



목 차

I. 개요

1. 소개

II. 프로그래밍 가이드 문서

0_local_image_classification_requirement

0_local_image_classification.ipynb

1_local_platform_image_classification.ipynb

2_platform_process

III. 수행 절차

01) T3Q.cep_데이터수집 파이프라인_image_classification

02) T3Q.cep_데이터변환 파이프라인_image_classification

03) T3Q.dl_프로젝트 설정_실행환경 관리_image_classification

04) T3Q.dl_프로젝트 설정_전처리 모듈 관리_image_classification

05) T3Q.dl_프로젝트 설정_학습 알고리즘 관리_image_classification

06) T3Q.dl_학습플랫폼_데이터셋 관리_image_classification

07) T3Q.dl_학습플랫폼_전처리 모델 설계_image_classification

08) T3Q.dl_학습플랫폼_전처리 모델 관리_image_classification

09) T3Q.dl_학습플랫폼_학습모델 설계_image_classification

10) T3Q.dl_학습플랫폼_학습모델 관리_image_classification

11) T3Q.dl_추론플랫폼_추론모델 관리_image_classification

12) T3Q.dl_추론플랫폼_추론API관리_image_classification

13) T3Q.cep_실시간 추론 파이프라인_image_classification

I. 개요

1. 소개 : 손그림 이미지 분류

Google에서 제공하는 numpy 형태의 낙서 이미지 데이터셋을
10개의 목록으로 분류하는 예제

1. 데이터셋

The Quick, Draw! Dataset:
345가지 목록으로 이루어진
numpy 형식의 낙서 이미지 데이터셋

이 중 [10가지 목록](#) 사용

2. 전처리 및 학습

전처리:
데이터 [정규화](#), 훈련 및 평가 데이터셋
생성, 레이블 생성 및 [원-핫 인코딩](#)

학습:
[CNN](#)을 활용한 분류 학습

3. 추론 결과











개미, 사과, 버스, 나비, 컵, 봉투, 물고기,
기린, 전구, 돼지
총 10가지로 낙서 이미지 분류

4. 기대 효과

인간의 공통적인 낙서 그리는 패턴 파악
낙서 맞추기 게임 활용 가능(현재 구글
운영중)

- 개미, 사과, 버스, 나비, 컵, 봉투, 물고기, 기린, 전구, 돼지 총 10개의 numpy 파일로 이루어져 있음

numpy 파일

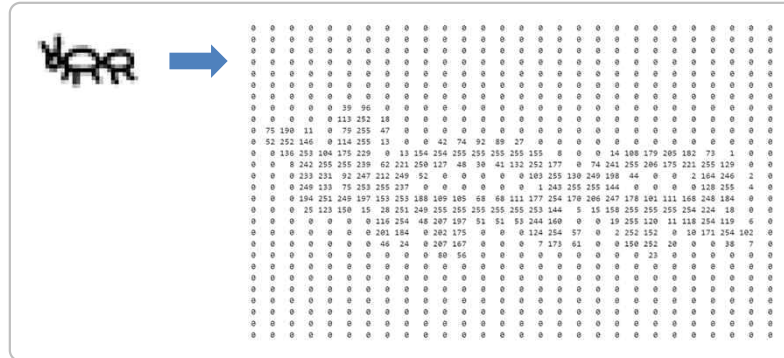
 ant NPY 파일 93.1MB	 apple NPY 파일 108MB
 bus NPY 파일 124MB	 butterfly NPY 파일 88.2MB
 cup NPY 파일 97.7MB	 envelope NPY 파일 100MB
 fish NPY 파일 100MB	 giraffe NPY 파일 95.0MB
 lightbulb NPY 파일 90.3MB	 pig NPY 파일 139MB

numpy 파일 시각화



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

이미지 픽셀화



원-핫 인코딩

- 표현하고 싶은 단어의 인덱스에 1의 값을 부여하고, 다른 인덱스에는 0을 부여하는 벡터 표현 방식
- 범주형 데이터를 계산할 수 있는 형태로 만들어 주는 작업

개미 데이터 원-핫 인코딩

개미	사과	버스	나비	컵	봉투	물고기	기린	전구	돼지
1	0	0	0	0	0	0	0	0	0

10개의 레이블 중 개미는 첫 번째에 해당
원-핫 인코딩을 통해 개미를 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]으로 표현



- 인간의 시신경을 모방하여 만든 딥러닝 구조로 이미지 분류에 주로 사용
- 입력 이미지와 필터를 합성곱 계산하여 이미지의 특징을 추출함
- 추출한 특징을 활용해 해당하는 이미지의 레이블을 찾아냄

