AI훈민정음_
Image_Anomaly_Detection
(노후 시설물 이미지 이상탐지)

목 차

I. 개요

1. 소개 : 노후 시설물 이미지 이상탐지

AI Hub에서 제공하는 노후 시설물 이미지 데이터셋 중
점자블록의 정상 및 교체/폐기 여부를 예측하는 예제

1. 데이터셋

노후 시설물 이미지:
정상 공공시설물 이미지와
수리된 공공시설물 이미지,
교체/폐기된 공공시설물 이미지
80만건을 9개 대분류, 43개 소분류로 제공

이 중 통행시설물 점자블록에 해당하는
데이터셋을 가져와 사용

2. 전처리 및 학습

전처리:
이미지 크기를 (224, 224)로 설정,
픽셀 값 정규화, SMOTE

학습:
VGG16을 활용한 전이학습

3. 추론 결과

점자블록 이미지가 정상 점자블록인지
교체/폐기가 필요한 점자블록인지 예측

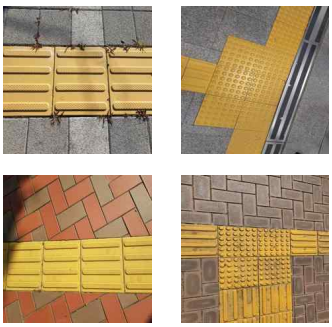
4. 기대 효과

점자블록 노후화 파악 가능
시각장애인을 위한 인도 상황 개선 자료로
활용 가능

데이터셋

- 정상 점자블록(8000개)와 교체/폐기가 필요한 점자블록(1500개)으로 이루어져 있음

정상 점자블록



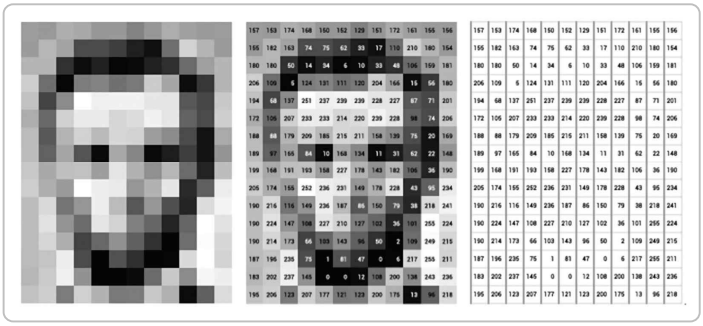
교체/폐기 점자블록

I. 개요

정규화

- 효율적인 모델 학습을 위해 각 픽셀을 0~1 사이의 값으로 만들어줌
- 한 픽셀이 0~255 사이의 값을 가짐으로 255로 나눠줌

이미지 픽셀화



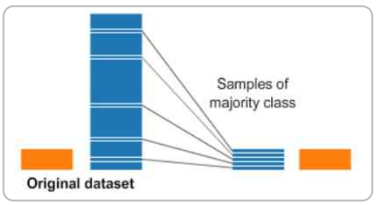
이미지 출처: 테크M 제58호(2018년 2월) 기사
https://www.techm.kr/news/articleView.html?idxno=4642

I. 개요

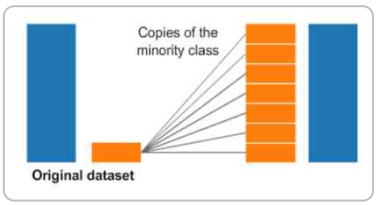
언더 샘플링/오버 샘플링

- 언더 샘플링은 불균형한 데이터셋에서 높은 비율을 차지하는 클래스의 데이터 수를 줄여 클래스의 불균형을 해결
- 오버 샘플링은 불균형한 데이터셋에서 낮은 비율을 차지하는 클래스의 데이터 수를 늘려 클래스의 불균형을 해결

언더 샘플링



오버 샘플링



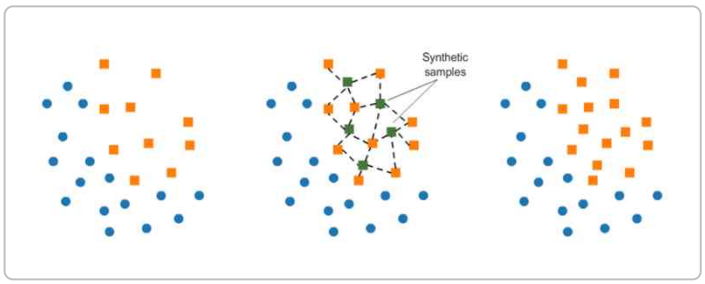
이미지 출처: http://www.incodom.kr/SMOTE

I. 개요

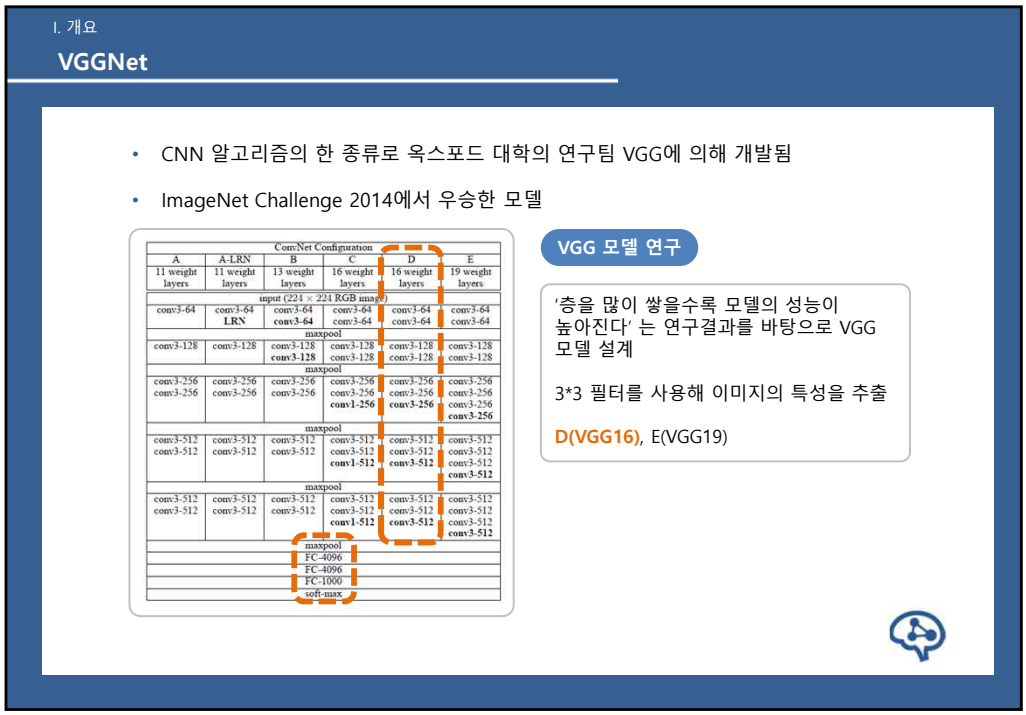
SMOTE

- imbalanced learn에서 제공하는 대표적인 오버 샘플링 기법 중 하나
- 최근접 이웃(k-NN) 알고리즘을 활용해 낮은 비율을 차지하는 클래스의 데이터 사이에 새로운 데이터를 생성

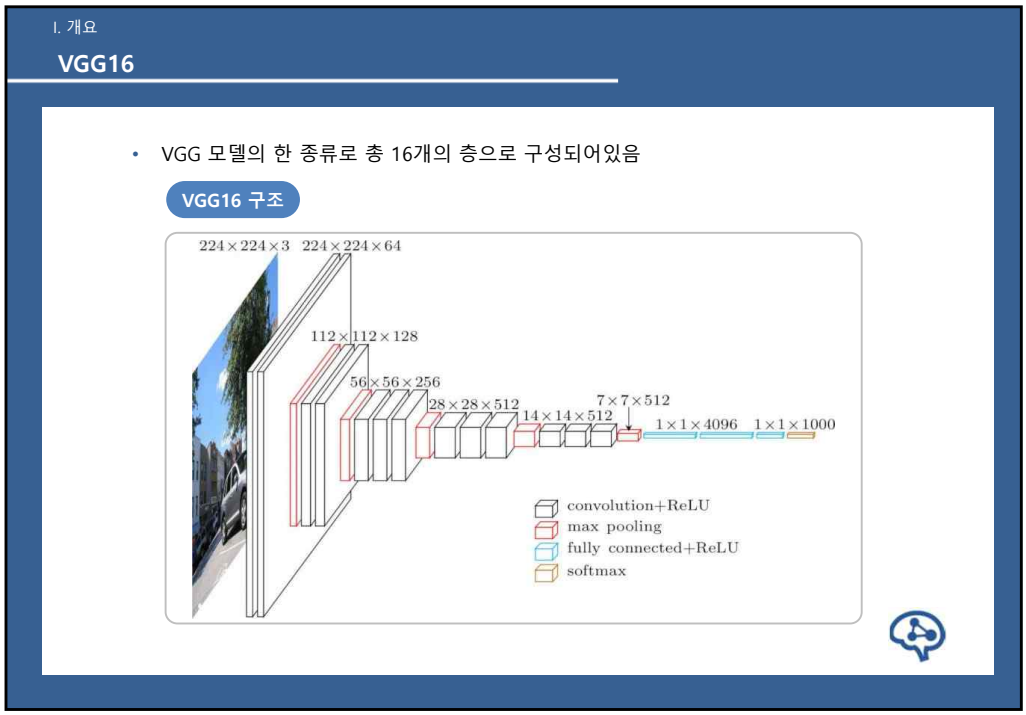
SMOTE



이미지 출처: http://www.incodom.kr/SMOTE



이미지 출처: Karen Simonyan & Andrew Zisserman (2015), VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION, Visual Geometry Group, Department of Engineering Science, University of Oxford



이미지 출처: https://neurohive.io/en/popular-networks/vgg16/
내용 출처: https://www.vlfeat.org/matconvnet/models/imagenet-vgg-verydeep-16.svg

Convolution/Pooling(이미지 객체가 라벨에 속할 확률 계산)

0) 인풋: 224 x 224 x 3 이미지(224 x 224 RGB 이미지) 입력 받음

1) 1층(conv1_1): 64개의 3 x 3 x 3 필터 커널 사용. zero padding은 1만큼 해줬고, 간격(stride)은 1로 설정함. 64장의 224 x 224 특성 맵(224 x 224 x 64)들이 생성됨

2) 2층(conv1_2): 64개의 3 x 3 x 64 필터 커널 사용. 64장의 224 x 224 특성 맵들(224 x 224 x 64)이 생성됨. 그 후 2 x 2 최대 pooling을 2간격으로 적용해 특성 맵의 사이즈를 112 x 112 x 64로 줄임

3) 3층(conv2_1): 128개의 3 x 3 x 64 필터 커널 사용. 128장의 112 x 112 특성 맵들(112 x 112 x 128)이 생성됨.

4) 4층(conv2_2): 128개의 3 x 3 x 128 필터 커널로 사용. 128장의 112 x 112 특성 맵들(112 x 112 x 128)이 생성됨. 그 후 pooling을 통해 특성 맵 사이즈를 56 x 56 x

128로 줄임.

