

Computer Vision

문서 타입 분류 대회

2024년 11월 6일

업스테이지AILab 4기

3조 진주영, 최정은, 김효원, 문기중, 이승민

목차

1. 프로젝트 개요
2. 팀 소개
3. 구현 프로세스와 전략
4. 구현 결과
5. 최종 결과물
6. 추가 개선 노력
7. 프로젝트 회고

1. 프로젝트 개요

Computer Vision 기반의 Classifier 모델을 훈련하고 왜곡이 포함된 이미지를 정확하게 분류하는 프로젝트입니다.

훈련 데이터



- 17개 클래스의 이미지
- 1,570장 (평가 데이터보다 적은 수)

평가 데이터



- Rotation, Flip, Noise 등 의도적으로 왜곡된 이미지
- 3,140장

2. 팀 소개



진주영(팀장)

지치지 않는 무한 리
액션



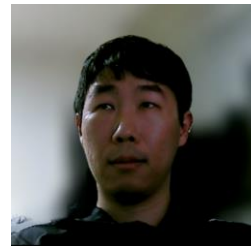
최정은

즐겁게 여러가지 시
도해보기



김호원

잘 모르는데
일단 해보기



문기중

뭔가 더 해보기

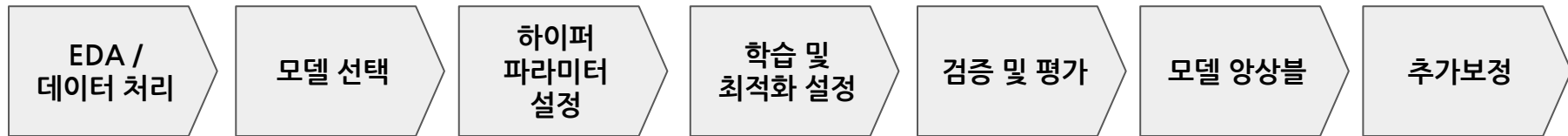


이승민

오디오 채우기

3. 구현 프로세스와 전략

프로세스



구현 전략

- 1 Test Set의 왜곡에 잘 대응할 수 있도록 데이터를 증강해서 모델 훈련
- 2 프로젝트 데이터의 특성을 잘 파악하고 효과적으로 분류할 수 있는 모델을 선정
- 3 모델이 잘 분류하지 못하는 데이터를 위한 추가보정
- 4 모델 세부 조정과 앙상블

4. 구현 결과 - 1) EDA / 데이터 처리 (오라벨, 불균형 수정)

라벨링이 잘못된 이미지 8개의 라벨을 정정

Index: 38
File ID: 0583254a73b48ece.jpg
Target: 11



Index: 192
File ID: 1ec14a140be633db.jpg
Target: 14



Index: 340
File ID: 38d1796b6ad99ddd.jpg
Target: 11



Index: 428
File ID: 45f0d2dfc7e47c03.jpg
Target: 3



Index: 723
File ID: 7100c5c7aecdac5.jpg
Target: 3



Index: 862
File ID: 8646f2c3280a4f49.jpg
Target: 7



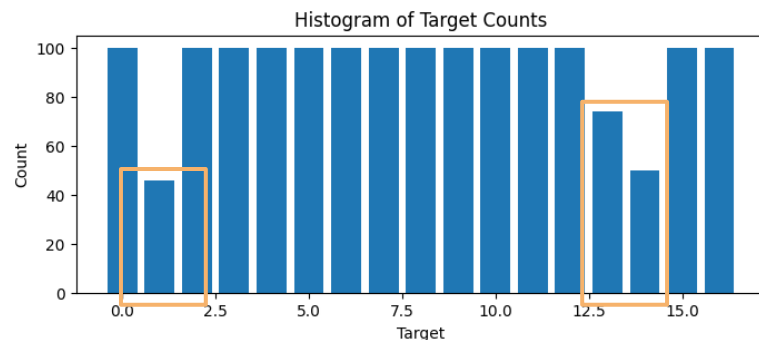
Index: 1095
File ID: aec62fcd7af97cd.jpg
Target: 3



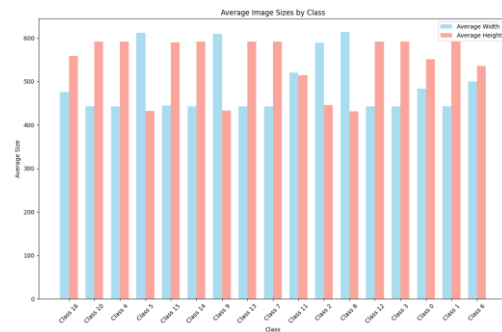
Index: 1237
File ID: c5182ab809478f12.jpg
Target: 4



