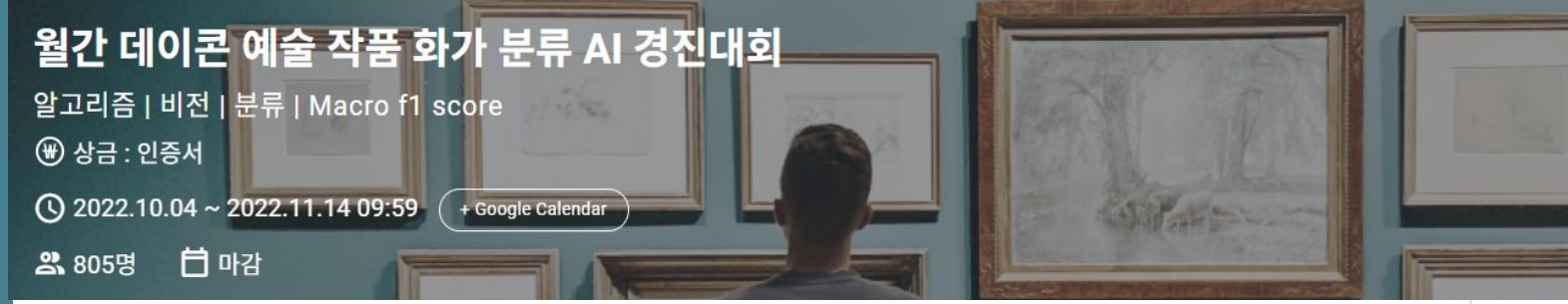


AI Bootcamp_트랙 학습 프로젝트 6조

Rainbow 🏳️‍🌈

권영수 김경은 김동억 최유라

목 차

01

대회 소개 및 일정

대회 소개

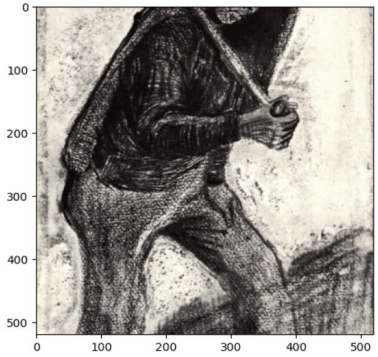
대 회 : 월간 데이콘 예술 작품 화가 분류 AI 경진대회
문 제 : 알고리즘 / 비전 / 분류 / Macro f1 score
기 간 : 2022.10.04 ~ 2022. 11.14

[배 경]
예술 작품을 화가 별로 분류하는 대회이며,
예술 작품의 일부분만 주어지는 테스트 데이터셋에 대해
올바르게 화가를 분류해낼 수 있는 예술 작품의 전문가인
AI 모델 개발

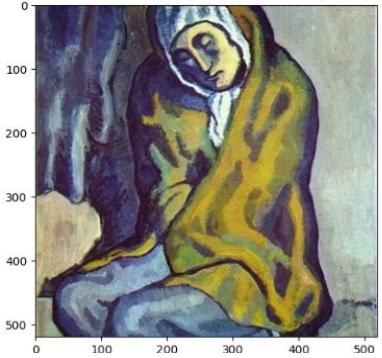
[주 제]
예술 작품을 화가 별로 분류하는 AI 모델 개발

[목 표]
일부분만 주어지는 예술 작품을 화가 별로 분류하는 AI 모델 개발

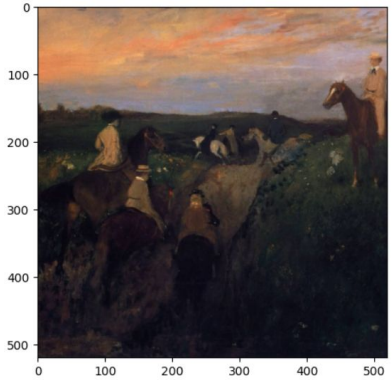
- 학습 데이터셋은 대표적인 화가 50명에 대한 예술 작품 (이미지) 제공
- 테스트 데이터셋은 대표적인 화가 50명에 대한 작품 (이미지)의 일부분(약 ¼)만 제공
- 학습에 활용할 수 있는 화가 50명에 대한 특징 정보 (csv) 추가 제공



-> 고흐



피카소 <-



-> 에드가 드가

일정

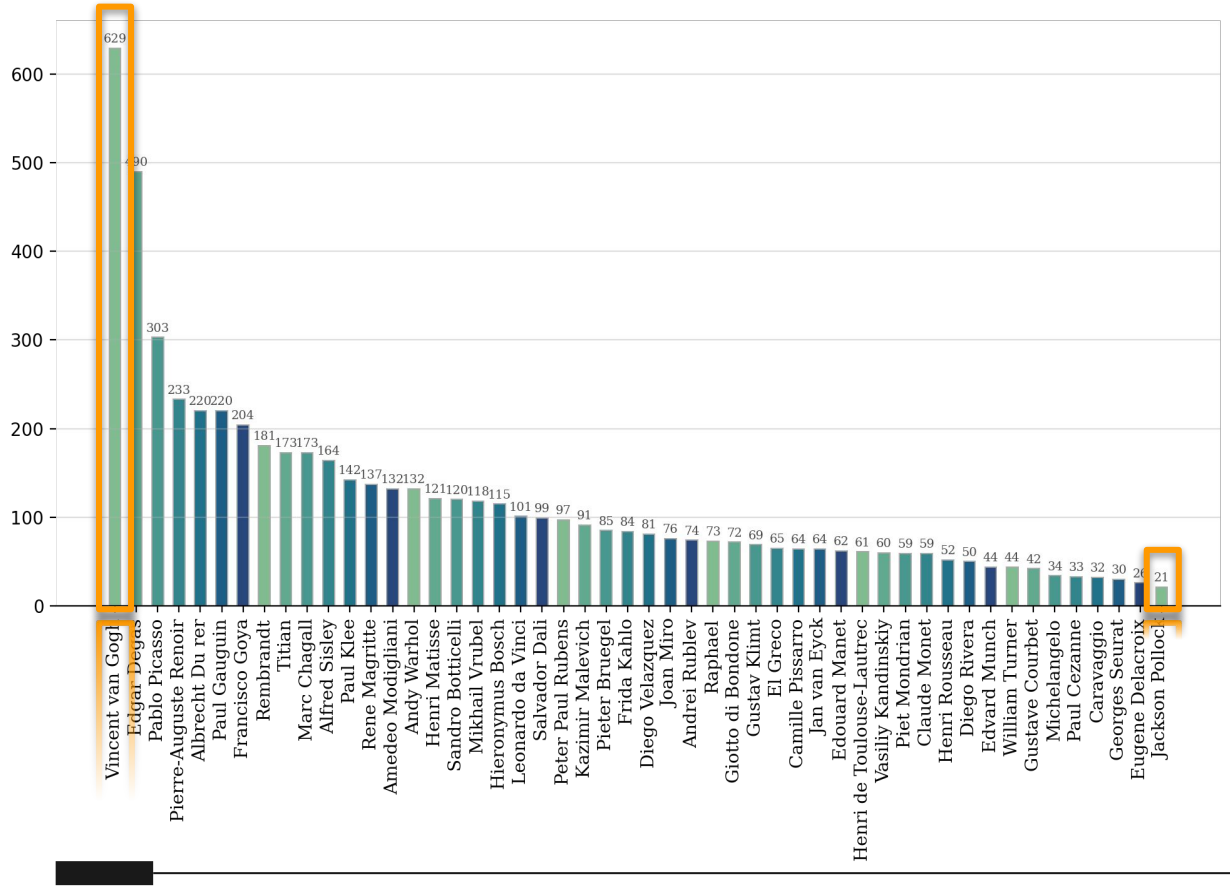

Step 1
공유된 베이스라인 코드 분석
Step 2
대회 상위권 코드 분석
Step 3
EDA
Step 4
베이스라인 모델 구축
Step 5
모델 개선
Step 6
성능 개선
Step 7
발표자료 작성
Step 8
마무리

