Computer Vision
이상치 탐지 알고리즘 경진대회

DL CV 2팀| 16기 이수찬, 17기 김지윤, 17기 이서연

2023-1 KUBIG CONTEST

Index.

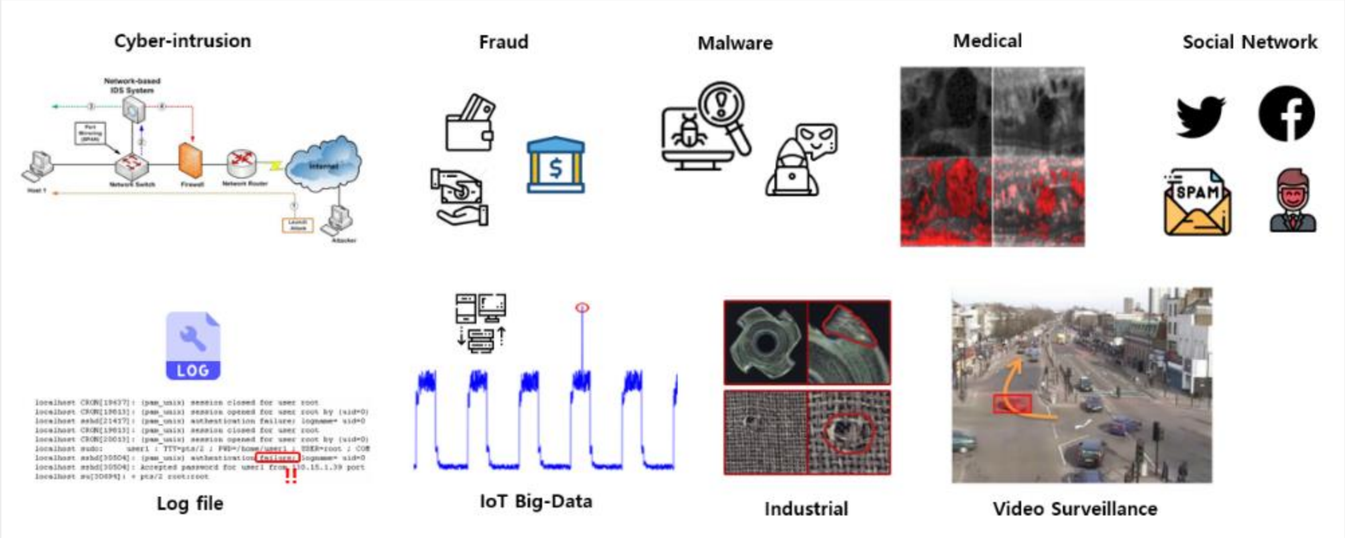
01 대회 개요

-이상치 탐지

-대회 목적

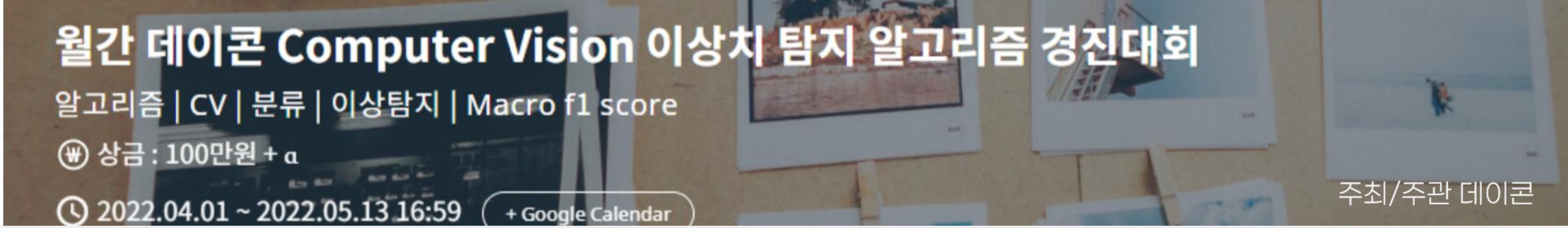
01. 이상치 탐지(Anomaly Detection)

"Normal(정상) sample과 Abnormal(비정상, 이상치, 특이치) sample을 구별해내는 문제"



