H&E Stained Histopathological Image Analysis for Gene Expression Prediction

CUAI CV project team 3

2024.11.26

발표자 : 김부영

스터디원 소개 및 만남 인증

회의실	날짜	예약시간
825호 창의자율연구실험실습실팀플룸03	2024- 11-22	10:00 ~ 13:00

스터디원 1 : 융합공학부 김부영

스터디원 2 : 역사학과 나영은

스터디원 3 : 전자전기공학부 오서윤

스터디원 4 : 응용통계학과 송채린

Content

1. Project Overview
2. Dataset Overview
3. Challenges
4. Regression models
5. Inception-ResNet-v2
6. Future directions

1. Project Overview

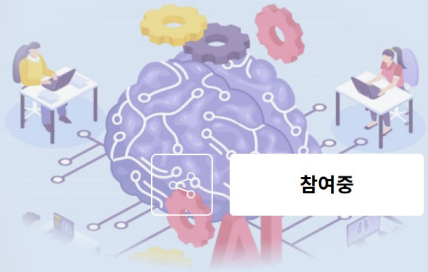
제1회 Medical AI (MAI) 경진대회

알고리즘 | 의료 | 유전자 | 비전 | 회귀 | PCC

상금 : 1,000만원

2024.10.02 ~ 2024.10.31 09:59 + Google Calendar

700명 마감



Objective:
Develop an AI model that predicts **gene expression data** from **H&E-stained tissue images**.

2. Dataset Overview

Dataset Info.

•train [폴더] :
학습용 H&E 염색된 조직 이미지 샘플 6992개
TRAIN_0000.png ~ TRAIN_6991.png

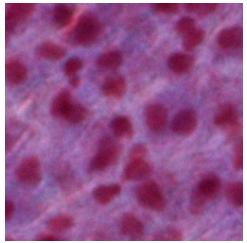
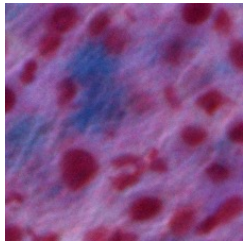
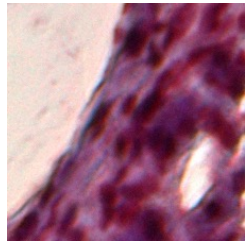
•test [폴더] :
평가용 H&E 염색된 조직 이미지 샘플 2277개
TEST_0000.png ~ TEST_2276.png

•train.csv [파일] :
ID : 샘플 ID
path : H&E 염색된 조직 이미지의 경로
AL645608.7 ~ AL592183.1 : 각 유전자의 발현 정보 (유전자 총 3467개 존재)

평가: Pearson Correlation Coefficient(PCC)

•test.csv [파일] :
ID : 샘플 ID
path : H&E 염색된 조직 이미지의 경로

•sample_submission.csv [파일] :
ID : 샘플 ID
AL645608.7 ~ AL592183.1 : 예측한 각 유전자의 발현 정보 (유전자 총 3467개 존재)

3. Challenges

Unexplored Task: Image Regression
•Tackling a novel challenge of applying regression techniques to image data.

Key Challenges:
1.Limited Dataset Size:
1. Unable to apply data augmentation due to the specific nature of the dataset.

2.Fixed and Distinct Features of Label Values in Genetic Data:
1. The labels exhibit unique characteristics tied closely to the dataset.

Proposed Approach:
•Transfer Learning:
- Leverage pre-trained models to compensate for the limited data availability.

•Model Exploration and Optimization:
- Experiment with various architectures and fine-tune them for optimal performance.

4. 회귀 모델

1. ResNet50(BASELINE) + Linear regression

Model Define

```
class BaseModel(nn.Module):
    def __init__(self, gene_size=CFG['label_size']):
        super(BaseModel, self).__init__()
        self.backbone = models.resnet50(pretrained=True)
        self.regressor = nn.Linear(1000, gene_size)

    def forward(self, x):
        x = self.backbone(x)
        x = self.regressor(x)
        return x
```