합성곱 계산

입력 이미지

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

필터

1	0	1
1	0	1
0	1	0

특징 맵

6	9	11
10	4	4
7	7	4

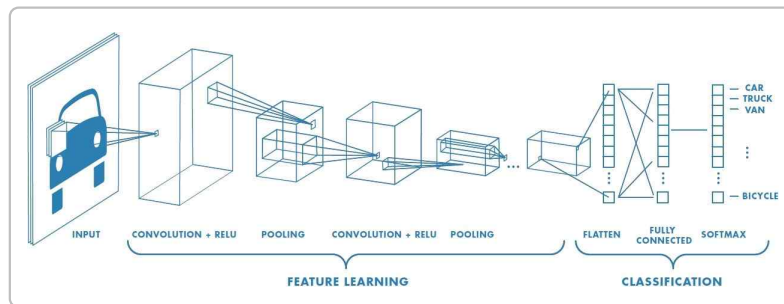
$$(1 \times 1) + (2 \times 0) + (3 \times 1) + (2 \times 1) + (1 \times 0) + (0 \times 1) + (3 \times 0) + (0 \times 1) + (1 \times 0) = 6$$



I. 개요 CNN

- 예제에서는 직접 층을 쌓아 CNN 모델 생성

일반적인 CNN 구조



이미지 출처: <https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html>

예제에서 사용한 CNN 모델 구조

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 32)	320
conv2d_1 (Conv2D)	(None, 28, 28, 64)	18496
dropout (Dropout)	(None, 28, 28, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 64)	36928
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
flatten (Flatten)	(None, 12544)	0
dense (Dense)	(None, 32)	401440
dropout_1 (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 10)	330
Total params: 457,514		
Trainable params: 457,514		
Non-trainable params: 0		

0_local_image_classification_requirement

- 로컬에 설치 되어야 할 패키지 버전
 - ✓ numpy 1.22.2
 - ✓ opencv-python 4.5.4.60
 - ✓ pandas 1.3.3
 - ✓ matplotlib 3.4.3
 - ✓ scikit-learn 1.1.1
 - ✓ tensorflow 2.9.0
 - ✓ tensorflow-gpu 2.6.0
 - ✓ keras 2.9.0

0_local_image_classification.ipynb

- 로컬 개발 코드
 - ✓ 로컬에서 주피터 노트북(Jupyter Notebook), 주피터 랩(JupyterLab) 또는 파이썬(Python)을 이용한다.
 - ✓ 사이킷 런(scikit-learn), 텐서플로우(tensorflow), 파이토치(pytorch)를 사용하여 딥러닝 프로그램을 개발한다.
 - ✓ 파일명: 0_local_image_classification.ipynb
 - 로컬 개발 워크플로우(workflow)
 - ✓ 로컬 개발 워크플로우를 다음의 4단계로 분리한다.
1. 데이터셋 준비(Data Setup)
 - 로컬 저장소에서 전처리 및 학습에 필요한 학습 데이터셋을 준비한다.
 2. 데이터 전처리(Data Preprocessing)
 - 데이터셋의 분석 및 정규화(Normalization) 등의 전처리를 수행한다.
 - 데이터를 모델 학습에 사용할 수 있도록 가공한다.
 - 추론과정에서 필요한 경우, 데이터 전처리에 사용된 객체를 meta_data 폴더 아래에 저장한다.
 3. 학습 모델 훈련(Train Model)
 - 데이터를 훈련에 사용할 수 있도록 가공한 뒤에 학습 모델을 구성한다.
 - 학습 모델을 준비된 데이터셋으로 훈련시킨다.
 - 정확도(Accuracy)나 손실(Loss) 등 학습 모델의 성능을 검증한다.
 - 학습 모델의 성능 검증 후, 학습 모델을 배포한다.
 - 배포할 학습 모델을 meta_data 폴더 아래에 저장한다.
 4. 추론(Inference)
 - 저장된 전처리 객체나 학습 모델 객체를 준비한다.
 - 추론에 필요한 테스트 데이터셋을 준비한다.
 - 배포된 학습 모델을 통해 테스트 데이터에 대한 추론을 진행한다.

0_local_image_classification.ipynb

```
=====
# Imports
import zipfile
import numpy as np
import cv2
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from keras.utils import np_utils
from tensorflow import keras
from tensorflow.keras import layers

1. 데이터셋 준비(Data Setup)

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

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

extract_zip_file.close()

my_path = './meta_data/dataset/'

# 카테고리
dataset=['ant','apple', 'bus', 'butterfly', 'cup', 'envelope','fish', 'giraffe', 'lightbulb','pig']
dataset_num= len(dataset) #10
```



```
# 경로에 있는 numpy를 load하고 dataset_numpy list에 추가한다.
dataset_numpy = []
for i in range (dataset_num):
    ad = my_path + str(dataset[i]) + '.npy'
    dataset_numpy.append(np.load(ad))
```

각 카테고리의 이미지 수와 이미지의 크기를 확인해보자

- 각 카테고리->각 numpy 파일
- 각 numpy 파일의 shape->(이미지 수, 이미지 크기)

확인해보면, 각 파일(카테고리) 별로 이미지의 수는 다르지만 각 이미지의 크기는 같음을 알 수 있다. 여기서 784는 numpy array의 크기를 말하는데, 28*28 크기의 이미지이기 때문에 총 784개의 값을 각 이미지가 가지게 된다.

```
print("( 이미지 수 , 이미지의 크기)")
for i in range (dataset_num):
    print(dataset_numpy[i].shape)
```

그런데 어떻게 numpy 파일이 이미지가 될 수 있을까?