"Deep Learning for Anomaly Detection: A Survey," 2019 arXiv

Cyber-Intrusion Detection, Fraud Detection,Malware Detection, Medical Anomaly Detection, Industrial Anomaly Detection 등에 활용



02. 대회 소개

MVtec AD Dataset에 들어있는 사물의 종류를 분류하고
정상 샘플과 비정상(이상치) 샘플을 분류
-> 불균형 데이터 셋을 학습하여 사물의 종류와 상태를 분류할 수 있는
컴퓨터 비전 알고리즘 개발

02 EDA 및 모델 분석

-데이터 소개

-모델 소개

01. 데이터 소개

1) Dataset 구성

```
classList = train_y['class'].unique()
classList
```

```
array(['transistor', 'capsule', 'wood', 'bottle', 'screw', 'cable',
       'carpet', 'hazelnut', 'pill', 'metal_nut', 'zipper', 'leather',
       'toothbrush', 'tile', 'grid'], dtype=object)
```

MVtec AD Dataset : 15 개의 class로 구성

1. **train [Folder]** : 4277개 이미지, 88개 label
2. **test[Folder]** : 2154개 이미지
3. **train_df(csv)** : train folder에 대한 정보(인덱스, 파일명, 클래스 등)
4. **test_df(csv)** : test folder에 대한 정보(인덱스, 파일명)
5. **sample_submission(csv)** : 제출 파일

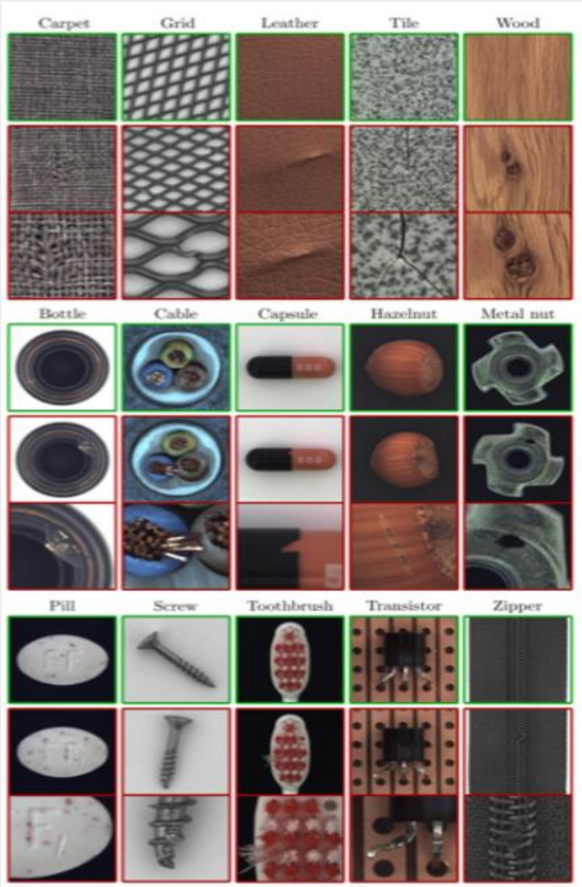


Figure 2: Example images for all five textures and ten object categories of the MVTec AD dataset. For each category, the top row shows an anomaly-free image. The middle row shows an anomalous example for which, in the bottom row, a close-up view that highlights the anomalous region is given.

01. 데이터 소개

2) train[Folder],train_df

train [Folder]
- 10001.png
- 10002.png
- •••
- 14276.png

train Folder]

```
train_y = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/open/train_df.csv")
train_y
```

	index	file_name	class	state	label
0	0	10000.png	transistor	good	transistor-good
1	1	10001.png	capsule	good	capsule-good
2	2	10002.png	transistor	good	transistor-good
3	3	10003.png	wood	good	wood-good
4	4	10004.png	bottle	good	bottle-good
...	...	...	...	...	...
4272	4272	14272.png	transistor	good	transistor-good
4273	4273	14273.png	transistor	good	transistor-good
4274	4274	14274.png	grid	good	grid-good
4275	4275	14275.png	zipper	good	zipper-good
4276	4276	14276.png	screw	good	screw-good

