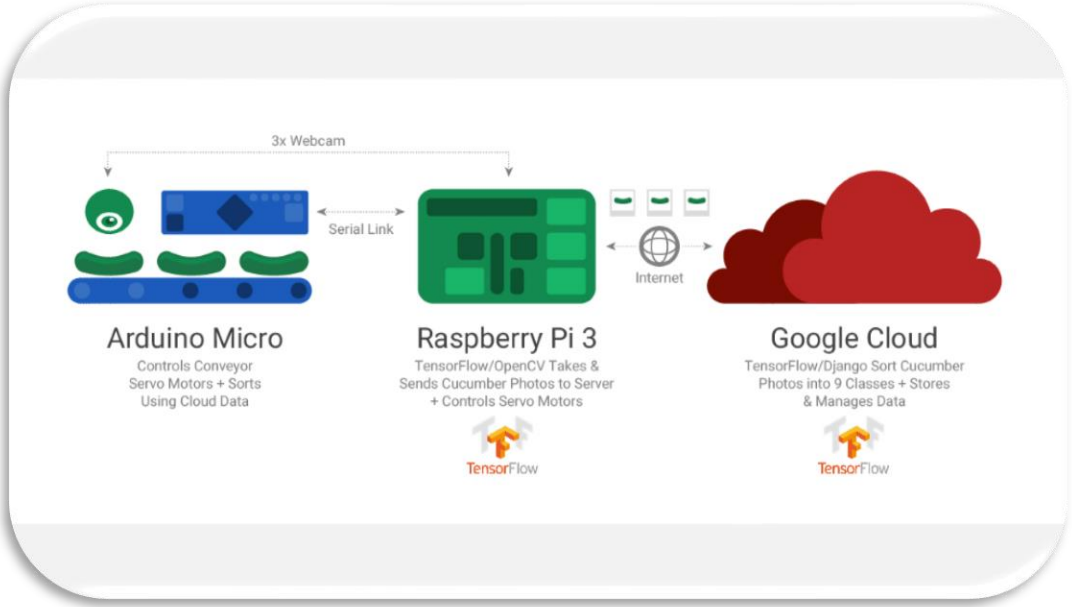
북한에 보낸 200톤 귤 선별 작업

[출처] 채널A 뉴스

컴퓨터로 선별하면
인건비는 줄이고
정확도를 높일 수 있지 않을까?

유사 사례 - 딥 러닝을 적용한 일본의 오이 농가

"크기, 두께, 색상, 질감, 작은 흠집, 구부러짐 여부, 가시가 있는지 여부 등을 살펴야 합니다.
분류 체계를 익히는데 여러 달이 걸리고, 가장 바쁜 시기에 아르바이트를 고용할 수 없습니다."



=> 딥 러닝을 활용하면 귤 분류 체계를 익히는 시간을 줄일 수 있다!

(3) 시스템 동작 과정

- 가정: 2가지 상태
  (Good/ Bad)