5) 5층(conv3_1): 256개의 3 x 3 x 128 필터 커널 사용. 256장의 56 x 56 특성 맵들(56 x 56 x 256) 생성됨

6) 6층(conv3_2): 256개의 3 x 3 x 256 필터 커널 사용. 256장의 56 x 56 특성 맵들(56 x 56 x 256) 생성됨

7) 7층(conv3_3): 256개의 3 x 3 x 256 필터 커널 사용. 256장의 56 x 56 특성 맵들(56 x 56 x 256) 생성됨. 그 후 pooling을 통해 특성 맵 사이즈를 28 x 28 x 256로 줄임

8) 8층(conv4_1): 512개의 3 x 3 x 256 필터 커널 사용. 512장의 28 x 28 특성 맵들(28 x 28 x 512)이 생성됨

9) 9층(conv4_2): 512개의 3 x 3 x 512 필터 커널 사용. 512장의 28 x 28 특성 맵들(28 x 28 x 512)이 생성됨

10) 10층(conv4_3): 512개의 3 x 3 x 512 필터 커널 사용. 512장의 28 x 28 특성 맵들(28 x 28 x 512)이 생성됨. 그 후 pooling을 통해 특성 맵 사이즈를 14 x 14 x 512로 줄임

11) 11층(conv5_1): 512개의 3 x 3 x 512 필터 커널 사용. 512장의 14 x 14 특성 맵들(14 x 14 x 512)이 생성됨

12) 12층(conv5_2): 512개의 3 x 3 x 512 필터 커널 사용. 512장의 14 x 14 특성 맵들(14 x 14 x 512)이 생성됨

13) 13층(conv5-3): 512개의 3 x 3 x 512 필터 커널 사용. 512장의 14 x 14 특성 맵들(14 x 14 x 512)이 생성됨. 그 후 pooling을 통해 특성 맵 사이즈를 7 x 7 x 512로 줄임

Fully Connected Layer(이미지 분류하는데 사용되는 계층)

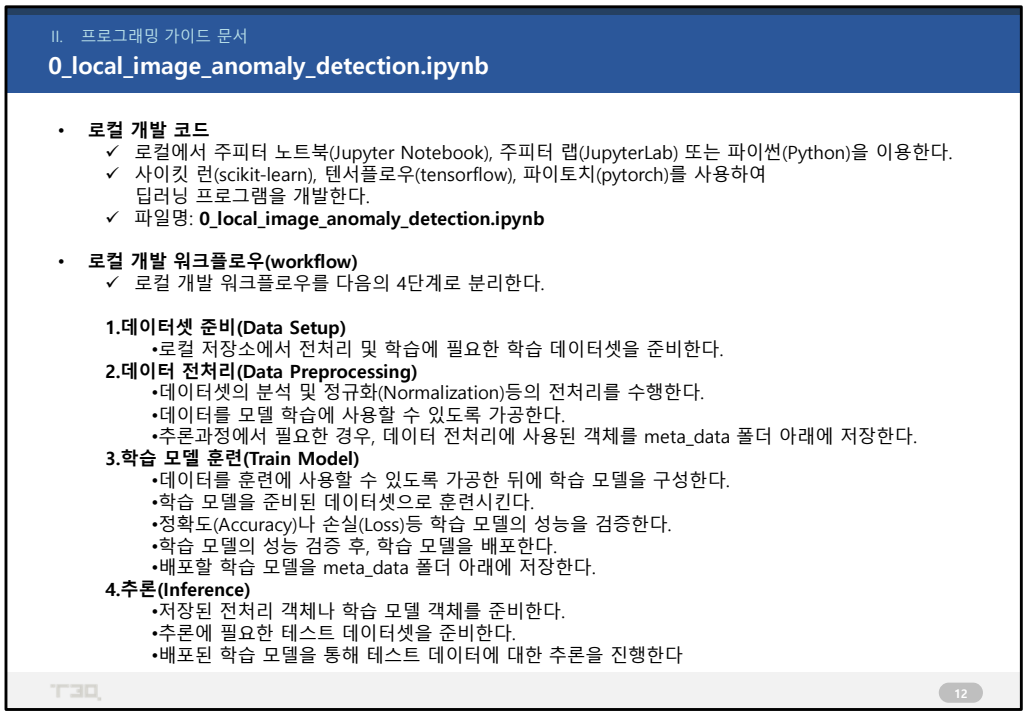
14) 14층(fc1): 7 x 7 x 512의 특성 맵 flatten 해줌. 7 x 7 x 512 = 25088개의 뉴런이 되고, fc1층의 4096개의 뉴런과 fully connected 됨. 훈련 시 dropout이 적용됨.

15) 15층(fc2): 4096개의 뉴런으로 구성되어 fc1층의 4096개의 뉴런과 fully connected 됨. 훈련 시 dropout이 적용됨

16) 16층(fc3): 1000개의 뉴런으로 구성됨. fc2층의 4096개의 뉴런과 fully connected됨. 1000개의 뉴런으로 구성되어 1000개의 클래스로 분류하는 것이 가능

0_local_image_anomaly_detection_requirement

- 로컬에 설치 되어야 할 패키지 버전
  - ✓ matplotlib 3.3.3
  - ✓ tensorflow-gpu 2.4.1
  - ✓ pillow 8.4.0
  - ✓ imbalanced-learn 0.8.1
  - ✓ scikit-learn 0.24.2
  - ✓ pandas 1.1.5
  - ✓ numpy 1.19.5



0_local_image_anomaly_detection.ipynb

```
==========================================
# imports
import os
import pandas as pd
import numpy as np
from glob import glob
import zipfile

from sklearn.utils import shuffle

import matplotlib.pyplot as plt
import random

from PIL import Image

import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras import layers
from tensorflow.keras.models import Model
from tensorflow.keras import optimizers
from tensorflow.keras.utils import to_categorical

from imblearn.over_sampling import SMOTE
```

1. 데이터셋 준비(Data Setup)

```
zip_target_path = './meta_data/dataset'
os.makedirs(zip_target_path, exist_ok=True)

# dataset.zip 파일을 dataset 폴더에 압축을 풀어준다.
zip_source_path = './dataset.zip'

extract_zip_file = zipfile.ZipFile(zip_source_path)
extract_zip_file.extractall(zip_target_path)

extract_zip_file.close()

train_dir = zip_target_path + '/'

#datatset으로부터 각 image들의 주소들의 목록을 가져온다.
#정상 이미지
images=[]
labels=[]
test_img=[]
test_label=[]
for (root, dirs, files) in os.walk(train_dir+"original"):
    for file_name in files:
        if len(test_img)!=500:  #앞의 500개의 이미지는 test dataset
            test_img.append(train_dir+"original/"+file_name)
            test_label.append(0) #정상 레이블 : 0
        else:
            images.append(train_dir+"original/"+file_name)
            labels.append(0)
#비정상 이미지
for (root, dirs, files) in os.walk(train_dir+"discard"):
    for file_name in files:
        if len(test_img) != 1000: #앞의 500개 이미지는 test dataset
            test_img.append(train_dir+"discard/"+file_name)
            test_label.append(1) #비정상 레이블 : 1
        else:
            images.append(train_dir+"discard/"+file_name)
            labels.append(1)

images, labels = shuffle(images,labels)
test_img, test_label = shuffle(test_img, test_label)
```

2. 데이터 전처리(Data Preprocessing)

```
# Train dataset
x = []
y = []
for i in range (len(images)):
    img = Image.open(images[i])
    img = img.resize((224,224))
    img = np.array(img)
    x.append(img)
    y.append(labels[i])

# Test dataset
test_x_arr = []
test_y_arr = []
for i in range (len(test_img)):
    img = Image.open(test_img[i])
    img = img.resize((224,224))
    img = np.array(img)
    test_x_arr.append(img)
    test_y_arr.append(test_label[i])

x_train=np.array(x)
y_train=np.array(y)
x_test=np.array(test_x_arr)
y_test=np.array(test_y_arr)

# x_train, x_test, y_train, y_test = train_test_split(x_arr, y_arr, test_size=0.2, shuffle = True)
y_train = y_train[...,tf.newaxis]
y_test = y_test[...,tf.newaxis]
x_train_cnn = x_train.astype(np.float32)/255.
x_test_cnn = x_test.astype(np.float32)/255.
y_train_cnn = to_categorical(y_train)
y_test_cnn = to_categorical(y_test)
```

3. 학습 모델 훈련(Train Model)

```
train_rows=len(x_train_cnn)
x_train_cnn_smote = x_train_cnn.reshape(train_rows,-1)
sm = SMOTE(random_state=42)
x_smote, y_smote = sm.fit_resample(x_train_cnn_smote, y_train)
x_smote_cnn = x_smote.reshape(-1,224,224,3)
y_smote_cnn = to_categorical(y_smote)

pretrained_model = VGG16(weights='imagenet', include_top = False, input_shape=(224,224,3))
for layer in pretrained_model.layers[:15]:
    layer.trainable = False
for layer in pretrained_model.layers[15:]:
    layer.trainable = True
last_layer = pretrained_model.get_layer('block5_pool')
last_output = last_layer.output

x = layers.GlobalMaxPooling2D()(last_output)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(2, activation='softmax')(x)

model_smote = Model(pretrained_model.input, x)
model_smote.summary() #모델 정보를 확인할 수 있는 함수이다.

model_smote.compile(loss='categorical_crossentropy',
              optimizer=optimizers.SGD(learning_rate=1e-4, momentum=0.9),
              metrics=['accuracy'])

epochs = 5
batch_size = 4
history = model_smote.fit(x_smote_cnn, y_smote_cnn, batch_size=batch_size, epochs=epochs,
validation_split=0.3)
```

```
# Plot accuracy and loss curves for both training and validation data
acc = history.history['accuracy']
loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']

fig, ax = plt.subplots(1,2,figsize=(10,5))
ax[0].plot(acc,label='accuracy')
ax[0].plot(val_acc,label='val_accuracy')
ax[0].set_title("Accuracy")
ax[0].legend()
ax[1].plot(loss, label='Loss')
ax[1].plot(val_loss,label='val_loss')
ax[1].set_title("Loss")
ax[1].legend()
```

4. 추론(Inference)

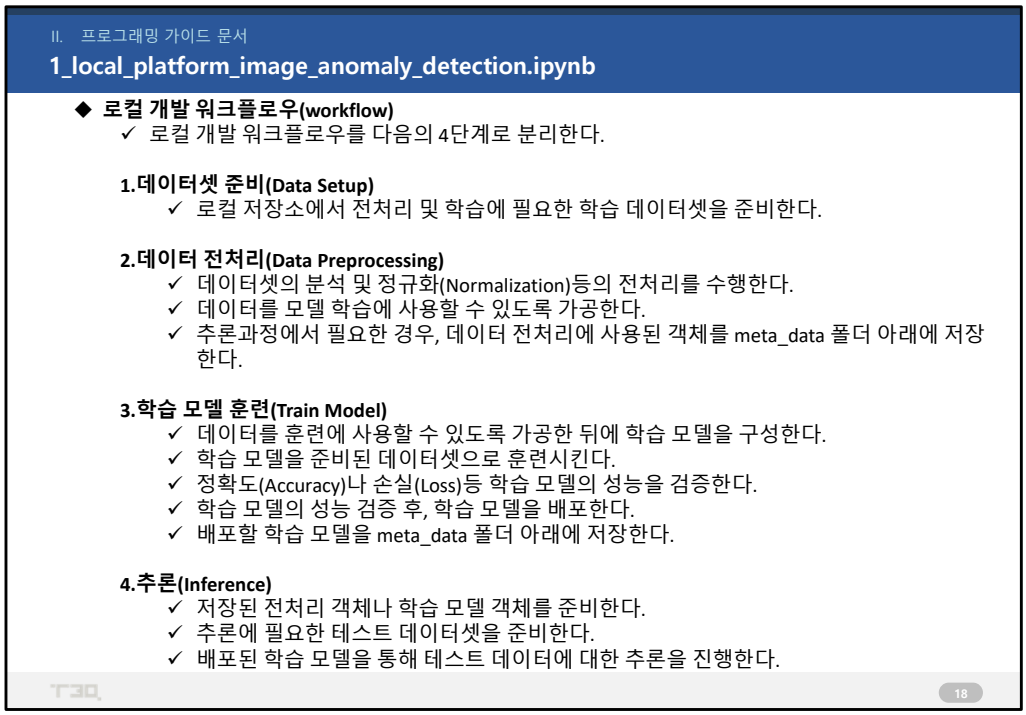
```
zip_test_target_path = './meta_data/test_dataset'
os.makedirs(zip_test_target_path, exist_ok=True)

# dataset.zip 파일을 dataset 폴더에 압축을 풀어준다.
zip_test_source_path = './test_dataset.zip'
extract_zip_file = zipfile.ZipFile(zip_test_source_path)
extract_zip_file.extractall(zip_test_target_path)
extract_zip_file.close()

test_files = glob(zip_test_target_path + '/*.jpeg')
test_files

image_dataset = []
for test_file in test_files:
    image = Image.open(test_file)
    image = image.resize((224, 224))
    img = np.array(image)
    img = img.astype(np.float32)/255.
    img = img.reshape(224, 224, 3)
    image_dataset.append(img)

image_dataset = np.array(image_dataset)
data_label=['original','discard']
prediction = np.argmax(model_smote.predict(image_dataset), axis = 1)
[data_label[i] for i in prediction]
```