훈련 데이터에서 클래스별 데이터 개수를 분석 데이터를 증강하여 불균형 해결

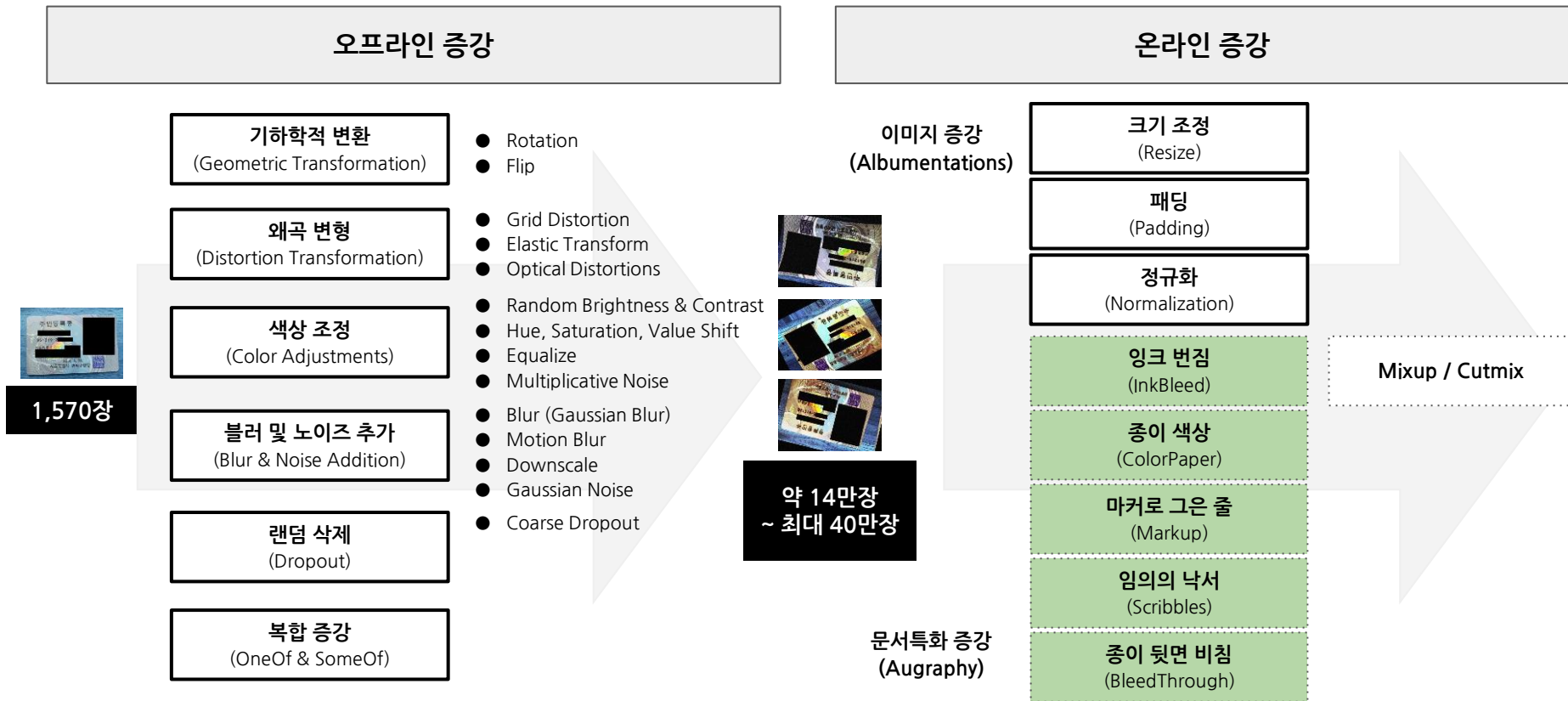


이미지 사이즈 평균 확인: 400~600



4. 구현 결과 - 1) EDA / 데이터 처리 (데이터 증강)

오프라인 증강으로 훈련 데이터의 수를 약 14~40만장으로 증가시키고 추가로 온라인 증강을 적용하였습니다.



4. 구현 결과 - 2) 모델 선택

프로젝트 데이터에 가장 잘 맞는 모델을 찾기 위해 최대한 다양한 모델을 실험하였습니다.