02

EDA

EDA - 작가 정보(그림 분포 및 Imbalance)

Number of Pictures



- 반 고흐의 그림이 629점으로 가장 많음
- 잭슨 폴락의 그림은 21점으로 가장 적음
- 반 고흐가 총 그림 수 10%를 차지함
- 반 고흐, 에드가 드가, 피카소 이 3명이 화가 3%인 반면 그림 수는 24%를 차지함
    - 즉, 특정 인물들에게 그림이 몰려있음

EDA – 작가 정보(장르)

	name	years	genre	nationality
0	Amedeo Modigliani	1884 - 1920	Expressionism	Italian
1	Vasiliy Kandinskiy	1866 - 1944	Expressionism,Abstractionism	Russian
2	Diego Rivera	1886 - 1957	Social Realism,Muralism	Mexican
3	Claude Monet	1840 - 1926	Impressionism	French
4	Rene Magritte	1898 - 1967	Surrealism,Impressionism	Belgian

Artist_info.csv 를 보면 화가들 중 2개 혹은 더 많은 장르를 소화하는 화가들이 있음
-> 문제는 같은 화가가 그렸어도 장르가 다른 그림은 스타일이 달라짐

EX)
Vasiliy Kandinskiy(Wassily Kandinsky, 바실리 칸단스키)는 유명한 러시아 화가는 표현주의(Expressionism)와 추상미술(Abstractionism)의 주요 인물임

Vasiliy Kandinskiy 가 그렸지만 다른 장르로 그린 그림

Abstractionism(추상) Expressionism(표현)

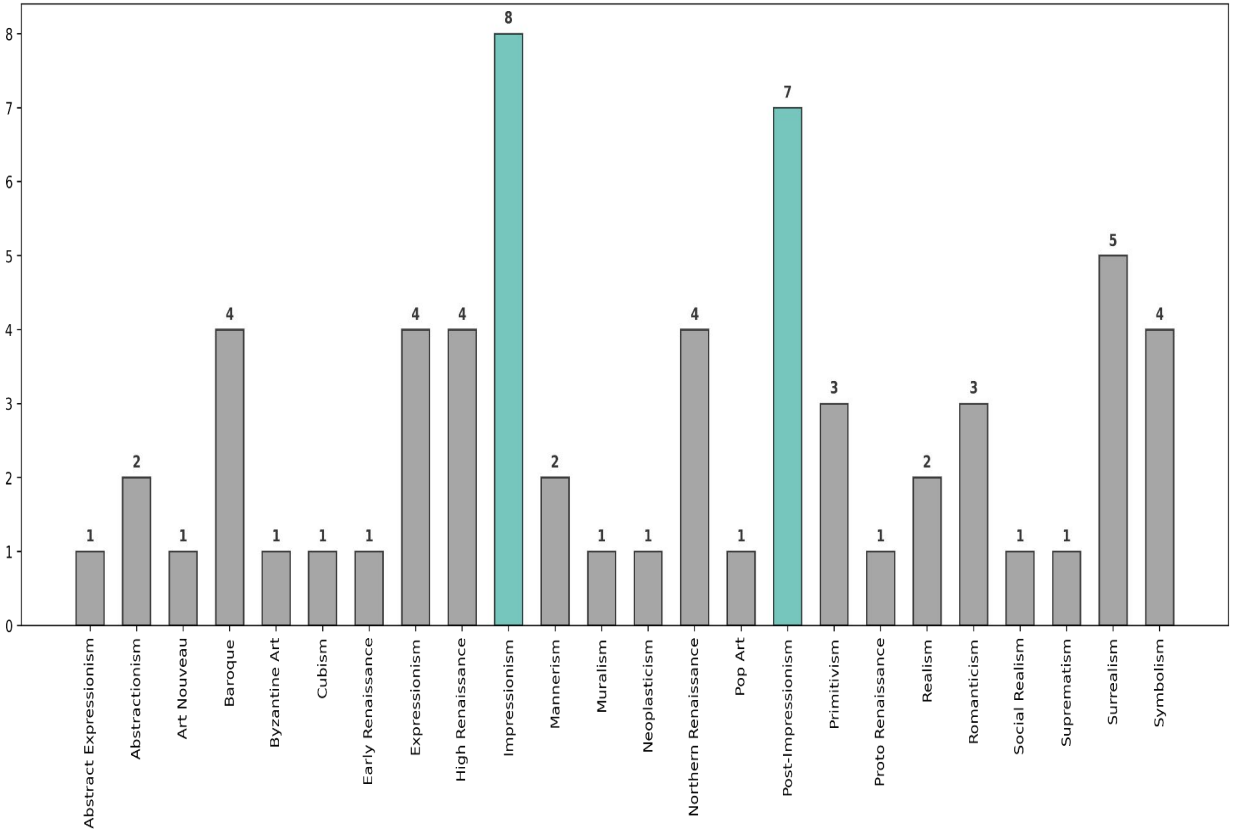
장르(주의)가 다르니 그림이 달라지기에 Vasiliy Kandinskiy 학습 데이터가 추상주의 그림 수가 적고 표현주의 그림이 많다고 가정해보면 표현주의 그림을 잘 분류하고 추상주의는 잘 분류 못할 확률이 높음

-> 결국 모델은 Vasiliy Kandinskiy 그림 중 표현주의만 잘 분류하고 추상주의 그림을 보고 Vasiliy Kandinskiy 를 강하게 부정 할 수 있음

-> train과 inference에서 괴리감이 있을 수 있음

EDA – 작가 정보(장르 분포)

Genre by Artist



- Impressionism (인상주의) 다음으로는 Post-Impressionism (탈인상주의) 가 가장 많음

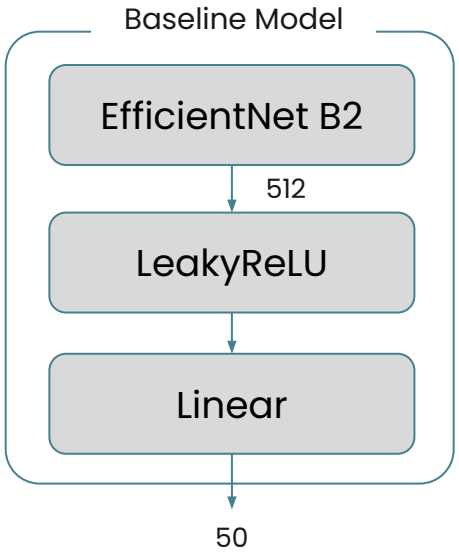
- 이 장르들은 19-20세기 근대시대에 유행한 주의임
- 가장 그림을 많이 가진 반고흐와 에드가 드가가 포함되는 주의임