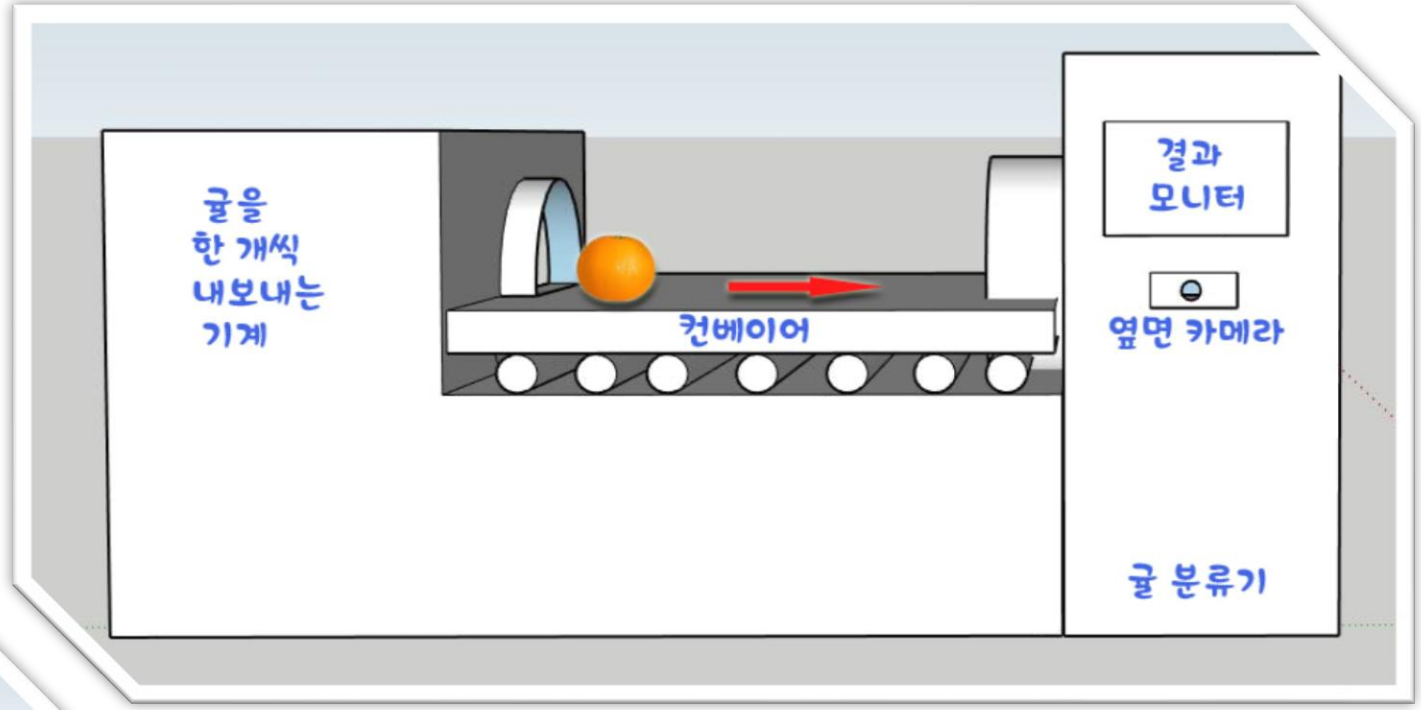
Good

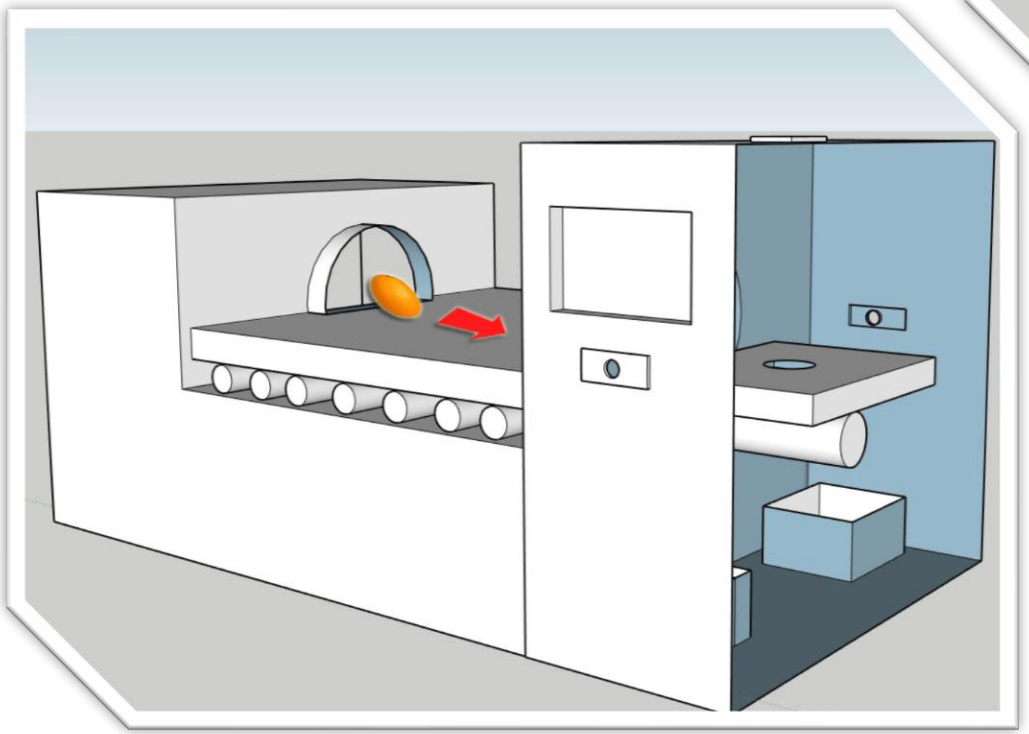



Bad



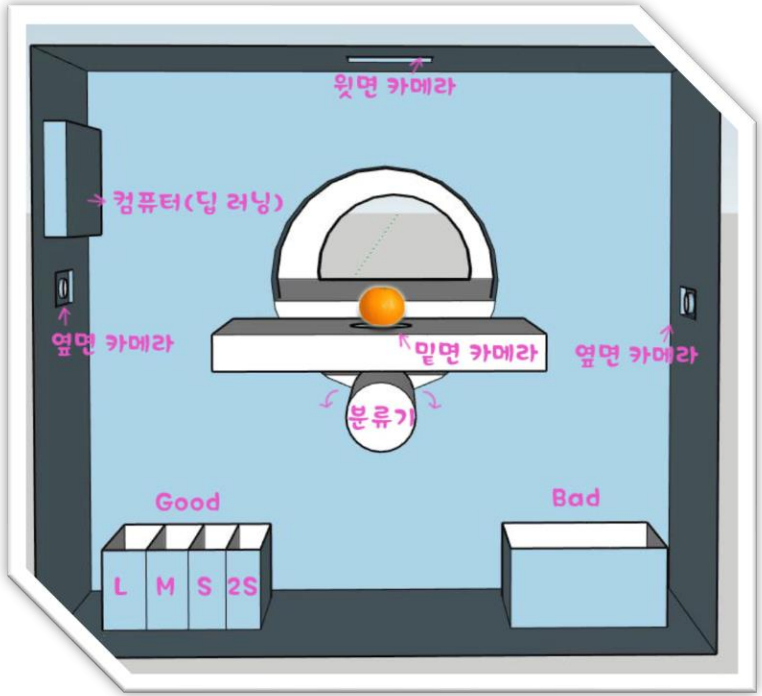

가상의 분류 기계





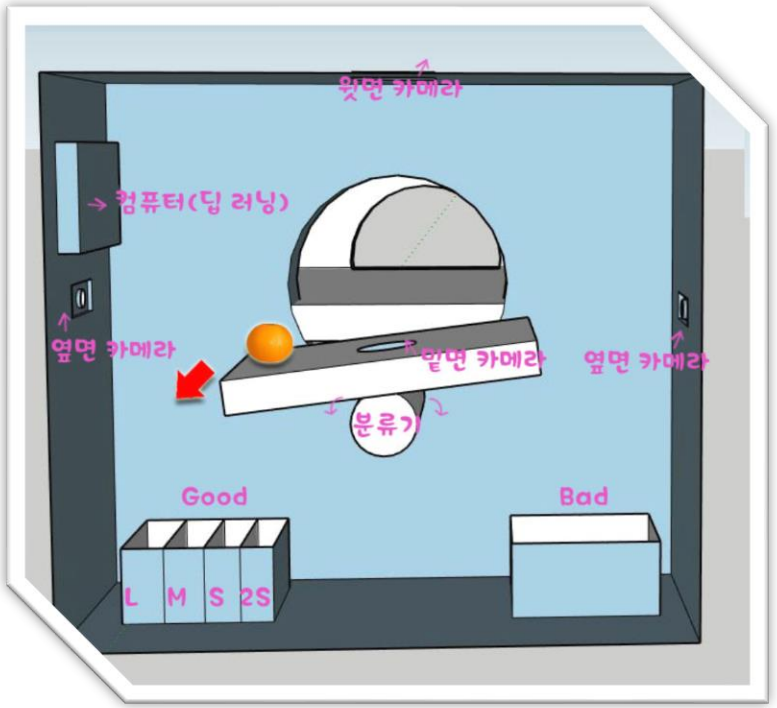
기계는 만들지 X
이런 기계가 있다고 가정

가상 분류기 동작

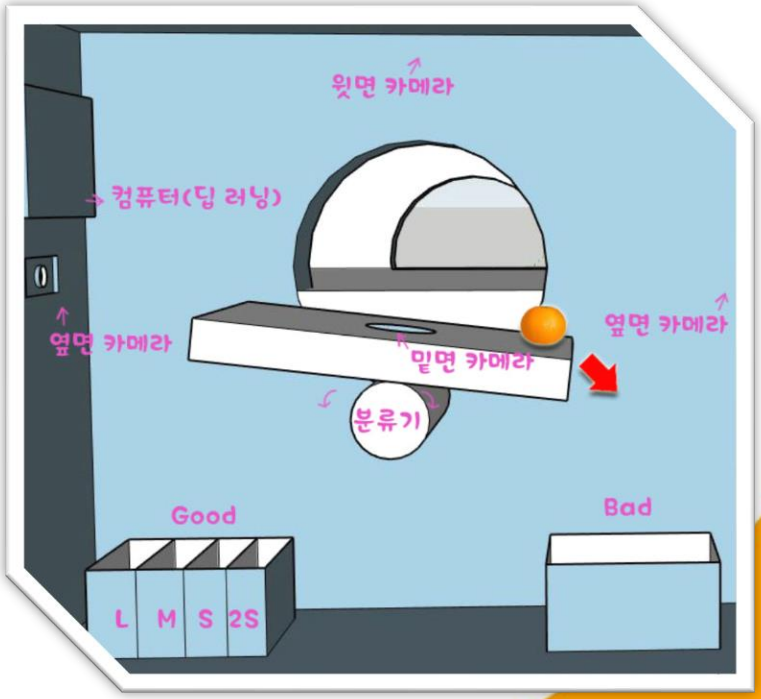


4개의 카메라로
귤 4면을 찍음
↓
딥 러닝으로
귤 상태 파악
↓
판의 기울기를
조절하여 분류

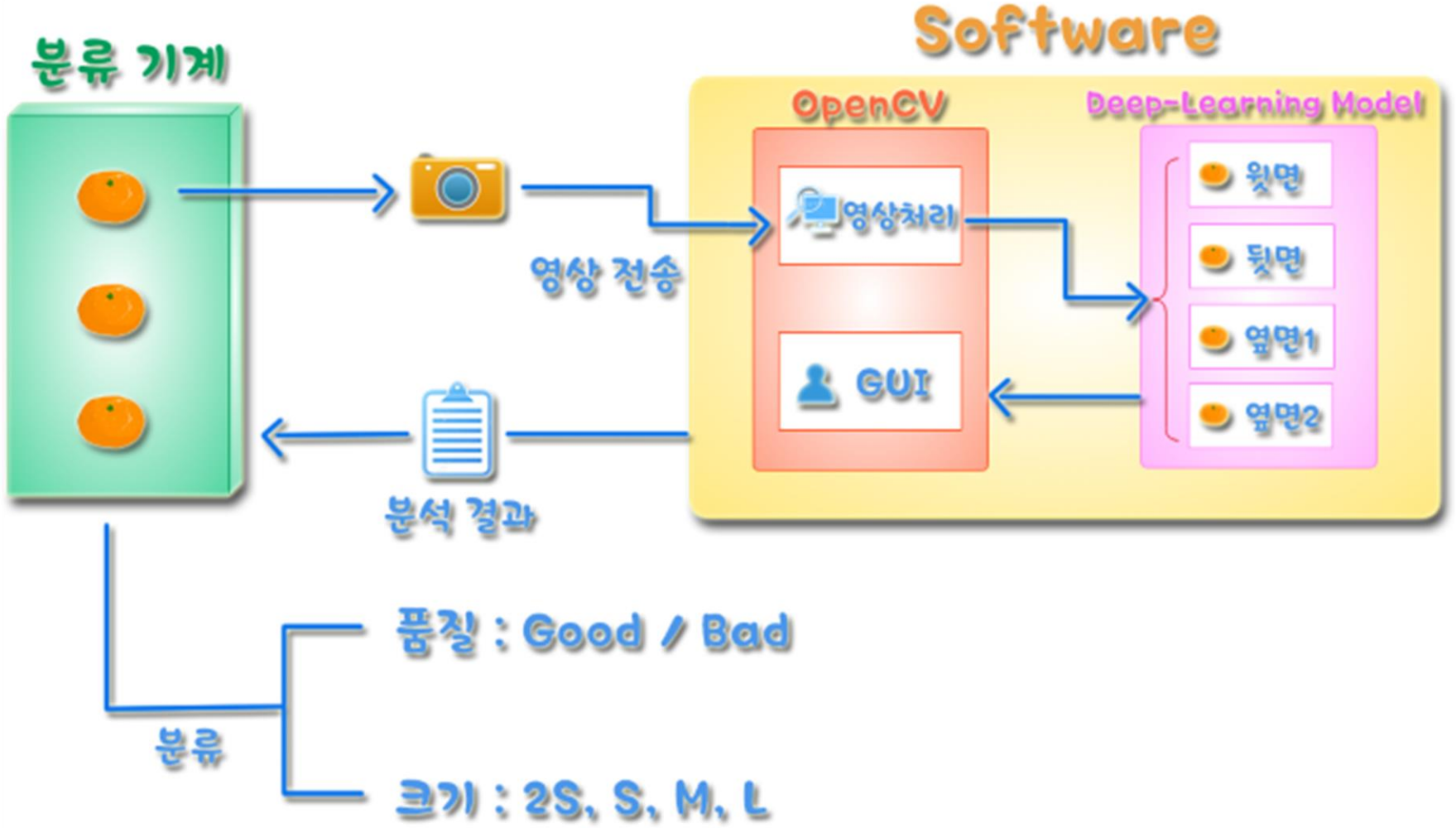
Good



Bad



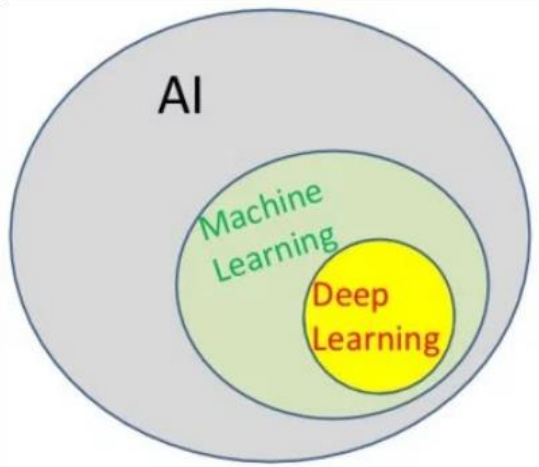
동작 과정 요약


분류 기계
Software
OpenCV
Deep-Learning Model
영상처리
GUI
윗면
뒷면
옆면1
옆면2
영상 전송
분석 결과
분류
품질 : Good / Bad
크기 : 2S, S, M, L

2. 핵심 개념

(1)Machine Learning

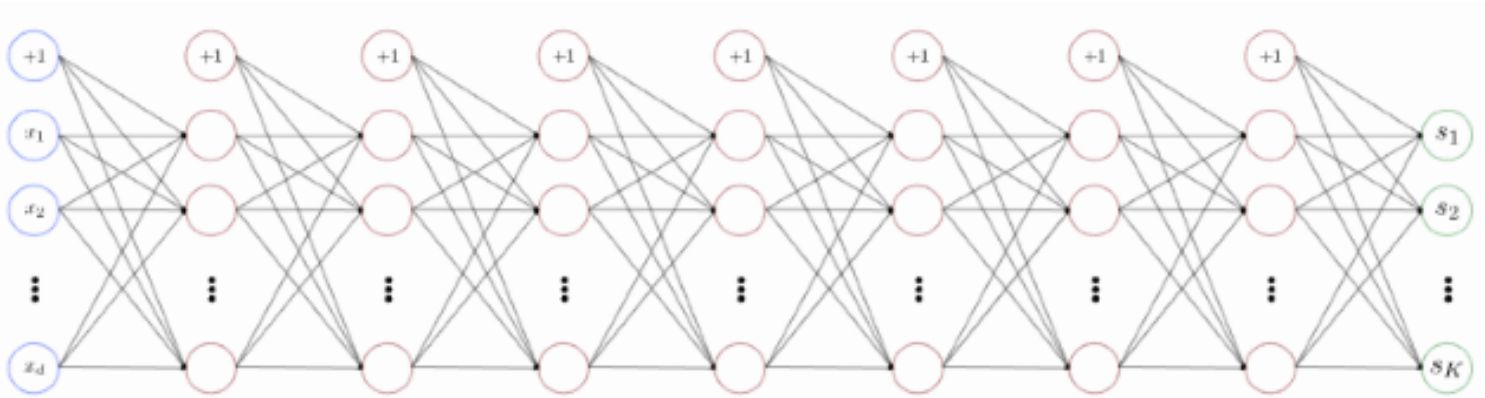
경험적 데이터를 통해 **컴퓨터 스스로** 새로운 지식과 능력을 개발
학습된 알고리즘으로 **새로운 데이터를 처리**하는 데 중점