	철학	특징	활용모델
Residual Networks	깊은 네트워크를 위한 안정성 및 성능 향상	<p>Residual Connections: 네트워크가 깊어질수록 발생하는 그라디언트 소실 문제를 해결하기 위해, <u>residual connection</u>을 통해 각 층의 입력을 그대로 다음 층으로 전달</p> <p>ResNeXt: ResNet의 개선형으로, <u>cardinality(그룹 수)를 늘려 feature map을 더욱 풍부하게</u> 만들어 성능과 효율성 개선</p>	<div> <div>resnet34</div> <div>resnet50</div> <div>resnet101</div> <div>resnet152</div> </div> <div> <div>resnext</div> <div>densenet121</div> </div>
Dense Networks	피쳐맵 재사용으로 매개변수 효율 극대화	Dense Connections: <u>이전 레이어의 출력을 모두 다음 레이어의 입력으로 사용</u> 해서 feature map 재사용성을 극대화하고 매개변수 수 축소 (Resnet의 스킵 연결에서 영감을 받아서 확장)	
Efficient Networks	성능-자원 균형을 최적화	EfficientNet: <u>compound scaling</u> 을 통해 모델의 너비, 깊이, 해상도를 <u>균형 있게 조정</u> 해 최적의 성능을 내도록 설계	
Attention-based Models (Vision Transformers)	Transformer 기반의 전역적 특성 학습	<p>ViT (Vision Transformer): CNN 대신 전적으로 attention 메커니즘을 사용해 전역 정보를 효과적으로 학습</p> <p>CaIT (Class-Attention in Image Transformers): ViT 기반 모델로, Class Attention 레이어를 추가하여 보다 정밀한 클래스 특성 추출</p> <p>Swin Transformer : <u>이미지의 지역적 윈도우를 적용</u>해 기존 Transformer보다 메모리 효율성과 성능을 모두 개선</p>	<div> <div>efficient_b0</div> <div>efficient_b3</div> <div>efficient_b4</div> </div> <div> <div>vit</div> <div>cait</div> <div>swin2_tiny</div> <div>swin2_base</div> </div>
Transformer-Inspired CNN	CNN 구조를 Transformer 스타일 설계로 업데이트	ConvNeXt 계열: <u>ConvNet을 Transformers와 비슷한 방식으로 재 설계</u> 하여 더 높은 성능을 목표로 한 모델. v2는 MAE를 통해 fine-tuning 도입	<div> <div>convnext</div> <div>convnextv2_base</div> </div>

4. 구현 결과 - 3) 하이퍼파라미터 설정



데이터 및 입력 설정	Image Size	모델별 최적화된 사이즈 (224, 384 등)
	Batch Size	32 그대로 사용
훈련 설정	Epochs	100으로 설정하되, Early Stopping 적용
	Early Stopping	Patience는 5로 설정. 단, 기준은 F1이 아니라 Validation Accuracy로 설정
학습률 및 스케줄러	Learning Rate	1e-3, 1e-4, 1e-5 중에서 선택
	Scheduler	StepLR, CyclicLR, CosineAnnealingLR 모델 별 번갈아 가며 적용
정규화 및 규제	Weight Decay	2e-5
	Dropout Rate	0.2, 0.3, 0.4 중에 선택

4. 구현 결과 - 4) 학습 및 최적화 설정



손실함수

Cross-Entropy Loss : 모든 샘플에 동일한 가중치를 부여

Focal Loss : 어려운 샘플에 대한 손실을 증폭시키고 쉬운 샘플의 손실을 억제

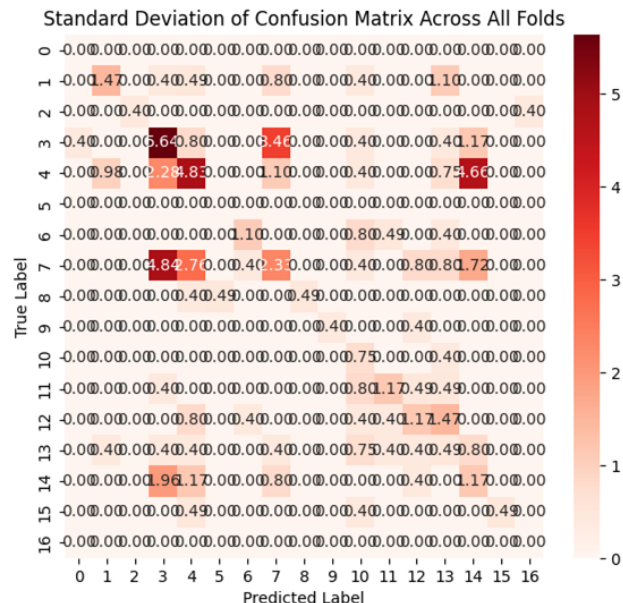
옵티마이저

Adam : SGD에 모멘텀과 학습률을 자동으로 조절하는 RMSprop 기능을 결합한 최적화 알고리즘. 각 가중치에 대해 학습률이 적응적으로 조정되기 때문에 빠르고 안정적

