H&E 염색된 조직 이미지로부터 유전자 발현 예측

조건우, 문지환, 백찬형

Department of Artificial Intelligence Engineering,
Chosun University, Korea

2024.12.10

조선대학교 CHOSUN UNIVERSITY

Contents

목차

I. 과제 배경

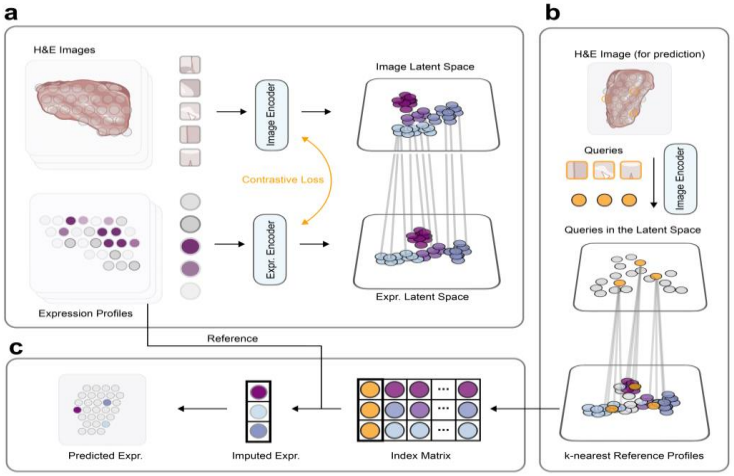
- 현대 의학의 발전과 함께 데이터 기반 의료 접근법이 주목받고 있다.
- 특히 유전자 데이터를 활용한 개인 맞춤형 의료가 주목받고 있는 상황이다.
- 하지만 이러한 의료 데이터는 복잡하고 대량의 데이터로 구성되어 있으며, 이를 효율적으로 처리하고 분석하기 위한 도구로 인공지능이 활용되어진다.
- 위와 같이 기존의 유전자 발현 분석은 시간과 비용이 많이 드는 복잡한 과정이다 따라서 AI를 활용하여 이 과정을 높은 정확도로 간소화 시키는 것이 주요 목표이다.

I. 과제 배경

Spatially Resolved Gene Expression Prediction from H&E Histology Images via Bi-modal Contrastive Learning

Ronald Xie[1,2,3,4] Kuan Pang[1,2,4] Sai W. Chung[1,5] Catia T. Perciani[1,5]
Sonya A. MacParland[1,5] Bo Wang[1,2,3*] Gary D. Bader[1,3,4,6*]

[1]University of Toronto, [2]Vector Institute, [3]University Health Network,
[4]The Donnelly Centre, [5]Toronto General Hospital Research Institute,
[6]Canadian Institute for Advanced Research (CIFAR)
{ronald.xie, kuan.pang, sai.chung,
catia.perciani, gary.bader}@mail.utoronto.ca,
Sonya.MacParland@uhnresearch.ca, bowang@vectorinstitute.ai



Journal of Life Science (생명과학회지)
Volume 22 Issue 8 / Pages.1034-1040 / 2012 / 1225-9918(pISSN) / 2287-3406(eISSN)
Korean Society of Life Science (한국생명과학회)

Quantitative Analyses of Cells using Photoshop after the H&E Staining of the Synovia of Osteoarthritis and Rheumatoid Arthritis Patients

H&E 염색 이미지의 포토샵 분석을 이용한 골관절염과 류마티스 관절염 활막 세포의 정량 분석

DOI QR Code

Park, Jin-Ah (Department of Biological Sciences, College of Natural Sciences, Kangwon National University) ;
Kim, Keun-Cheol (Department of Biological Sciences, College of Natural Sciences, Kangwon National University)
박진아 (강원대학교 자연과학대학 생명과학과) ; 김근철 (강원대학교 자연과학대학 생명과학과)

Received : 2012.04.27 Accepted : 2012.07.27 Published : 2012.08.30
https://doi.org/10.5352/JLS.2012.22.8.1034 Copy Citation KSCI

Download PDF

(Previous Next)

Abstract
Keywords
References

Abstract

Synovium is the soft tissue that lines the non-cartilaginous surfaces within joints. It has been reported that synovial cells are activated during the pathogenesis of rheumatoid arthritis. In this study, we quantitate and compare the cellular composition of synovia derived from individuals with non-inflammatory osteoarthritis (OA) and those with inflammatory rheumatoid arthritis (RA). Synovia from OA (n=8) and RA (n=5) patients were used for hematoxylin and eosin (H&E) staining. A light microscopic examination has shown that RA synovia were morphologically thickened and hypertrophied as compared to OA synovia. We also performed an immunohistochemistry (IHC) analysis to classify cell types in the synovia using CD68, CD90, or PGP9.5 markers. As a result, we obtained

- 국내외에서 H&E 유전체 이미지를 이용한 발현량 측정에 관해 다양한 연구가 이루어지고 있음.

II. 주제 설명

"H&E 염색된 조직 이미지로부터 유전자 발현 예측“

- **목표 :**

  H&E 염색된 조직 이미지를 입력으로 받아, 해당 이미지에서 유전자 발현 데이터를 정확히 예측하는 인공지능 모델을 개발하는 것

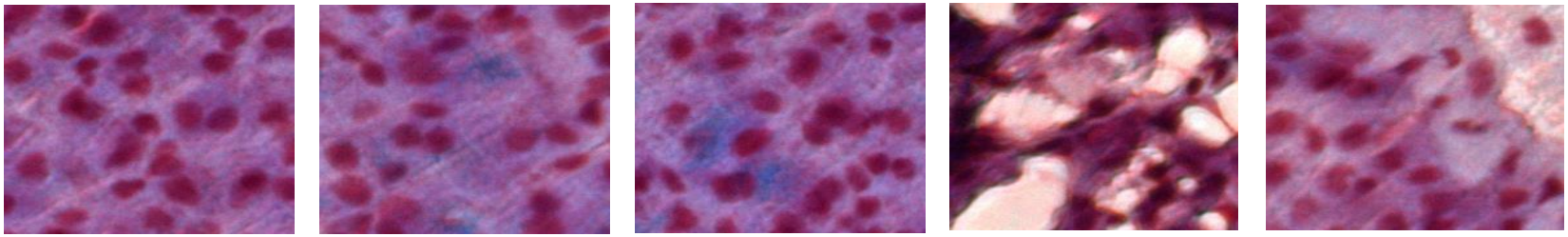
- **방식:**

  제공되어진 학습 데이터는 염색체 이미지와 유전자 발현 정보가 결합되어 있으며, 모델은 이 데이터를 학습하여 **이미지와 유전자 발현 간의 관계를 이해해야 함.**
  이후 평가 단계에서는 발현 정보가 마스킹 된 새로운 발현체 이미지를 받아서 해당 **이미지의 발현 정보를 가장 유사하게 맞추는 것이 목표**

