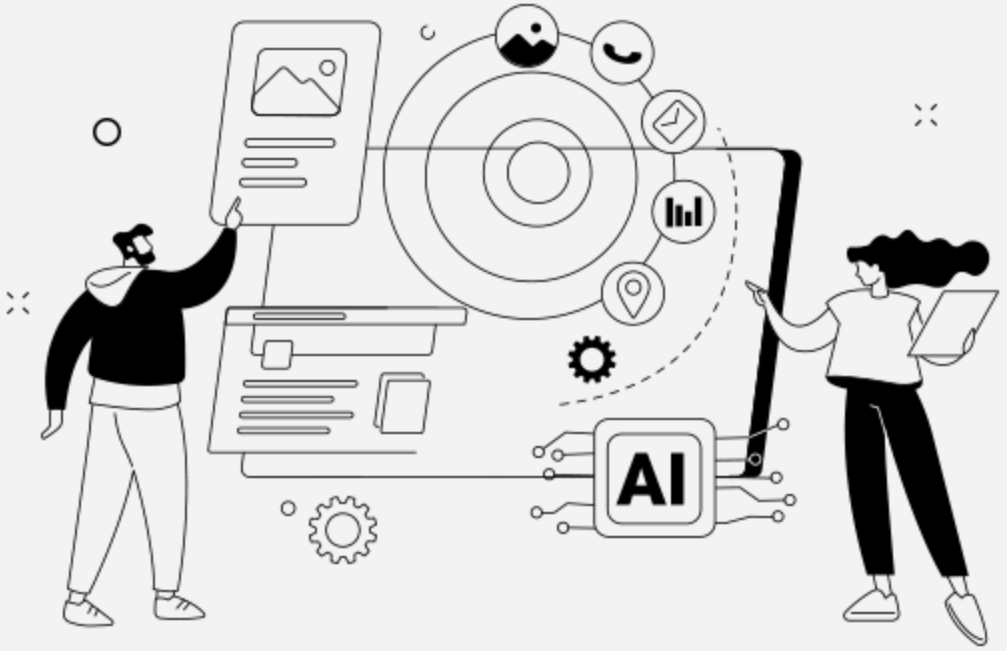
2022 데이터 크리에이터 캠프

Data Creator Camp



덕영고등학교 - 춘식이

과학기술정보통신부 NIA 한국지능정보사회진흥원

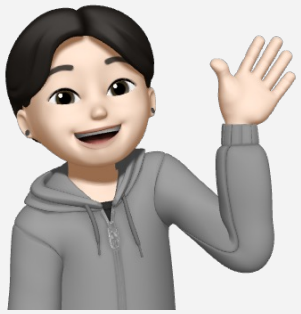
팀 소개

남우석

데이터 전처리

김재훈

모델 학습



안진영

미션 해결

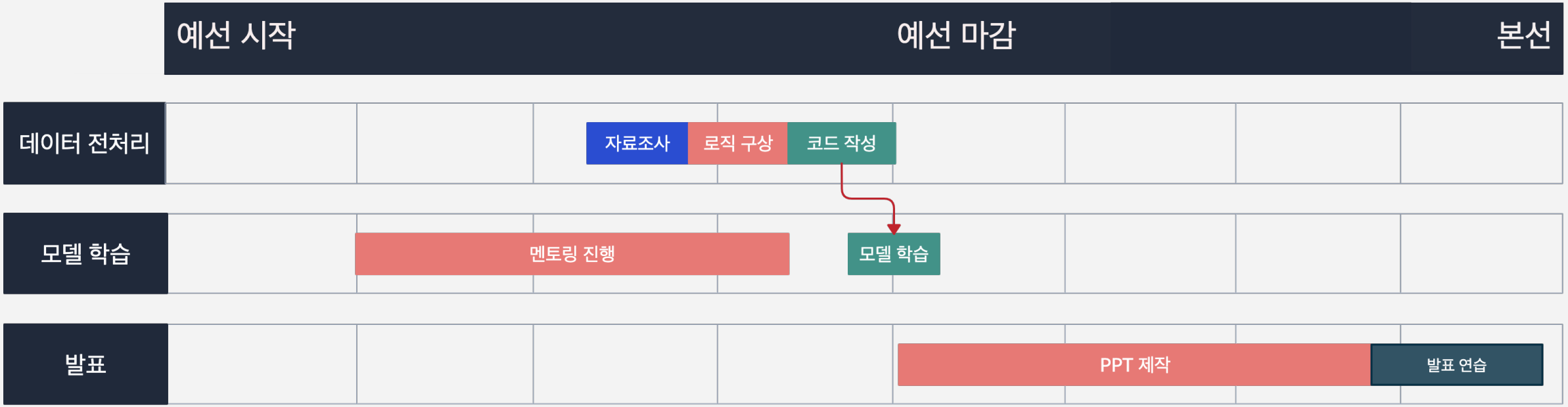
이채은

이슈 해결

전우진

QA

Workflow



위와 같은 워크플로우를 구축하여 학습 데이터셋을 만들고, 모델 학습 진행했습니다.

Mission 1

1-1. Training 데이터 셋의 데이터를 살펴보고 라벨 종류는 무엇이 있고, 각 라벨의 개수를 구하시오.

Answer:
라벨 종류는 **cap_and_hat, outerwear, tops, bottoms, shoes**로 총 5개이고,
라벨 별 이미지의 개수는 다음과 같다.

cap_and_hat: 196개,
outerwear: 4606개,
tops: 18350개,
bottoms: 6424개,
shoes: 424개

Mission 2

2-1. 이미지 크기를 적절히 조절하거나, 해상도를 조절하여 학습 데이터 셋을 구축하시오.

Answer:

모델의 정확도를 높이려면 데이터 셋에서 옷만 잘 잘라내는 것이 유리하다고 생각했습니다.

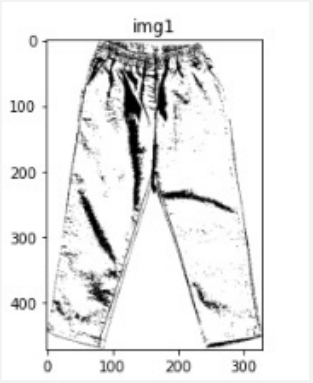
학습 데이터 셋을 구축하기 위해 **데이터 전처리를 진행**했습니다.

데이터 전처리 과정

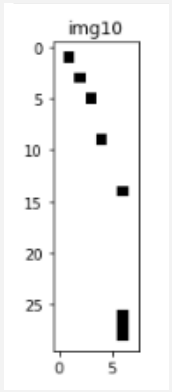
전처리 코드를 완성하기 위해 많은 시행착오를 거쳤습니다.

이미지의 모서리를 찾아내기 위해 **지역 이진화 함수**를 사용했습니다.
하지만 배경이 흰색인 탓에 흰 옷 이미지 이진화 시 문제가 생기는 것을 확인할 수 있었고,
어떻게 해결해야 할 지 고민하던 중 **적응형 이진화 함수**를 발견했습니다.
적응형 이진화는 이미지에 따라 스스로 임계 값을 다르게 할당할 수 있도록 구현된 알고리즘입니다.
이 알고리즘을 사용해 이진화를 했을 경우 흰 옷의 이진화가 알맞게 된 것을 확인할 수 있습니다.

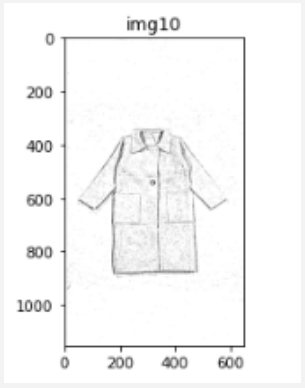
흰 옷 원본 이미지



Threshold 함수 사용
(갈색 옷)



Threshold 함수 사용
(흰 옷)

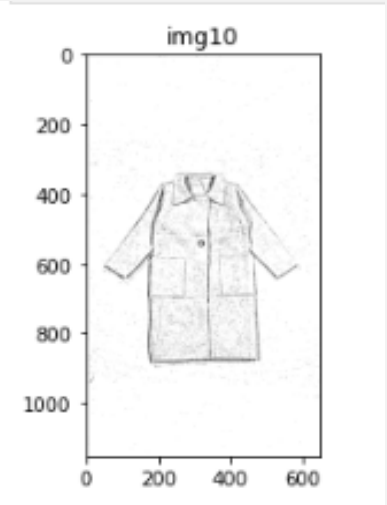


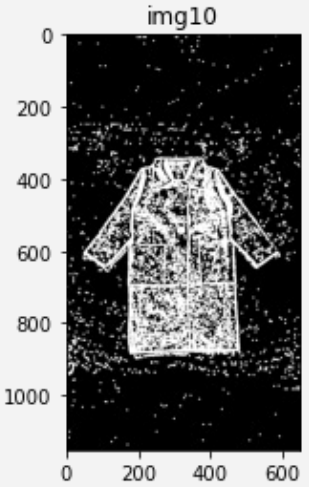
Adaptive Threshold 함수 사용
(흰 옷)

