

소고기 마블링을 이용한 품질 평가

최종보고

2024.12

10조
김성빈 팀장
임찬근 조원
김영진 조원
박가은 조원

배경

OBJECTIVE

문제점

카메라를 이용해 소고기 단면 인식

1. 소고기 등급의 비 일관성과 평가 기준의 모호성 문제

장점

1. 데이터 기반의 정확한 판정
2. 인건비 및 유통비용 감소 기대 -> 소비자 가격 감소 기대

데이터 특징

- 소고기 부위 중 등심 부위의 단면만을 이용해 딥러닝
- 마블링 이미지 자체를 딥러닝 시켜 소고기의 등급 분류
- 1++, 1, 3등급의 다양한 데이터

최종 목표

OBJECTIVE

사용자

카메라를 이용해 소고기 단면 인식
기존의 소고기 이미지 데이터를 이용하여 등급 구별

최종 출력

1++, 1, 3 등으로 출력 및 각 라벨별 확률 출력

모델 특징

- CNN 모델 사용
- 옵티마이저로 Adam 사용

이슈보고

| 발견된 이슈 |

이슈 1

- 데이터셋의 신뢰도 문제
- 데이터 양의 문제
- 육안으로도 구분하기 힘들거나 그림자 때문에 학습에 지장이 있을 수 있음.

이슈 2

- 신뢰도가 60%~70%수준

| 개선 방향 |

해결책 1

- 부적절한 데이터 제거
- 테스트 검증 데이터 비율 9:1
- 감안하고 데이터 입력

해결책 2

- 데이터 추가
- 뉴런 제거(Dropout) 축소
- 더 많은 레이어 쌓기

HOW?

이슈 해결과정

이슈



해결 과정

해결책 1

- 부적절한 데이터 제거-양이 많은 만큼 모두 하지는 못 했지만 어느정도 제거
- 약 2만장의 사진을 학습 데이터로 확보

해결책 2

- 데이터 추가
- 뉴런 제거(Dropout) 축소
- 더 많은 레이어 쌓기
- Epoch 수 추가



이슈 해결

PYTHON CODE

```
1 import tensorflow as tf
2 from tensorflow.keras.models import Model
3 from tensorflow.keras.layers import Dense, Flatten, Dropout, GlobalAveragePooling2D
4 from tensorflow.keras.applications import ResNet50
5 from tensorflow.keras.preprocessing.image import ImageDataGenerator
6 import cv2
7 import numpy as np
8 import os
9 import time
10
11 start_time = time.time()
12
13 # 데이터 경로 설정
14 base_dir = "C:\\Users\\kimyoungjin\\Desktop\\cow_datasets\\" # 데이터셋 폴더 경로 (수정 필요)
15
16 # 그레이스케일로 변환하는 전처리 함수
17 def preprocess_image(image):
18     gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
19
20     gray_image = np.stack((gray_image,) * 3, axis=-1)
21
22     gray_image = gray_image / 255.0
23     return gray_image
24
25 # 이미지 데이터 증강 및 로드
26 train_datagen = ImageDataGenerator(
27     rescale=1.0/255,
28     rotation_range=20,
29     width_shift_range=0.2,
30     height_shift_range=0.2,
31     shear_range=0.2,
32     zoom_range=0.2,
33     horizontal_flip=True,
34     validation_split=0.1,
35     preprocessing_function=preprocess_image
36 )
37
38 # 학습 데이터셋 로드
39 train_generator = train_datagen.flow_from_directory(
40     base_dir,
41     target_size=(128, 128), # 이미지 크기
42     batch_size=32,
43     class_mode='categorical',
44
45
46 # ResNet50 모델 불러오기 (사전 학습된 가중치 사용)
47 base_model = ResNet50(
48     weights='imagenet', # ImageNet 사전 학습 가중치
49     include_top=False, # 최상위 분류층 제외 (전이 학습에 적합)
50     input_shape=(128, 128, 3) # 입력 이미지 크기
51 )
52
53 # ResNet50 위에 사용자 정의 분류층 추가
54 x = base_model.output
55 x = GlobalAveragePooling2D()(x) # Global Average Pooling
56 x = Dense(128, activation='relu')(x)
57 x = Dropout(0.5)(x)
58 predictions = Dense(train_generator.num_classes, activation='softmax')(x)
59
60 model = Model(inputs=base_model.input, outputs=predictions)
61
62 # ResNet50의 기존 층을 동결 (사전 학습 가중치 유지)
63 for layer in base_model.layers:
64     layer.trainable = False
65
66 model.compile(
67     optimizer='adam',
68     loss='categorical_crossentropy',
69     metrics=['accuracy']
70 )
71
72 # 모델 학습
73 history = model.fit(
74     train_generator,
75     validation_data=validation_generator,
76     epochs=20,
77     steps_per_epoch=train_generator.samples // train_generator.batch_size,
78     validation_steps=validation_generator.samples // validation_generator.batch_size
79 )
80
81 # 모델 저장
82 model.save("cow_marbling_resnet_model_gray.h5")
83 print("Model saved as cow_marbling_resnet_model_gray.h5")
84 end_time = time.time()
85 print("Time elapsed: {:.2f} seconds".format(end_time - start_time))
```

```
Epoch 7/10
621/621 ----- 282s 454ms/step - accuracy: 0.6954 - loss:
0.6057 - val_accuracy: 0.6857 - val_loss: 0.6109
Epoch 8/10
621/621 ----- 0s 423us/step - accuracy: 0.7188 - loss:
0.7486 - val_accuracy: 0.5161 - val_loss: 0.8158
Epoch 9/10
621/621 ----- 593s 955ms/step - accuracy: 0.7028 - loss:
0.6021 - val_accuracy: 0.7151 - val_loss: 0.5576
```