III. 역할 분담

- 조건우 - 논문 분석, 모델링
- 문지환 - EDA(Exploratory Data Analysis), 모델링
- 백찬형 - EDA(Exploratory Data Analysis), 모델링

IV. 데이터셋 구성



```
[12] train_df.shape
```

```
(5593, 3469)
```

```
[8] train_df.head(10)
```

	ID	path	AL645608.7	HES4	TNFRSF18	TNFRSF4	SDF4	ACAP3	INTS11	MXRA8	...	MT-ATP8	MT-ATP6	MT
0	TRAIN_0000	./train/TRAIN_0000.png	0.000506	0.010635	-0.000213	-0.000846	1.512467	0.021131	0.024409	0.004109	...	-0.168265	2.858198	3.15
1	TRAIN_0001	./train/TRAIN_0001.png	0.000506	0.010635	-0.000213	-0.000846	1.508787	0.021131	0.024409	0.004109	...	-0.168265	2.392524	3.10
2	TRAIN_0002	./train/TRAIN_0002.png	-0.000415	0.005658	-0.000413	0.003148	0.109204	0.013978	0.049823	0.005327	...	0.189374	2.730253	3.21
3	TRAIN_0003	./train/TRAIN_0003.png	-0.000855	0.004366	0.000684	0.000865	0.503090	0.295115	0.303922	-0.004290	...	-0.158511	2.753111	2.89
4	TRAIN_0004	./train/TRAIN_0004.png	0.000506	0.010635	-0.000213	-0.000846	0.905195	0.021131	1.597454	0.004109	...	1.097993	3.760496	3.80
5	TRAIN_0005	./train/TRAIN_0005.png	-0.000855	0.004366	0.000684	0.000865	-0.027202	-0.004779	0.625543	-0.004290	...	-0.158510	2.811021	2.91
6	TRAIN_0006	./train/TRAIN_0006.png	-0.000855	0.004366	0.000684	0.000865	-0.027202	-0.004779	0.495353	-0.004290	...	-0.158510	2.604077	2.78
7	TRAIN_0007	./train/TRAIN_0007.png	-0.000855	0.004366	0.000684	0.000865	0.664473	-0.004779	0.004029	-0.004290	...	0.128436	2.667228	3.05
8	TRAIN_0008	./train/TRAIN_0008.png	0.000506	0.010635	-0.000213	-0.000846	0.083756	0.021131	0.024409	0.004109	...	1.244236	3.505259	3.96
9	TRAIN_0009	./train/TRAIN_0009.png	0.000506	0.010635	-0.000213	-0.000846	0.809897	0.021131	0.024409	0.004109	...	-0.168265	3.420015	3.36

10 rows × 3469 columns

주최측에서 제공한 “H&E 유전체 이미지와 이미제의 유전체 발현량 정보 CSV ”

- 다양한 H&E 유전체 이미지가 제공되어짐
- 총 5593개의 유전체 이미지와 1개의 이미지당 3469개의 유전체 feature가 존재

V. 모델(특징 추출기) 선정 과정

그래프 풀링을 활용한 공간전사체 데이터 스폿의 군집화 : Clustering Spots of Spatially Resolved Transcriptomics Data with Graph Pooling

Cited 0 time in Web of Science Cited 0 time in Scopus

Authors	이동주
Advisor	윤형진; 김광수
Issue Date	2023
Publisher	서울대학교 대학원
Keywords	공간전사체학 ; 군집화 ; 딥러닝 ; 그래프 신경망 ; 그래프 풀링
Description	학위논문(석사) -- 서울대학교대학원 : 공과대학 협동과정 바이오엔지니어링전공, 2023. 2. 윤형진 김광수.
Abstract	공간전사체학은 조직 슬라이드에서 세포의 위치 정보와 유전자 발현을 함께 측정할 수 있는 최신 기술이다. 사체 데이터는 스폿 단위로 유전자 발현을 측정하는데, 여러 가지 하위 분석을 위해서는 비슷한 스폿들을 군 요하다. 하지만 기존의 군집화 방법들은 공간 정보를 활용하지 않고 유전자 발현 정보만을 활용하여 군집화 라서 본 연구에서는 공간전사체학 기술로 얻을 수 있는 유전자 발현, 조직 이미지, 물리적 위치 정보를 모두 행하기 위해, 공간전사체 데이터를 그래프 형태로 모델링한 다음 그래프의 정점을 군집화하는 그래프 풀링 하였다. 유전자 발현 특성은 각 스폿의 정규화된 유전자 발현량을 사용하였으며, 스폿을 포함하는 H&E ResNet50 이미지 분류 모델의 전이학습을 통해 조직 이미지 특성을 추출하였다. 두 특성 행렬은 전역적 기준으로 최적의 랭크를 선정하는 비음수 행렬 분해를 통해 적절한 차원으로 축소하였다. 그래프를 구성하는 용했는데, 하나는 조직 이미지 특성을 유전자 발현과 함께 그래프 정점요소로 사용하는 방법이고 다른 하나 와 물리적 거리를 결합하여 간선 연결에 활용하는 방법이다. 그래프 풀링 신경망 모델의 풀링 레이어로는 DMoNPool을 활용하였다. 모델의 군집 결과와 조직학자의 표기의 ARI를 평가 지표로 삼아, 그래프의 구성 망 모델의 여러 조합에 대한 정량 평가를 수행하였다. 실험 결과에 따르면 조직 이미지를 활용하지 않고 들로 한 그래프를 DMoNPool 레이어 기반의 그래프 신경망 모델로 군집화 했을 때 가장 성능이 좋았으며, 이는 Leiden, stLearn, spaGCN보다 평균적으로 더 높은 성능이었다. 이를 통해 그래프 풀링 기반의 군집화가 공간 전사체 스폿의 군집화에 충분히 활용될 수 있음을 확인했다.