데이터 전처리 과정

다만 이미지가 제대로 crop 되지 않은 것을 확인할 수 있습니다.
이미지를 확인해보니 노이즈가 많아
경계 값을 제대로 도출하지 못한 것으로 보입니다.

그래서 이미지의 노이즈를 줄이는 방법을 생각해보게 되었습니다.



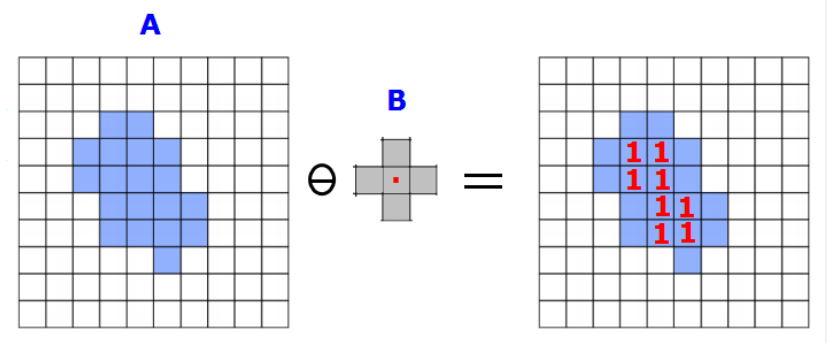


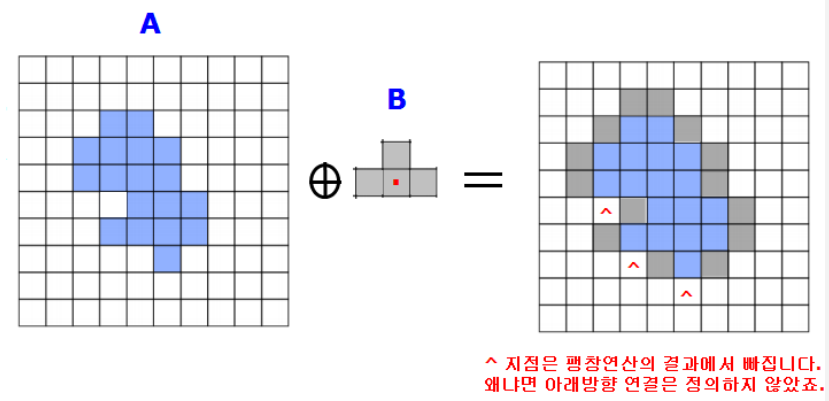
데이터 전처리 과정

처음은 **Morphology 연산**으로 시작했습니다.
Morphology 연산이란 '형태학' 이라는 뜻인데,
사진, 영상 분야에서의 노이즈 제거 등에 쓰이는 형태학적 연산을 말합니다.
대표적인 연산으로는 침식 연산과 팽창 연산이 있습니다.

침식은 말 그대로 이미지를 깎아 냅니다.
이미지를 0과 1로 전환하고, 커널을 생성하여
이 커널 안에 들어오지 못하는 이미지는 삭제해버리는 것입니다.

팽창은 침식의 반대입니다.
커널에 픽셀이 걸치기만 해도 1로 바꿔버리는 것입니다.





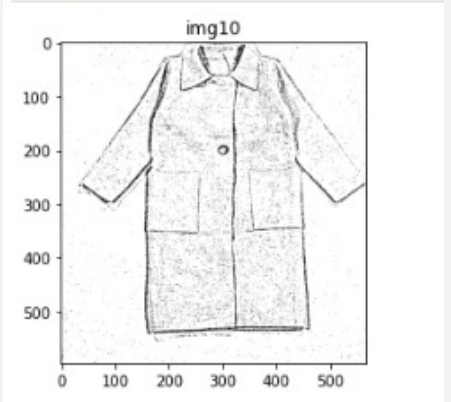
이미지 출처: https://bkshin.tistory.com

데이터 전처리 과정

저희는 Morphology 연산 중 **닫힘 연산**과 **Gradient 연산**을 사용하였습니다.