- CNN모델 학습 및 데이터 증강
- 데이터 크기 128*128사이즈로 통일
- 3번에 걸친 ‘합성곱, 풀링’과 fully connected레이어
- ResNet50 사용

PYTHON CODE

```
test_loss, test_acc = model.evaluate(validation_generator)
print(f"Test accuracy: {test_acc}")
```

```
In [4]: runcell(0, 'C:/Users/kimyoungjin/Desktop/24-2/deeplearning/test
accuracy.py')
69/69 ————— 19s 282ms/step - accuracy: 0.6990 - loss:
0.5775
Test accuracy: 0.705482542514801
```

- 테스트 데이터를 이용해 다시 정확도 측정
- 최종 정확도 약 70%
- 데이터 수가 더 많다면 정확도가 올라갈 여지가 있음

PYTHON CODE

cow_marbling_resnet_model.h5

- 학습한 모델을 파일로 저장

```
1 import cv2
2 import numpy as np
3 from tensorflow.keras.models import load_model
4 from tensorflow.keras.preprocessing.image import img_to_array
5 import time
6
7 model = load_model(r"C:\Users\kimyoungjin\cow_marbling_resnet_model.h5") # h5 폴더 경로 (수정 필요)
8
9 camera = cv2.VideoCapture(0)
10 if not camera.isOpened():
11     print("카메라를 열 수 없습니다.")
12     exit()
13
14 print("카메라가 열렸습니다. 'q'를 눌러 종료하세요.")
15
16 last_prediction_time = 0
17 interval = 0.1 # 0.1초 간격으로 출력
18
19 while True:
20     ret, frame = camera.read()
21     if not ret:
22         print("프레임을 읽을 수 없습니다. 종료합니다.")
23         break
24     processed_frame = frame.copy()
25     # 이미지를 그레이스케일로 변환
26     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
27     # 밝은 영역 강조 (Threshold)
28     _, thresh = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY)
29     # 윤곽선 검출
30     contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
31
32     # 가장 큰 윤곽선만 선택
33     largest_contour = None
34     largest_area = 0
35
36     for contour in contours:
37         area = cv2.contourArea(contour)
38         if area > 500: # 노이즈 제거: 500 픽셀 이하 영역 제외
39             if area > largest_area: # 가장 큰 윤곽선 선택
40                 largest_area = area
41                 largest_contour = contour
42
43     if largest_contour is not None:
44         x, y, w, h = cv2.boundingRect(largest_contour)
45         cv2.rectangle(processed_frame, (x, y), (x + w, y + h), (0, 0, 255), 2) # 빨간색 사각형
46         roi = frame[y:y + h, x:x + w]
47         current_time = time.time()
48         if current_time - last_prediction_time >= interval:
49             last_prediction_time = current_time
50
51             resized_roi = cv2.resize(roi, (128, 128))
52             roi_array = np.expand_dims(img_to_array(resized_roi) / 255.0, axis=0)
53             predictions = model.predict(roi_array) # 모델 예측
54             class_idx = np.argmax(predictions[0])
55             if class_idx == 2: # 등급 출력
56                 grade = '3등급'
57             elif class_idx == 0:
58                 grade = '1등급'
59             else:
60                 grade = '1++등급'
61
62             print(f"Predicted class: {class_idx}")
63             print(f"Predicted probabilities: {predictions}")
64             print(f"등급: {grade}")
65             # 결과를 박스 위에 표시
66             cv2.putText(processed_frame, grade, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
67
68     cv2.imshow("Marbling Detection", processed_frame)
69     if cv2.waitKey(1) & 0xFF == ord('q'):
70         break
71
72 camera.release()
73 cv2.destroyAllWindows()
```

- 모델에 맞게 이미지 데이터 맞추기
- 라벨 출력, 라벨별 확률 출력, 분류
- 실시간 분류

PREDICTION

1++등급



```
Predicted class: 1  
Predicted probabilities: [[0.37061393 0.6252376 0.00414851]]  
1++등급
```

1등급



```
Predicted class: 0  
Predicted probabilities: [[0.5072679 0.45625317 0.03647896]]  
1등급
```