2 Related Works

2.1 Existing histology expression prediction approaches

Several existing approaches have shown promising results in predicting expression from histology images including HE2RNA[20], ST-Net[9], HisToGene[14], hist2rna[13], Hist2ST[26] and others[7, 24].

ST-Net and HisToGene are two of the most popular methods for predicting spatially resolved expression from H&E images. Both of these approaches frame the task of expression prediction as regression tasks trained in a feed-forward fashion. ST-Net uses a resnet50 image encoder followed by a fully connected layer where as HisToGene leverages a vision transformer backbone and an increased field of view.

Methods that predict tissue-level expression generally achieve good correlation but lack the ability to generate spatially resolved expression profiles (HE2RNA). Existing methods that are capable of generating spatially resolved expression predictions were either not quantitatively evaluated (hist2RNA), limited in terms of the predicted panel (ST-Net, Hist2ST, HisToGene), or prone to overfitting [24].

Both HisToGene and Hist2ST utilize spot-spatial relations to improve performance. However, our work challenges the necessity of this information, particularly in tissues with distinct and repetitive spatial patterns like human liver tissue. The implicit assumption that spatially adjacent regions should have similar representations compared to spatially distant regions may not be beneficial for performance in such cases. Hard coding position information could also lead to overfitting in data-scarce scenarios.

발현 정보를 추출할수 있는 정보량의 원천 => 이미지 데이터 이기때문에

서로 다른 발현제 이미지로부터 발현정보 feature를 잘 뽑아내는게 우선이라고 생각함.

기존의 논문들에서는 보통 resnet50을 특징 추출기로 사용함을 논문 분석을 통해 확인.

V. 모델(특징 추출기) 선정 과정

Torch vision 라이브러리에서 제공하는 cnn기반 다양한 모델들 기반으로 비교 실험 진행

계열	모델
ResNet 계열	resnet18, resnet34, resnet50, resnet101, resnet152
VGG 계열	vgg11, vgg13, vgg16, vgg19 등과 BatchNorm 버전
DenseNet 계열	densenet121, densenet169, densenet201, densenet161
Inception 계열	inception_v3
AlexNet	alexnet
SqueezeNet	squeezenet1_0, squeezenet1_1
MobileNet	mobilenet_v2, mobilenet_v3_large, mobilenet_v3_small
ShuffleNet	shufflenet_v2_x0_5, shufflenet_v2_x1_0, shufflenet_v2_x1_5, shufflenet_v2_x2_0
EfficientNet	efficientnet_b0, efficientnet_b1, ..., efficientnet_b7
RegNet	regnet_y_400mf, regnet_y_800mf, ..., regnet_y_32gf, regnet_x_400mf, ..., regnet_x_32gf
GoogLeNet	googlenet
MNASNet	mnasnet0_5, mnasnet0_75, mnasnet1_0, mnasnet1_3
Wide ResNet	wide_resnet50_2, wide_resnet101_2
ResNeXt	resnext50_32x4d, resnext101_32x8d

V. 모델(특징 추출기) 선정 과정

```python
def train(model, optimizer, train_loader, val_loader, scheduler, device):
    model.to(device)
    criterion = nn.MSELoss().to(device)

    best_loss = float('inf')
    best_model = None
```

```python
#Run!!
model = BaseModel()
#print(model)
model.eval()
optimizer = torch.optim.AdamW(params = model.parameters(), lr = CFG["LEARNING_RATE"])
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', factor=0.5, patience=2, threshold_mode='abs', min_lr=1e-8, verbose=True)

infer_model = train(model, optimizer, train_loader, val_loader, scheduler, device)
```

```
Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to /root/.cache/torch/hub/checkpoints/resnet50-0676ba61.pth
100%|██████████| 97.8M/97.8M [00:00<00:00, 159MB/s]
r50
Epoch [1], Train Loss: [0.06522], Val Loss: [0.04912]
Epoch [2], Train Loss: [0.04805], Val Loss: [0.04785]
Epoch [3], Train Loss: [0.04717], Val Loss: [0.04716]
Epoch [4], Train Loss: [0.04675], Val Loss: [0.04668]
Epoch [5], Train Loss: [0.04634], Val Loss: [0.04648]
Epoch [6], Train Loss: [0.04576], Val Loss: [0.04643]
Epoch [7], Train Loss: [0.04537], Val Loss: [0.04733]
Epoch [8], Train Loss: [0.04532], Val Loss: [0.04638]
Epoch [9], Train Loss: [0.04510], Val Loss: [0.04654]
Epoch [10], Train Loss: [0.04492], Val Loss: [0.04658]
```