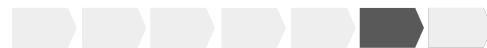
AdamW : Adam 옵티마이저의 변형 버전. weight decay를 효과적으로 적용. 과적합을 줄이고 일반화 성능을 향상.

클래스 [3, 7], [4, 14], [3, 4, 7, 14]

만으로 훈련한 모델 추가



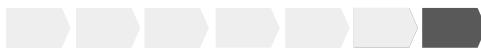
4. 구현 결과 - 6) 단일 모델 실험 결과



순위	모델	하이퍼 파라미터	F1 Score (Public)
1	15_effb3_epo13	Efficientnet_b3/ 이미지 260, 배치 32, Epochs = 14, Loss_fn = Focal loss	0.9392
2	r+p_1107	ResNet50 / 이미지 244, 배치 32, Epochs = 21, Label Smoothing = 0.1, 패딩 추가	0.9389
3	22_resnext	ResNeXt / 이미지 260, 배치 32, Epochs = 8, Label Smoothing = 0.1, 패딩 추가, Scheduler =StepLR	0.9349
4	r50_8/13	ResNet50 / 이미지 244, 배치 32, Epochs = 8, learning rate = 1e-3	0.9339
5	swin_base	이미지 224 배치 24 epochs = 15 dropout = 0.3 weight_decay = 2e-5 label_smoothing = 0.1	0.9321
6	r152_7/12	ResNet152 / 이미지 244, 배치 32, Epochs = 7, learning rate = 1e-3	0.9309
7	21_cosine	Resnet101/ 이미지 224, 배치 32, Epochs = 8, Scheduler =StepLR, 일부 에포크에 Mixup 및 Cutmix 추가	0.9285
8	19_eff_augr	Efficientnet_b0/ 이미지 260, 배치 32, Epochs = 10, learning rate = 1e-4, Scheduler =StepLR	0.9283
9	20_effb4_all	Efficientnet_b4/ 이미지 260, 배치 32, Epochs = 7, Scheduler =StepLR, smoothing, 일부 Transform 변형	0.9268
10	effib4_20	Efficientnet_b4/ 이미지 224, 배치 16, Epochs = 8, Scheduler =StepLR, 에포크별 추가 학습	0.9268

...이외 77회 이상 Submission/실험

4. 구현 결과 - 7) 모델 앙상블 및 추가 보정



Soft voting


	모델	F1 Score
1	efficientnetb0//b4/resnet50/101/resnext/densenet121 /각 클래스별 학습 모델 추가(efficientnetb3)	0.9464
2	efficientnetb0/b4/resnet50//101/resnext/densenet121	0.9414
3	efficientnetb3/b4/densenet121/resnet101/resnet50	0.9283
4	efficientnetb0/b4/densenet121/resnet101	0.9320
5	efficientnetb0/b4/densenet121	0.9445
6	efficientnetb0/b4	0.9459

Hard voting

	모델	F1 Score
1	efficientnetb0/b3/b4/resnet50/resnext/swin-base	0,9442
2	efficientnetb0/b3/b4/resnet101/resnext	0.9394
3	efficientnetb0/b3/b4/resnet50/resnet101/resnext	0.9314
4	efficientnetb0/b3/b4/resnet101/resnet50	0.9337
5	efficientnetb0/b3/b4/resnet101	0.9420
6	efficientnetb0/b4	0.9384

개별 모델의 높은 점수에도 불구하고 앙상블 결과는 앙상블 프로세스 중 오류나 모델의 편향의 문제로 인해 큰 폭의 개선을 보이지 못했던 것 같음.