1_local_platform_image_anomaly_detection.ipynb

◆ **플랫폼 업로드를 쉽게하기 위한 로컬 개발 코드**
- ✓ T3Q.ai(T3Q.cep + T3Q.dl): 빅데이터/인공지능 통합 플랫폼
- ✓ 플랫폼 업로드를 쉽게하기 위하여 로컬에서 아래의 코드(파일1)를 개발한다.
- ✓ 파일 1(파일명): 1_local_platform_image_anomaly_detection.ipynb

▪ **전처리 객체 또는 학습모델 객체**
- ✓ 전처리 객체나 학습모델 객체는 meta_data 폴더 아래에 저장한다.

▪ **데이터셋(학습 데이터/테스트 데이터)**
- ✓ 학습과 테스트에 사용되는 데이터를 나누어 관리한다.
- ✓ 학습 데이터: dataset 폴더 아래에 저장하거나 dataset.zip 파일 형태로 저장한다.
- ✓ 테스트 데이터: test_dataset 폴더 아래에 저장하거나 test_dataset.zip 파일 형태로 저장한다.




```
# 파일명: image_anomaly_detection_preprocess.py

'''
from image_anomaly_detection_preprocess_sub import exec_process
'''
import logging

logging.basicConfig(level=logging.INFO)

def process_for_train(pm):

    exec_process(pm)

    logging.info('[hunmin log] the end line of the function [process_for_train]')


def init_svc(im, rule):
    return {}


def transform(df, params, batch_id):
```

```
logging.info('[hunmin log] df : {}'.format(df))
logging.info('[hunmin log] df.shape : {}'.format(df.shape))
logging.info('[hunmin log] type(df) : {}'.format(type(df)))
logging.info('[hunmin log] the end line of the function [transform]')

return df
```

```
# 파일명: image_anomaly_detection_preprocess_sub.py

import os
import numpy as np
import pandas as pd
import zipfile
import logging


def exec_process(pm):
    logging.info('[hunmin log]  the start line of the function [exec_process]')
    logging.info('[hunmin log] pm.source_path : {}'.format(pm.source_path))

    # 저장 파일 확인
    list_files_directories(pm.source_path)

#####################################################################
# 1_1_local_platform_image_anomaly_detection_preprocess_train.ipynb 파일의
# 1.데이터셋 준비(Data Setup) 부분을 여기에 작성한다.
#####################################################################
    # pm.source_path의 dataset.zip 파일을
    # pm.target_path의 dataset 폴더에 압축을 풀어준다.
```

```
    my_zip_path = os.path.join(pm.source_path,'dataset.zip')

    extract_zip_file = zipfile.ZipFile(my_zip_path)
    extract_zip_file.extractall(pm.target_path)

    extract_zip_file.close()


 # 저장 파일 확인
    list_files_directories(pm.target_path)

    logging.info('[hunmin log]  the finish line of the function [exec_process]')



# 저장 파일 확인
def list_files_directories(path):
    # Get the list of all files and directories in current working directory
    dir_list = os.listdir(path)
    logging.info('[hunmin log] Files and directories in {} :'.format(path))
    logging.info('[hunmin log] dir_list : {}'.format(dir_list))
```

```
# 파일명: image_anomaly_detection_train.py

'''
from image_anomaly_detection_train_sub import exec_train, exec_init_svc, exec_inference
'''
import logging


def train(tm):

    exec_train(tm)
    logging.info('[hunmin log] the end line of the function [train]')


def init_svc(im):

    params = exec_init_svc(im)
    logging.info('[hunmin log] the end line of the function [init_svc]')

    return { **params }
```

```
def inference(df, params, batch_id):

    result = exec_inference(df, params, batch_id)
    logging.info('[hunmin log] the end line of the function [inference]')

    return { **result }
```

```
# 파일명: image_anomaly_detection_train_sub.py

import os
import numpy as np
import pandas as pd
import tensorflow as tf
import io
import base64

# Preprocessing
from sklearn.utils import shuffle
from tensorflow.keras.utils import to_categorical
from imblearn.over_sampling import SMOTE

# Evaluation Metrics
from sklearn import metrics

# Deep Learning Model - Keras
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras import layers
from tensorflow.keras.models import Model
from tensorflow.keras import optimizers
```

```
from tensorflow.keras.models import load_model

from PIL import Image
import logging

logging.info(f'[hunmin log] tensorflow ver : {tf.__version__}')

# 사용할 gpu 번호를 적는다.
os.environ["CUDA_VISIBLE_DEVICES"]='0,1'


gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        tf.config.experimental.set_visible_devices(gpus, 'GPU')
        logging.info('[hunmin log] gpu set complete')
        logging.info('[hunmin log] num of gpu: {}'.format(len(gpus)))

    except RuntimeError as e:
        logging.info('[hunmin log] gpu set failed')
        logging.info(e)

########### 플랫폼 train(tm) 부분의 코드 입니다. ###########
def exec_train(tm):
    logging.info('[hunmin log] train start')

    ###############################################################
    ## 1. 데이터셋 준비(Data Setup)
    ###############################################################
    logging.info('[hunmin log] data load')
    images, labels, test_img, test_label = data_load(tm)
```

```
    ######################################################################
    ## 2. 데이터 전처리(Data Preprocessing)
    ######################################################################
    logging.info("[hunmin log] generate dataset")
    x_smote_cnn, y_smote_cnn, x_test_cnn, y_test_cnn = generate_datasets(images, labels,
test_img, test_label)

    ######################################################################
    ## 3. 학습 모델 훈련(Train Model)
    ######################################################################
    logging.info("[hunmin log] model build and compile")

    # 단일 gpu 혹은 cpu학습
    if len(gpus) < 2:
        model_smote = model_build_and_compile()
    # multi-gpu
    else:
        strategy = tf.distribute.MirroredStrategy()
        logging.info('[hunmin log] gpu devices num
{}'.format(strategy.num_replicas_in_sync))
        with strategy.scope():
            model_smote = model_build_and_compile()
```

```
    logging.info("[hunmin log] model fit")
    history = model_fit(model_smote, x_smote_cnn, y_smote_cnn)

    ######################################################################
    ## 학습 모델 저장
    ######################################################################
    logging.info("[hunmin log] export model")
    model_smote.save(os.path.join(tm.model_path, 'model.h5'))

 # 저장 파일 확인
    list_files_directories(tm.model_path)

    ######################################################################
    ## 플랫폼 시각화
    ######################################################################
    ## : 이 부분은 로컬에서 plotting하는 부분이지만
    ## 로컬환경과 플랫폼환경에서의 코드가 많이 달라 로컬코드에서는 생략되었습니다.
    ## 이 부분을 주석처리하고 실행하여도 플랫폼에서의 오류는 없습니다.
    ######################################################################
    logging.info("[hunmin log] plot metrics")
    '''
    plot_metrics(tm, history, model_smote, x_test_cnn, y_test_cnn)
    '''
    logging.info("[hunmin log] train complete")

########### 플랫폼 init_svc부분의 코드입니다. ###########
def exec_init_svc(im):
    model = call_model(im)
    return {'model' : model}
```

```
########### 플랫폼 inference부분의 코드입니다. ###########
def exec_inference(df, params, batch_id):
    # 파라미터(모델, 추론출력에 필요한 데이터) 로드
    model = params['model']

    logging.info('[hunmin log] data transform')
    data = data_transform(df)

    # 추론
    logging.info('[hunmin log] inference')
    predict = data_predict(model, data)
    logging.info('[hunmin log] model prediction : {}'.format(predict))

    # 역변환
    logging.info('[hunmin log] prediction decoding')
    predict = inverse_transform(predict)

    # json형식으로 변환할 수 있는 dictionary로 반환
    result = { 'inference' : predict }
    logging.info('[hunmin log] response : {}'.format(result))
    return result
```

```
#####################################################################
## exec_train(tm) 호출 함수
#####################################################################
def data_load(tm):
    train_dir = tm.train_data_path
    #datatset으로부터 각 image들의 주소들의 목록을 가져온다.
    #정상 이미지
    images=[]
    labels=[]
    test_img=[]
    test_label=[]
    for (root, dirs, files) in os.walk(train_dir + "/original"):
        for file_name in files:
            if len(test_img)!=500:  #앞의 500개의 이미지는 test dataset
                test_img.append(train_dir + "/original/" + file_name)
                test_label.append(0) #정상 레이블 : 0
            else:
                images.append(train_dir + "/original/" + file_name)
                labels.append(0)
    #비정상 이미지
    for (root, dirs, files) in os.walk(train_dir + "/discard"):
        for file_name in files:
            if len(test_img) != 1000: #앞의 500개 이미지는 test dataset
                test_img.append(train_dir + "/discard/" + file_name)
                test_label.append(1) #비정상 레이블 : 1
            else:
                images.append(train_dir + "/discard/" + file_name)
                labels.append(1)
    images, labels = shuffle(images, labels)
    test_img, test_label = shuffle(test_img, test_label)
    return images, labels, test_img, test_label
```

```
def generate_datasets(images, labels, test_img, test_label):
    # Train dataset
    x = []
    y = []
    for i in range(len(images)):
        img = Image.open(images[i])
        img = img.resize((224,224))
        img = np.array(img)
        x.append(img)
        y.append(labels[i])

    # Test dataset
    test_x_arr = []
    test_y_arr = []
    for i in range(len(test_img)):
        img = Image.open(test_img[i])
        img = img.resize((224,224))
        img = np.array(img)
        test_x_arr.append(img)
        test_y_arr.append(test_label[i])

    x_train=np.array(x)
```

```
y_train=np.array(y)
x_test=np.array(test_x_arr)
y_test=np.array(test_y_arr)

y_train = y_train[...,tf.newaxis]
y_test = y_test[...,tf.newaxis]

x_train_cnn = x_train.astype(np.float32)/255.
x_test_cnn = x_test.astype(np.float32)/255.

y_train_cnn = to_categorical(y_train)
y_test_cnn = to_categorical(y_test)

train_rows=len(x_train_cnn)
x_train_cnn_smote = x_train_cnn.reshape(train_rows, -1)

sm = SMOTE(random_state=42)
x_smote, y_smote = sm.fit_resample(x_train_cnn_smote, y_train)

x_smote_cnn = x_smote.reshape(-1,224,224,3)
y_smote_cnn = to_categorical(y_smote)


return x_smote_cnn, y_smote_cnn, x_test_cnn, y_test_cnn
```

```
def model_build_and_compile():
    pretrained_model = VGG16(weights='imagenet', include_top = False,
input_shape=(224,224,3))
    for layer in pretrained_model.layers[:15]:
        layer.trainable = False

    for layer in pretrained_model.layers[15:]:
        layer.trainable = True

    last_layer = pretrained_model.get_layer('block5_pool')
    last_output = last_layer.output

    x = layers.GlobalMaxPooling2D()(last_output)
    x = layers.Dense(512, activation='relu')(x)
    x = layers.Dropout(0.5)(x)
    x = layers.Dense(2, activation='softmax')(x)

    model_smote = Model(pretrained_model.input, x)

    model_smote.compile(loss='categorical_crossentropy',
              optimizer=optimizers.SGD(learning_rate=1e-4, momentum=0.9),
              metrics=['accuracy'])
```

```
    logging.info('[hunmin log] model_smote.summary() : ')
    model_smote.summary(print_fn = logging.info)

    return model_smote

def model_fit(model_smote, x_smote_cnn, y_smote_cnn):
    epochs = 5
    batch_size = 4 * len(gpus) if len(gpus) > 0 else 4
    history = model_smote.fit(x_smote_cnn, y_smote_cnn, batch_size=batch_size,
epochs=epochs, validation_split=0.3)

    return history


# 시각화
def plot_metrics(tm, history, model, x_test, y_test):
    from sklearn.metrics import confusion_matrix

    accuracy_list = history.history['accuracy']
    loss_list = history.history['loss']

    for step, (acc, loss) in enumerate(zip(accuracy_list, loss_list)):
        metric={}
        metric['accuracy'] = acc
        metric['loss'] = loss
        metric['step'] = step
        tm.save_stat_metrics(metric)
```

```
    predict_y = np.argmax(model.predict(x_test), axis = 1).tolist()
    actual_y = np.argmax(y_test, axis = 1).tolist()

    eval_results={}
    eval_results['predict_y'] = predict_y
    eval_results['actual_y'] = actual_y
    eval_results['accuracy'] = history.history['val_accuracy'][-1]
    eval_results['loss'] = history.history['val_loss'][-1]

    # calculate_confusion_matrix(eval_results)
    eval_results['confusion_matrix'] = confusion_matrix(actual_y, predict_y).tolist()
    tm.save_result_metrics(eval_results)
    logging.info('[hunmin log] accuracy and loss curve plot for platform')


################################################################################
## exec_init_svc(im) 호출 함수
################################################################################
def call_model(im):
    model_path = os.path.join(im.model_path, 'model.h5')
```

```
    model = load_model(model_path)
    return model

#########################################################################
#######
## exec_inference(df, params, batch_id) 호출 함수
#########################################################################
#######
def data_transform(df):
    data = df.iloc[0, 0]
    image_bytes = io.BytesIO(base64.b64decode(data))
    image = Image.open(image_bytes)
    image = image.resize((224, 224))
    img = np.array(image)
    img = img.astype(np.float32)/255.
    img = img.reshape(-1, 224, 224, 3)
    return img

def data_predict(model, data):
    pred = model.predict(data)
    prediction = np.argmax(pred, axis = 1)
    return prediction
```

```
def inverse_transform(prediction):
    data_label = ['original', 'discard']
    return data_label[prediction[0]]

# 저장 파일 확인
def list_files_directories(path):
    # Get the list of all files and directories in current working directory
    dir_list = os.listdir(path)
    logging.info('[hunmin log] Files and directories in {} :'.format(path))
    logging.info('[hunmin log] dir_list : {}'.format(dir_list))



# PM 클래스: pm 객체
class PM:
    def __init__(self):
        self.source_path = './'
```

```
        self.target_path = './meta_data/dataset'

# TM 클래스: tm 객체
class TM:
    def __init__(self):
        self.train_data_path = './meta_data/dataset'
        self.model_path = './meta_data'

# IM 클래스: im 객체
class IM:
    def __init__(self):
        self.model_path = './meta_data'


# pm 객체
pm = PM()
print('pm.source_path:', pm.source_path)
print('pm.target_path: ', pm.target_path)

# tm 객체
tm = TM()
print('tm.train_data_path: ', tm.train_data_path)
print('tm.model_path: ', tm.model_path)

# im 객체
im = IM()
print('im.model_path: ', im.model_path)
```

```
# inferecne(df, params, batch_id) 함수 입력
params = {}
batch_id = 0

import io
import pandas as pd

# csv파일에 있는 테스트 샘플
# 1번째 행: 정상 데이터 / 2번째 행: 비정상 데이터
data = [[0,
92,115,120,94,84,102,106,79,84,102,102,83,101,126,133,103,92,112,118,85,84,103,104,81,10
2,126,134,104,88,121,128,100,84,107,113,87],

[1,51,75,93,75,51,79,96,79,55,87,100,83,56,83,108,85,56,83,100,81,56,79,100,81,56,88,101,83,
56,88,105,83,56,84,93,83]]
df = pd.DataFrame(data)
print('df: ', df)
print('df.dtypes:', df.dtypes)
df.columns
```