닫힘 연산은 팽창 연산 후 침식 연산을 적용하여 주변보다 어두운 노이즈를 제거하는데 효과적입니다.

Gradient 연산이란
팽창 연산을 적용한 이미지에서 침식 연산을 적용한 이미지를 빼면 경계 픽셀만 얻게 되는데
이 연산을 Gradient 연산이라고 합니다.



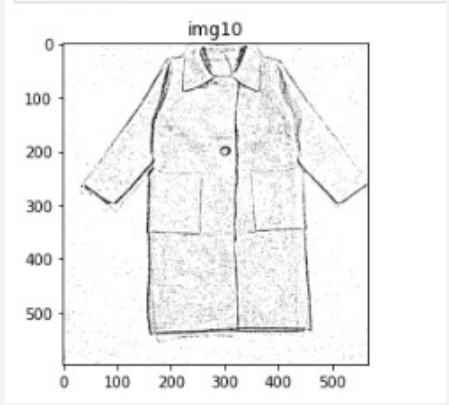
저희는 닫힘 연산을 사용하여 1차적으로 노이즈를 없애고
Gradient 연산을 사용하여 2차적으로 노이즈를 없애며 옷의 경계를 뚜렷하게 했습니다.

데이터 전처리 과정

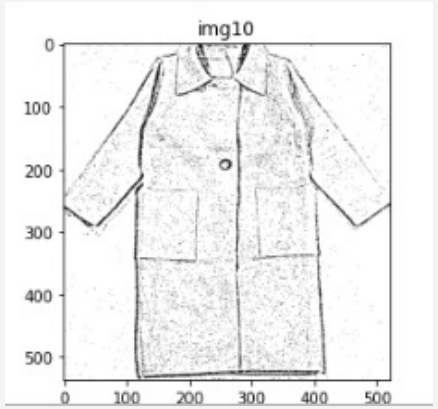
처음보단 많이 좋아졌지만 아직 여백이 있는 걸 확인할 수 있습니다.

노이즈의 형태를 보니 **소금 & 후추 노이즈**와 비슷하여
Median Blur 함수를 사용하여 Blur 처리를 하게 하였습니다.

결과를 보니, 옷만 Crop 된 것을 확인할 수 있습니다.



블러 처리 전



블러 처리 후

원본 이미지

최종 전처리 코드

2-1. 이미지 크기를 적절히 조절하거나, 해상도를 조절하여 학습 데이터 셋을 구축하시오.

```python
def crop(img_num):
    file = f"{BASE_DIR}/dataset/Item-Image/img{img_num}.jpg"
    img1 = cv2.imread(file)

    # Image Shape
    h, w = img1.shape[:2]
    h1, h2 = int(h * 0.2), int(h * 0.7)
    w1, w2 = int(w * 0.05), int(w * 0.95)
    img = img1[h1: h2, w1: w2]

    # Resolution
    img = cv2.resize(img, None, fx=0.75, fy=0.75, interpolation=cv2.INTER_AREA)

    # Gray Scale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Adaptive Threshold
    img2 = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 15, 2)

    #Blur
    blur = cv2.medianBlur(img2, 5, dst=None)

    # Add Kernel
    kernel = np.ones((3,3),np.uint8)

    # Morphology
    closing = cv2.morphologyEx(blur, cv2.MORPH_CLOSE, kernel)
    gradient = cv2.morphologyEx(closing, cv2.MORPH_GRADIENT, kernel)

    # 경계값 도출
    contours = cv2.findContours(gradient, cv2.RETR_TREE, cv2.CHAIN_APPROX_TC89_L1)[0]

    x1 = [] #x-min
    y1 = [] #y-min
    x2 = [] #x-max
    y2 = [] #y-max
    for i in range(1, len(contours)):
        ret = cv2.boundingRect(contours[i])
        x1.append(ret[0])
        y1.append(ret[1])
        x2.append(ret[0] + ret[2])
        y2.append(ret[1] + ret[3])

    x1_min = min(x1)
    y1_min = min(y1)
    x2_max = max(x2)
    y2_max = max(y2)
    cv2.rectangle(gradient, (x1_min, y1_min), (x2_max, y2_max), (0, 255, 0), 3)

    origial_img = gray[y1_min:y2_max, x1_min:x2_max]
    crop_img = img2[y1_min:y2_max, x1_min:x2_max]

    crop_img = cv2.resize(crop_img, (32, 32))

    #img_merge = np.hstack((origial_img, crop_img))

    #return img_merge

    return crop_img
```