03

Baseline Model

Baseline Model

- EfficientNet B2 모델 선택
  - 컴퓨팅 파워와 학습 속도를 고려하여 선택
- EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, 2019 논문
  - depth, width, resolution을 효율적으로 조절하는 compound scaling을 제안



MODEL	EfficientNet B2
EPOCH	1000
LEARNING RATE	1e-3
BATCH SIZE	16
OPTIMIZER	Adam
CRITERION	CrossEntropyLoss
METRIC	Macro F1 Score
EARLY STOPPING	10회

Baseline Model 결과

train loss : 0.5161

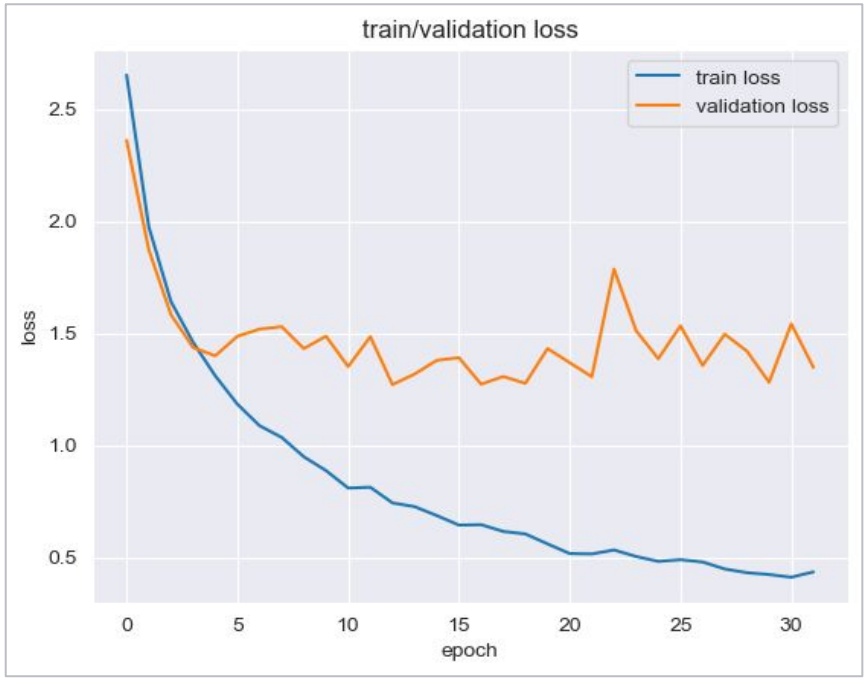
validation loss : 1.3064

f1 score : 0.6652

public 점수 : 0.6067131031

private 점수 : 0.6057289787

epoch : 22



04

성능 개선

성능 개선 요약

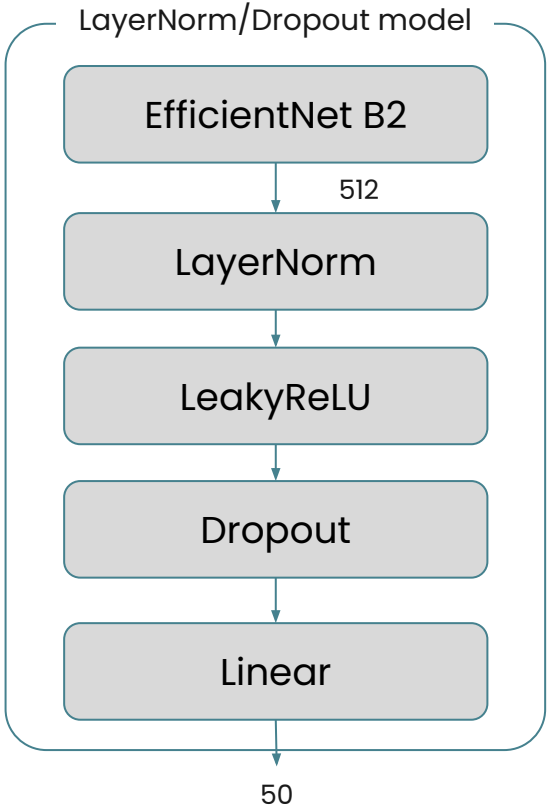
데이터 준비 → 모델 구성 → 모델 훈련 및 성능 검증

- **오버피팅 방지**
  - 데이터 증강
  - CutMix
- **Imbalance 보완**
  - Stratify 적용
  - Sampler
  - Style Transfer

- **모델 개선**
  - Tiny ViT
  - Layernorm
  - GELU
- **오버피팅 방지**
  - Dropout
- **일반화된 모델**
  - k-Fold
  - 다른 모델간의 앙상블

- **학습 성능 개선**
  - LRScheduler
  - AdamW
  - SAM Optimizer
  - Combination
- **Imbalance 보완**
  - Focal Loss
- **평가 점수 향상**
  - TTA

모델 개선 - **LayerNorm**, 오버피팅 방지 - **Dropout**



현재 모델 결과

train loss : 0.4852
validation loss : 1.2672
f1 score : 0.6769
public 점수 : 0.6294
private 점수 : 0.6237

baseline model

train loss : 0.5161
validation loss : 1.3064
f1 score : 0.6652
public 점수 : 0.6067
private 점수 : 0.6057

오버피팅 방지 – 데이터 증강

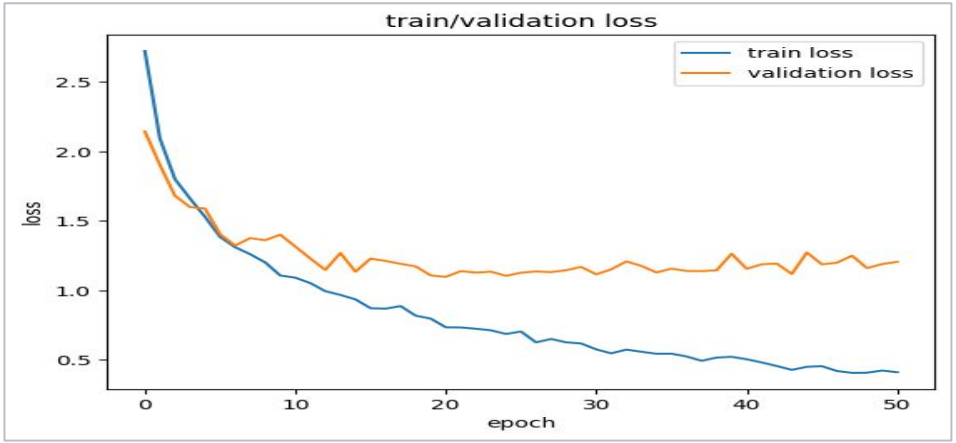
- Resize(520, 520)
  - 대회의 test 이미지는 기본 크기의 ¼ 크기이므로 260의 2배로 resize를 진행
- RandomCrop(260, 260)
  - EfficientNet B2의 기본 input size
- Transpose : 행렬 스왑
- HorizontalFlip : 좌우 반전
- VerticalFlip : 상하 반전
- ShiftScaleRotate : 이동, scale 조절, 회전
- HueSaturationValue : 색조 변환
- RandomBrightnessContrast : 명도 대비
- ChannelShuffle : 채널간 shuffle
- CoarseDropout : 검은 픽셀 추가