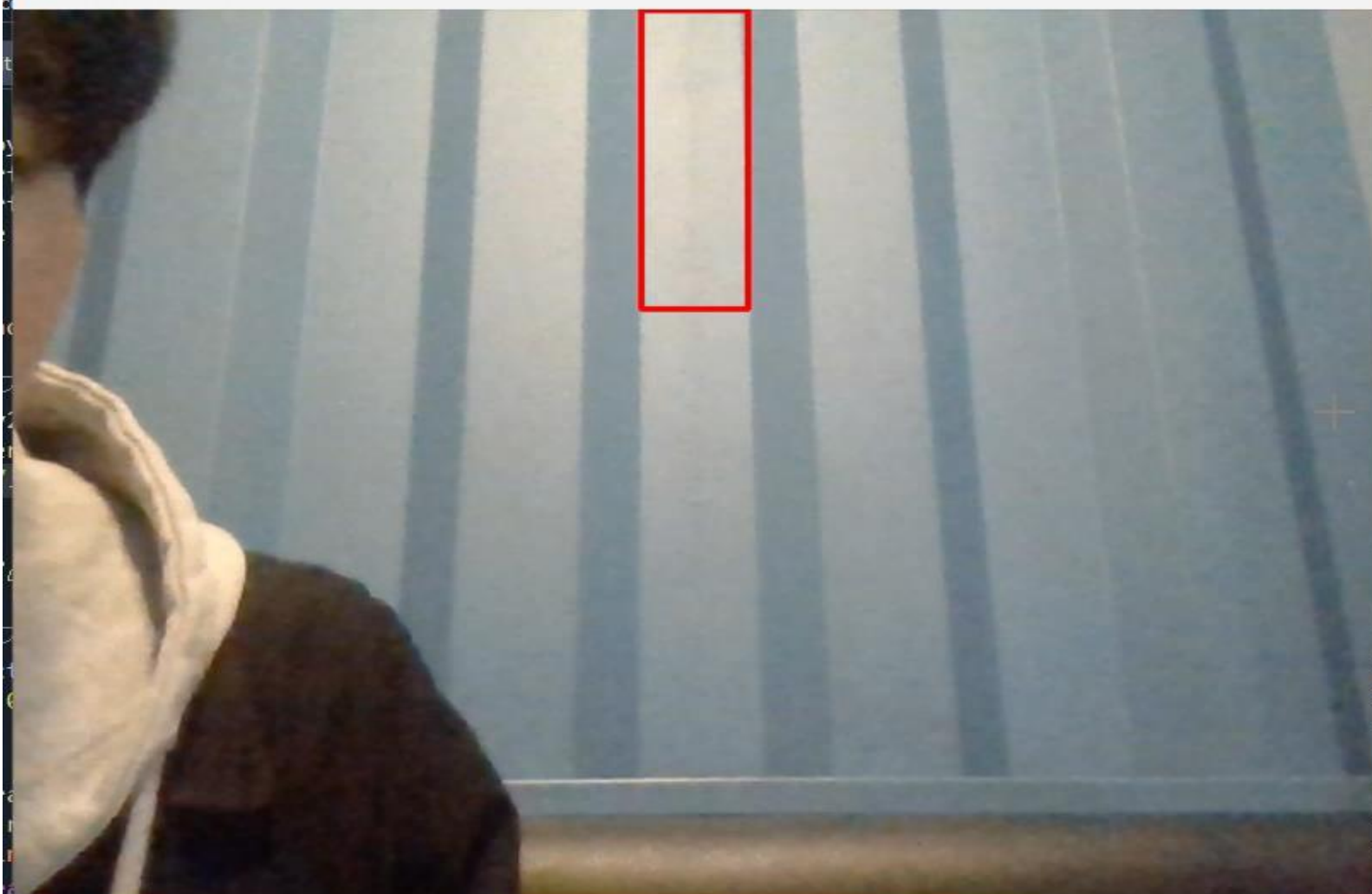
3등급



```
Predicted class: 2  
Predicted probabilities: [[1.7227841e-04 2.0559764e-06 9.9982578e-01]]  
3등급
```

VIDEO

Marbling Detection



```
프레임 복사
ed_frame = frame.copy()

이를 그레이스케일로 변환
cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

영역 강조 (Threshold)
sh = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY)

검출
```

Console 1/A

ected class: 1
ected probabilities: [[0.11281288 0.7269723 0.1602148]
1++등급
0s 97ms/step

ected class: 1
ected probabilities: [[0.11413166 0.7328572 0.15301117]
1++등급
0s 103ms/step

1/1
Predicted class: 1
Predicted probabilities: [[0.11426418 0.7384269 0.14730891]
등급: 1++등급
0s 94ms/step

1/1
Predicted class: 1
Predicted probabilities: [[0.11419198 0.7374691 0.14833891]
등급: 1++등급
0s 125ms/step

THANK YOU

들어주셔서 감사합니다.

THANK YOU.