다양한 논문에서 사용한 resnet50를 이용해 특징 추출후 linear regressor를 통해 측정된 mse loss

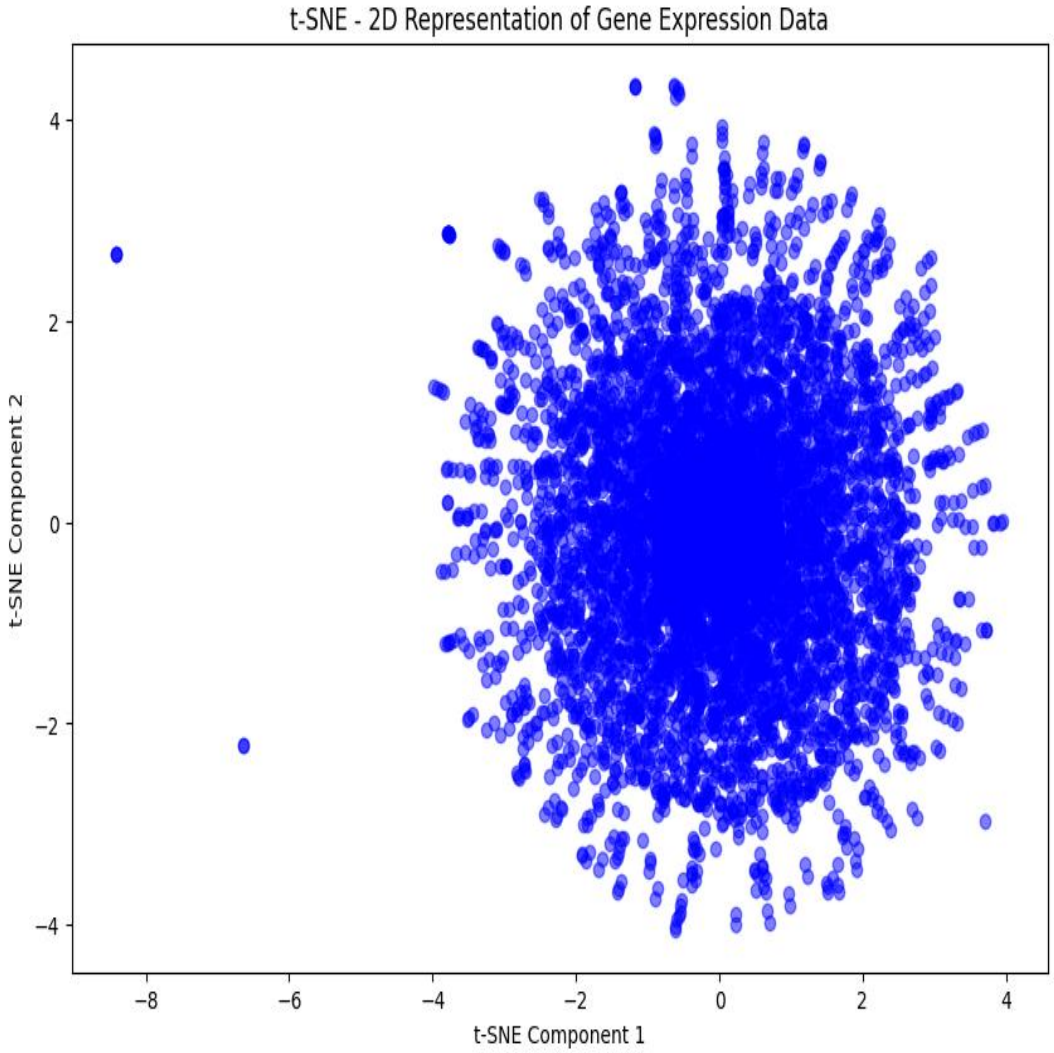
V. 모델(특징 추출기) 선정 과정

```
Downloading: "https://download.pytorch.org/models/googlenet-1378be20.pth" to /root/.cache/torch/hub/checkpoints/googlenet-1378be20.pth
100%|██████████| 49.7M/49.7M [00:00<00:00, 166MB/s]
googlenet
Epoch [1], Train Loss: [0.06575], Val Loss: [0.04722]
Epoch [2], Train Loss: [0.04890], Val Loss: [0.04715]
Epoch [3], Train Loss: [0.04792], Val Loss: [0.04680]
Epoch [4], Train Loss: [0.04739], Val Loss: [0.04641]
Epoch [5], Train Loss: [0.04694], Val Loss: [0.04632]
Epoch [6], Train Loss: [0.04636], Val Loss: [0.04625]
Epoch [7], Train Loss: [0.04580], Val Loss: [0.04626]
Epoch [8], Train Loss: [0.04618], Val Loss: [0.04646]
Epoch [9], Train Loss: [0.04638], Val Loss: [0.04599]
Epoch [10], Train Loss: [0.04558], Val Loss: [0.04615]
```

```
Downloading: "https://download.pytorch.org/models/efficientnet_b1_rwightman-bac287d4.pth" to /root/.cache/torch/hub/checkpoints/efficientnet_b1_rwightman-bac287d4.pth
100%|██████████| 30.1M/30.1M [00:00<00:00, 99.4MB/s]
efficientnet_b1
Epoch [1], Train Loss: [0.08031], Val Loss: [0.04789]
Epoch [2], Train Loss: [0.04899], Val Loss: [0.04705]
Epoch [3], Train Loss: [0.04775], Val Loss: [0.04673]
Epoch [4], Train Loss: [0.04710], Val Loss: [0.04632]
Epoch [5], Train Loss: [0.04669], Val Loss: [0.04607]
Epoch [6], Train Loss: [0.04638], Val Loss: [0.04598]
Epoch [7], Train Loss: [0.04607], Val Loss: [0.04597]
Epoch [8], Train Loss: [0.04581], Val Loss: [0.04594]
Epoch [9], Train Loss: [0.04552], Val Loss: [0.04598]
Epoch [10], Train Loss: [0.04527], Val Loss: [0.04599]
```

cnn 기반 모델 라이브러리인 torchvision의 모델들 실험 결과 googlenet, efficientnetb1이 resnet50보다 낮은 mse loss를 달성 => cnn기반 아키텍쳐의 모델은 googlenet, efficientnetb1 선정