현재 모델 결과

train loss : 0.5034
validation loss : 1.1546
f1 score : 0.6903
public 점수 : 0.6585
private 점수 : 0.6482

이전 결과

train loss : 0.4852
validation loss : 1.2672
f1 score : 0.6769
public 점수 : 0.6294
private 점수 : 0.6237



학습 성능 개선 - **LRScheduler**

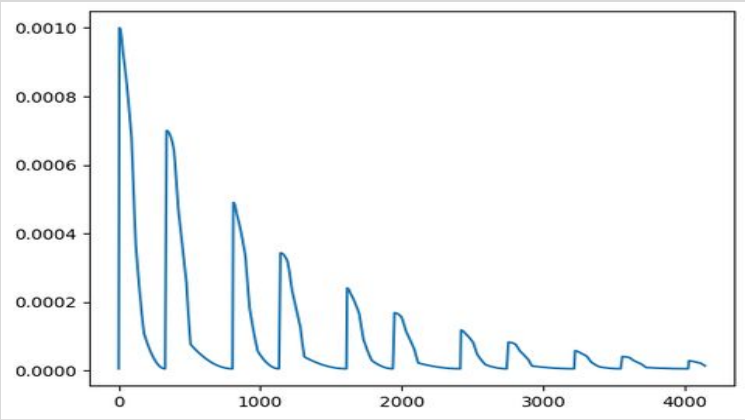
- CosineAnnealingWarmUpRestarts 사용
  - T_0 : 5 epoch
  - eta_max : 0.001
  - T_up : 50 iter
  - T_multi : 1
  - gamma : 0.7
  - T_min : 5e-6

현재 모델 결과

train loss : 0.2683
validation loss : 0.6997
f1 score : 0.7737
public 점수 : 0.7435
private 점수 : 0.7327

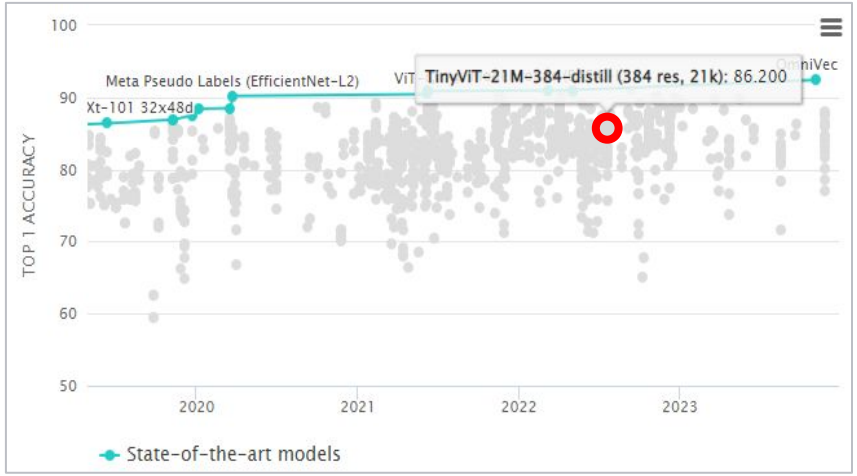
이전 결과

train loss : 0.5034
validation loss : 1.1546
f1 score : 0.6903
public 점수 : 0.6585
private 점수 : 0.6482



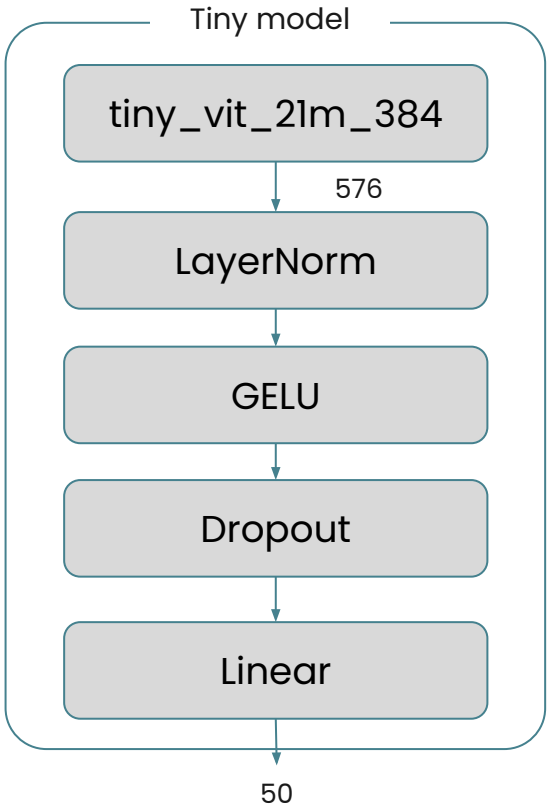
모델 개선 - **TinyViT**

- TinyViT-21M-384-distill (384 res, 21k) 사용
  - ViT 기반 작은 모델
  - Knowledge distillation 사용
  - Top1 Accuracy : 86.2%
  - Params 수 : 21M
  - GFLOPs : 13.8



	Model	Top-1 (%)	Top-5 (%)	#Params (M)	MACs (G)	Throughput (images/s)	Input	Arch.
5-10M #Params	MoblieViT-S [46]	78.4	-	6	1.8	2,661	256	Hybrid
	ViTAS-DeiT-A [60]	75.5	92.4	6	1.3	3,504	224	Trans
	GLiT-Tiny [9]	76.3	-	7	1.5	3,262	224	Trans
	Mobile-Former-214M [14]	76.7	-	9	0.2	3,105	224	Hybrid
	CrossViT-9 [10]	77.1	-	9	2.0	2,659	224	Trans
	TinyViT-5M (ours)	**79.1**	94.8	5.4	1.3	3,060	224	Hybrid
	TinyViT-5M⚗ (ours)	**80.7**	95.6	5.4	1.3	3,060	224	Hybrid
11-20M	ResNet-18 [28]	70.3	86.7	12	1.8	8,714	224	CNN
	PVT-Tiny [66]	75.1	-	13	1.9	2,791	224	Trans
	ResT-Small [81]	79.6	94.9	14	2.1	2,037	224	Trans
	LeViT-256 [24]	81.6	-	19	1.1	7,386	224	Hybrid
	CoaT-Lite Small [71]	81.9	95.6	20	4.0	1,138	224	Trans
	TinyViT-11M (ours)	**81.5**	95.8	11	2.0	2,468	224	Hybrid
	TinyViT-11M⚗ (ours)	**83.2**	96.5	11	2.0	2,468	224	Hybrid
>20M	DeiT-S [64]	79.9	95.0	22	4.6	2,276	224	Trans
	T2T-ViT-14 [74]	81.5	95.7	21	4.8	1,557	224	Trans
	AutoFormer-S [11]	81.7	95.7	23	5.1	1,341	224	Trans
	Swin-T [43]	81.2	95.5	28	4.5	1,393	224	Trans
	CrossViT-15 [10]	82.3	-	28	6.1	1,306	224	Trans
	EffNet-B5 [62]	83.6	96.7	30	9.9	330	456	CNN
	TinyViT-21M (ours)	**83.1**	96.5	21	4.3	1,571	224	Hybrid
	TinyViT-21M⚗ (ours)	**84.8**	97.3	21	4.3	1,571	224	Hybrid
	TinyViT-21M⚗ ↑384 (ours)	**86.2**	97.8	21	13.8	394	384	Hybrid
	TinyViT-21M⚗ ↑512 (ours)	**86.5**	97.9	21	27.0	167	512	Hybrid