process_for_train(pm)

train(tm)

transform(df, params, batch_id)

params = init_svc(im)

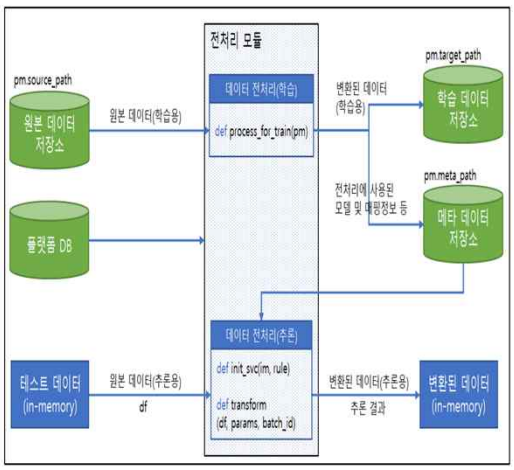
inference(df, params, batch_id)

2_platform_process

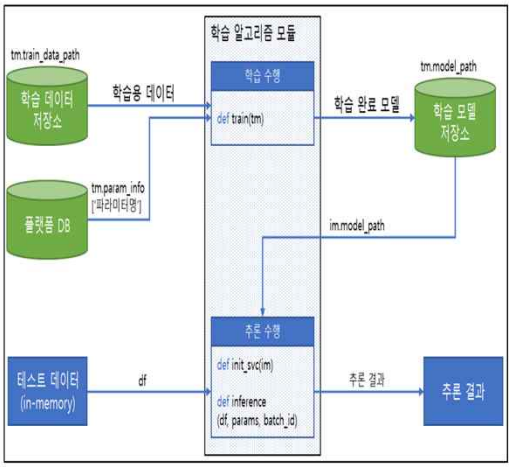
- 파일명 : image_anomaly_detection_preprocess.py
  - ✓ from image_anomaly_detection_preprocess_sub import exec_process
  - ✓ def process_for_train(pm):
  - ✓ def init_svc(im, rule):
  - ✓ def transform(df, params, batch_id):
- 파일명: image_anomaly_detection_preprocess_sub.py
  - ✓ def exec_process(pm):
- 파일명: image_anomaly_detection_train.py
  - ✓ from image_anomaly_detection_train_sub import exec_train, exec_init_svc, exec_inference
  - ✓ def train(tm):
  - ✓ def init_svc(im):
  - ✓ def inference(df, params, batch_id):
- 파일명: image_anomaly_detection_train_sub.py
  - ✓ def exec_train(tm):
  - ✓ def exec_init_svc(im):
  - ✓ def exec_inference(df, params, batch_id):
    1. 데이터셋 준비(Data Setup)
    2. 데이터 전처리(Data Preprocessing)
    3. 학습 모델 훈련(Train Model)
    4. 추론(Inference)

II. 프로그래밍 가이드 문서

2_platform_process



[전처리 모듈 업무흐름도]



[알고리즘 모듈 업무흐름도]

0. 빅데이터/인공지능 통합 플랫폼 [T3Q.ai]
   - 빅데이터 플랫폼 [T3Q.cep]
   - 인공지능 플랫폼 [T3Q.dl]
   - 빅데이터/인공지능 통합 플랫폼 [T3Q.ai (T3Q.cep + T3Q.dl)]

1. 머신러닝(Machine Learning)과 딥러닝(Deep Learning) 프로그래밍 패턴
   (1) 데이터셋 불러오기(Dataset Loading)
   (2) 데이터 전처리(Data Preprocessing)
      - 데이터 정규화(Normalization)
      - 학습과 테스트 데이터 분할(Train/Test Data Split) 등
   (3) 학습 모델 구성(Train Model Build)
   (4) 학습(Model Training)
   (5) 학습 모델 성능 검증(Model Performance Validation)
   (6) 학습 모델 저장(배포) 하기(Model Save)
   (7) 추론 데이터 전처리((Data Preprocessing)
   (8) 추론(Inference) 또는 예측(Prediction)
   (9) 추론 결과 데이터 후처리(Data Postprocessing)

2. 빅데이터/인공지능 통합 플랫폼[T3Q.ai]에서 딥러닝 프로그래밍 하고 인공지능 서비스 실시간 운용하기
   - 6개의 함수로 딥러닝 프로그래밍 하고 인공지능 서비스 실시간 운용하기

   (1) process_for_train(pm) 함수
      - 데이터셋 준비(Dataset Setup)에 필요한 코드 작성
   (2) init_svc(im, rule) 함수
      - 전처리 객체 불러오기 에 필요한 코드 작성(생략 가능)
   (3) transform(df, params, batch_id) 함수
      - 추론 데이터 전처리((Data Preprocessing) 에 필요한 코드 작성(생략 가능)

(4) train(tm) 함수
- 데이터셋 불러오기(Dataset Loading)
- 데이터 전처리(Data Preprocessing)
- 학습 모델 구성(Train Model Build)
- 학습(Model Training)
- 학습 모델 성능 검증(Model Performance Validation)
- 전처리 객체 저장
- 학습 모델 저장(배포) 하기에 필요한 코드 작성

(5) init_svc(im) 함수
- 전처리 객체 불러오기
- 학습모델 객체 불러오기에 필요한 코드 작성

(6) inference(df, params, batch_id) 함수
- 추론 데이터 전처리((Data Preprocessing)
- 추론(Inference) 또는 예측(Prediction)
- 추론 결과 데이터 후처리(Data Postprocessing)에 필요한 코드 작성

==========

3. 전처리 모듈 관리, 학습 알고리즘 관리 함수 설명

1) 프로젝트 설정/전처리모듈 관리 함수

```
def process_for_train(pm):
    """
    (1) 입력: pm
      # pm.source_path: 학습플랫폼/데이터셋 관리 메뉴에서 저장한 데이터를 불러오는 경로
      # pm.target_path: 처리 완료된 데이터를 저장하는 경로
    (2) 출력: None
    (3) 설명:
      # 데이터셋 관리 메뉴에서 저장한 데이터를 불러와서 필요한 처리를 수행
      # 처리 완료된 데이터를 저장하는 기능, pm.target_path에 저장
      # train(tm) 함수의 tm.train_data_path를 통해 데이터를 불러와서 전처리와 학습을 수행
    """

def init_svc(im, rule):
    """
      (1) 입력: im, rule
      (2) 출력: None
      (3) 설명:
        # process_for_train(pm) 함수에서 저장한 전처리 객체와 데이터에 적용된 룰(rule)을
          불러오는 기능
        # 전처리 객체, 룰(rule) 불러오기 기능 없이 처리
    """
    return {}

def transform(df, params, batch_id):
    """
      (1) 입력: df, params, batch_id
        # df: 추론모델관리와 추론API관리, 실시간 추론을 통해 전달되는 추론 입력 데이터
          (dataframe 형태)
        # params: init_svc(im, rule) 함수의 리턴(return) 값을 params 변수로 전달
      (2) 출력: df
      (3) 설명:
        # df(추론 입력 데이터)에 대한 전처리를 수행한 후 전처리 된 데이터를
          inference(df, ...) 함수의 입력 df에 전달하는 기능
        # df(추론 입력 데이터)를 전처리 없이 inference(df, params, batch_id) 함수의 입력 df
          에 리턴(return)
      """
      return df
```