1. 이미지의 위, 아래, 양 옆을 강제로 제거한다.
2. 이미지의 해상도를 낮추고, Gray Scale로 전환한다.
3. Adaptive Threshold 함수를 이용해 이미지 이진화를 한다.
4. 노이즈를 낮추기 위해 1차적으로 Blur 처리를 한다.
5. 2차적으로 커널을 생성하여 Morphology 연산 중 closing 연산과 gradient 연산을 거친다.
6. Find Contours 함수를 이용해 경계 값을 도출한 후 그 경계 값으로 사각형을 생성해 이미지 trimming 좌표를 생성한다.
7. 적응형 이진화 된 이미지를 6번에서 구한 좌표로 자르고 32 x 32 사이즈로 Resize 후 Return 한다.

Mission 2

2-2. Color는 자세한 정보지만, 데이터가 크고, Gray는 덜 자세하지만 데이터가 작아 학습에 유리하다. 어떤 데이터셋이 분류문제에서 더 좋은 결과를 보이는가?

Answer:

Color는 3차원 데이터셋으로 정보가 너무 많아 정확도가 떨어질 수 있고,

상품 분류는 색이 아닌 옷의 형태로 구분되기 때문에 색상 데이터는 필요하지 않습니다.

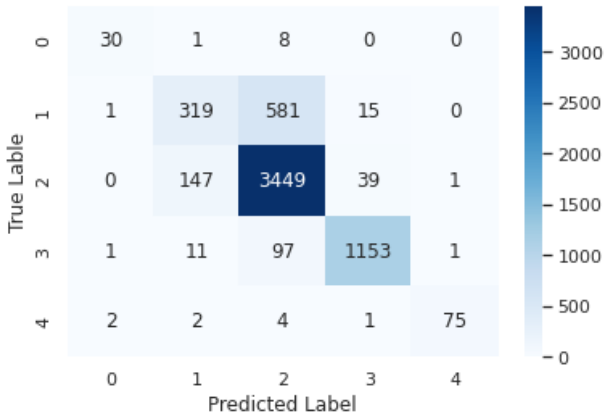
그렇기에 Gray Scale 1차원 데이터셋으로 학습을 진행하였습니다.

Mission 3

3-1. 분류 문제를 수행하여 Validation 데이터의 라벨 별 정확도를 제시하시오.

Answer:

```
[86] cm = confusion_matrix(np.argmax(yValidation, axis = -1), np.argmax(pred, axis=-1))
     sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
     plt.xlabel('Predicted Label')
     plt.ylabel('True Lable')
     plt.show()
```




```
print(classification_report(np.argmax(yValidation, axis = -1), np.argmax(pred, axis=-1)))
```

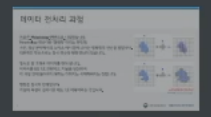
	precision	recall	f1-score	support
0	0.88	0.77	0.82	39
1	0.66	0.35	0.46	916
2	0.83	0.95	0.89	3636
3	0.95	0.91	0.93	1263
4	0.97	0.89	0.93	84
accuracy			0.85	5938
macro avg	0.86	0.77	0.81	5938
weighted avg	0.84	0.85	0.83	5938

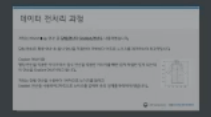
Mission 3

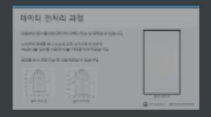
3-2. 정확도를 올리는 작업을 수행하고 작업 수행과정을 설명하시오.

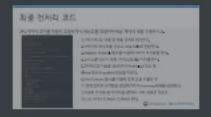
Answer:

모델 학습 시 레이어 설정과 최적화 함수, 하이퍼파라미터 등을 조작하며 모델의 성능을 늘렸습니다.

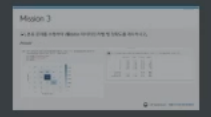

8


9


10


11


12


13


14

Mission 3

3-2. 정확도를 올리는 작업을 수행하고 작업 수행과정을 설명하시오.

