토마토 숙성도 이미지 분류

양한솔
류경아
이송현
김수환

1. 개요

스마트팜과 같이 농업에도 인공지능이 사용되고 있습니다.
특히, 자율재배 부분에서는 3D카메라와 인공지능으로 생육 상태 데이터를 수집하여 분석에 활용한다고 합니다.
익지 않은 토마토는 독성이 존재한다고 하여 재배할 때 특히 더 유의해야 한다고 생각했습니다.
그래서 저희 조는 토마토의 이미지를 수집해서 숙성 여부를 분류하는 딥러닝 모델을 만들기로 하였습니다.

2. 이미지 수집

```
from icrawler.builtin import GoogleImageCrawler
# import argparse

print("₩nGoogle Image Crawler₩n----------------------------------₩nVer1 By Zikx₩n----------------------
word = input("Search : ")
dir_name = input("dir name :")
# parser = argparse.ArgumentParser()
# parser.add_argument("-search", "--searchImages",nargs='*', required=True)

# args = parser.parse_args()

# searchImages = args.searchImages

def main():
    google_crawler = GoogleImageCrawler(
        feeder_threads=1,
        parser_threads=1,
        downloader_threads=4,
        storage={'root_dir': dir_name})

    google_crawler.crawl(keyword=word, offset=0, max_num=1000,
                         min_size=(200,200), max_size=None, file_idx_offset=0)


if __name__=="__main__":
    main()
```

3. Cnn 모델 만들기

(1) 모델 만들기

train -> 800개 , test -> 294개
층 4개(3X3 convolution, 2X2 pooling, 활성화 함수는 relu)
첫 번째 층 필터 32개
두 번째 층 필터 64개
세 번째 층 필터 128개
네 번째 층 필터 256개(fully connect layer)

(1) 모델 만들기

```
W3 = tf.Variable(tf.random_normal([3,3,64, 128], stddev=0.01))
L3 = tf.nn.conv2d(L2, W3, strides=[1,1,1,1], padding='SAME')
L3 = tf.nn.relu(L3)
L3 = tf.nn.max_pool(L3, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')
L3 = tf.nn.dropout(L3, keep_prob)

print(L3)
```

```
Tensor("dropout_2/mul_1:0", shape=(?, 4, 4, 128), dtype=float32)
```

```
W4 = tf.Variable(tf.random_normal([4 * 4 * 128, 256], stddev=0.01))
L4 = tf.reshape(L3, [-1, 4 *4 * 128])
L4 = tf.matmul(L4, W4)
L4 = tf.nn.relu(L4)
print(L4)
```

```
Tensor("Relu_3:0", shape=(?, 256), dtype=float32)
```

```
W5 = tf.Variable(tf.random_normal([256,2], stddev=0.01))
model = tf.matmul(L4, W5)
model
```

```
<tf.Tensor 'MatMul_1:0' shape=(?, 2) dtype=float32>
```

```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=model, labels=Y))
optimizer = tf.train.AdamOptimizer(0.001).minimize(cost)
```