2) 프로젝트 설정/학습 알고리즘 관리 함수

```
    def train(tm):
        """
        (1) 입력: tm
          # tm.train_data_path: pm.target_path에 저장한 데이터를 불러오는 경로
          # tm.model_path: 전처리 객체와 학습 모델 객체를 저장하는 경로
        (2) 출력: None
        (3) 설명:
          # pm.target_path에 저장한 데이터를 tm.train_data_path를 통해 데이터를 불러오는 기능
          # 데이터 전처리와 학습 모델을 구성하고 모델 학습을 수행
          # 학습 모델의 성능을 검증하고 배포할 학습 모델을 저장
          # 전처리 객체와 학습 모델 객체를 저장, tm.model_path에 저장
          # init_svc(im) 함수의 im.model_path를 통해 전처리 객체와 학습 모델 객체를 준비
        """

    def init_svc(im):
        """
        (1) 입력: im
          # im.model_path: tm.model_path에 저장한 전처리 객체와 학습 모델 객체 등을
            불러오는 경로
        (2) 출력: 전처리 객체와 학습 모델 객체 등을 딕셔너리(dictionary) 형태로 리턴(return)
        (3) 설명:
          # tm.model_path에 저장한 전처리 객체와 학습 모델 객체 등을 불러오는 기능
          # 전처리 객체, 룰(rule) 불러오기 기능 없이 처리
          # 전처리 객체와 학습 모델 객체 등을 딕셔너리(dictionary) 형태로 리턴(return)
          # 리턴(return) 값을 inference(df, params, batch_id) 함수의 입력 params 변수로 전달
        """

        return { "model": model, "param": param }

    def inference(df, params, batch_id):
        """
        (1) 입력: df, params, batch_id
          # df: transform(df, params, batch_id)함수의 리턴(return) 값으로 전달된 df, 추론 입력
            데이터 (dataframe 형태)
          # params  init_svc(im) 함수의 return 값을 params 변수로 전달
              ## 학습 모델 객체 사용 예시        model=params['model']
              ## 전처리(pca) 객체 사용 예시      pca=params['pca']
        (2) 출력: 추론 결과를 딕셔너리(dictionary) 형태로 리턴(return)
        (3) 설명:
          # 전처리 객체를 사용하여 df(추론 입력 데이터)에 대한 전처리 수행
          # 배포된 학습 모델(model)을 사용하여 df(추론 입력 데이터)에 추론(예측)을 수행
          # 추론 결과를 딕셔너리(dictionary) 형태로 리턴(return)
        """
        return {"inference": result}
```

==========

4. 전처리 모듈 관리, 학습 알고리즘 관리 함수 설명(AI 훈민정음 프로젝트)

1) 프로젝트 설정/전처리모듈 관리 함수(AI 훈민정음 프로젝트)

```
import logging

def process_for_train(pm):
    """
    (1) 입력: pm
        # pm.source_path: 학습플랫폼/데이터셋 관리 메뉴에서 저장한 데이터를 불러오는 경로
        # pm.target_path: 처리 완료된 데이터를 저장하는 경로
    (2) 출력: None
    (3) 설명:
        # 데이터셋 관리 메뉴에서 저장한 데이터를 불러와서 필요한 처리를 수행
        # 처리 완료된 데이터를 저장하는 기능, pm.target_path에 저장
        # train(tm) 함수의 tm.train_data_path를 통해 데이터를 불러와서 전처리와 학습을 수행
    (4) 추가 설명:
        # 함수 구조는 원형대로 유지
        # 실질적인 기능을 하는 함수를 서브모듈 함수(exec_process)로 정의하여 사용함
        # 함수명                          서브함수명
        # process_for_train(pm)          exec_process(pm)
        # 함수의 정상적인 동작 체크를 위해 마지막 라인(the end line)에 로그 출력 수행
    """

    exec_process(pm)

    logging.info('[hunmin log] the end line of the function [process_for_train]')

def init_svc(im, rule):
    """
    (1) 입력: im, rule
    (2) 출력: None
    (3) 설명:
      # process_for_train(pm) 함수에서 저장한 전처리 객체와 데이터에 적용된 룰(rule)을
        불러오는 기능
      # 전처리 객체, 룰(rule) 불러오기 기능 없이 처리
    """
    return {}

def transform(df, params, batch_id):
    """
    (1) 입력: df, params, batch_id
      # df: 추론모델관리와 추론API관리, 실시간 추론을 통해 전달되는 추론 입력 데이터
       (dataframe 형태)
      # params: init_svc(im, rule) 함수의 리턴(return) 값을 params 변수로 전달
    (2) 출력: df
    (3) 설명:
      # df(추론 입력 데이터)에 대한 전처리를 수행한 후 전처리 된 데이터를 inference(df, ...)
        함수의 입력 df에 전달하는 기능
      # df(추론 입력 데이터)를 전처리 없이 inference(df, params, batch_id) 함수의 입력 df에
        리턴(return)
    (4) 추가 설명:
      # 함수 구조는 원형대로 유지
      # 함수의 정상적인 동작 체크를 위해 마지막 라인(the end line)에 로그 출력 수행
    """
    logging.info('[hunmin log] the end line of the function [transform]')
    return df
```

2) 프로젝트 설정/ 학습 알고리즘 관리 함수(AI 훈민정음 프로젝트)

```
import logging

def train(tm):
    """
    (1) 입력: tm
        # tm.train_data_path: pm.target_path에 저장한 데이터를 불러오는 경로
        # tm.model_path: 전처리 객체와 학습 모델 객체를 저장하는 경로
    (2) 출력: None
    (3) 설명:
        # pm.target_path에 저장한 데이터를 tm.train_data_path를 통해 데이터를 불러오는 기능
        # 데이터 전처리와 학습 모델을 구성하고 모델 학습을 수행
        # 학습 모델의 성능을 검증하고 배포할 학습 모델을 저장
        # 전처리 객체와 학습 모델 객체를 저장, tm.model_path에 저장
        # init_svc(im) 함수의 im.model_path를 통해 전처리 객체와 학습 모델 객체를 준비
    (4) 추가 설명:
        # 함수 구조는 원형대로 유지
        # 실질적인 기능을 하는 함수를 서브모듈 함수(exec_train)로 정의하여 사용함
        # 함수명                    서브함수명
        # train(tm)                 exec_train(tm)
        # 함수의 정상적인 동작 체크를 위해 마지막 라인(the end line)에 로그 출력 수행
    """

    exec_train(pm)

    logging.info('[hunmin log] the end line of the function [train]')

def init_svc(im):
    """
    (1) 입력: im
        # im.model_path: tm.model_path에 저장한 전처리 객체와 학습 모델 객체 등을 불러오는
          경로
    (2) 출력: 전처리 객체와 학습 모델 객체 등을 딕셔너리(dictionary) 형태로 리턴(return)
    (3) 설명:
        # tm.model_path에 저장한 전처리 객체와 학습 모델 객체 등을 불러오는 기능
        # 전처리 객체, 룰(rule) 불러오기 기능 없이 처리
        # 전처리 객체와 학습 모델 객체 등을 딕셔너리(dictionary) 형태로 리턴(return)
        # 리턴(return) 값을 inference(df, params, batch_id) 함수의 입력 params 변수로 전달
    (4) 추가 설명:
        # 함수 구조는 원형대로 유지
        # 실질적인 기능을 하는 함수를 서브모듈 함수(exec_init_svc)로 정의하여 사용함
        # 함수명                    서브함수명
        # init_svc(im)              exec_init_svc(im)
        # 함수의 정상적인 동작 체크를 위해 마지막 라인(the end line)에 로그 출력 수행

    """
    params = exec_init_svc(im)
    logging.info('[hunmin log] the end line of the function [init_svc]')
    return {**params}
```

```
def inference(df, params, batch_id):
        """
        (1) 입력: df, params, batch_id
          # df: transform(df, params, batch_id)함수의 리턴(return) 값으로 전달된 df, 추론 입력
            데이터(dataframe 형태)
          # params: init_svc(im) 함수의 리턴(return) 값을 params 변수로 전달
            ## 학습 모델 객체 사용 예시        model=params['model']
            ## 전처리(pca) 객체 사용 예시      pca=params['pca']
        (2) 출력: 추론 결과를 딕셔너리(dictionary) 형태로 리턴(return)
        (3) 설명:
          # 전처리 객체를 사용하여 df(추론 입력 데이터)에 대한 전처리 수행
          # 배포된 학습 모델(model)을 사용하여 df(추론 입력 데이터)에 추론(예측)을 수행
          # 추론 결과를 딕셔너리(dictionary) 형태로 리턴(return)
        (4) 추가 설명:
          # 함수 구조는 원형대로 유지
          # 실질적인 기능을 하는 함수를 서브모듈 함수(exec_inference)로 정의하여 사용함
          # 함수명                                    서브함수명
          # inference(df, params, batch_id)         exec_inference(df, params, batch_id)
          # 함수의 정상적인 동작 체크를 위해 마지막 라인(the end line)에 로그 출력 수행
        """

        result = exec_inference(df, params, batch_id)
        logging.info('[hunmin log] the end line     of the function [inference]')
        return {**result}
```

수행절차 소개

01) T3Q.cep_데이터수집 파이프라인_image_anomaly_detection : 해당 예제에서는 수행 절차 없음

02) T3Q.cep_데이터변환 파이프라인_ image_anomaly_detection : 해당 예제에서는 수행 절차 없음

03) T3Q.dl_프로젝트 설정_실행환경 관리_ image_anomaly_detection

04) T3Q.dl_프로젝트 설정_전처리 모듈 관리_ image_anomaly_detection

05) T3Q.dl_프로젝트 설정_학습 알고리즘 관리_ image_anomaly_detection

06) T3Q.dl_학습플랫폼_데이터셋 관리_ image_anomaly_detection

07) T3Q.dl_학습플랫폼_전처리 모델 설계_ image_anomaly_detection

08) T3Q.dl_학습플랫폼_전처리 모델 관리_ image_anomaly_detection

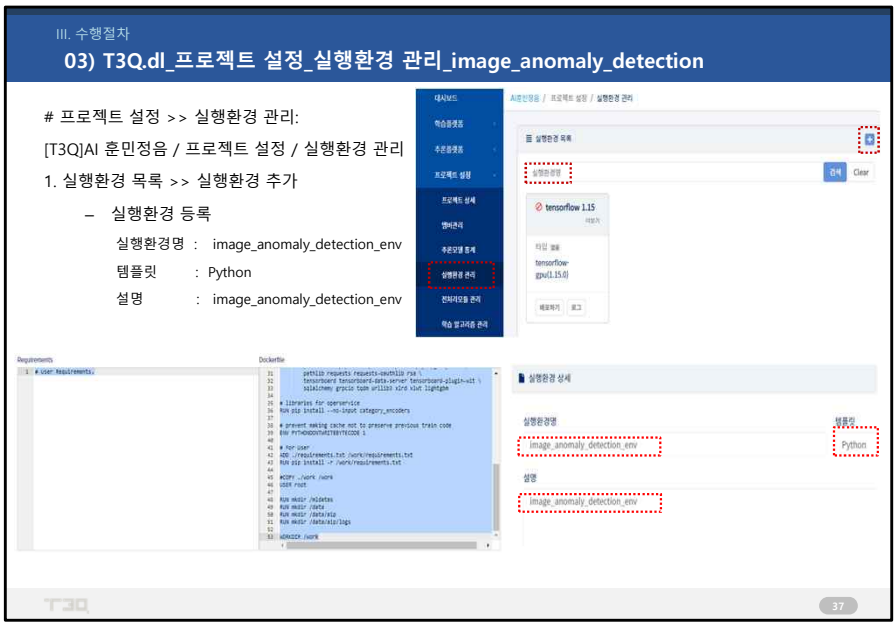
09) T3Q.dl_학습플랫폼_학습모델 설계_ image_anomaly_detection