AI, ML, DL의 관계

(2) Deep Learning

인공신경망(Neural Network) 알고리즘을 사용하는 머신 러닝의 한 부류
주로 **이미지 인식, 음성 인식** 등의 분야에서 사용



심층 신경망 예시

딥 러닝으로 문제 해결하는 과정

① 어떤 문제를 해결할 것인가 결정
예) 귤의 상태를 분류한다.

② Dataset 모으기
예) 정상 귤 사진과 상한 귤 사진을 6천 장 모았다.

③ Dataset을 train, validation, test set으로 나눈다.

④ 딥 러닝 모델 만들기
예) keras에서 모델 만들기(layer 정의)

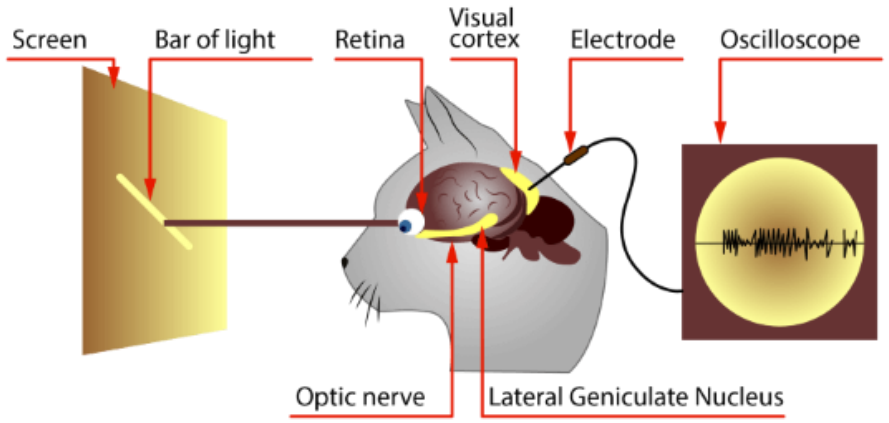
```
model = models.Sequential()
model.add(layers.Dense(256, activation='relu', input_dim=4 * 4 * 512))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

⑤ 모델에 Dataset을 훈련 시키고 테스트하기
예) keras에서 모델 훈련 시키기

```
model.compile(optimizer = optimizers.RMSprop(lr=1e-5),
              loss = 'categorical_crossentropy',
              metrics = ['acc'])