모델 개선 - TinyViT



현재 모델 결과

train loss : 0.3039
validation loss : 0.5419
f1 score : 0.8214
public 점수 : 0.7949
privite 점수 : 0.7916

이전 결과

train loss : 0.2683
validation loss : 0.6997
f1 score : 0.7737
public 점수 : 0.7435
privite 점수 : 0.7327

학습 성능 개선 - **AdamW**

[**Adam**]
Momentum + RMSProp 의 장점을 취합해 놓은 것

[**AdamW**]
AdamW는 Adam 옵티마이저의 변형

[**Adam** vs **AdamW**]
- Adam에서는 기울기(g)를 정의할때 weight decay를 적용하여 가중치(θ)를 업데이트할 때 간접적으로 반영
- AdamW는 기울기에 적용하던 weight decay를 가중치를 계산할 때 직접적으로 반영되도록 변경

현재 모델 결과

train loss : 0.0722
validation loss : 0.5830
f1 score : 0.8378
public 점수 : 0.8133
private 점수 : 0.7986

이전 결과

train loss : 0.3039
validation loss : 0.5419
f1 score : 0.8214
public 점수 : 0.7949
privite 점수 : 0.7916

오버피팅 방지 - **CutMix**

- 2019년 naver에서 제출한 논문
- 서로 다른 이미지의 부분 영역을 합쳐서 하나의 이미지로 만드는 기법
- Mix된 영역 만큼 loss값을 조정해 주어야 함
- 훈련할때 50% 확률로 CutMix를 진행하여 훈련

현재 모델 결과

train loss : 0.9142
validation loss : 0.5076
f1 score : 0.8198
Public 점수 : 0.8038
Private 점수 : 0.7885

ViT+AdamW 결과

train loss : 0.0722
validation loss : 0.5830
f1 score : 0.8378
public 점수 : 0.8133
private 점수 : 0.7986

원본 CutOut CutMix

Imbalance 보완 - Focal Loss

[Imbalance 상황에서 CrossEntropy의 문제점]

- class가 적은 쪽은 어려운 문제, 많은 쪽은 쉬운 문제로 정의할 수 있음
- Imbalance상황에서는 loss 합산시 어려운 문제보다 쉬운 문제의 loss가 압도적으로 커짐
- 결국 쉬운 문제 위주로 학습

[해결책 1 - Balanced Cross Entropy]

- loss계산할때 가중치를 곱해주어서 balance를 조절 (알파)
- 잘 학습될 경우 어려운 문제가 쉬운 문제가 되면서 밸런스가 깨짐

[해결책 2 - Focal Loss]

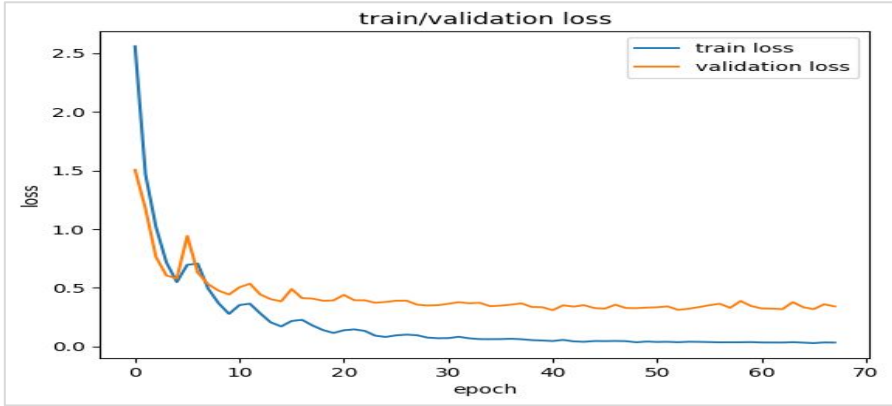
- 가중치를 임의로 정하지 말고 (1-p)**`감마`로 조절 (p = exp(-crossentopy()))
- `감마`값을 조절하면서 loss를 적절히 조율할 수 있음

현재 모델 결과

train loss : 0.0353
validation loss : 0.3287
f1 score : 0.8424
public 점수 : 0.8107
privite 점수 : 0.8079

ViT+AdamW 결과

train loss : 0.0722
validation loss : 0.5830
f1 score : 0.8378
public 점수 : 0.8133
private 점수 : 0.7986



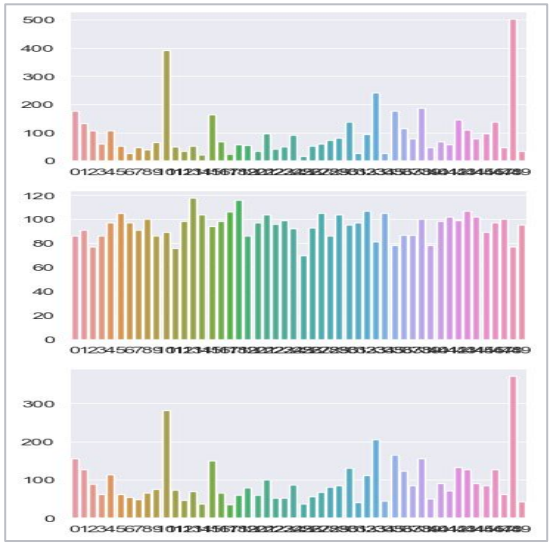
Imbalance 보완 - Sampler

- DataLoader의 sampler에 weight를 주어 데이터 추출 조절
- 많은 쪽은 downsampling, 적은 쪽은 upsampling
- 중복 추출되는 문제는 transformer에 의존함
- 적절한 비중이 필요

기본

총 작품수 / 작가의 작품수

총 작품수 / log(작가의 작품수)

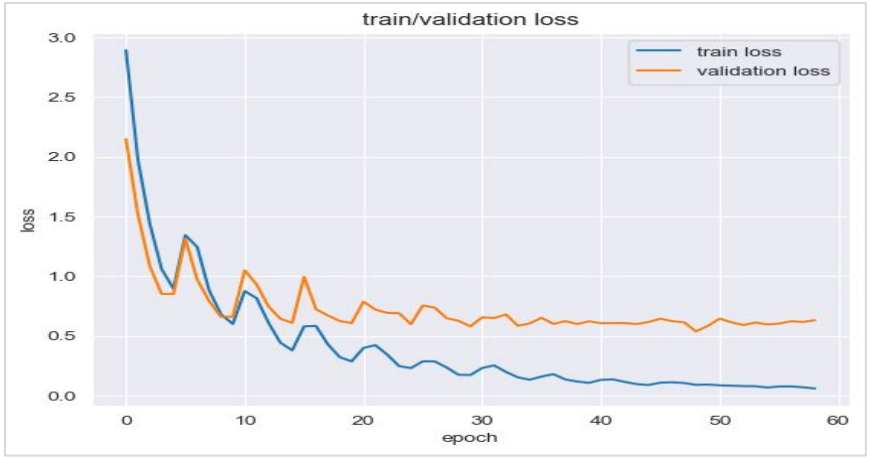


현재 모델 결과

train loss : 0.0888
validation loss : 0.5362
f1 score : 0.8332
public 점수 : 0.8171
private 점수 : 0.8122