10) T3Q.dl_학습플랫폼_학습모델 관리_ image_anomaly_detection

11) T3Q.dl_추론플랫폼_추론모델 관리_ image_anomaly_detection

12) T3Q.dl_추론플랫폼_추론API관리_ image_anomaly_detection

13) T3Q.cep_실시간 추론 파이프라인_ image_anomaly_detection



실행환경 추가 내용 및 절차

1) Requirements

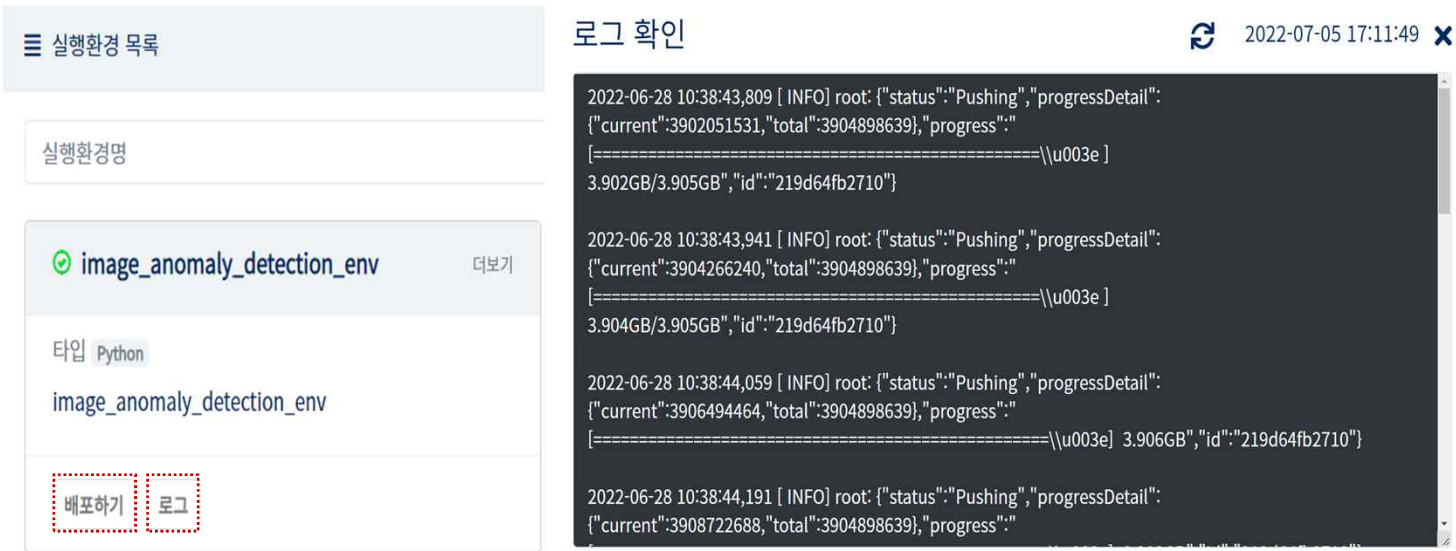
```
==========
# User Requirements.
==========
```

2) Dockerfile

```
==========
FROM tensorflow/tensorflow:2.4.1-gpu

ARG DEBIAN_FRONTEND=noninteractive

RUN apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
A4B469963BF863CC

RUN apt-get update && apt-get install -y wget ₩
    python3.8 ₩
    python3-pip ₩
    python3-dev ₩
    python3.8-dev ₩
    postgresql ₩
    libpq-dev

RUN pip3 install --upgrade pip

# libraries for operservice
RUN pip install --no-input kubernetes pygresql pyjwt pyarrow pandas ₩
        flask flask-sqlalchemy flask-cors flask-bcrypt flask-migrate flask-restful
flask-rest-jsonapi

# opencv
```

```
RUN apt-get -y install libgl1-mesa-glx
```

```
# generic libraries
RUN pip install --no-input numpy==1.19.5 ₩
        torch scikit-learn imbalanced-learn xgboost ₩
        fastai keras keras-preprocessing keras-vis ₩
        matplotlib pillow nltk ₩
        opencv-contrib-python opencv-python openpyxl imageio pretty_midi ₩
        pickleshare pip-tools protobuf psutil psycopg2 PyYAML ₩
        pathlib requests requests-oauthlib rsa ₩
        tensorboard tensorboard-data-server tensorboard-plugin-wit ₩
        sqlalchemy grpcio tqdm urllib3 xlrd xlwt lightgbm


# libraries for operservice
RUN pip install --no-input category_encoders

# prevent making cache not to preserve previous train code
ENV PYTHONDONTWRITEBYTECODE 1

# For User
ADD ./requirements.txt /work/requirements.txt
RUN pip install -r /work/requirements.txt

#COPY ./work /work
USER root

RUN mkdir /mldatas
RUN mkdir /data
RUN mkdir /data/aip
RUN mkdir /data/aip/logs

WORKDIR /work
```



04) T3Q.dl_프로젝트 설정_전처리 모듈 관리_image_anomaly_detection

- 프로젝트 설정>>전처리모듈관리
  - 전처리모듈 관리>> [전처리 모듈 추가] 실행
  - 전처리모듈 등록 시 순서와 입력 정보
  ① 기본정보
    - 전처리명: image_anomaly_detection_premodule
    - 실행환경 선택 : image_anomaly_detection_env
    - GPU지원 : GPU 지원(체크)
  ② 입력 형태: file
  ③ 출력 형태: default
  ④ 파라미터
  ⑤ 소스 코드 : T3Q.ai_platform_image_anomaly_detection_preprocess.zip [파일업로드]
    - LICENSE.txt
    - README.txt
    - image_anomaly_detection_preprocess.py (실행모듈 선택)
    - image_anomaly_detection_preprocess_sub.py
    - platform_process.txt

- 전처리모듈 등록 시 순서와 입력 정보

① 기본정보

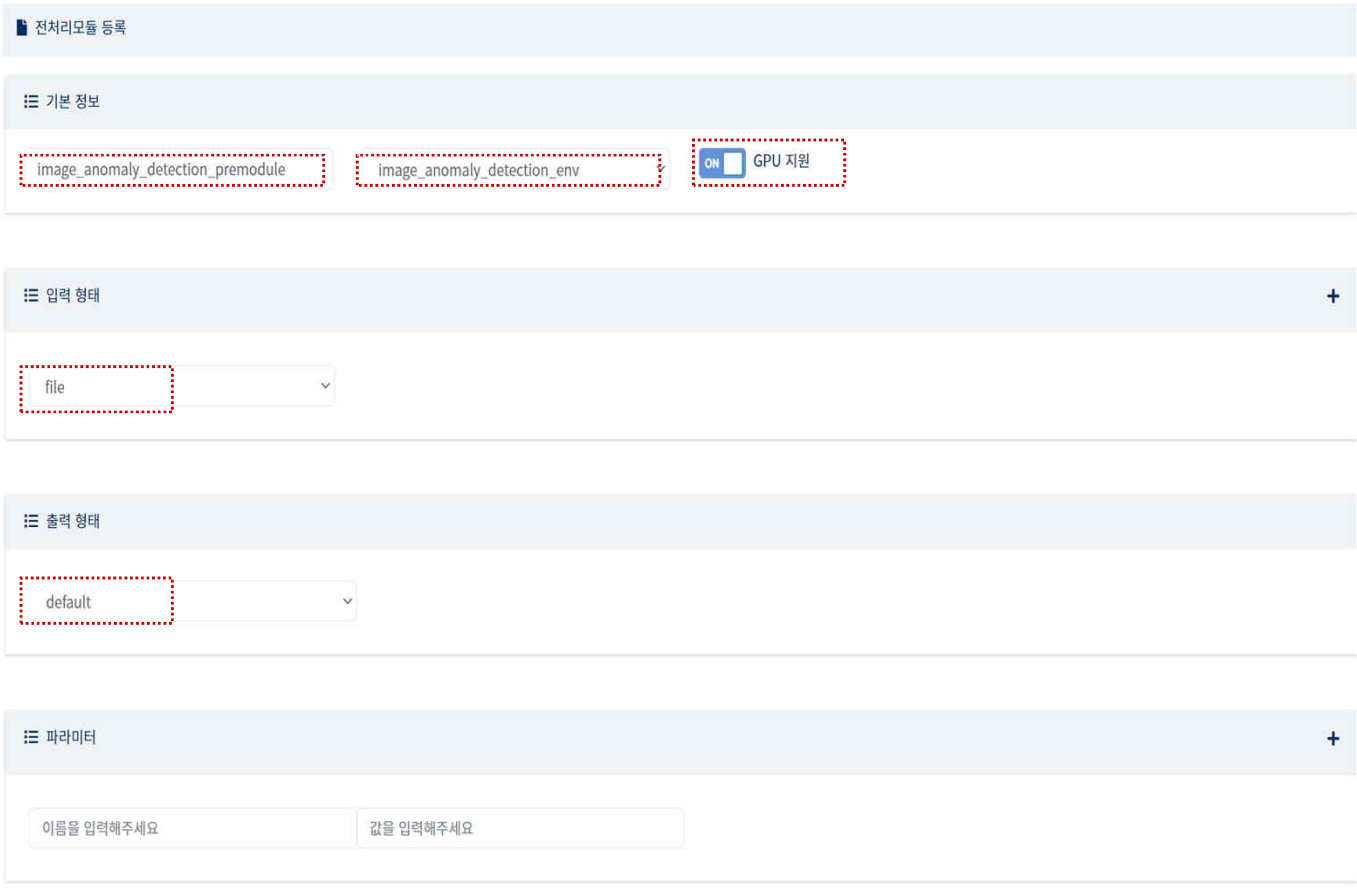
전처리명: image_anomaly_detection_premodule

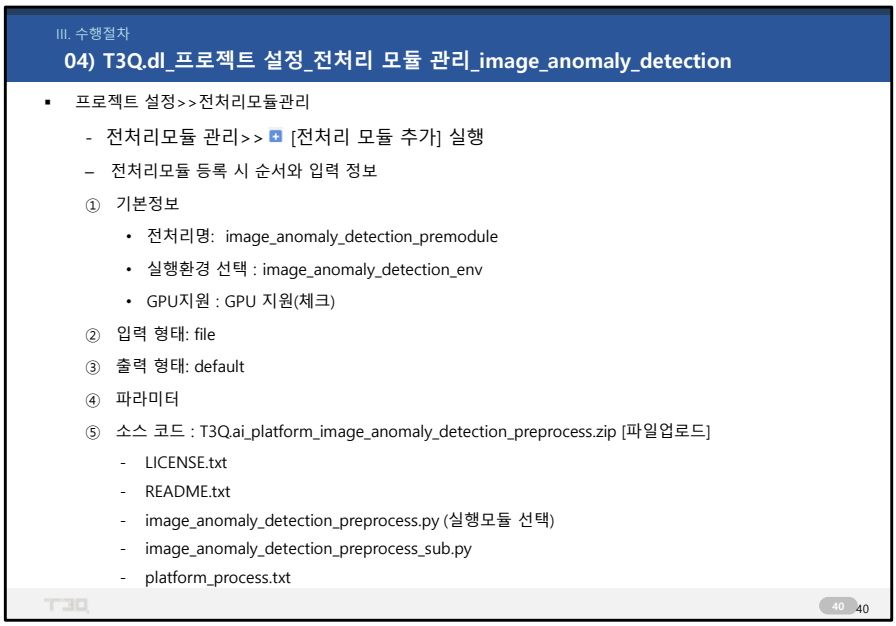
실행환경 선택 : image_anomaly_detection_env

GPU지원 : GPU 지원(체크)