history = model.fit(train_features, train_labels,
                    epochs=30,
                    batch_size=20,
                    validation_data=(validation_features, validation_labels))
```

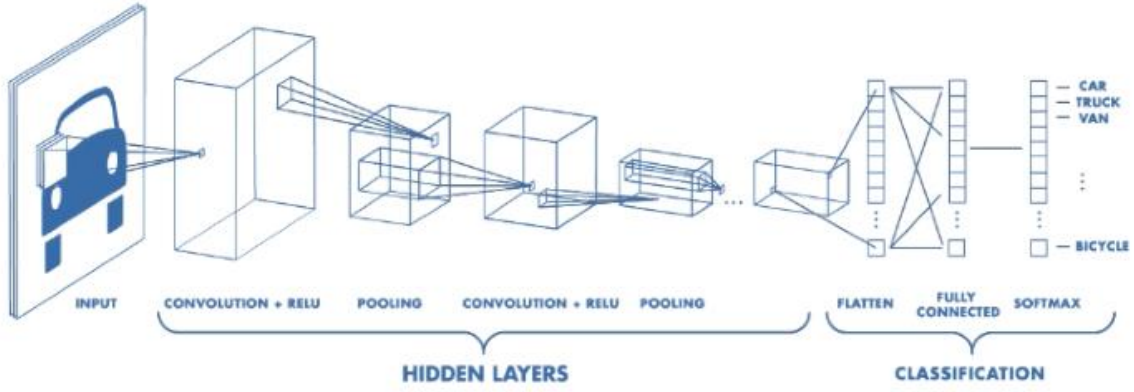
⑥ 훈련이 완료된 모델을 실전에 사용하기

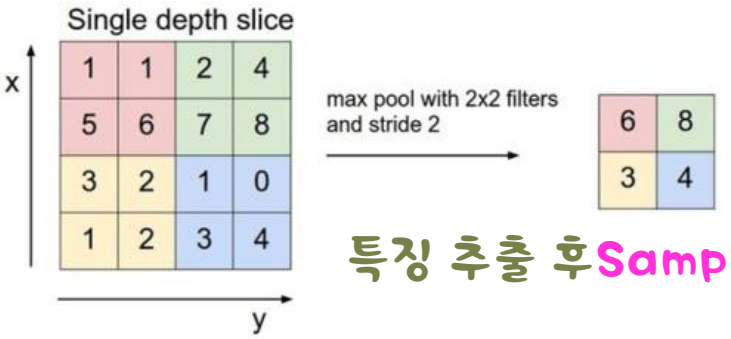
CNN (Convolution Neural Network)



사물을 인식할 때
특정 부분의 모양에 따라
반응하는 뉴런이 다르다

- Filter를 이용하여 이미지의 특징을 추출
- 대표적인 딥 러닝 모델





특징 추출 후Sampling 과정을 거친다

예) keras로 간단한 CNN 구현하기

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

3. 시스템 구현

(1)프로그래밍 언어

- 구조 및 문법이 쉬움
- 머신 러닝에 필요한 라이브러리 쉽게 사용 가능

→ 최근 머신 러닝에서 각광 받고 있음

(2) 프로그래밍 환경

구글이 연구 및 교육 목적으로
딥 러닝을 쉽게 할 수 있도록
제공한 서비스

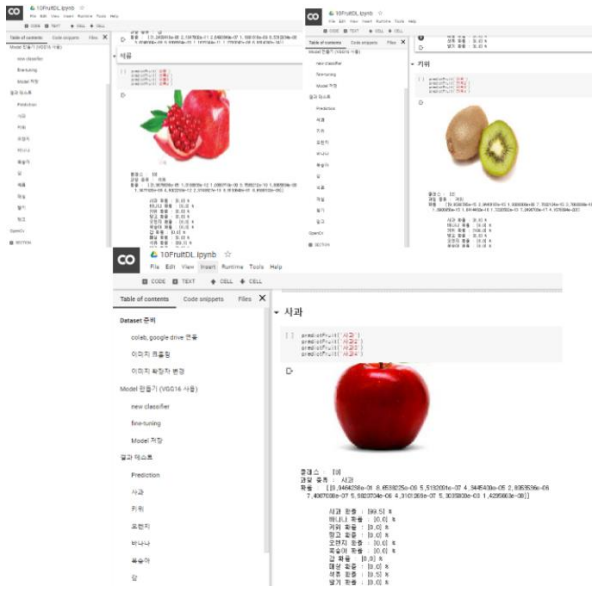
Python과 수치 해석 라이브러리를
쉽게 사용할 수 있게 해주는 플랫폼

(3) 라이브러리

Keras

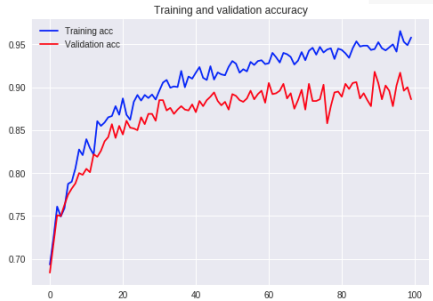
- Python 기반의 딥 러닝 라이브러리
- 간결하고 직관적으로 딥 러닝 모델 개발

Keras로 처음 구현한 딥 러닝 모델



- 사과, 바나나, 오렌지, 키위, 감, 망고, 석류, 복숭아, 매실, 딸기
- 바나나, 사과 ,오렌지를 가장 잘 분류함

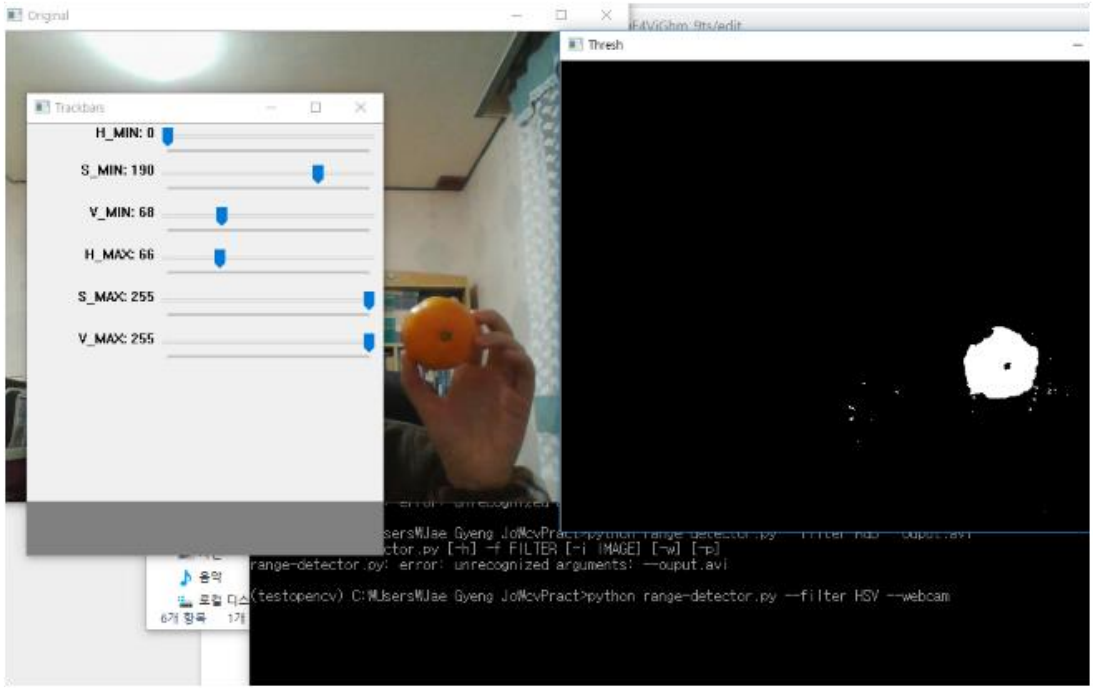
연습 모델 결과 (10가지 과일 분류)




```
[ ]  test_generator = test_datagen.flow_from_directory(
         test_dir,
         target_size=(150, 150),
         batch_size=20,
         class_mode='categorical')