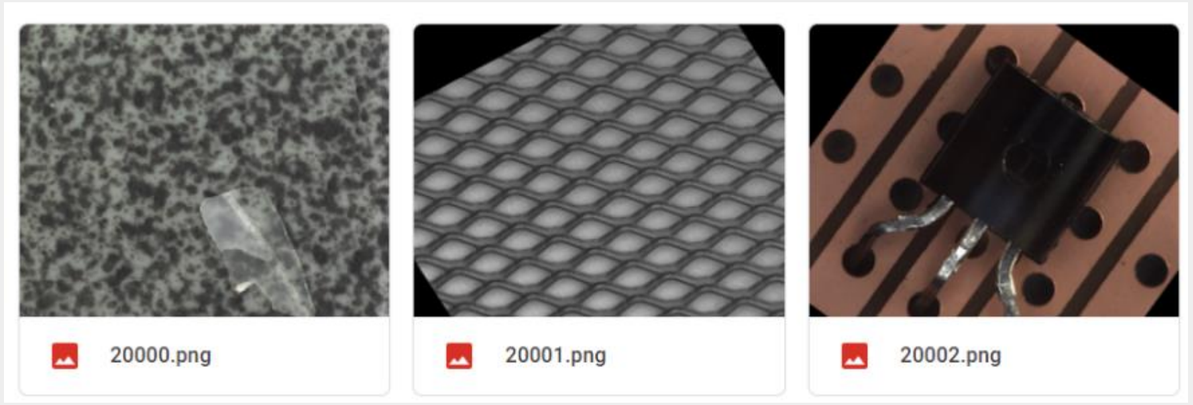
4277 rows × 5 columns

train_df

01. 데이터 소개

3) test[Folder], test_df

test {Folder]
- 20001.png
- 20002.png
- •••
- 22153.png



test [Folder]

```
test_y = pd.read_csv(path +"open/test_df.csv")
test_y
```

	index	file_name
0	0	20000.png
1	1	20001.png
2	2	20002.png
3	3	20003.png
4	4	20004.png
...	...	...
2149	2149	22149.png
2150	2150	22150.png
2151	2151	22151.png
2152	2152	22152.png
2153	2153	22153.png

2154 rows × 2 columns

test_df

02. 모델 소개

Efficientnet :

Rethinking model scaling for CNN

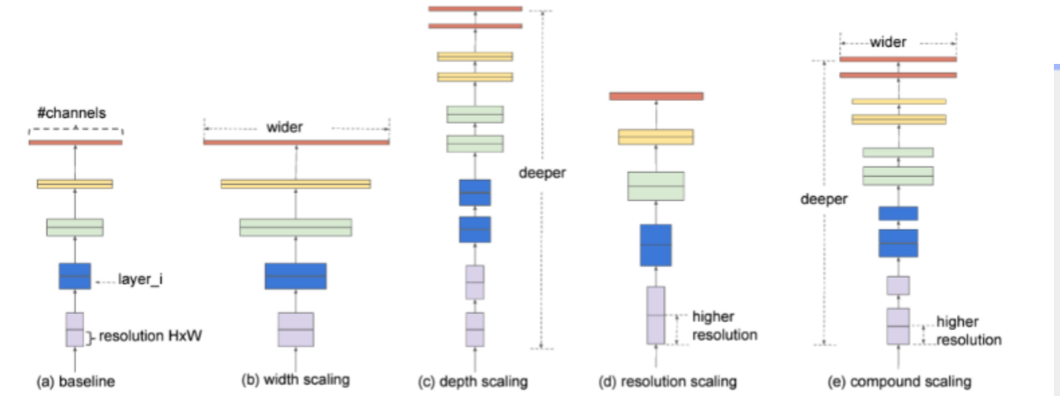


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

<compound scaling>

-모델을 크게 만드는 방법 :