public점수	0.437392041
private점수	0.4349610449

4. 회귀 모델

2. SENet

Model Define

```
class SEBlock(nn.Module):
    def __init__(self, channels, reduction=16):
        super(SEBlock, self).__init__()
        self.global_avg_pool = nn.AdaptiveAvgPool2d(1)
        self.fc1 = nn.Linear(channels, channels // reduction, bias=False)
        self.relu = nn.ReLU(inplace=True)
        self.fc2 = nn.Linear(channels // reduction, channels, bias=False)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        b, c, _, _ = x.size()
        se = self.global_avg_pool(x).view(b, c)
        se = self.fc1(se)
        se = self.relu(se)
        se = self.fc2(se)
        se = self.sigmoid(se).view(b, c, 1, 1)
        return x * se

class SEResNetBlock(nn.Module):
    def __init__(self, original_block, reduction=16):
        super(SEResNetBlock, self).__init__()
        self.original_block = original_block
        channels = original_block.bn3.num_features if hasattr(original_block, 'bn3') else original_block.bn2.num_features
        self.se_block = SEBlock(channels, reduction)

    def forward(self, x):
        x = self.original_block(x)
        x = self.se_block(x)
        return x

class SENet50Regression(nn.Module):
    def __init__(self, gene_size=CFG['label_size'], pretrained=True):
        super(SENet50Regression, self).__init__()
        resnet = models.resnet50(pretrained=pretrained)
        self.base = nn.Sequential(*list(resnet.children())[:-2])

        # SE block 추가
        for name, layer in self.base.named_children():
            if isinstance(layer, nn.Sequential):
                for i, block in enumerate(layer):
                    layer[i] = SEResNetBlock(block)

        self.pool = nn.AdaptiveAvgPool2d(1)
        self.regressor = nn.Linear(resnet.fc.in_features, gene_size)

    def forward(self, x):
        x = self.base(x)
        x = self.pool(x)
        x = torch.flatten(x, 1)
        x = self.regressor(x)
        return x
```

제출 일시	public점수 private점수
2024-11-13 22:10:01	0.4586608791 0.4777116757

4. 회귀 모델

3. ViT

```python
class ViTSmallDatasetModel(nn.Module):
    def __init__(self, num_outputs=1, freeze_layers=True):
        super(ViTSmallDatasetModel, self).__init__()
        # 사전 학습된 ViT 모델 로드
        self.vit = timm.create_model('vit_base_patch16_224', pretrained=True)

        # 일부 레이어를 고정 (freeze)하여 소규모 데이터셋에서도 효율적으로 학습
        if freeze_layers:
            for param in self.vit.parameters():
                param.requires_grad = False

        # 마지막 헤드 레이어만 학습되도록 설정 (회귀용으로 사용)
        self.vit.head = nn.Linear(self.vit.head.in_features, num_outputs)

        # 고정 해제: 마지막 레이어만 학습되도록
        for param in self.vit.head.parameters():
            param.requires_grad = True

    def forward(self, x):
        return self.vit(x)
```

제출 일시	public점수 private점수
2024-11-13 15:49:46	0.4666792702 0.4720495287

4. 회귀 모델

4. EfficientNetB0

```python
# EfficientNetB0 모델 설정 (회귀용)
def build_model():
    base_model = EfficientNetB0(include_top=False, input_shape=(224, 224, 3))
    x = GlobalAveragePooling2D()(base_model.output)
    output = Dense(3467, activation='linear')(x)
    model = Model(inputs=base_model.input, outputs=output)
    return model

model = build_model()
```

제출 일시	public점수 private점수
2024-11-12 23:33:07	0.4753527598 0.4844446715

4. 회귀 모델

5. Inception-ResNet-v2

```
class BaseModel(nn.Module):
    def __init__(self, gene_size=CFG['label_size']):
        super(BaseModel, self).__init__()
        # Inception-ResNet-V2 (Inception-ResNet-V4로도 불림) 모델 호출
        self.backbone = timm.create_model('inception_resnet_v2', pretrained=True)

        # 기존 모델의 최종 출력 특징 수를 가져옵니다.
        in_features = self.backbone.classif.in_features
        # Inception 모델의 마지막 레이어를 제거하여 고유한 출력 레이어로 변경
        self.backbone.classif = nn.Identity()  # 기존 classifier 레이어를 제거

        # 최종 출력 레이어로 원하는 레이블 크기(gene_size)에 맞게 Linear 레이어 추가
        self.regressor = nn.Linear(in_features, gene_size)

    def forward(self, x):
        x = self.backbone(x)
        x = self.regressor(x)
        return x
```

제출 일시	public점수 private점수
2024-11-18 13:37:08	0.5025639088 0.5105486941

5. Inception - ResNet v2 tuning

Enhanced Model Tuning with Pretrained Inception-ResNet-v2

- **Initial Selection:**
  - Among various pretrained models, **Inception-ResNet-v2** demonstrated the highest PCC (Pearson Correlation Coefficient) under identical conditions.

- **Optimization with OPTUNA:**
  - Hyperparameters such as **learning rate** and **batch size** were optimized using the OPTUNA framework for better performance.

- **Refinement of Regression Layer:**
  - Replaced the regression layers with a configuration of **SiLU activation**, **Dropout**, and a **Linear output layer**.
  - Fine-tuned the learning rate using OPTUNA for these updated layers.

- **Training Approach:**
  - Conducted training at multiple epochs (10, 15, 20, 30) to evaluate performance improvements across iterations.