Answer:
데이터 전처리 :
정확도를 올리기 위해서는 모델 학습의 기초인 전처리가 잘 되어있어야 합니다.
전처리 과정은 위에서 설명했으니 생략하겠습니다.

모델 학습 :
모델 학습 시 레이어 설정과 최적화 함수, 하이퍼파라미터 등을 조작하며 모델의 성능을 늘렸습니다.

과학기술정보통신부 NIA 한국지능정보사회진흥원

네 발표 시작하겠습니다.

```
Epoch 190/200
186/186 [==============================] - 2s 9ms/step - loss: 0.3055 - accuracy: 0.8746 - val_loss: 0.4145 - val_a
Epoch 191/200
186/186 [==============================] - 2s 9ms/step - loss: 0.3210 - accuracy: 0.8705 - val_loss: 0.3994 - val_a
Epoch 192/200
186/186 [==============================] - 2s 9ms/step - loss: 0.3041 - accuracy: 0.8750 - val_loss: 0.4002 - val_a
Epoch 193/200
186/186 [==============================] - 2s 9ms/step - loss: 0.3480 - accuracy: 0.8618 - val_loss: 0.4209 - val_a
Epoch 194/200
186/186 [==============================] - 2s 9ms/step - loss: 0.3721 - accuracy: 0.8555 - val_loss: 0.4097 - val_a
Epoch 195/200
186/186 [==============================] - 2s 9ms/step - loss: 0.3153 - accuracy: 0.8692 - val_loss: 0.4020 - val_a
Epoch 196/200
186/186 [==============================] - 2s 9ms/step - loss: 0.3249 - accuracy: 0.8690 - val_loss: 0.3949 - val_a
Epoch 197/200
186/186 [==============================] - 2s 9ms/step - loss: 0.3077 - accuracy: 0.8738 - val_loss: 0.4017 - val_a
Epoch 198/200
186/186 [==============================] - 2s 9ms/step - loss: 0.3039 - accuracy: 0.8766 - val_loss: 0.3979 - val_a
Epoch 199/200
186/186 [==============================] - 2s 9ms/step - loss: 0.3011 - accuracy: 0.8778 - val_loss: 0.3970 - val_a
Epoch 200/200
186/186 [==============================] - 2s 9ms/step - loss: 0.3019 - accuracy: 0.8760 - val_loss: 0.3929 - val_a
```

16

```
1 from datetime import datetime
```

클래스 불균형 이슈

5개의 클래스는 서로 다른 개수로 구성되어 있습니다.

특히, tops의 개수가 몇배 더 많이 구성되어 있는데
이를 고려하지 않고 학습 시 모델의 성능이 하락할 수 있습니다.

그렇기에 Weighted Cross Entropy, Focal Loss와 같은 함수를 사용하여
부족한 클래스에 가중치를 곱해 Loss 값을 높여주거나
정규화 과정을 통한다면 클래스 불균형이 어느 정도 해결될 것이라고 생각합니다.

그러나 시간이 부족하여 이 이슈를 해결하지 못한 채 모델 학습을 진행했지만,
클래스 불균형을 해결하는 방법에 대해 공부하며 관련 지식을 쌓았습니다.

Mission 3

3-3. 오류가 나온 이미지에 대해 왜 오류가 나왔는지
그동안 미션 수행에서 얻은 경험과 지식을 통해 설명하시오.

Answer:

오류가 나온 이미지는 대부분 라벨링이 잘못 매칭된 경우였습니다.
이 부분 제외시키고 학습하였더니 정확도가 많이 증가했습니다.

```
====] - 2s 8ms/step - loss: 0.4313 - accuracy: 0.8272 - val_loss: 0.4623 - val_accuracy: 0.8242
====] - 2s 8ms/step - loss: 0.4273 - accuracy: 0.8302 - val_loss: 0.4594 - val_accuracy: 0.8255
====] - 2s 8ms/step - loss: 0.4242 - accuracy: 0.8301 - val_loss: 0.4485 - val_accuracy: 0.8302
====] - 2s 8ms/step - loss: 0.4188 - accuracy: 0.8325 - val_loss: 0.4504 - val_accuracy: 0.8352
```