test_loss, test_acc = model.evaluate_generator(test_generator, steps=50)
print('test acc:', test_acc)
```

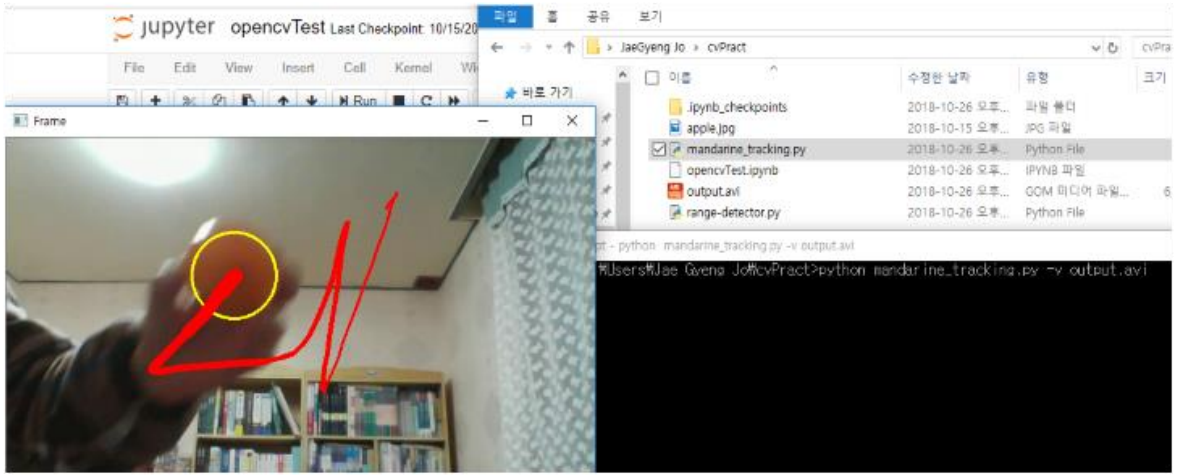
```
Found 7695 images belonging to 10 classes.
/usr/local/lib/python3.6/dist-packages/PIL/Image.py:872: UserWarning
  'to RGBA images')
test acc: 0.8750000011920929
```

1) 영상 처리



->Masking

- 실시간 이미지 프로세싱 오픈 소스 라이브러리

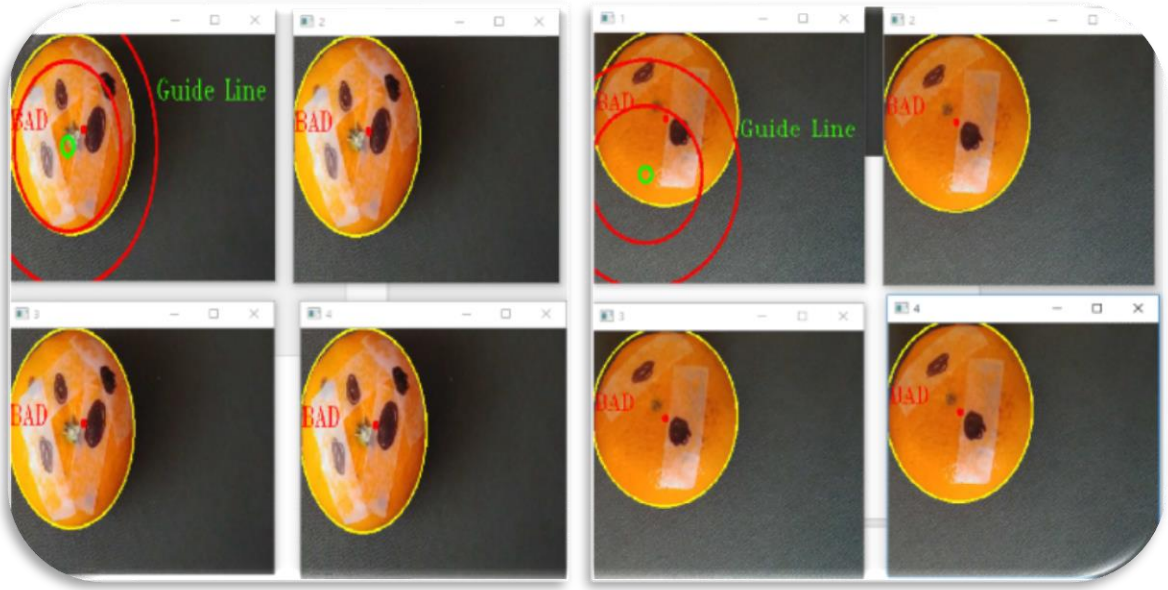


-> Tracking

- 귤을 tracking하고, 딥 러닝 모델이 분석할 이미지 데이터를 생성하기 위해서 사용

※기계가 없으므로

0.5초마다 4장의 귤 이미지를 자동으로 캡쳐

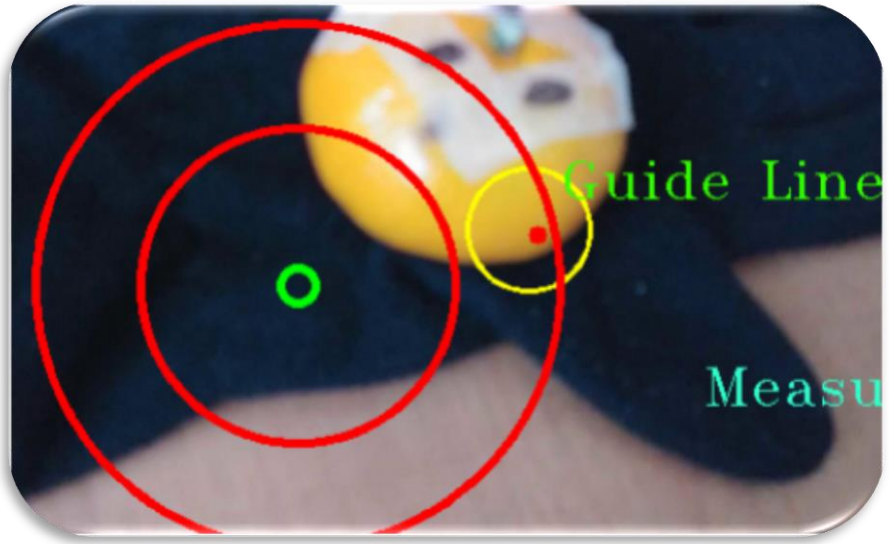


→ 딥 러닝 모델이 각각 분석하여
모두 GOOD 판정을 받았을 때만
정상 귤로 분류

귤이 선택된 영역의 반지름으로
귤의 크기를 판단

2) GUI

실행 결과를 보여주기 위해 GUI를 구성



Guide Line:
카메라의 위치에 따라
귤 크기가 다르게
인식되는 것을 방지

안내 및 결과 이미지:
설명, 진행 상황, 결과 보고

(4) 딥 러닝 모델

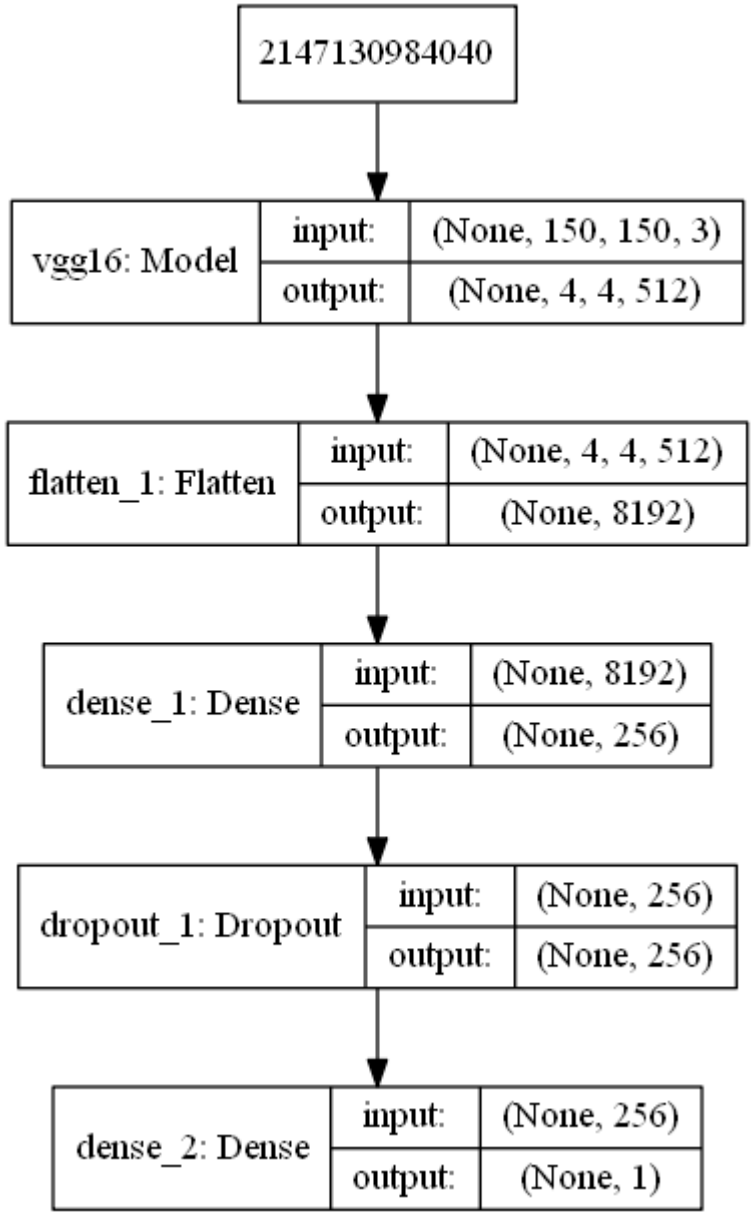
- Binary Classification
  -> Good/Bad

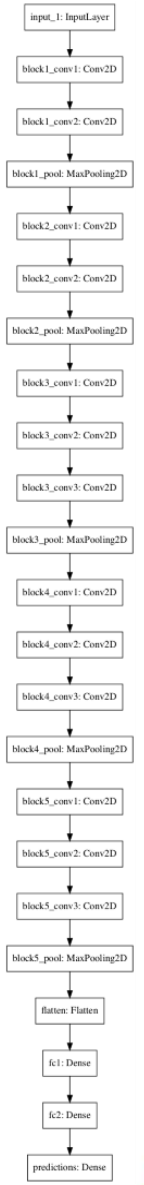
- Sequential model 구조