5. Inception - ResNet v2 tuning

동일 조건에서 PCC 결과가 가장 좋았던 (pretrained) Inception-Resnet v2를 더 튜닝

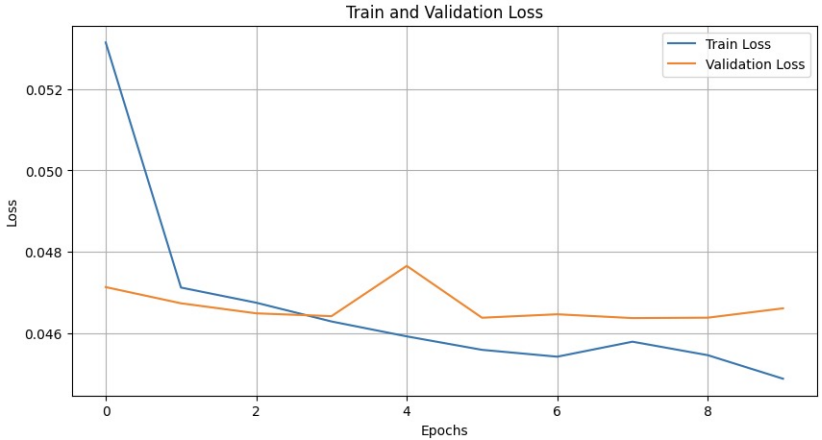
1. OPTUNA로 learning rate, batch size 최적화
2. 회귀 계층을 SiLU + dropout + linear 로 변경, OPTUNA로 learning rate 최적화 후, epoch 10, 15, 20, 30 으로 train

```python
class BaseModel(nn.Module):
    def __init__(self, gene_size=CFG['label_size'], dropout_rate=0.5):
        super(BaseModel, self).__init__()

        # Backbone: Inception-ResNet-V2
        self.backbone = timm.create_model('inception_resnet_v2', pretrained=True)

        # Get the number of input features from the last layer
        in_features = self.backbone.classif.in_features

        # Remove the original classification layer
        self.backbone.classif = nn.Identity()

        # Regressor: Add dropout and activation function
        self.regressor = nn.Sequential(
            nn.SiLU(),  # Activation function
            nn.Dropout(p=dropout_rate),  # Dropout layer
            nn.Linear(in_features, gene_size)  # Linear regression layer
        )

    def forward(self, x):
        x = self.backbone(x)
        x = self.regressor(x)
        return x
```

5. Inception - ResNet v2 tuning 결과

1. OPTUNA로 learning rate, batch size 최적화

```
[I 2024-11-17 16:06:25,713] Trial 14 finished with value: 0.046598626978018066 and parameters: {'learning_rate': 0.007276417425771936, 'batch_size': 32}. Best is trial 11 with value: 0.045830197632312775.
Epoch [10], Train Loss : [0.04696] Val Loss : [0.06022]
Best trial:
  Validation Loss: 0.045830197632312775
  Params:
    learning_rate: 0.0003666091461681633
    batch_size: 32
```



제출 일시	public점수 private점수
2024-11-18 13:37:08	0.5025639088 0.5105486941

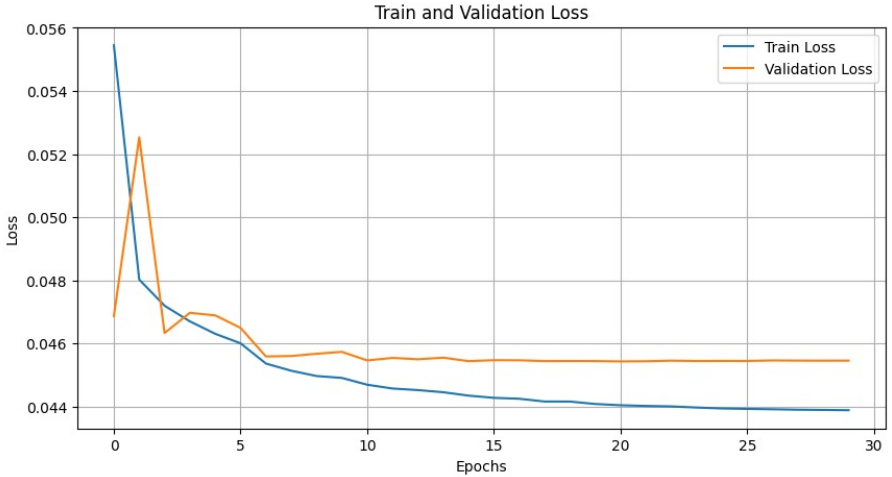
5. Inception - ResNet v2 tuning 결과

2. 회귀 계층을 SiLU + dropout + linear 로 변경, OPTUNA로 learning rate 최적화 후, epoch 10, 15, 20, 30 으로 train

inception- resnet- v2-ver4_tuning.csv

inception-resnet-v2 _ dropout_ optuna edit

Epoch 30 | 2024-11-23 03:20:32 | 0.5014278354 | 0.5061351962



Best trial:
Value: 0.045531538569114426
Params:
learning_rate: 0.000762730617772347