5. 최종 결과물

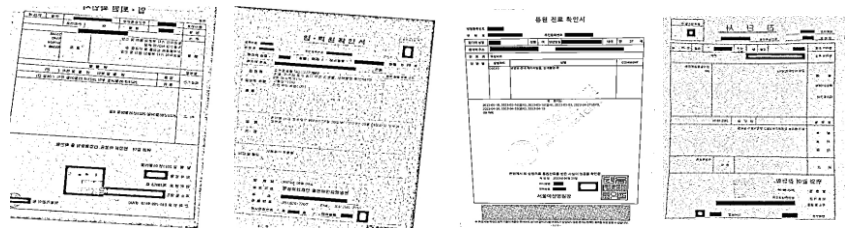
Final Submission	Model Name	Submitter	F1 score F1 score (Final)
<input checked="" type="checkbox"/>	updated_sv	J	0.9464 0.9376
<input type="checkbox"/>	sv(re/ef/den.../r50)	J	0.9432 0.9313
<input type="checkbox"/>	r+p_1107		0.9389 0.9237
<input type="checkbox"/>	sv(ef/re/3,7...7,14)	J	0.9441 0.9445
<input checked="" type="checkbox"/>	sv(ef/re/3,7/4,14)	J	0.9467 0.9353

efficient 모델과 resnet 모델,
[3,7] , [4,14] , [3,4,7,14] 클래스만 학습한 모델
들을 soft voting한 제출이 가장 높게 나왔다.

6. 추가 개선 노력

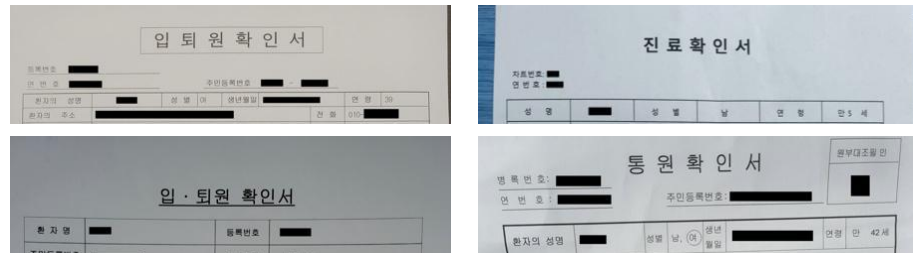
데이터셋 이진화 적용

비슷한 이미지인 클래스3,4,7,14에 대해서 train 및 test 데이터에 binary를 적용한 훈련 및 검증
서류의 텍스트를 선명하게 하고자 했던 시도



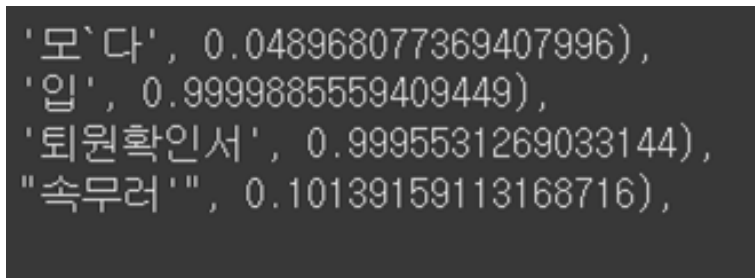
서류 제목 크롭 데이터 추가

훈련 데이터의 상단만 크롭한 이미지를 추가. 서류 제목에 포커스를 두고
자 하였던 시도.



Easy OCR

easy ocr 라이브러리를 활용. 이미지에서 텍스트 데이터를 추출하고, 이미지와 같이 벡터화하여 분류를 해보려고 시도.
test 이미지에 노이즈가 있는 건 인식 불가.



7. 프로젝트 회고

배운 점

눈으로 보기에 test셋과 다른 augmentation이라도 모델이 학습을 더 잘한다는 것을 알게 됨

여러 모델을 적용해보고 싶었는데, 주어진 시간과 자원 내에서 최대한 해봐서 좋았음

점수 올리기 보다도 다양한 모델, 파라미터를 사용해 시도해본 점과 팀원들과 매일 피드백하며 진행한 점이 실력 향상에 큰 도움이 됐음

사이클을 돌아가면서 점수가 나오지 않는 이유를 찾아보고 가설을 세운 뒤 모델을 돌려서 가설을 확인하는 과정을 경험해봐서 좋았음.

아쉬운 점

모델을 fine tuning하는 방식으로 혼동 클래스를 학습하지 못한 것.

많은 모델 시도에 시간이 많이 걸려서 마지막에 앙상블을 더 많이 하지 못했는데 그래서 성능을 더 올리지 못한 것

각 모델별 스코어를 많이 올린 편이었는데, 앙상블에 더 많은 시간을 쓰지 못한 점이 가장 아쉬움

정확히 알고 쓰지는 못한 점, 그리고 결과가 나왔을 때 왜 이렇게 나왔는지 분석하기가 어려웠던 것