(2) 학습시키기

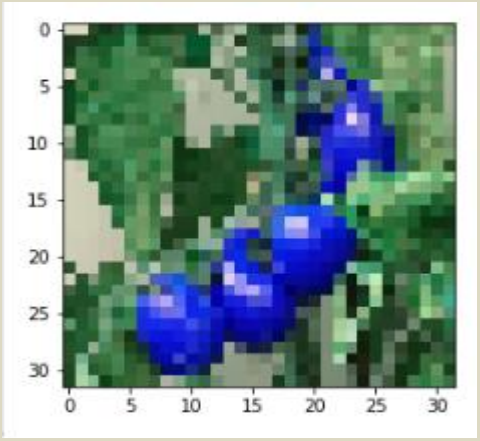
에폭 수20
정확도 90% 이상

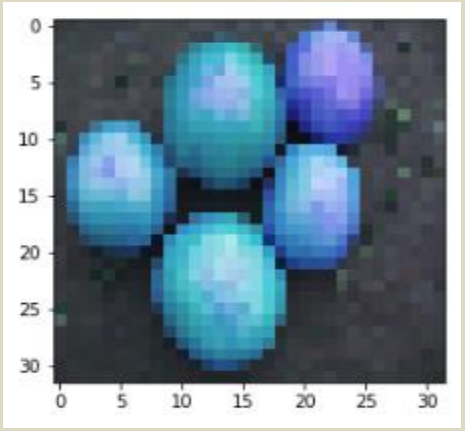
```
전체 입력 데이터 : (880, 32, 32, 3)
전체 출력 데이터 : (880, 2)
data_step = 0, Avg. cost = 0.693
data_step = 7, Avg. cost = 0.591
epoch: 0 total.cost = 5.336
data_step = 0, Avg. cost = 0.527
data_step = 7, Avg. cost = 0.235
epoch: 1 total.cost = 3.017
data_step = 0, Avg. cost = 0.195
data_step = 7, Avg. cost = 0.112
epoch: 2 total.cost = 1.243
data_step = 0, Avg. cost = 0.045
data_step = 7, Avg. cost = 0.023
epoch: 3 total.cost = 0.596
data_step = 0, Avg. cost = 0.029
data_step = 7, Avg. cost = 0.064
epoch: 4 total.cost = 0.471
data_step = 0, Avg. cost = 0.012
data_step = 7, Avg. cost = 0.096
epoch: 5 total.cost = 0.444
data_step = 0, Avg. cost = 0.017
data_step = 7, Avg. cost = 0.014
epoch: 6 total.cost = 0.554
data_step = 0, Avg. cost = 0.021
data_step = 7, Avg. cost = 0.063
epoch: 7 total.cost = 0.848
data_step = 0, Avg. cost = 0.043
data_step = 7, Avg. cost = 0.015
epoch: 8 total.cost = 0.355
data_step = 0, Avg. cost = 0.012
data_step = 7, Avg. cost = 0.004
epoch: 9 total.cost = 0.382
data_step = 0, Avg. cost = 0.004
data_step = 7, Avg. cost = 0.006
epoch: 10 total.cost = 0.579
data_step = 0, Avg. cost = 0.005
data_step = 7, Avg. cost = 0.014
```

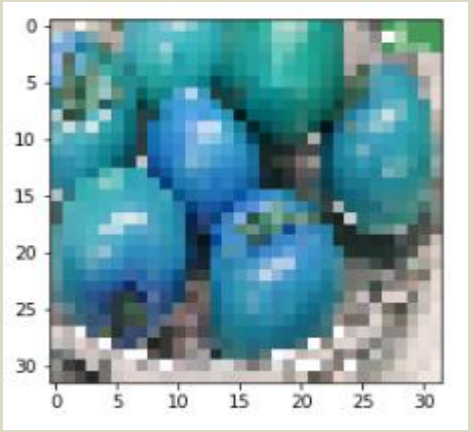
```
epoch: 11 total.cost = 0.197
data_step = 0, Avg. cost = 0.008
data_step = 7, Avg. cost = 0.005
epoch: 12 total.cost = 0.278
data_step = 0, Avg. cost = 0.003
data_step = 7, Avg. cost = 0.008
epoch: 13 total.cost = 0.171
data_step = 0, Avg. cost = 0.003
data_step = 7, Avg. cost = 0.006
epoch: 14 total.cost = 0.207
data_step = 0, Avg. cost = 0.002
data_step = 7, Avg. cost = 0.006
epoch: 15 total.cost = 0.228
data_step = 0, Avg. cost = 0.002
data_step = 7, Avg. cost = 0.007
epoch: 16 total.cost = 0.150
data_step = 0, Avg. cost = 0.002
data_step = 7, Avg. cost = 0.005
epoch: 17 total.cost = 0.176
data_step = 0, Avg. cost = 0.002
data_step = 7, Avg. cost = 0.006
epoch: 18 total.cost = 0.171
data_step = 0, Avg. cost = 0.001
data_step = 7, Avg. cost = 0.007
epoch: 19 total.cost = 0.149
```

```
f.argmax(Y,1))
t, tf.float32))

.reshape(-1,32,32,3),

:0.8}))
```

(3) 예측해보기







예측 레이블 : red

예측 레이블 : green

예측 레이블 : green

"익은 토마토"

"안 익은 토마토"

"안 익은 토마토"