1) depth 2) width 3) resolution(image size) 조정

-> Efficientnet은이 3가지의 최적의 조합을 찾아냄

-> compound scaling을 제안하여 SOTA 달성

-> 기존 ConvNet보다 8.4배 작으면서 6.1배 빠르고 더 높은 정확도를 가짐

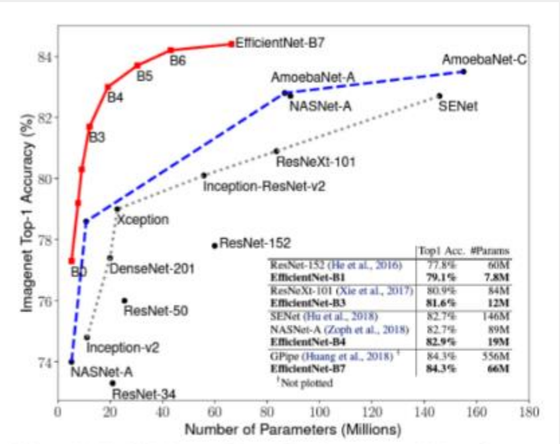


Figure 1. **Model Size vs. ImageNet Accuracy.** All numbers are for single-crop, single-model. Our EfficientNets significantly outperform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.3% top-1 accuracy but being 8.4x smaller and 6.1x faster than GPipe. EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152. Details are in Table 2 and 4.

02. 모델 소개

Efficientnet b0~b7 : a family of models

-> 다양한 scale 에서의 최적의 조합을 보여줌

<모델별 권장 input size(resloution)>

	Model	Recommended Image Size
EfficientNet V1	B0	224
	B1	240
	B2	260
	B3	300
	B4	380
	B5	456
	B6	528
	B7	600
EfficientNet V2	S	384
	M	480
	L	480
	B0	224
	B1	240
	B2	260
	B3	300

모델별 권장 사항

1) resolution(image size) : 8 or 16의 배수

2) depth,width : 8의 배수의 channel size 요구

3) resource limit :

resolution 병목 현상 -> resolution 고정 후 depth, width 조정

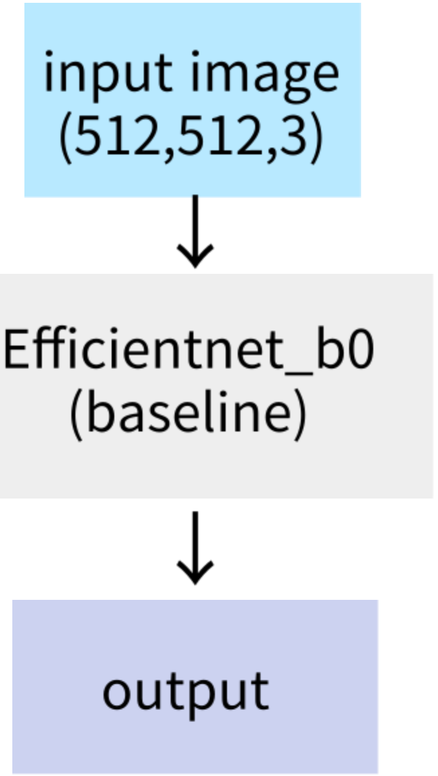
03 데이터 전처리

-Augmentation

-Normalization

+) basline 코드 및 추가 도입 시도

Baseline



1. image augmentation 진행
2. image normalization 진행
3. Efficientnet_b3으로 교체
4. AdamW optimizer 사용
5. post-processing 진행

01. Augmentation

1) image input(resolution) :

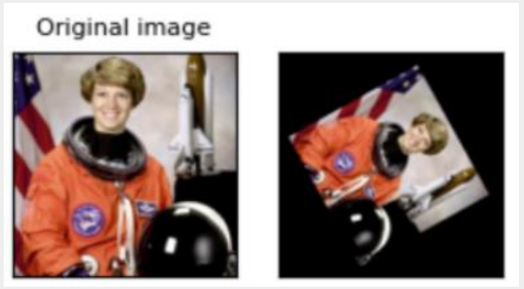
-efficient 모델별 권장 input size(resolution)로 resize

-input image shape = (img_size, img_size,3) 확인

2) image Augmentation :