VI. 전처리, 손실함수 구성



고차원의 유전체 feature를 t-SNE를 통해 차원축소를 진행해 보았을 때 이상치도 보임

큰 이상치들은 일반화 성능에 있어 큰 노이즈를 줌

Outlier scaler의 도입

VI. 전처리, 손실함수 구성

```
[12] train_df.shape
```

```
(5593, 3469)
```

```
[8] train_df.head(10)
```

	ID	path	AL645608.7	HES4	TNFRSF18	TNFRSF4	SDF4	ACAP3	INTS11	MXRA8	...	MT-ATP8	MT-ATP6	MT
0	TRAIN_0000	./train/TRAIN_0000.png	0.000506	0.010635	-0.000213	-0.000846	1.512467	0.021131	0.024409	0.004109	...	-0.168265	2.858198	3.1
1	TRAIN_0001	./train/TRAIN_0001.png	0.000506	0.010635	-0.000213	-0.000846	1.508787	0.021131	0.024409	0.004109	...	-0.168265	2.392524	3.1
2	TRAIN_0002	./train/TRAIN_0002.png	-0.000415	0.005658	-0.000413	0.003148	0.109204	0.013978	0.049823	0.005327	...	0.189374	2.730253	3.2
3	TRAIN_0003	./train/TRAIN_0003.png	-0.000855	0.004366	0.000684	0.000865	0.503090	0.295115	0.303922	-0.004290	...	-0.158511	2.753111	2.8
4	TRAIN_0004	./train/TRAIN_0004.png	0.000506	0.010635	-0.000213	-0.000846	0.905195	0.021131	1.597454	0.004109	...	1.097993	3.760496	3.8
5	TRAIN_0005	./train/TRAIN_0005.png	-0.000855	0.004366	0.000684	0.000865	-0.027202	-0.004779	0.625543	-0.004290	...	-0.158510	2.811021	2.9
6	TRAIN_0006	./train/TRAIN_0006.png	-0.000855	0.004366	0.000684	0.000865	-0.027202	-0.004779	0.495353	-0.004290	...	-0.158510	2.604077	2.7
7	TRAIN_0007	./train/TRAIN_0007.png	-0.000855	0.004366	0.000684	0.000865	0.664473	-0.004779	0.004029	-0.004290	...	0.128436	2.667228	3.0
8	TRAIN_0008	./train/TRAIN_0008.png	0.000506	0.010635	-0.000213	-0.000846	0.083756	0.021131	0.024409	0.004109	...	1.244236	3.505259	3.9
9	TRAIN_0009	./train/TRAIN_0009.png	0.000506	0.010635	-0.000213	-0.000846	0.809897	0.021131	0.024409	0.004109	...	-0.168265	3.420015	3.3

10 rows × 3469 columns

유전체 3469의 유전체 feature에서 특정 feature 들의 발현량의 값 범위가 매우 큰것을 확인

정규화 or 표준화를 통해 값의 scale을 일정하게 변환해주자 (log transform, z-score, norm)

=> 최종 선택 min max norm

VI. 전처리, 손실함수 구성

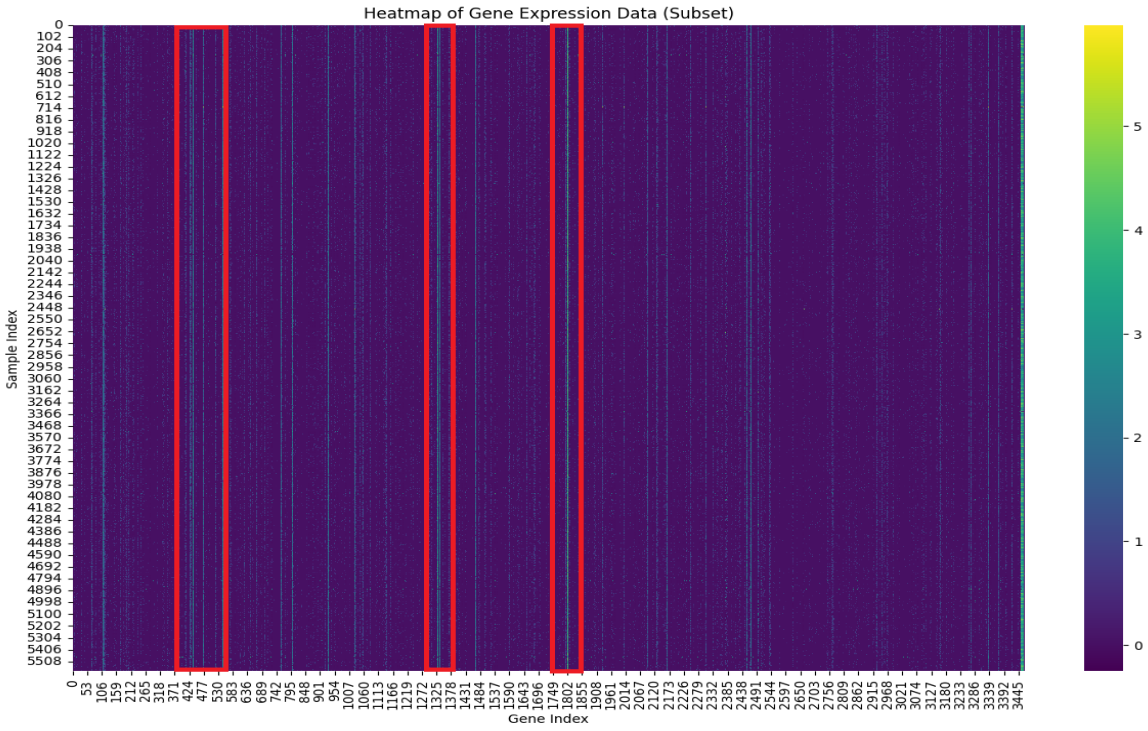
```
[12] train_df.shape
```

```
(5593, 3469)
```

```
[8] train_df.head(10)
```

	ID	path	AL645608.7	HES4	TNFRSF18	TNFRSF4	SDF4	ACAP3	INTS11	MXRA8	...	MT-ATP8	MT-ATP6	MT
0	TRAIN_0000	./train/TRAIN_0000.png	0.000506	0.010635	-0.000213	-0.000846	1.512467	0.021131	0.024409	0.004109	...	-0.168265	2.858198	3.1
1	TRAIN_0001	./train/TRAIN_0001.png	0.000506	0.010635	-0.000213	-0.000846	1.508787	0.021131	0.024409	0.004109	...	-0.168265	2.392524	3.1
2	TRAIN_0002	./train/TRAIN_0002.png	-0.000415	0.005658	-0.000413	0.003148	0.109204	0.013978	0.049823	0.005327	...	0.189374	2.730253	3.2
3	TRAIN_0003	./train/TRAIN_0003.png	-0.000855	0.004366	0.000684	0.000865	0.503090	0.295115	0.303922	-0.004290	...	-0.158511	2.753111	2.8
4	TRAIN_0004	./train/TRAIN_0004.png	0.000506	0.010635	-0.000213	-0.000846	0.905195	0.021131	1.597454	0.004109	...	1.097993	3.760496	3.8
5	TRAIN_0005	./train/TRAIN_0005.png	-0.000855	0.004366	0.000684	0.000865	-0.027202	-0.004779	0.625543	-0.004290	...	-0.158510	2.811021	2.9
6	TRAIN_0006	./train/TRAIN_0006.png	-0.000855	0.004366	0.000684	0.000865	-0.027202	-0.004779	0.495353	-0.004290	...	-0.158510	2.604077	2.7
7	TRAIN_0007	./train/TRAIN_0007.png	-0.000855	0.004366	0.000684	0.000865	0.664473	-0.004779	0.004029	-0.004290	...	0.128436	2.667228	3.0
8	TRAIN_0008	./train/TRAIN_0008.png	0.000506	0.010635	-0.000213	-0.000846	0.083756	0.021131	0.024409	0.004109	...	1.244236	3.505259	3.9
9	TRAIN_0009	./train/TRAIN_0009.png	0.000506	0.010635	-0.000213	-0.000846	0.809897	0.021131	0.024409	0.004109	...	-0.168265	3.420015	3.3