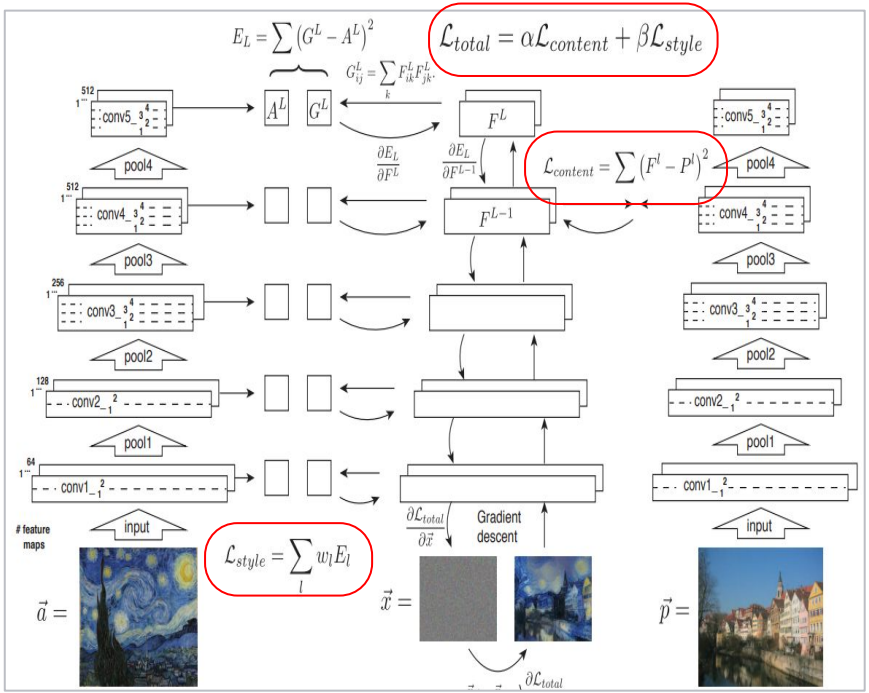
ViT+AdamW 결과

train loss : 0.0722
validation loss : 0.5830
f1 score : 0.8378
public 점수 : 0.8133
private 점수 : 0.7986



Imbalance 보완 - Style Transfer 활용

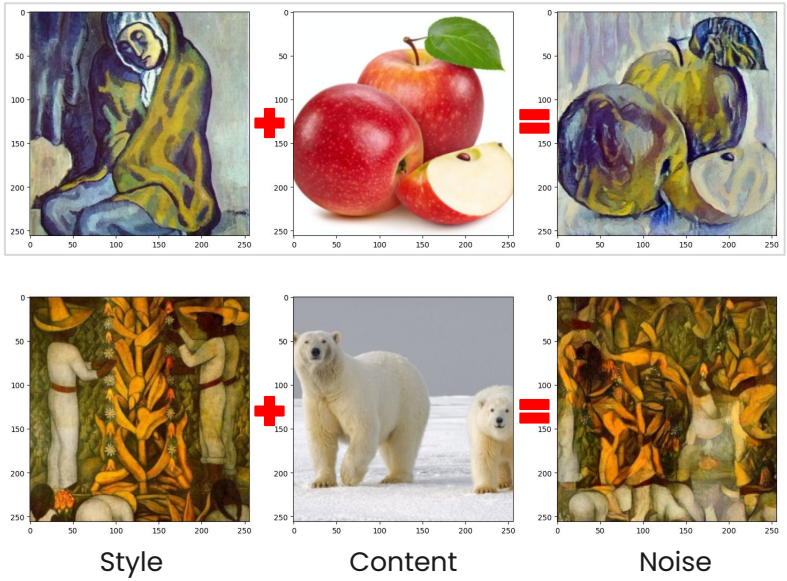
- 그림의 style을 옮겨주는 기법(2016년 논문)
- 이미지 구성
  - 작가의 그림 style
  - 작가의 화풍을 입힐 그림 content
  - style과 content를 학습하는 noise
- Loss
  - Lcontent
    - noise와 content의 Loss
    - conv4-2의 feature map L2norm
  - Lstyle
    - noise와 style의 Loss
    - 각 레이어의 feature map 간 내적 (gram) 후 L2norm
  - Ltotal = α*Lcontent +β*Lstyle
- 모델
  - vgg19 model 이용



Style Transfer Loss

Imbalance 보완 - Style Transfer 결과

- 작품수 하위 11명의 작가에 대해 5개 작품 추가
- 작품수 하위 6명의 작가에 대해 5개 작품 추가
- 총 85장의 작품을 생성



현재 모델 결과

train loss : 0.0525
validation loss : 0.6050
f1 score : 0.8374
public 점수 : 0.8202
private 점수 : 0.8126

ViT+AdamW 결과

train loss : 0.0722
validation loss : 0.5830
f1 score : 0.8378
public 점수 : 0.8133
private 점수 : 0.7986

학습 성능 개선 - SAM Optimizer

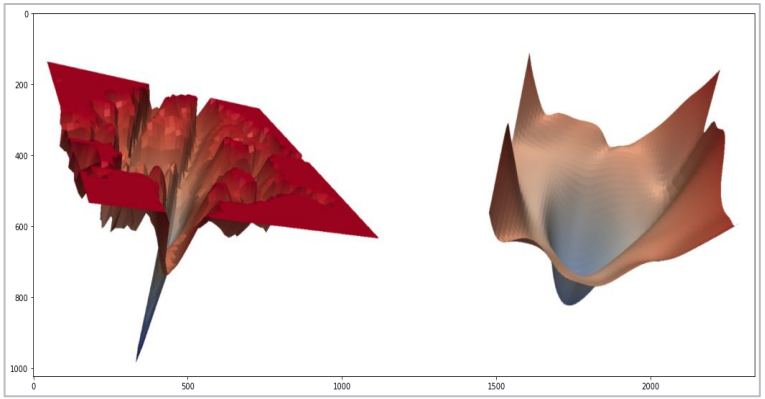
- 구글 리서치의 연구 결과
- 2020년 ImageNet 데이터셋에서 EfficientNet-L2-475에 적용하여 SOTA 달성
- Loss의 최적화와 Loss의 Sharpness 최적화로 구성됨
  - 2번의 forward와 2번의 backward 수행
- 점수의 상승 없이 학습시간이 1.5배 이상 증가하여 이후 모델에는 적용하지 않음

현재 모델 결과

train loss : 0.0969
validation loss : 0.5526
f1 score : 0.8253
public 점수 : 0.8050
private 점수 : 0.7945

ViT+AdamW 결과

train loss : 0.0722
validation loss : 0.5830
f1 score : 0.8378
public 점수 : 0.8133
private 점수 : 0.7986