② 입력 형태: file

③ 출력 형태: default

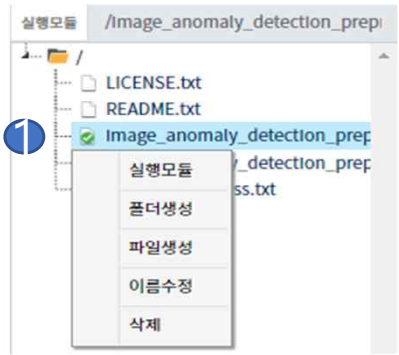




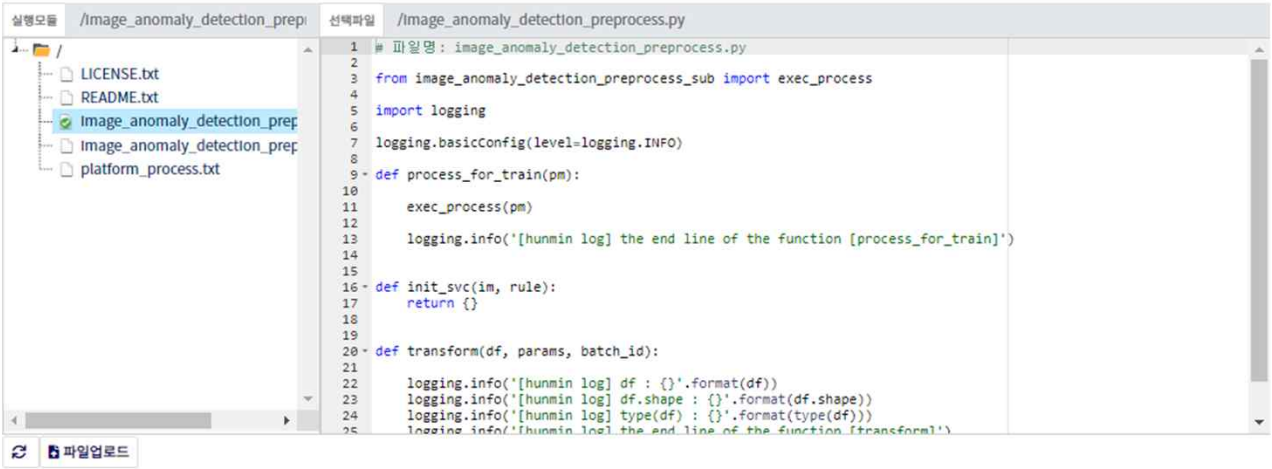
전처리모듈 등록 시 순서와 입력 정보

⑤ 소스 코드 : T3Q.ai_platform_image_anomaly_detection_preprocess.zip [파일업로드] 누름

- LICENSE.txt
- README.txt
- image_anomaly_detection_preprocess.py(실행모듈 선택)
- image_anomaly_detection_preprocess_sub.py
- platform_process.txt



② 소스 코드



05) T3Q.dl_프로젝트 설정_학습 알고리즘 관리_image_anomaly_detection

- 프로젝트 설정 >> 학습 알고리즘 관리
  - 학습 알고리즘 관리>>
    - [알고리즘 추가] 실행

- 학습 알고리즘 등록 시 순서와 입력 정보

① 기본정보: 다음과 같이 입력
-알고리즘명: image_anomaly_detection_train
-설명 : image_anomaly_detection_train
-카테고리 : Anomaly Detection
-실행환경 : image_anomaly_detection_env
-GPU 지원 : 체크

② 공통 파라미터: 아래 1가지 항목 사용
- 초기화 방법 : 사용

③ 모델 파라미터

④ 시각화 설정: 아래 4개 항목만 체크
- Accuracy : 체크
- Loss : 체크
- Confusion Matrix : 체크
- Precision/Recall/F1-score : 체크

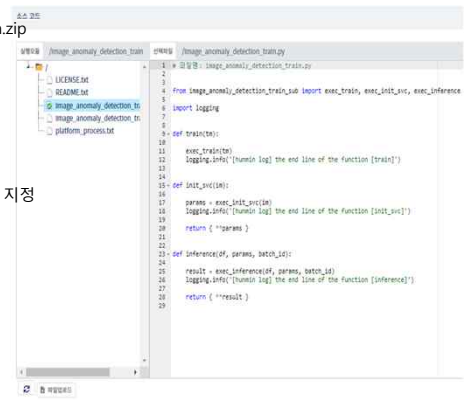
05) T3Q.dl_프로젝트 설정_학습 알고리즘 관리_image_anomaly_detection

- 프로젝트 설정 >> 학습 알고리즘 관리
  - 학습 알고리즘 관리>> [알고리즘 추가] 실행
  - 학습 알고리즘 등록 시 순서와 입력 정보

⑤ 소스코드 업로드 : [파일업로드]

: T3Q.ai_platform_image_anomaly_detection_train.zip
- LICENSE.txt
- README.txt
- image_anomaly_detection_train.py
- image_anomaly_detection_train_sub.py
- platform_process.txt

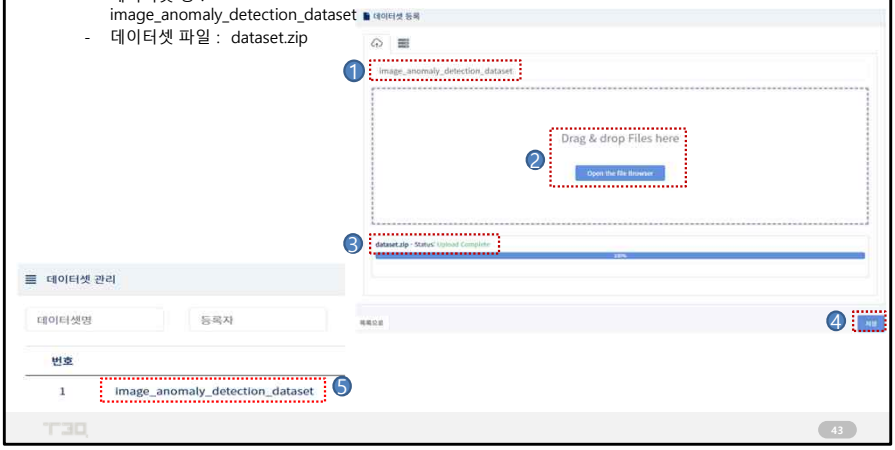
: 실행 모듈 설정 >> [저장]
- /image_anomaly_detection_train.py 실행 모듈 지정
- [저장]을 누른다.



III. 수행절차

06) T3Q.dl_학습플랫폼_데이터셋 관리_image_anomaly_detection

- 학습플랫폼 >> 데이터셋 관리
- 데이터셋 등록 절차
  - 데이터셋 관리>> 등록 실행
  - 데이터셋 명 : image_anomaly_detection_dataset
  - 데이터셋 파일 : dataset.zip

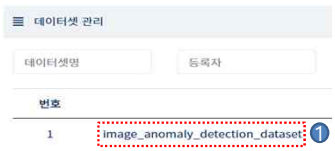


07) T3Q.dl_학습플랫폼_전처리 모델 설계_image_anomaly_detection

- 학습플랫폼 >> 전처리 모델 설계
- 전처리 모델 설계 절차

① 학습플랫폼 >> 데이터셋 관리
   - 데이터셋 명 : image_anomaly_detection_dataset 선택

② image_anomaly_detection_dataset 아래의 [전처리 모델 설계] 선택



III. 수행절차

07) T3Q.dl_학습플랫폼_전처리 모델 설계_image_anomaly_detection

- 학습플랫폼 >> 전처리 모델 설계

② 전처리 모델 설계 절차

Step1. 기본 정보

- 모델명: image_anomaly_detection_premodel

Step2. 데이터셋 등록:

- 참조데이터셋 : image_anomaly_detection_dataset

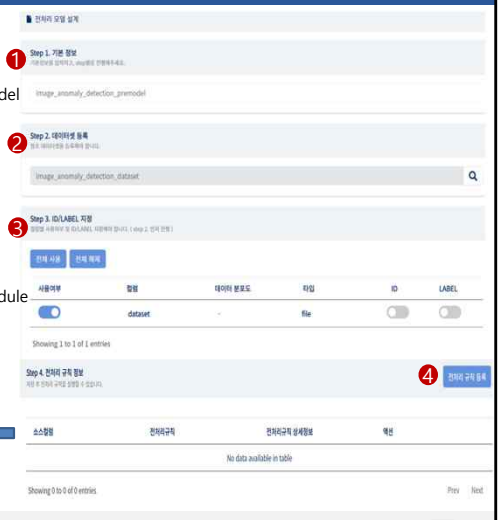
Step3. ID/LABEL 지정

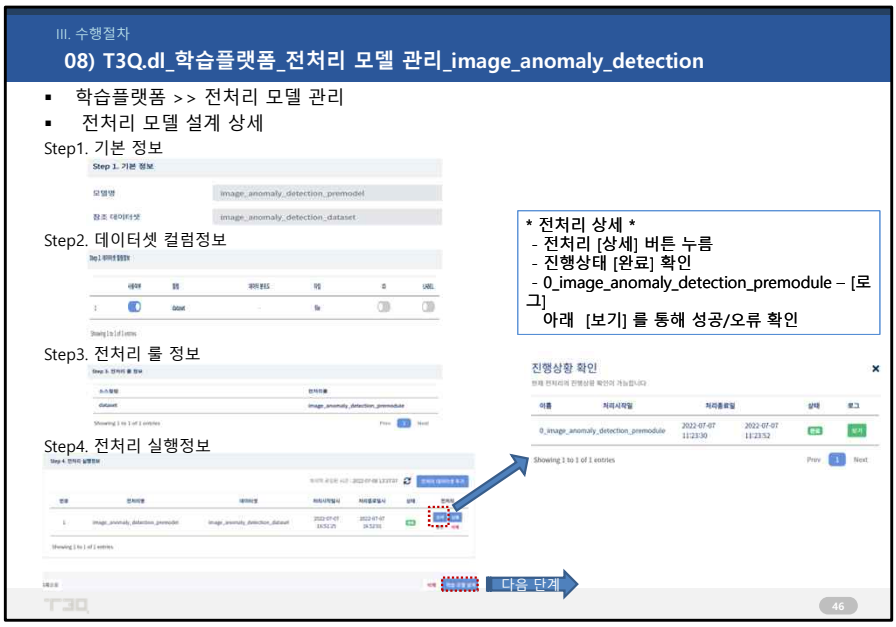
Step4. 전처리 규칙 정보

③ - [전처리 규칙 등록] 선택

전처리 규칙 등록 시 절차

- 소스컬럼 : dataset(file) 선택
- 변환 함수 : image_anomaly_detection_premodule 선택 후 [저장]