- test[Folder] 이미지 분석 결과 : image rotation이 많이 존재

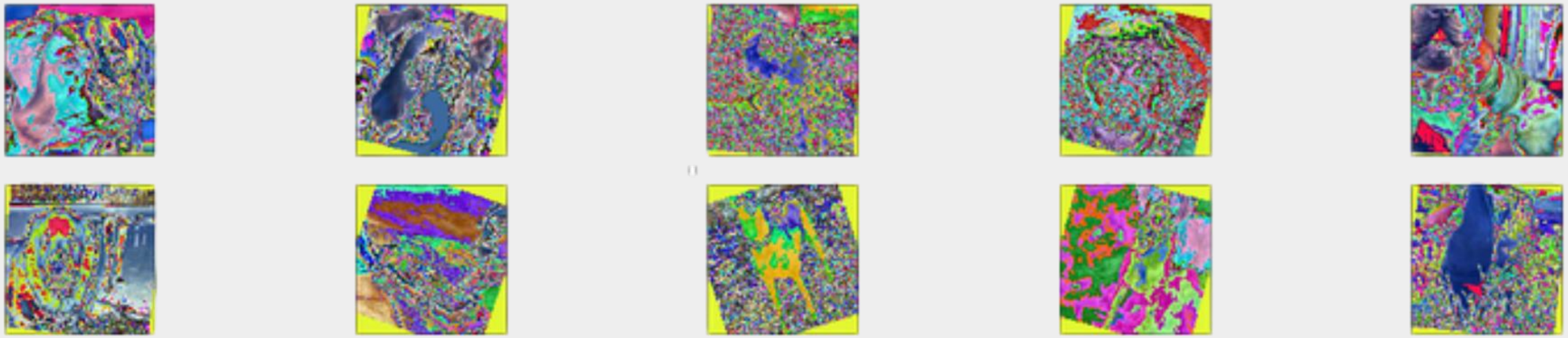
- RandomAffine vs RandomRotation



-> train image에 RandomAffine((-180,180)) 사용

02. Normalization

3) image Normalization : 성능 향상을 위해 평균과 표준편차로 정규화 실행



04 모델링

-모델링

-post-processing

-제언

01. 모델

Efficientnet_b3으로 교체

기존 Baseline code에서 성능을 높이기 위해 model size를 증가

-> input size에 따른 Colab 메모리 한계로 Efficientnet_b3으로 결정

Table 2. **EfficientNet Performance Results on ImageNet** (Russakovsky et al., 2015). All EfficientNet models are scaled from our baseline EfficientNet-B0 using different compound coefficient ϕ in Equation 3. ConvNets with similar top-1/top-5 accuracy are grouped together for efficiency comparison. Our scaled EfficientNet models consistently reduce parameters and FLOPS by an order of magnitude (up to 8.4x parameter reduction and up to 16x FLOPS reduction) than existing ConvNets.

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.1%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.6%**	**95.7%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.9%**	**96.4%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.6%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.3%**	**97.0%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

We omit ensemble and multi-crop models (Hu et al., 2018), or models pretrained on 3.5B Instagram images (Mahajan et al., 2018).

<모델 size별 성능>

01. 모델

	public 점수 private 점수
1차 시도 : transforms.RandomAffine((-45,-45)) + efficientnet_b3 + epoch수(30)	0.7628774442 0.7665225537
2차 시도 : transforms.RandomAffine((-180,180)) +efficientnet_b3 +epoch수(40)	0.794914952 0.8085756346
3차 시도 : 2차 시도 + epoch수(50)로 진행	0.8139326135 0.8202689987
4차 시도 : 3차 시도 + epoch수(40)+post-processing 진행	0.9080061477 0.9045483911

02. post-processing(후처리)