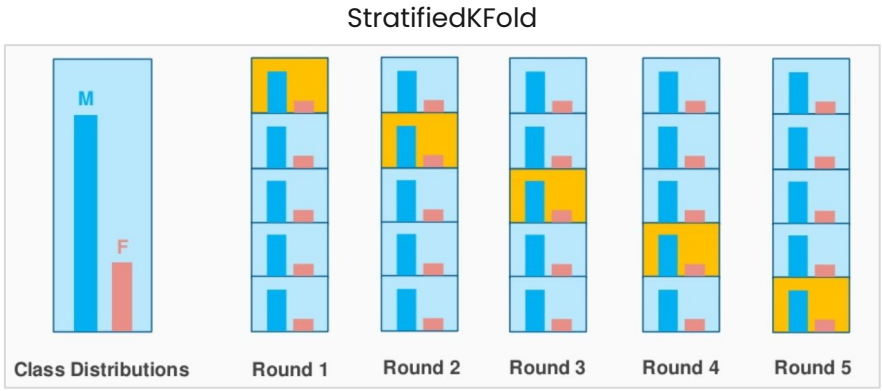
Loss 변화

학습성능 개선 - **Combination**

S: Sampler, F: Focal Loss, C: CutMix, T: Style Transfer, A: Color Augmentation

Combination	Train loss	Validation loss	F1 score	Public score	Private score
S+F+C	0.4367	0.3063	0.8425	0.8117	0.8062
S+F+C+T	0.3816	0.3440	0.8375	0.8210	0.8087
S+F+C+A	0.5124	0.2909	0.8461	0.8139	0.8154
S+F+T	0.0819	0.4218	0.8226	0.7981	0.7944
S+F+A	0.1190	0.3509	0.8348	0.7889	0.7976

일반화된 모델 - **k-Fold** 구조

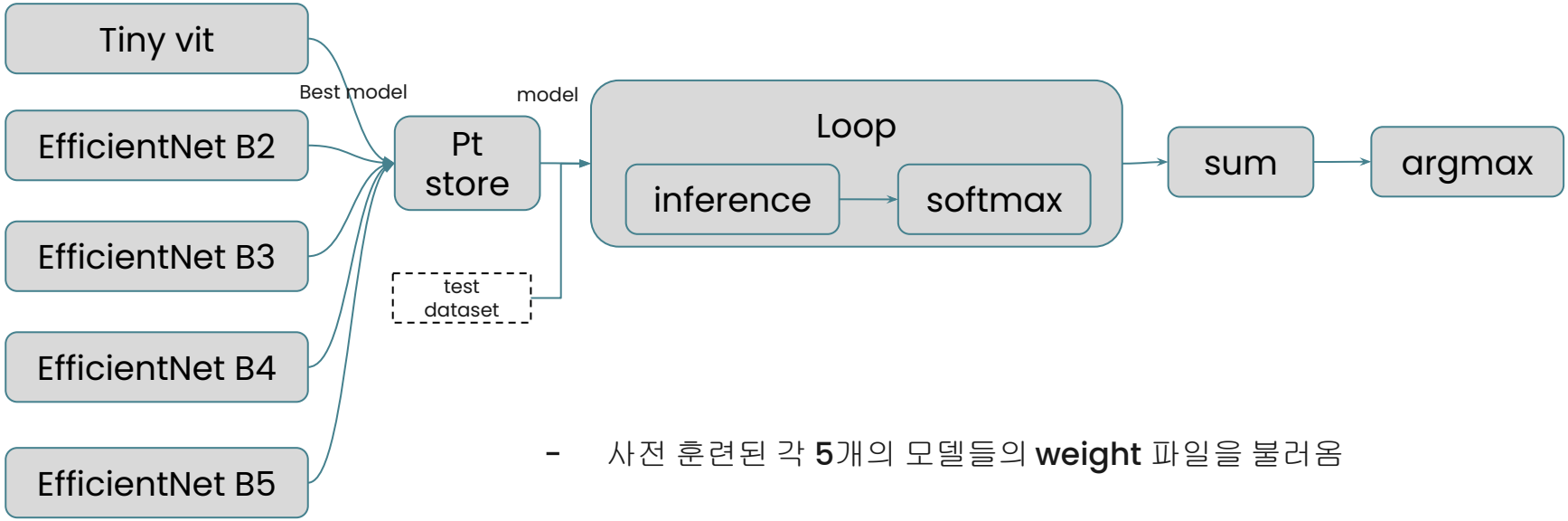


일반화된 모델 - k-Fold 결과

k-Fold	Train loss	Validation loss	F1 score	Public score	Private score
Model 1	0.7591	0.4867	0.8393	-	-
Model 2	0.7585	0.4382	0.8414	-	-
Model 3	0.9172	0.6001	0.8119	-	-
Model 4	0.8001	0.5264	0.8362	-	-
Model 5	8522	0.5466	0.8248	-	-
K-Fold	-	-	-	0.8412	0.8422

일반화된 모델 - **Ensemble** 구조

EfficientNet B2~B5 and TinyVit 총 5개의 모델에 대해서 Ensemble 적용



- 사전 훈련된 각 **5**개의 모델들의 **weight** 파일을 불러옴
- **Load** 된 각 모델들을 **Inference**
- 결과값에 각 모델중 **Accuracy**가 **max**값을 갖는 모델의 결과를 저장

일반화된 모델 - **Ensemble**에 사용한 모델 비교

EfficientNet B2~B5

EfficientNet은 Neural Architecture Search(신경망 구조 탐색)을 사용하여 만들어진 효율적인 컨볼루션 신경망(CNN)

EfficientNet은 다음 세 가지 요소를 조절하여 모델의 성능과 효율성을 균형있게 유지

1. 네트워크 깊이(Depth): 층의 수를 조절하여 모델의 깊이를 변화

2. 너비(Width): 각 층의 채널 수를 조절하여 네트워크의 폭을 조절

3. 해상도(Resolution): 이미지 입력의 해상도를 변경하여 모델의 해상도를 조절

	B2	B3	B4	B5
Input	260	300	380	456
Output	1280	1408	1792	2048

Tiny Vit

Tiny ViT는 Vision Transformer(비전 트랜스포머) 아키텍처의 간소화된 버전

1. 작은 모델 크기: Tiny ViT는 더 적은 수의 파라미터를 가지고 있어 메모리 요구량이 적고, 속도가 빠름

2. 이미지 특성을 토큰으로 변환: 입력 이미지를 작은 패치(patch)로 나누고 각 패치를 표현하기 위해 초기 임베딩을 수행합니다. 이후에는 트랜스포머의 attention mechanism을 활용하여 이러한 패치들 간의 관계를 학습