5. Inception - ResNet v2 tuning 결과

2. 회귀 계층을 SiLU + dropout + linear 로 변경, OPTUNA로 learning rate 최적화 후, epoch 10, 15, 20, 30 으로 train

inception- resnet- v2-ver4_tuning_10.csv inception-resnet-v2 _ dropout _ optuna _epoch10 edit	Epoch 10	2024-11-23 11:35:51	0.5056858836 0.5175981194
inception- resnet- v2-ver4_tuning_15.csv inception-resnet-v2 _ dropout _ optuna _epoch15 edit	Epoch 15	2024-11-23 11:37:20	0.5063581951 0.5097910925
inception- resnet- v2-ver4_tuning_20.csv edit	Epoch 20	2024-11-23 12:10:33	0.4976268571 0.5060357977
inception- resnet- v2-ver4_tuning.csv inception-resnet-v2 _ dropout_ optuna edit	Epoch 30	2024-11-23 03:20:32	0.5014278354 0.5061351962

6. 발전 방향

Future Directions

1.Model Development:

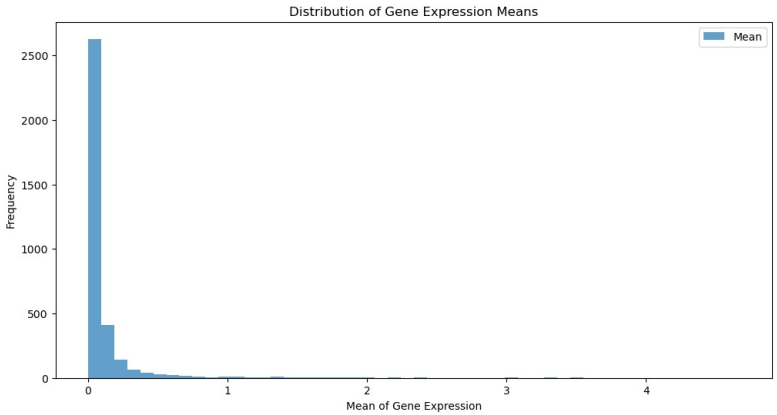
1. Implement strategies to **prevent overfitting** by incorporating regularization techniques or data augmentation methods tailored to the task.
2. Explore **task-specific architectural modifications** to enhance model performance.

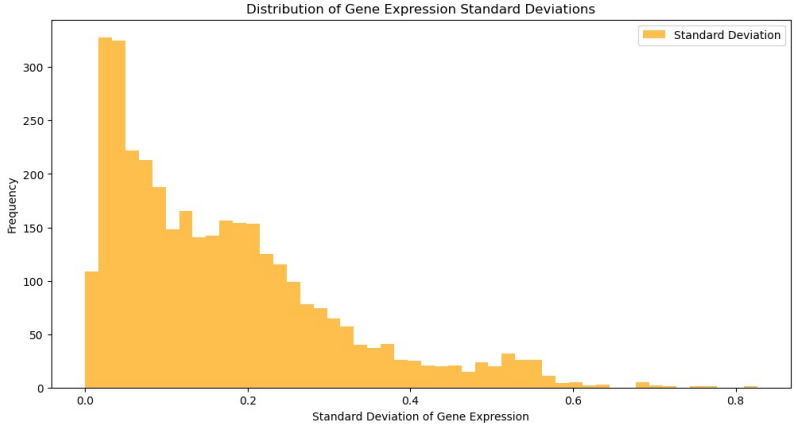
2.Data Analysis and Domain Integration:

1. Investigate biological domain knowledge related to **gene expression data** to devise innovative approaches and incorporate insights from domain-specific research.

6. 발전 방향

Analysis of Gene Expression Value Distribution





6. 발전 방향

Analysis of Highly Correlated Gene Pairs

```
상관 계수가 0.8 이상인 유전자 페어:
             Gene1       Gene2  Correlation
100916     ANGPTL7  AC012074.1     0.999627
101126     ANGPTL7        OXTR     0.997991
101401     ANGPTL7        UGT8     0.883277
101428     ANGPTL7      RNF175     0.999089
101833     ANGPTL7       PDE1C     0.893330
...            ...         ...          ...
11879232  DCAF12L1       PDE1C     0.948075
11879318  DCAF12L1  AC078845.1     0.803578
11881124  DCAF12L1  AL031668.2     0.828285
11992348   MT-ATP6      MT-ND4     0.808448
12006212    MT-ND4     MT-ATP6     0.808448

[264 rows x 3 columns]
```