방향성 1

1) class 불균형으로 인한 good label의 과한 예측 방지

방향성 2

2) 1)의 결과를 이용해도 헷갈려하는
class(zipper,toothbrush)에 대해 추가 학습

02. post-processing(후처리)

post-processing code : 방향성 1) good label로의 과한 예측 방지

	index	file_name	class	state	label
0	0	10000.png	transistor	good	25
1	1	10001.png	capsule	good	5
2	2	10002.png	transistor	good	25
3	3	10003.png	wood	good	27
4	4	10004.png	bottle	good	1
5	5	10005.png	wood	good	27
6	6	10006.png	capsule	good	5
7	7	10007.png	screw	good	19

train_bad_df

a. state가 good인 클래스들은 그대로, good이 아닌 클래스들은 일괄적으로 (class)-bad로 통일

b. 1) bad로 예측하거나 2) good으로 예측했지만 softmax값이 0.999999보다 작은 인덱스 추출

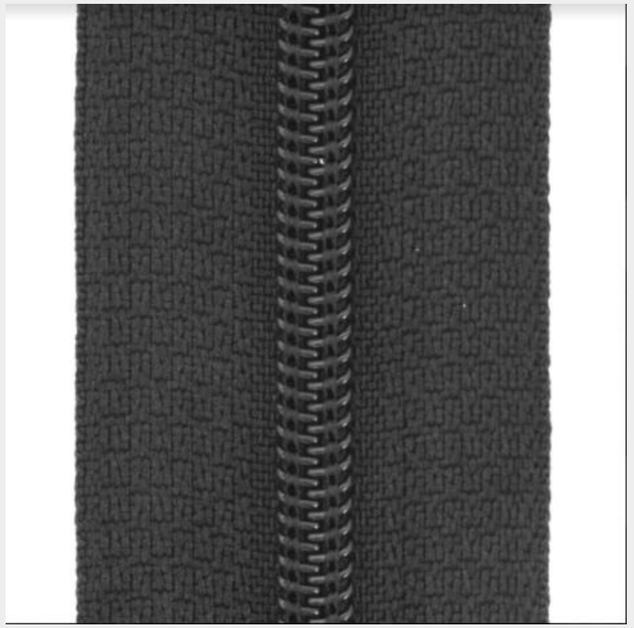
c. 원래 예측값이 good으로 되어 있는 경우 해당 레이블이 아닌 2번째로 높았던 레이블로 예측하게 함.

02. post-processing(후처리)

post-processing code : 방향성 2)

-zipper class : 기존 예측값과 3개의 zipper만 학습한 단일 모델 이용하여 Hard Voting

-toothbrush class : 해당 클래스만 학습한 단일 모델 이용



<zipper>

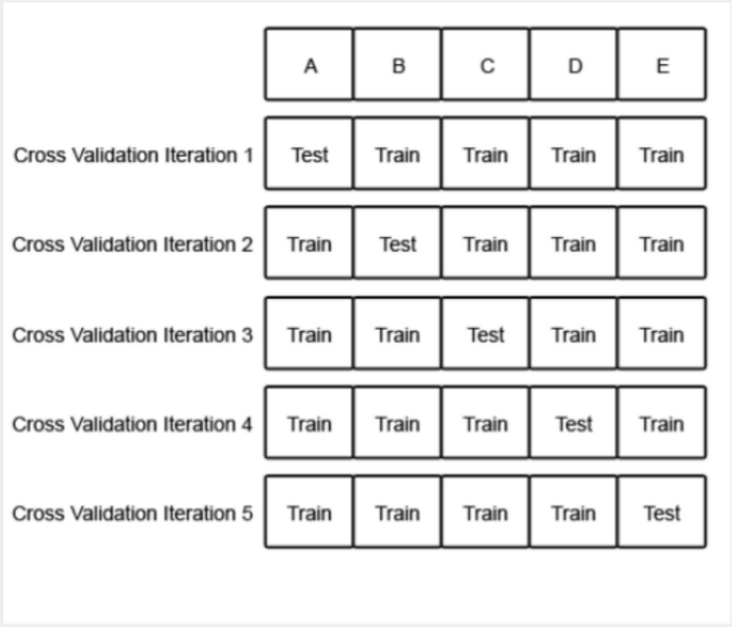
<toothbrush>

03. 제언

모델 성능 향상을 위한 제언

1) 전처리에서 이상치 데이터 불균형 문제를 해결하기 위한 방법 추가

2) 모델 학습 과정에서 K-fold 교차 검증을 사용



<K-fold 교차 검증>

Thank You

DL CV 2팀| 16기 이수찬, 17기 김지윤, 17기 이서연

2023-1 KUBIG CONTEST