10 rows × 3469 columns



5593개의 이미지 데이터 모두에서 높은 발현량을 보이는 특정 유전체 feature의 존재를 확인

높은 발현량을 가지는 feature를 일반화 시키면 단순 mse loss는 커질것?

규제를 가한 새로운 loss함수를 선택해보자

VI. 전처리, 손실함수 구성

$$\text{Huber Loss} = \frac{1}{n}\sum_{i=1}^{n}\begin{cases}\frac{1}{2}(y_i-\hat{y}_i)^2, & \text{if } |y_i-\hat{y}_i| \le \delta \\ \delta\cdot|y_i-\hat{y}_i| - \frac{1}{2}\delta^2, & \text{if } |y_i-\hat{y}_i| > \delta\end{cases}$$

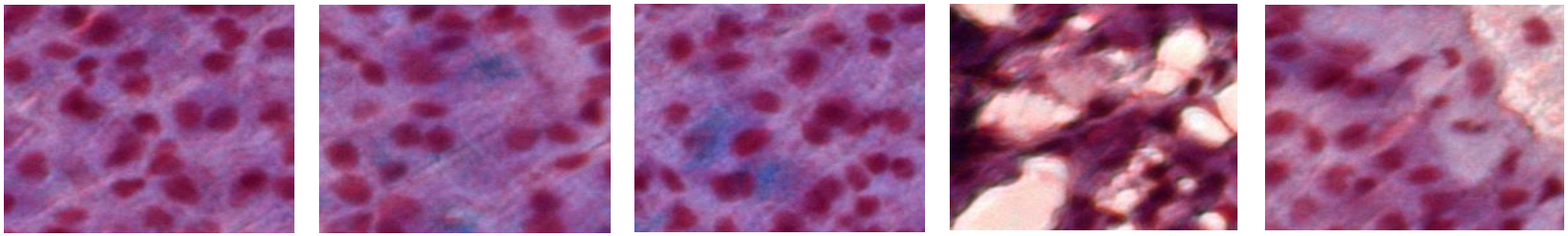
$$\text{Log-Cosh Loss} = \frac{1}{n}\sum_{i=1}^{n}\log(\cosh(y_i-\hat{y}_i))$$

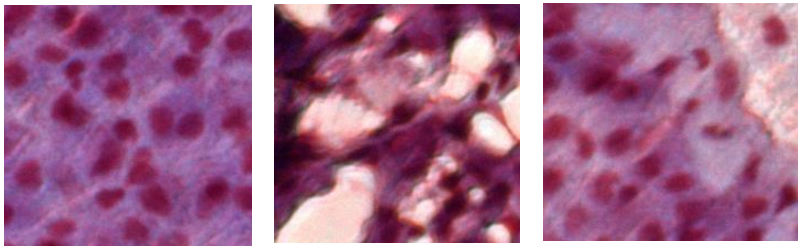
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

가설과 달리 규제를 가한 다양한 손실함수보다 단순 mse가 더 좋은 일반화 성능을 보임

=> 최종적으로 mse loss 기본 손실 함수로 결정

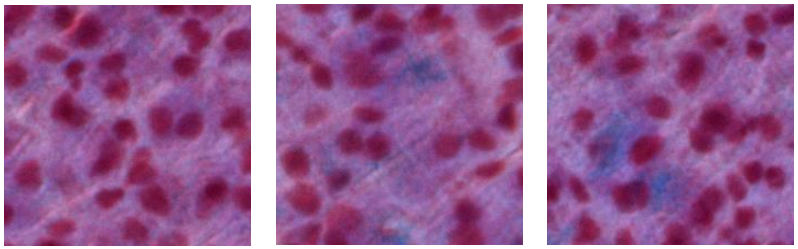
VII. 최종 모델, 손실함수 선정 (모델)





CNN:

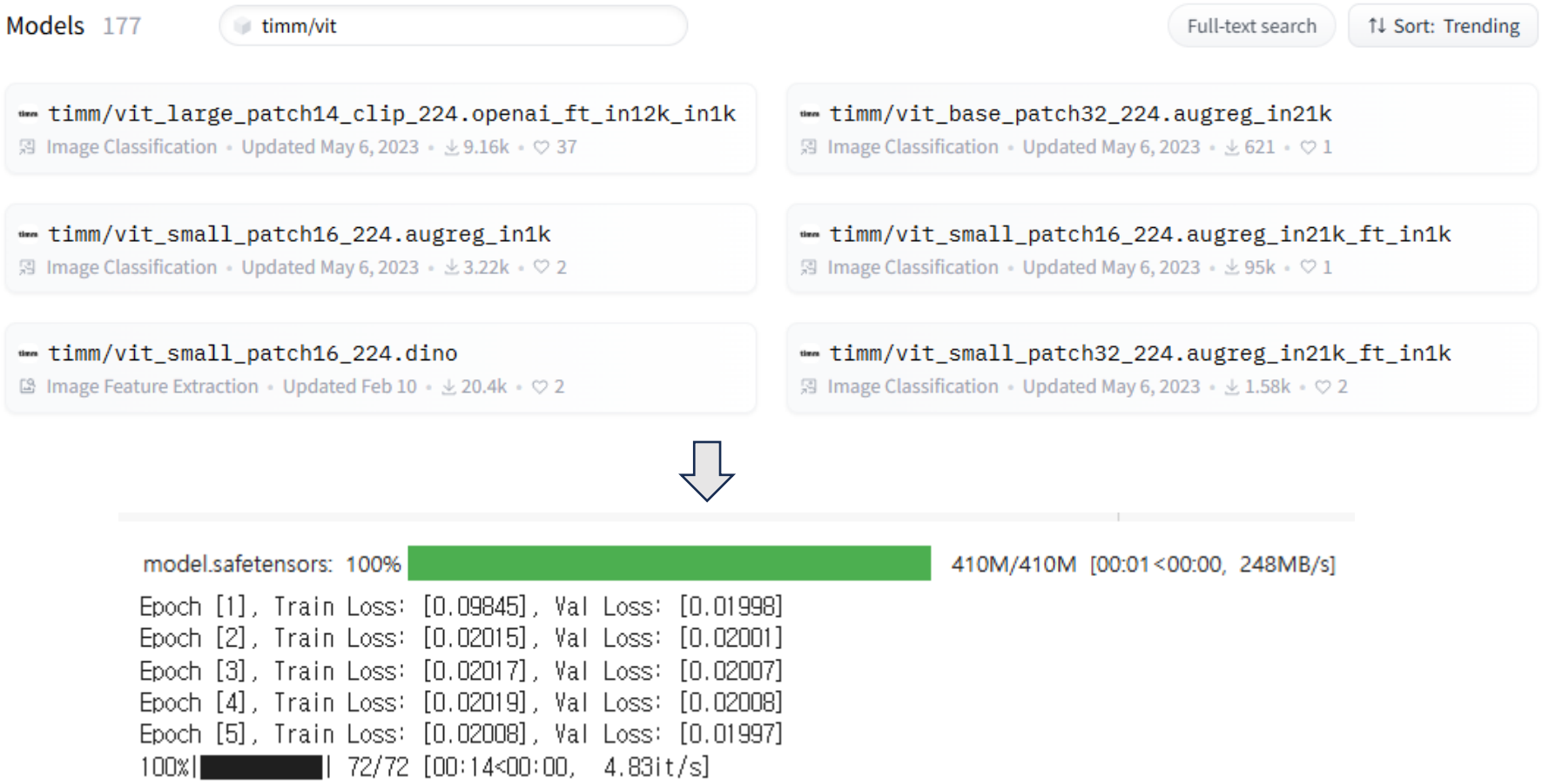
- 국소적인 명확한 차이가 있는 첫 번째 이미지와 같은 경우에 유리.
- 빠르고 경량화된 구조로 텍스처와 지역 패턴을 정확히 학습 가능.



ViT:

- 전역적인 관계(순서, 회전, 공간적 구성)가 중요한 두 번째 이미지와 같은 경우에 유리.
- 더 많은 데이터를 필요로 하지만, 복잡한 전역 정보와 구조적 패턴을 잘 학습.

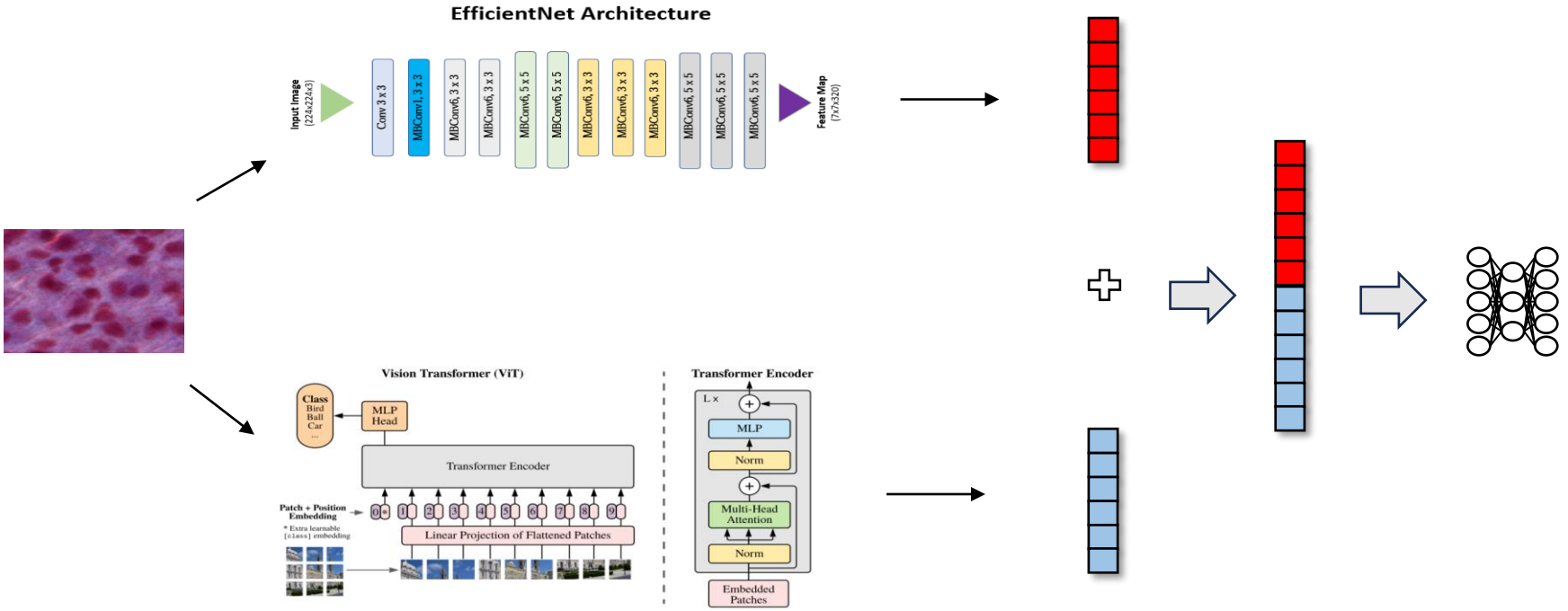
VII. 최종 모델, 손실함수 선정 (모델)