3. 학습된 특성 추출: 트랜스포머 레이어를 통해 이미지의 전역적인 특성을 추출하고, 이를 사용하여 분류 작업을 수행

일반화된 모델 - **Ensemble 결과**

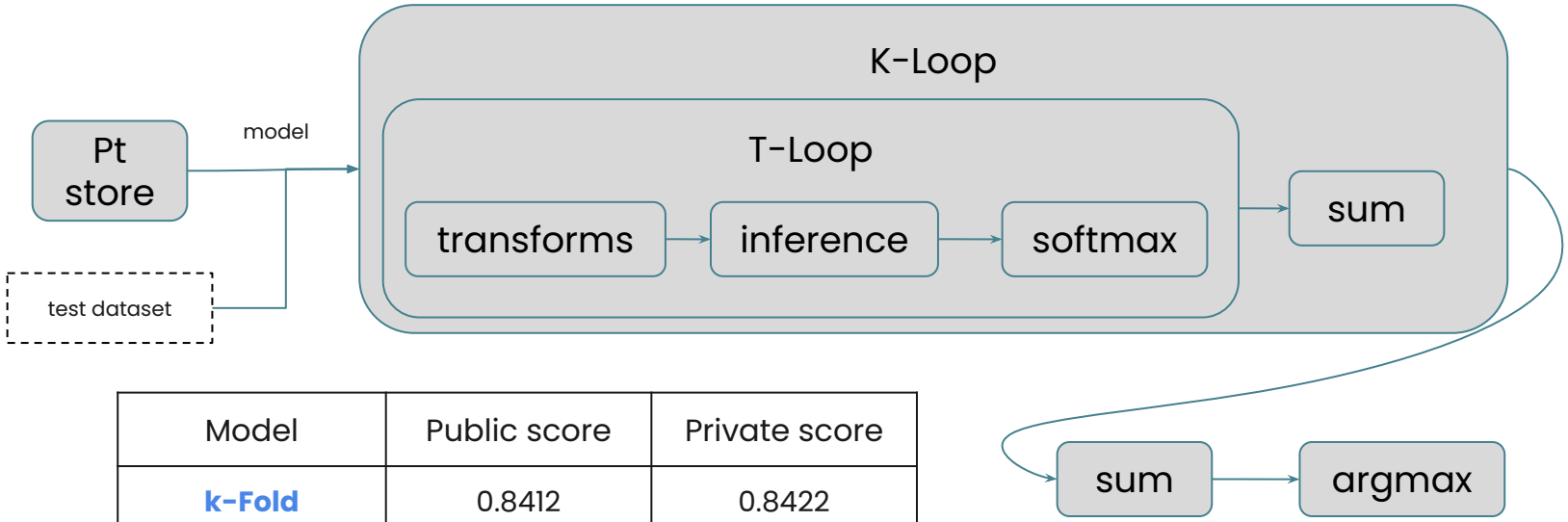
a. 앙상블 적용 전

Model	Train loss	Validation loss	F1 score	Public score	Private score
EfficientNet B2	0.2402	0.9830	0.7025	0.7061	0.6984
EfficientNet B3	0.2116	0.9855	0.7237	0.6932	0.6701
EfficientNet B4	0.2210	0.9022	0.7435	0.7435	0.7327
EfficientNet B5	0.1945	0.8469	0.7536	0.7221	0.7083
TinyViT	0.2493	0.4537	0.8126	0.8007	0.7840

b. 앙상블 적용 후

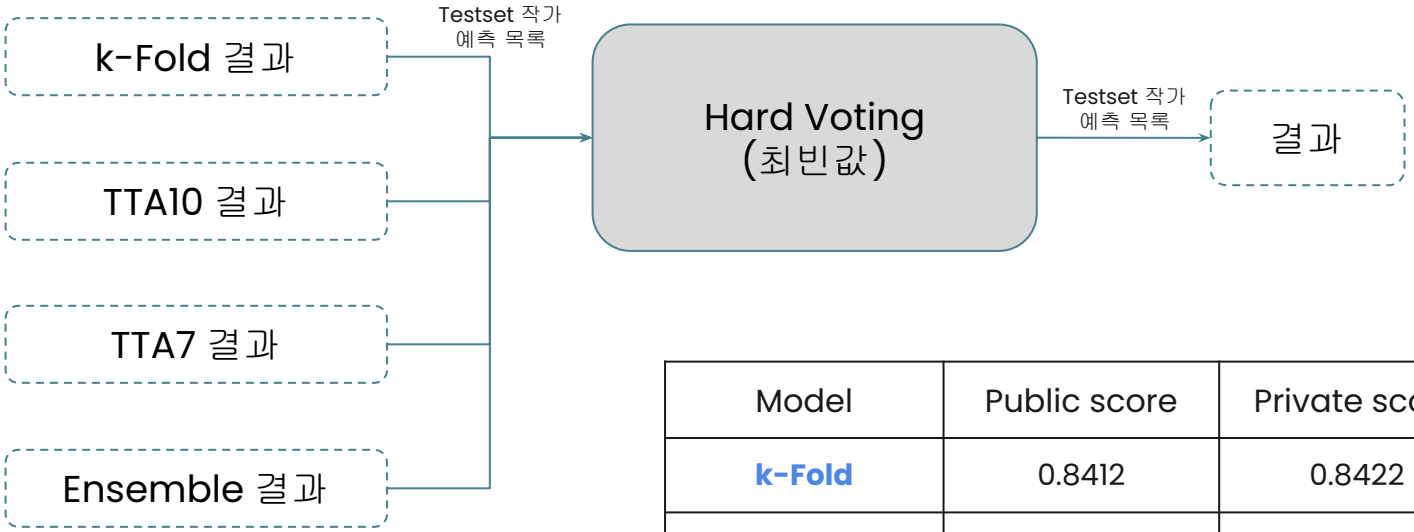
Ensemble(EfficientNet B2~B5)	-	-	-	0.7910	0.7737
Ensemble(EfficientNet B2~ B5 + Tiny Vit)	-	-	-	0.8197	0.8144

평가 점수 향상 - TTA



Model	Public score	Private score
k-Fold	0.8412	0.8422
TTA10	0.8418	0.8424
TTA7	0.8412	0.8447
TTA5	0.8412	0.8416

평가 점수 향상 - **Hard Voting**



Model	Public score	Private score
k-Fold	0.8412	0.8422
TTA10	0.8418	0.8424
Hard Voting	0.8423	0.8461

05

최종 결과

최종 결과

Imbalance 처리

- Stratify 옵션 사용
- dataloader의 sampler 기능 적용
- Focal Loss를 이용함
- 추가 그림 생성

모델 구축

- TinyViT를 최종 모델로 선택
- LayerNorm
- GELU
- Dropout

오버피팅 방지와 학습 성능 개선

- 데이터 증강과 CutMix 사용
- Early Stopping을 이용하여 훈련 종료
- LR Scheduler를 도입
- AdamW 사용

일반화된 모델과 점수 향상

- k-Fold 로 앙상블 진행
- CNN과 ViT 모델과의 Ensemble 진행
- TTA 적용

06

보완 사항

학습 데이터 보완

- Generation AI 적용
- VAE
- CGGAN
  - 논문에 나온 구조를 바탕으로 conv layer를 확대하여 생성 시도
  - 512x512 이미지 사이즈로 생성되도록 시도하였으나 제대로 생성이 되질 않음
  - 판별자는 빠른 수렴을 보인 반면 생성자의 학습이 개선되지 않았음
- Conditional GAN
  - CGGan으로 어느정도 성과를 보이면 진행할 예정이었으나 CGGAN 성능이 안나와서 취소함
- CycleGAN
  - 화풍을 입히는 예제도 있으며 고해상도 이미지를 대비하여 논문이 작성됨
  - Style Transfer와 비교하여 퀄리티를 높이기 위한 노력과 학습에 드는 시간을 고려하여 Style Transfer 선택
- Stable Diffusion
  - 시도는 해보았지만 시간 부족으로 제대로 훈련을 진행 못 해봄

모델 보완

- 모델
- 다양한 모델 적용
  - 하드웨어 사양과 학습 시간을 고려하여 작은 parameter를 가진 모델로 테스트를 진행
  - Basic한 ViT를 직접 만들어 보고 공유되어 있는 모델과 성능 비교를 해도 좋았을것 같음
- 장르 정보에 대한 활용
- 장르에 대한 예측 모델을 만들고 훈련시에는 Force Teaching을 시킴으로써 장르를 도출하는 모델을 만드는 것도 시도해봄직함
- EDA에서 보았던 여러 장르를 가진 작가에 대한 처리 필요

Q&A

감사합니다