▪ 학습플랫폼 >> 전처리 모델 관리
▪ 전처리 모델 설계 상세
Step1. 기본 정보
- 모델명: image_anomaly_detection_premodel
- 참조데이터셋 : image_anomaly_detection_dataset
Step2. 데이터셋 컬럼정보
Step3. 전처리 룰 정보
Step4. 전처리 실행정보
전처리 상세
- 전처리 상세 버튼 누름
- 진행상황 확인
- 0_image_anomaly_detection_premodule – [로그] 아래 [보기] 누름
[학습 모델 설계] 선택하여 다음 단계 진행

로그 확인

마지막 로딩된 시간 : 2022-07-06 14:15:44

```
'/data/aip/logs'}
2022-06-28 10:12:50,662 [ WARN] root: datasource_repo_id : 123, datasource_repo_obj : <DataSourceRepo 123>, repo_type : file
2022-06-28 10:12:50,708 [ INFO] root: module_path=/data/aip/logs/t3qai/premodule/premodule_513/1
2022-06-28 10:12:50,722 [ INFO] root: dp_module=<module 'image_anomaly_detect_preprocess' from
'/data/aip/logs/t3qai/premodule/premodule_513/1/image_anomaly_detect_preprocess.py'>
2022-06-28 10:12:50,723 [ INFO] root: [hunmin log]  the start line of the function [exec_process]
2022-06-28 10:12:50,723 [ INFO] root: [hunmin log] pm.source_path : /data/aip/file_group/t3qai/411
2022-06-28 10:12:50,724 [ INFO] root: [hunmin log] Files and directories in /data/aip/file_group/t3qai/411 :
2022-06-28 10:12:50,724 [ INFO] root: [hunmin log] dir_list : ['dataset.zip']
2022-06-28 10:13:13,054 [ INFO] root: [hunmin log] Files and directories in /data/aip/dataset/t3qai/pm/pm_678/ds_685 :
2022-06-28 10:13:13,055 [ INFO] root: [hunmin log] dir_list : ['discard', 'original']
2022-06-28 10:13:13,055 [ INFO] root: [hunmin log]  the finish line of the function [exec_process]
2022-06-28 10:13:13,059 [ INFO] root: [hunmin log] the end line of the function [process_for_train]
2022-06-28 10:13:13,102 [ INFO] root: ### preprocessing finished ###
2022-06-28 10:13:13,102 [ INFO] root: Status : total:1, success:1, error:0
```

학습플랫폼 >> 학습모델 설계

1. AI 훈민정음 >> 학습플랫폼 >> 학습모델 설계 상세 과정
   1) Step 1. 기본 정보
      학습모델명 image_anomaly_detection_trainmodel
      전처리모델
        [사용] 체크
        image_anomaly_detection_premodel image_anomaly_detection_premodel

   2) Step 2. 모델 설계
      문제유형 Anomaly detection
      알고리즘 image_anomaly_detection_train
      평가방법
        #Train-Test Split : 80
        None

   3) Step 3. 상세 설계
      알고리즘 선택 시 상세 설계가 가능합니다. (step 2. 먼저 진행)

      (1) 공통 파라미터
          초기화방법 : Xavier uniform

   4) [저장] 누름

2. AI 훈민정음 >> 학습플랫폼 >> [학습모델 관리] 에서 등록된 학습 모델 확인

학습플랫폼 >> 학습모델 관리

1. 학습플랫폼 >> 학습모델 관리

2 학습모델 상세

1) Step 1. 학습모델 기본정보
모델명 image_anomaly_detection_trainmodel
전처리 모델명 image_anomaly_detection_premodel
image_anomaly_detection_premodel

2) Step 2. 학습모델 알고리즘 정보
문제유형 Anomaly detection
알고리즘 image_anomaly_detection_train
평가방법 / 평가값 none 없음

3) Step 3. 학습수행 관리정보
(1) 공통 파라미터

초기화방법	Xavier uniform
랜덤seed	777
CPU	16 코어
Memory	32 Gi
GPU	2 코어

(2) 학습상태
학습상태 시작전 [- ~ -]

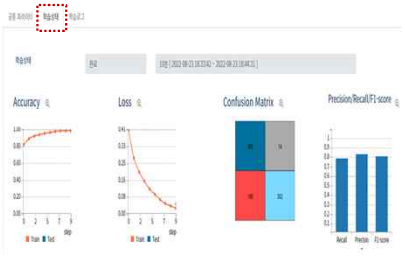
Accuracy Loss Confusion Matrix Precision/Recall/F1-score

[실행] 버튼 누름

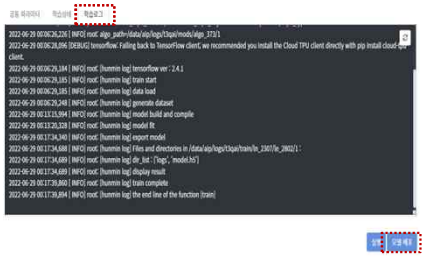
III. 수행절차

10) T3Q.dl_학습플랫폼_학습모델 관리_image_anomaly_detection

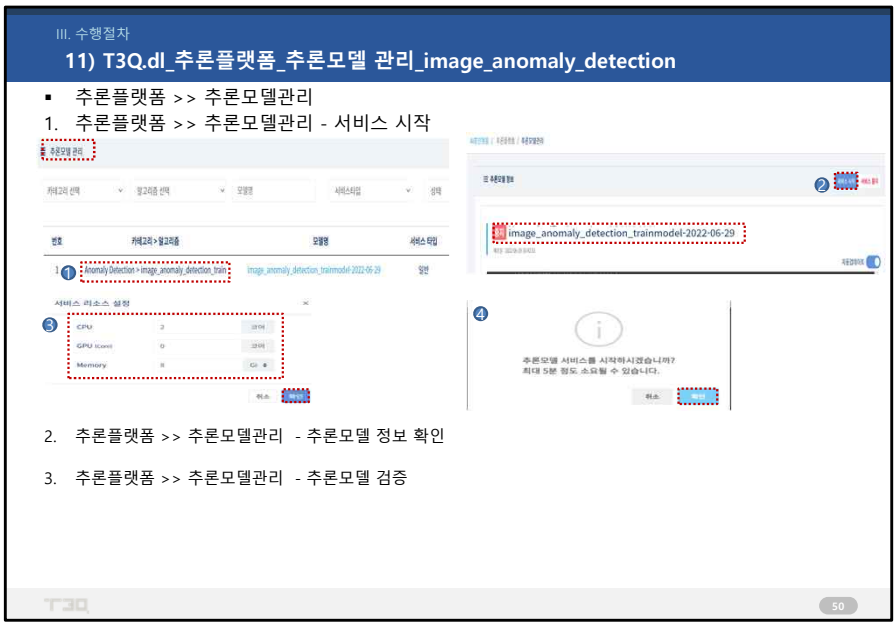
- 학습플랫폼 >> 학습모델 관리
- 학습모델 상세
- [실행] 완료 후
  - [학습상태]



- [학습로그]

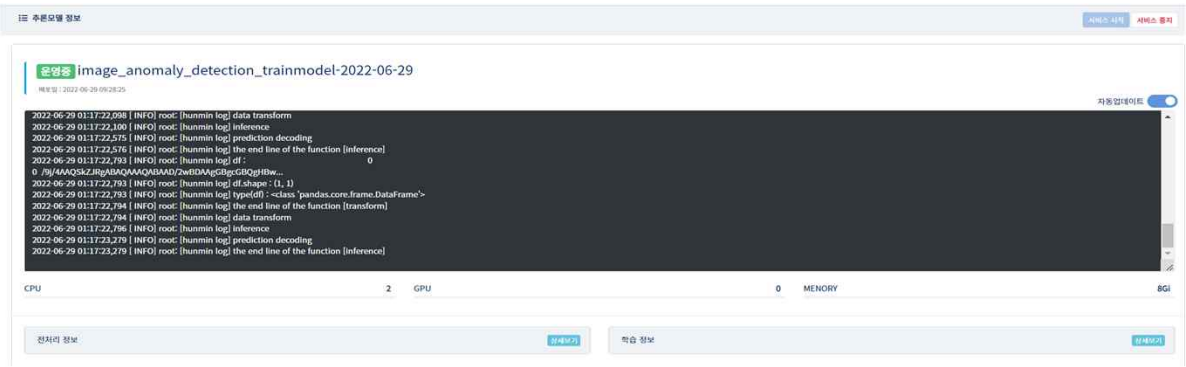


- 모델 배포 : [모델 배포] 버튼 누르고 [배포] 누름



추론플랫폼 >> 추론모델관리

1. 추론플랫폼 >> 추론모델관리 - 서비스 시작
2. 추론플랫폼 >> 추론모델관리 – 추론모델 정보 확인



3. 추론플랫폼 >> 추론모델관리 - 추론모델 검증
[추론모델테스트]-[요청] 에 아래의 값을 넣고, [테스트] 눌러 수행
요청 [['/data/aip/file_group/pm/pm_334/ds_441/image/1/1230.png']]
요청 -> (11_1) Request_image_clustering.txt' 파일 안에 있는 base64 형식의 이미지의 내용을 복사 붙여넣기 한다

12) T3Q.dl_추론플랫폼_추론API관리_image_anomaly_detection

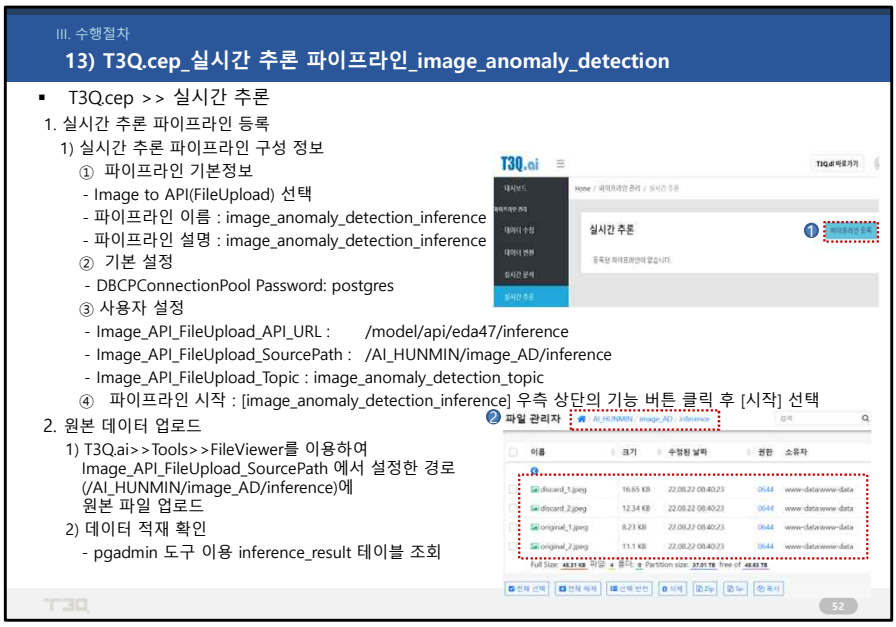
- 추론플랫폼 >> 추론API관리

1. 추론플랫폼 >> 추론API관리 – 신규 등록

2. 추론플랫폼 >> 추론API관리 –추론 API 상세

▪추론플랫폼 >> 추론API관리
1.추론플랫폼 >> 추론API관리 – 신규 등록
2.추론플랫폼 >> 추론API관리 – 추론 API 상세

=> 요청 :{ “data”:” [[‘테스트 데이터 값 입력=’]]”}=>[API 호출] 클릭
=> 응답 : {"data":[” 결과 "]}



(2) 데이터 적재 확인
T3Q.ai >> Tools>> PgAdmin

#inference_origin
inference_result

테이블 조회

#select * from inference_origin;

select * from inference_result;

데이터 저장 확인

==========
SELECT * FROM public.inference_result
where url like '%/model/api/eda47/inference%'
order by start_time desc