이상치 제외 전

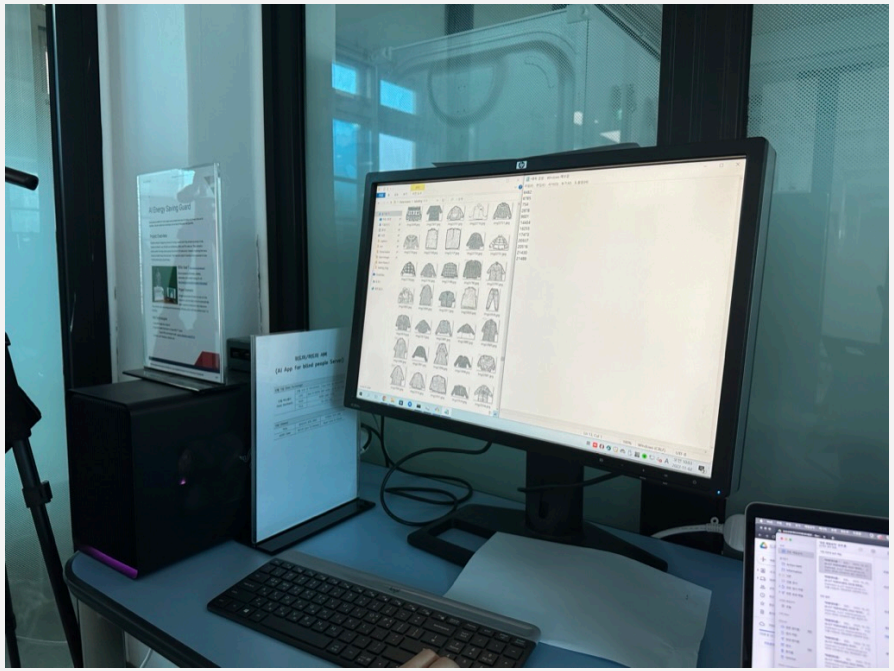
```
- 2s 8ms/step - loss: 0.2554 - accuracy: 0.9006 - val_loss: 0.5014 - val_accuracy: 0.8327
- 2s 8ms/step - loss: 0.2613 - accuracy: 0.8977 - val_loss: 0.4945 - val_accuracy: 0.8297
- 2s 8ms/step - loss: 0.2567 - accuracy: 0.8998 - val_loss: 0.4899 - val_accuracy: 0.8292
- 2s 8ms/step - loss: 0.2523 - accuracy: 0.9035 - val_loss: 0.5044 - val_accuracy: 0.8300
```

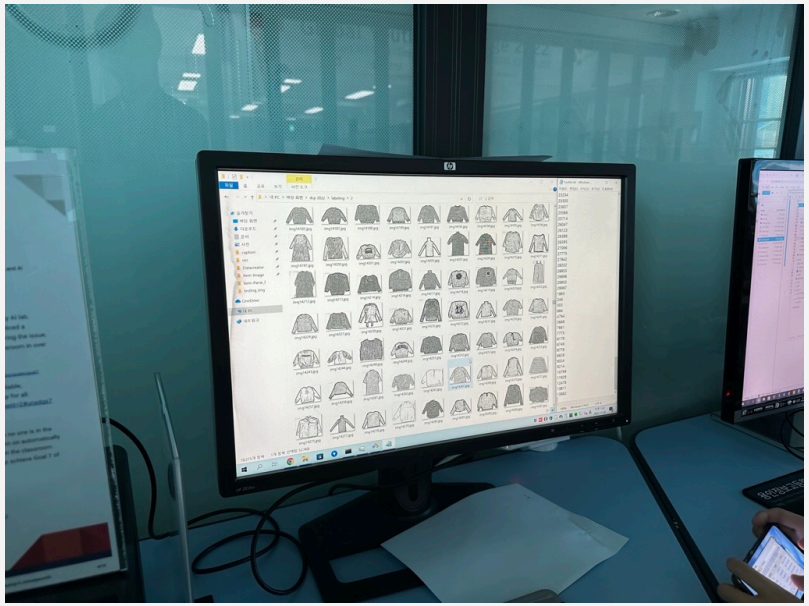
이상치 제외 후

라벨 이슈

라벨 별 이미지를 확인하다 보니, 라벨이 잘못 설정되어 있는 이미지를 발견하게 되었습니다.

이상치 제거 알고리즘을 만들 수도 있었지만,
시간도 촉박하고 라벨링이 잘못된 데이터가 많지 않아서 직접 제외 작업을 진행했습니다.





감사합니다

2022 DATA CREATOR CAMP

과학기술정보통신부 NIA 한국지능정보사회진흥원