Timm라이브러리 이용 transformer기반 vitt모델들 실험 결과 resnet50보다 낮은 mse loss를 달성

=> vit기반 timm/vit_small_patch16_224.augreg_in1k 모델을 도입

VII. 최종 모델, 손실함수 선정 (모델)



지역적 local 특징에 특화된 cnn 기반 efficientnet b1 과 global한 특징에 특화된 transformer 기반 vit모델을 통해 나온 각 특징을 특징수준의 앙상블 을 통해 과적합 개선

출처 :
https://gaussian37.github.io/dl-concept-vit/
https://wisdomml.in/efficientnet-and-its-performance-comparison-with-other-transfer-learning-networks/

VII. 최종 모델, 손실함수 선정 (손실함수)

1. 리더보드

• 평가 산식 : Pearson Correlation Coefficient (PCC)

$$\text{Score} = \max\left(\frac{\text{MeanCorr}_{\text{Cells}} + \text{MaxCorr}_{\text{Genes}} + \text{HEG}_{\text{Corr}} + \text{HVG}_{\text{Corr}}}{4}, 0\right)$$

$\text{MeanCorr}_{\text{Cells}} = \frac{1}{n}\sum_{i=1}^{n} \text{Corr}(\mathbf{Pred}_i, \mathbf{True}_i)$: 샘플 i에 대한 예측값과 실제값 사이의 상관계수의 평균

$\text{MaxCorr}_{\text{Genes}} = \max\left(\text{Corr}(\mathbf{Pred}_j, \mathbf{True}_j)\right)$: 유전자 j에 대한 상관계수 중 가장 큰 값

HEG_{Corr}: HEG 리스트에 속한 유전자들에 대한 상관계수의 평균

HVG_{Corr}: HVG 리스트에 속한 유전자들에 대한 상관계수의 평균

※ HEG, HVG 상관계수 계산에 활용되는 유전자(Gene) 리스트는 공개하지 않습니다.

평가 산식 해석

1. **MeanCorr**$Cells$:
   - 샘플 i에 대한 예측값과 실제값 사이의 상관계수(Pearson Correlation Coefficient, PCC)의 평균.
   - **샘플 단위로 일관된 상관관계**를 유지하도록 모델링해야 함.
2. **MaxCorr**$Genes$:
   - 각 유전자 j에 대해 계산된 상관계수 중 가장 큰 값.
   - 특정 유전자에서 높은 정확도의 예측이 이루어지도록 모델이 해당 유전자를 잘 학습해야 함.
3. **HEG**$Corr$, **HVG**$Corr$:
   - 특정한 HEG, HVG 리스트에 속한 유전자들에 대해 상관계수의 평균.
   - **중요 유전자 리스트(HEG, HVG)**가 모델 학습과 평가에 있어 핵심적으로 작용하므로, 해당 유전자들에 초점을 맞춘 데이터 처리와 학습이 필요.

VII. 최종 모델, 손실함수 선정 (손실함수)

```python
# ========================
# HEG and HVG Extraction Function
# ========================
def extract_HEG_HVG(expression_data, top_N=100):
    gene_means = np.mean(expression_data, axis=0)
    gene_std = np.std(expression_data, axis=0)
    gene_cv = gene_std / (gene_means + 1e-8)

    HEG_indices = np.argsort(gene_means)[-top_N:]
    HVG_indices = np.argsort(gene_cv)[-top_N:]

    return HEG_indices, HVG_indices

# Extract HEG and HVG indices
top_N = 10
HEG_indices, HVG_indices = extract_HEG_HVG(train_label_vec, top_N=top_N)
```

```python
# Pearson Correlation Coefficient Loss 정의
class PearsonLoss(nn.Module):
    def __init__(self):
        super(PearsonLoss, self).__init__()

    def forward(self, y_pred, y_true):
        # y_pred와 y_true의 평균을 계산
        y_pred_mean = torch.mean(y_pred, dim=0)
        y_true_mean = torch.mean(y_true, dim=0)

        # 분자 계산
        vx = y_pred - y_pred_mean
        vy = y_true - y_true_mean
        numerator = torch.sum(vx * vy, dim=0)

        # 분모 계산
        denominator = torch.sqrt(torch.sum(vx ** 2, dim=0)) * torch.sqrt(torch.sum(vy ** 2, dim=0))

        # 각 차원별 Pearson 상관계수 계산
        correlation = numerator / (denominator + 1e-8)

        # 상관계수의 평균을 구하고 손실로 사용
        loss = 1 - torch.mean(correlation)

        return loss
```

```python
# ========================
# Total Loss Function with Pearson Loss
# ========================
def total_loss_function(output, target, heg_weight=5.0, hvg_weight=2.0):
    mse_loss = nn.MSELoss()(output, target)
    pearson_loss = PearsonLoss()(output, target)

    heg_loss = nn.MSELoss()(output[:, HEG_indices], target[:, HEG_indices]) * heg_weight
    hvg_loss = nn.MSELoss()(output[:, HVG_indices], target[:, HVG_indices]) * hvg_weight

    total_loss = huber_loss
    return total_loss, mse_loss, heg_loss, hvg_loss, pearson_loss
```

VII. 최종 모델, 손실함수 선정 (최종)

```
Epoch [1], Train Loss: [0.02241]
Epoch [2], Train Loss: [0.02013]
Epoch [3], Train Loss: [0.02003]
Epoch [4], Train Loss: [0.01988]
Epoch [5], Train Loss: [0.01971]
Epoch [6], Train Loss: [0.01962]
Epoch [7], Train Loss: [0.01947]
Epoch [8], Train Loss: [0.01943]
Epoch [9], Train Loss: [0.01936]
Epoch [10], Train Loss: [0.01934]
Interence: 100%|██████████| 72/72 [00:07<00:00,  9.15it/s]
```

최종적으로 초반 resnet50 기반 모델 과 접근론보다 다양한 실험을 통해 loss값을 낮추고 일반화 성능을 끌어올림.