- Pre-trained model 사용
  (VGG16-CNN기반)

  -> 이미 훈련되어 있어
  사물의 특성 잘 찾아냄

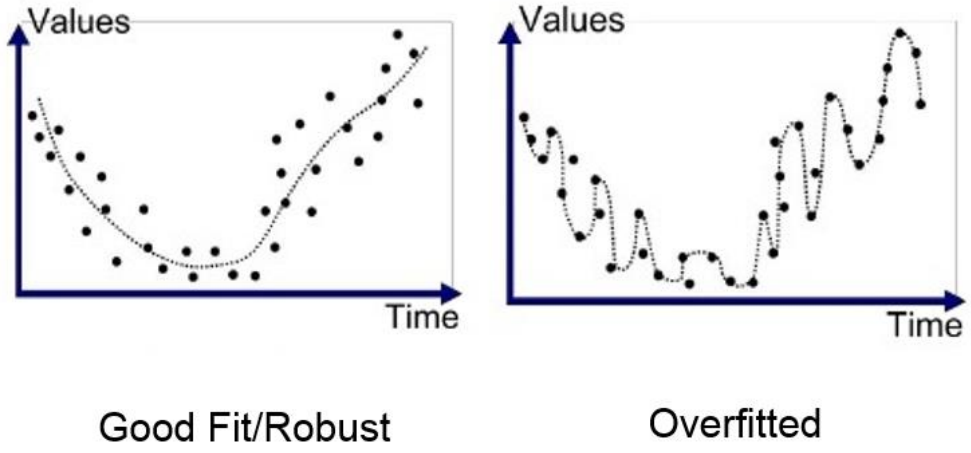
  -> 적은 데이터셋으로 처음부터
  훈련 시키는 것은 힘들기 때문에 사용

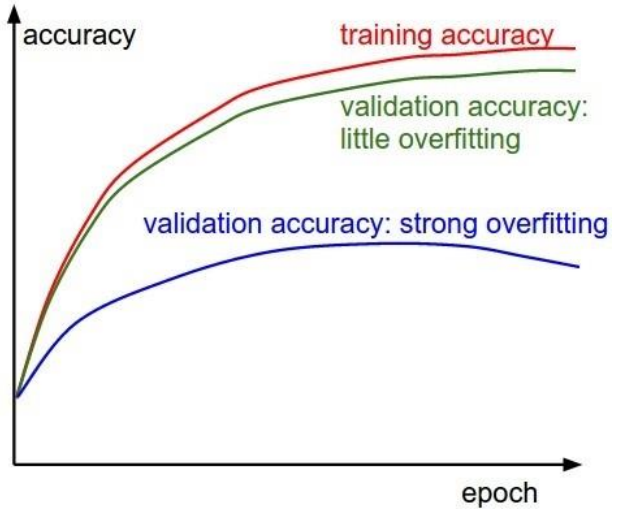




모델 훈련 과정

- 훈련 데이터가 적을 때,
  모델이 훈련하면서 데이터를 모두 외워 성능 저하가 발생하는
  Overfitting 문제를 줄이도록 훈련





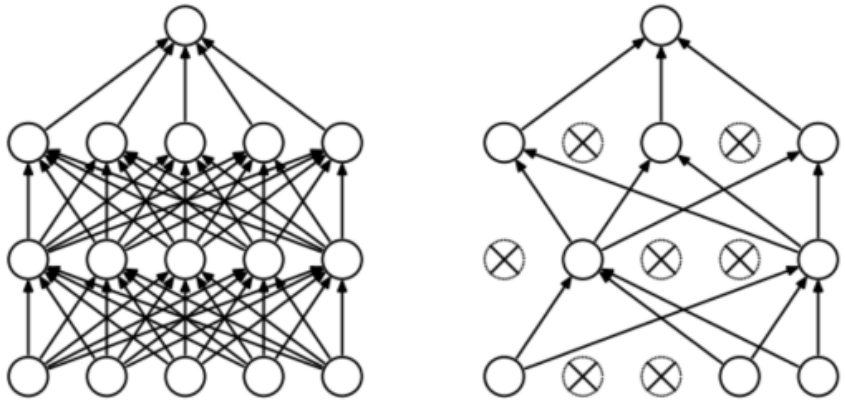
- VGG16와 다른 pre-trained model들을 이용하여 훈련

-> 가장 성능이 좋은 모델을 선정하여 선과 시스템에 사용함

Overfitting

- 훈련 데이터를 여러 번 반복 학습을 하고 나면
  어느 시점부터 성능이 높아지지 않고 감소
- Overfitting 될수록 새로운 데이터에 취약
- Overfitting 완화 기법

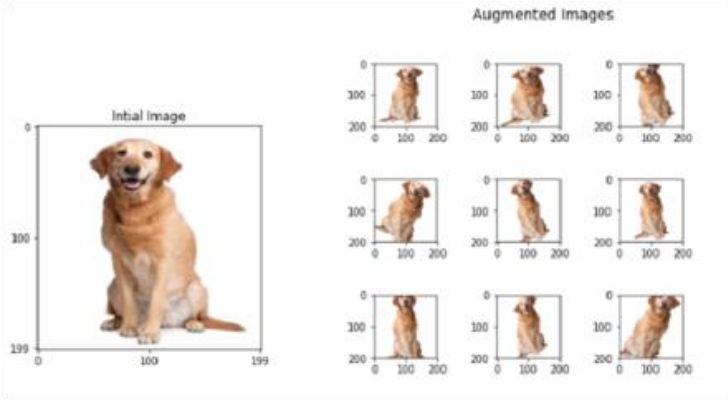
Dropout



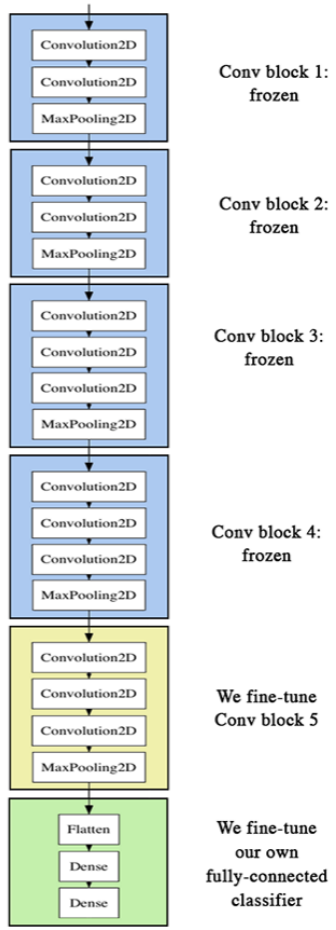
(a) Standard Neural Net

(b) After applying dropout.

Data Augmentation



Fine-Tuning

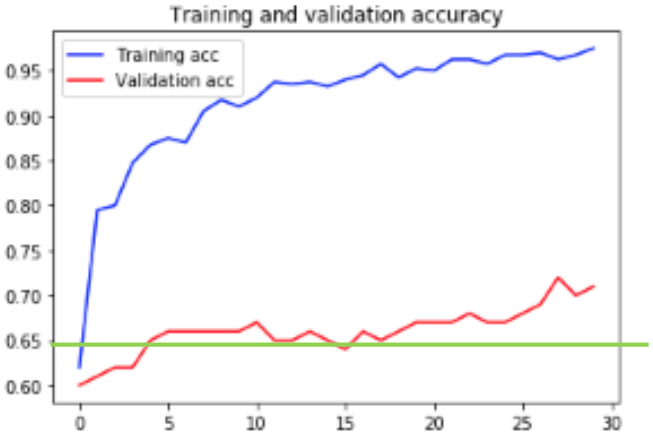


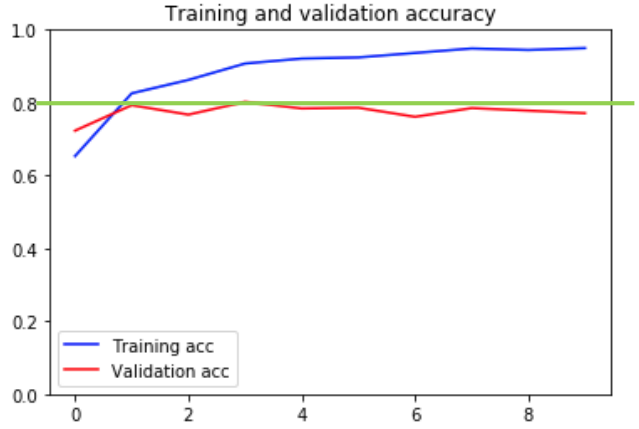
Overfitting 기법에 따른 모델 훈련 결과

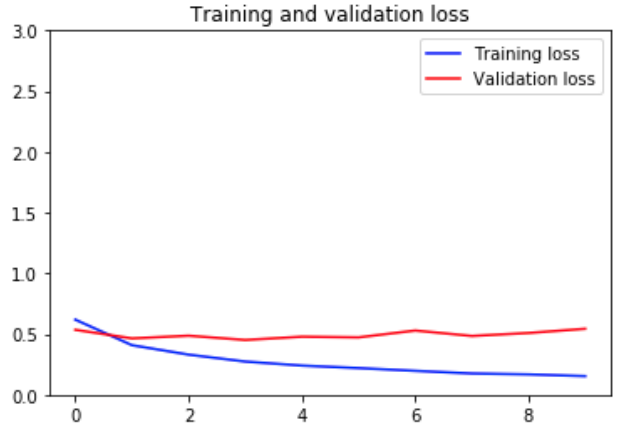
1)Dropout layer

```
model = models.Sequential()
model.add(layers.Dense(256, activation='relu', input_dim=4 * 4 * 512))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

훈련 데이터
500장





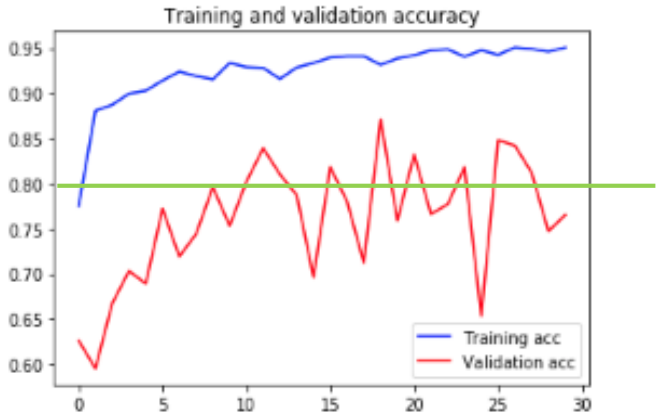


훈련 데이터
2000장

2) Data augmentation

```
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=20,
      width_shift_range=0.1,
      height_shift_range=0.1,
      shear_range=0.1,
      zoom_range=0.1,
      horizontal_flip=True,
      fill_mode='nearest')
```

```
model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

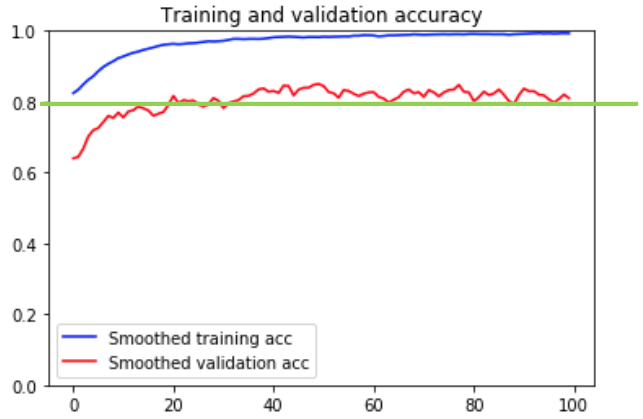


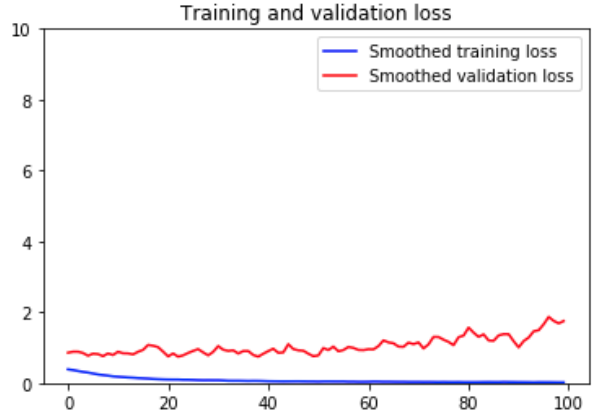
3) Fine-Tuning

```
conv_base.trainable = True

set_trainable = False
for layer in conv_base.layers:
    if layer.name == 'block5_conv1':
        set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False
```

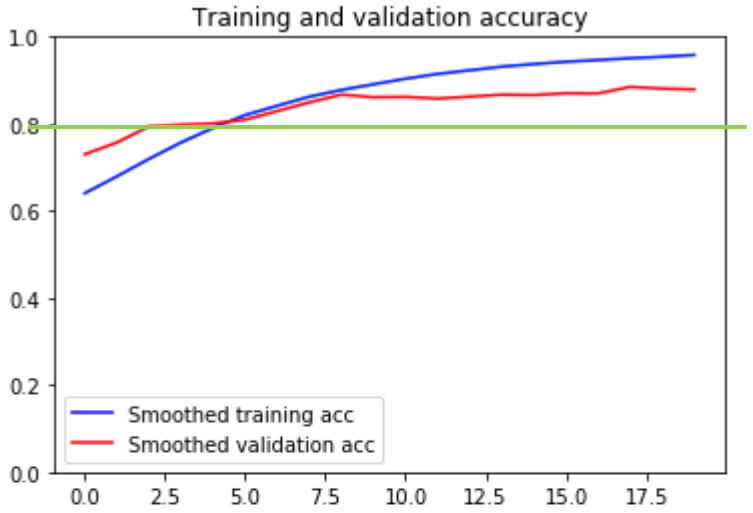
```
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(150, 150),
    batch_size=20,
    class_mode='binary')

test_loss, test_acc = model.evaluate_generator(tes
print('test acc:', test_acc)
```

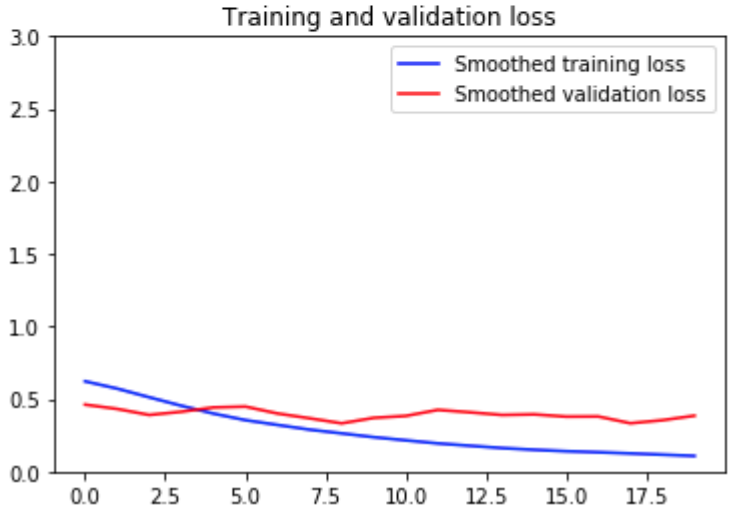
```
Found 1000 images belonging to 2 classes.
test acc: 0.7930000019073487
```





4) Data augmentation, Dropout, L2 regularization






```
model.summary()
```

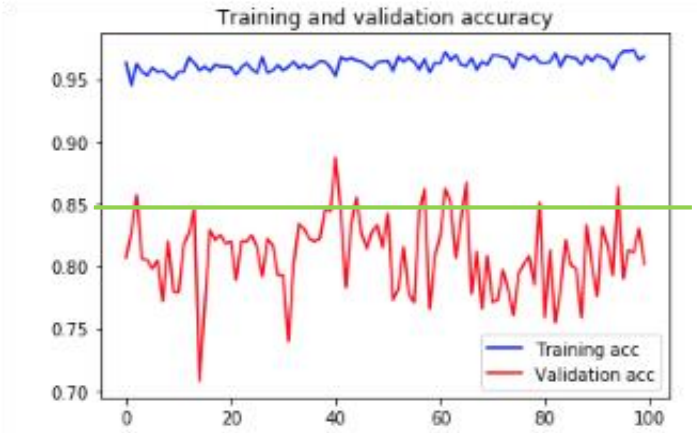
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               2097408
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 16,812,353
Non-trainable params: 0
_________________________________________________________________
```

In [12]:
```
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(150, 150),
    batch_size=20,
    class_mode='binary')

test_loss, test_acc = model.evaluate_generator(test_generator, steps=50)
print('test acc:', test_acc)
```

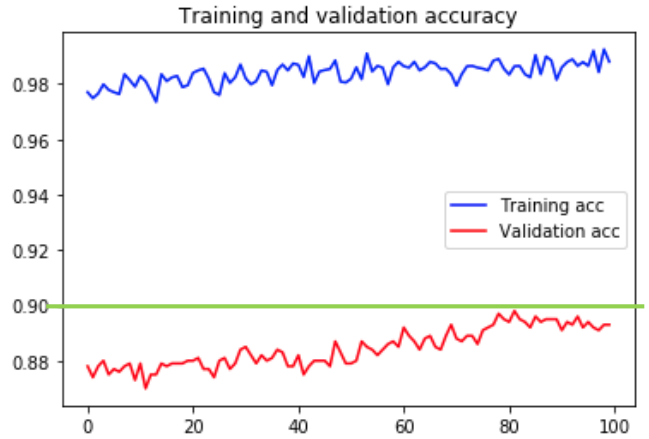
```
Found 1000 images belonging to 2 classes.
test acc: 0.8710000026226044
```

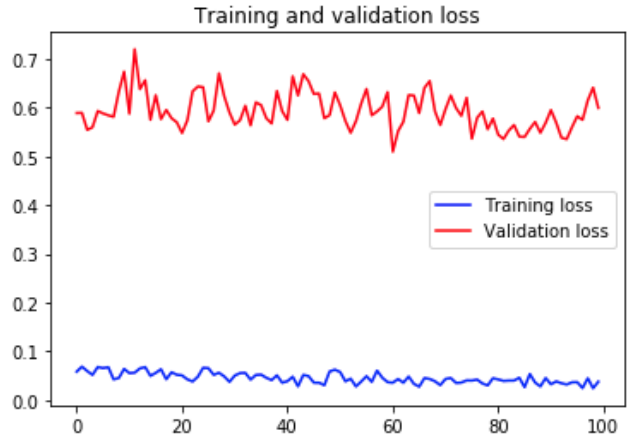
다른 pre-trained model 사용 결과

InceptionResNetV2



MobileNet





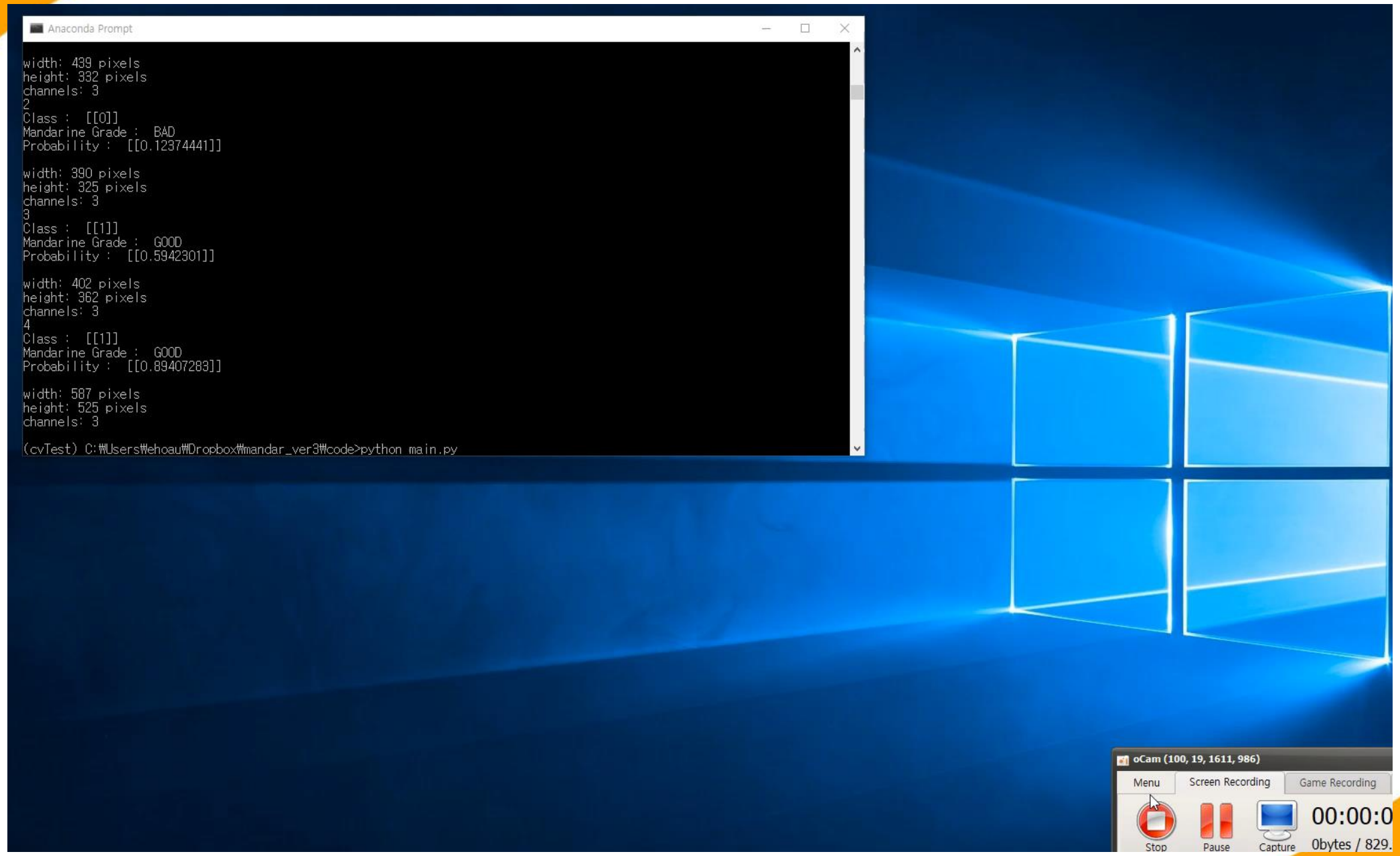
성능은 VGG16와 비슷하나 모델 로드가 느리고
이미지 처리가 오래 걸리므로 실시간 분석에 적합하지 않음

(5)구현 결과

클래스 : [[0]]
귤 등급 : BAD
확률 : [[12.74729]]

클래스 : [[0]]
귤 등급 : BAD
확률 : [[2.1010294]]

클래스 : [[1]]
귤 등급 : GOOD
확률 : [[99.53425]]


Anaconda Prompt
width: 439 pixels
height: 332 pixels
channels: 3
2
Class : [[0]]
Mandarine Grade : BAD
Probability : [[0.12374441]]
width: 390 pixels
height: 325 pixels
channels: 3
3
Class : [[1]]
Mandarine Grade : GOOD
Probability : [[0.5942301]]
width: 402 pixels
height: 362 pixels
channels: 3
4
Class : [[1]]
Mandarine Grade : GOOD
Probability : [[0.89407283]]
width: 587 pixels
height: 525 pixels
channels: 3
(cvTest) C:₩Users₩ehoau₩Dropbox₩mandar_ver3₩code>python main.py
oCam (100, 19, 1611, 986)
Menu
Screen Recording
Game Recording
Stop
Pause
Capture
00:00:0
0bytes / 829.

4.결론

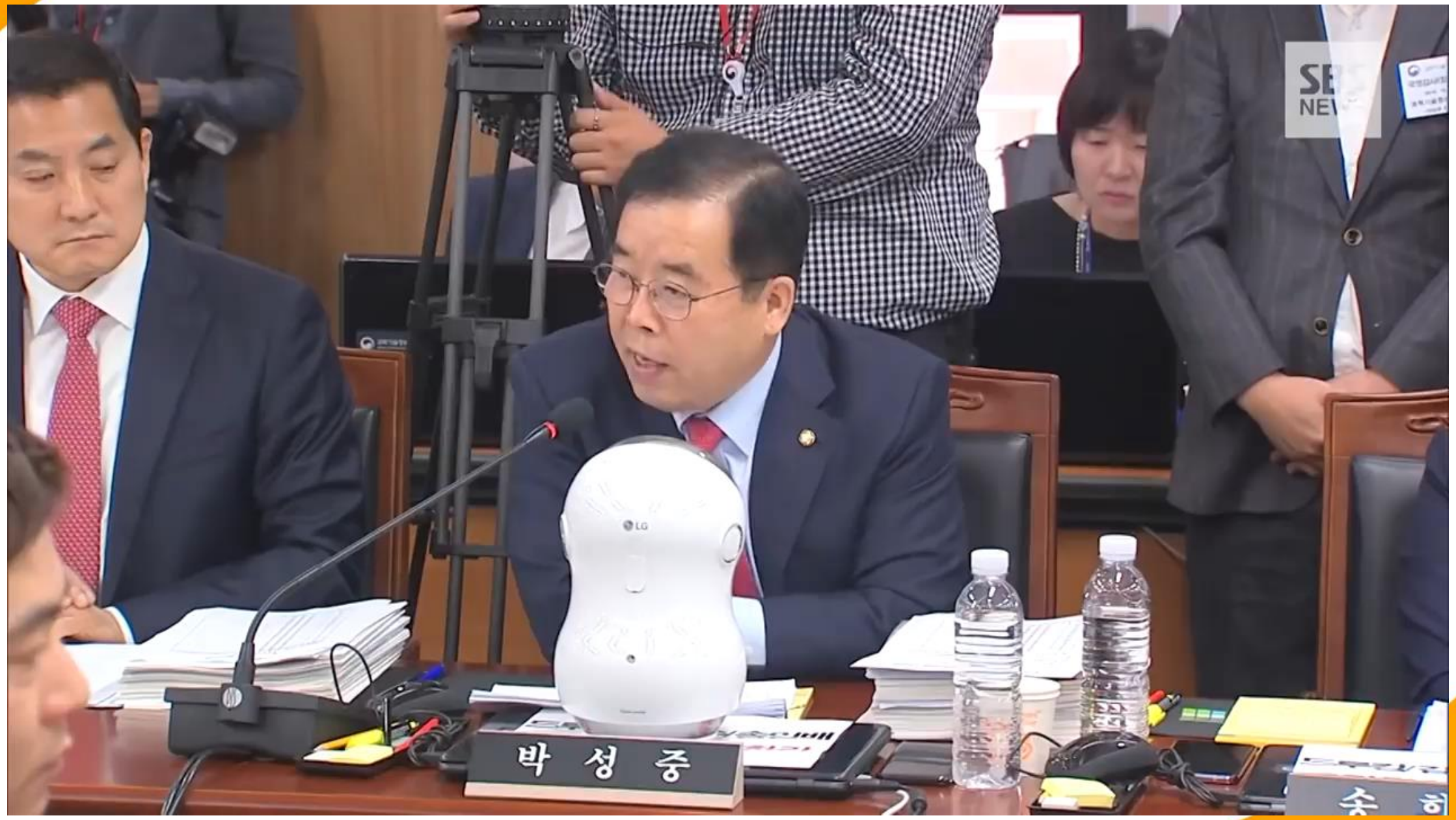
- 6천장 정도의 적은 데이터셋 사용

  -> 80% 이상의 정확도

  -> 데이터를 더 모을수록 정확도가 더 높아질 것

  => 정확도를 개선하면 실제 농가에서도 사용 가능할 것임

※ 정확도를 높이려면 다양한 변수를 고려하여 데이터를 모아야 할 것임



- 재사용성이 높은 모델

  -> 훈련 데이터만 다른 과일로 바꿔주면 됨

- 컴퓨터만 있으면 귤 분류 가능

  -> Raspberry Pi 같은 값싼 하드웨어로 만든 기계와 결합 가능

  => 적은 투자 비용으로 인건비를 줄일 수 있음

- 한 번에 귤의 4면을 분석하므로 사람보다 정확하게 분류 가능


Q&A


Thank
YOU