- 간단하게 말하자면, 28 * 28 크기의 이미지는 총 784개의 픽셀로 이루어져있는데, 각 픽셀의 색을 0~255의 값으로 표현할 수 있다. 그리고 그 값들을 하나의 배열로 표현한다면, numpy array가 되는 것이다.
- 이해를 위해 아래를 보자. 아래는 Envelope(봉투) numpy 파일의 첫번째 값이다. 해당 array 값을 가로, 세로 각 각 28개씩 print할 경우, 0은 배경, 그 외의 값들은 선의 형태를 보임을 알 수 있다.

```
np.set_printoptions(linewidth=116)
# dataset_numpy[5] 가 envelope numpy 이다. 그중 0번째 array 값을 print 했다.
print(dataset_numpy[5][0])
```

배열을 이미지로 시각화

- 앞에서 확인한 배열의 값을 더 잘 시각화 해보고자 한다. 각 카테고리 별로 5개의 이미지를 시각화 해보자.

```
def plot_samples(input_array, rows=1, cols=5, title=""):
    fig, ax = plt.subplots(figsize=(cols,rows))
    ax.axis('off')
    plt.title(title)

    for i in list(range(0, min(len(input_array),(rows*cols)) )):
        a = fig.add_subplot(rows,cols,i+1)
        imgplot = plt.imshow(input_array[i].reshape((28,28)), cmap='gray_r')
        plt.xticks([])
        plt.yticks([])

for i in range (dataset_num):
    plot_samples(dataset_numpy[i], title=dataset[i]+ '\n')
```

2. 데이터 전처리 (Data Preprocessing)

데이터 준비 (Preparing Data)

- 앞서 확인하고 분석한 numpy array들을 훈련에 사용할 수 있는 형태로 바꾸고자 한다.

데이터 정규화 (Normalization)

- 지금의 numpy 배열은 전부 0 ~ 255의 값을 가진다. 해당 값들을 전부 255로 나누어 0 ~ 1의 값을 가지도록 해준다.

데이터 합치기 & 레이블 생성

- 지금 총 10개의 numpy array가 dataset_numpy list의 요소들로 존재한다. 이를 하나의 numpy array로 합쳐준다. ('concatenate')
- numpy array는 입력 데이터(X)이다. 하지만 훈련을 위해서는 입력 데이터 뿐만 아니라 정답 데이터(Y), 즉 레이블 (label)이 존재해야 한다. 레이블 array를 생성해준다. 레이블은 앞서 정의했던 dataset array의 값의 index로 한다.
- `dataset=['ant','apple', 'bus', 'butterfly', 'cup', 'envelope','fish', 'giraffe', 'lightbulb','pig']`

훈련 (train) & 평가 (test) 데이터셋 생성

- 전체 데이터 중 일부는 훈련 (train)에 사용하고, 나머지 일부는 훈련된 모델의 성능을 평가 (test)하기 위해 사용하고자 한다. ('train_test_split')

모델 훈련에 사용할 수 있는 형태로 변경

- 입력 데이터 X는 numpy 배열의 차원을 바꿔준다. ('reshape')
- 정답(레이블) 데이터 Y는 one-hot-encoding을 수행한다.
- One-hot-encoding: 0과 1로 데이터를 구별하는 인코딩이다.
- 본래 정답 데이터는 총 카테고리의 값이 10가지가 존재하므로 0에서 9의 숫자로 이루어져있다. 이를 one-hot-encoding으로 표현하면, 2의 label을 가지는 'bus'의 경우에 [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]의 형태로 인코딩된다.
- Keras에서 one-hot-encoding 함수를 지원한다. ('to_categorical')

각 카테고리별로 존재하는 이미지의 수가 다르다.

카테고리별로 같은 수의 이미지를 훈련시키기 위해 훈련시키고자 하는 이미지의 개수를 정해준다.
idx = 1000

데이터 정규화(Normalization)&데이터 합치기&레이블 생성

X: 입력 이미지 배열 데이터

Y: 정답 레이블 데이터

X = (dataset_numpy[0][:idx,:]/255.) #첫번째 카테고리의 numpy array 정규화

Y = np.full(idx,0) #첫번째 카테고리의 정답 레이블 생성

첫번째 카테고리과 같은 방식으로 정규화 및 정답 레이블 생성

정규화된 입력 데이터와 새로 만든 정답 레이블 데이터를 각각 하나의 입력 데이터 배열, 정답 데이터 배열로 만들어준다.

for i in range (1,dataset_num):

 X = np.concatenate((X,dataset_numpy[i][:idx,:]/255.), axis=0).astype('float32')

 Y = np.concatenate((Y,np.full(idx,i)), axis=0).astype('float32')

훈련&평가 데이터셋 생성

- 전체 데이터셋 중 8:2의 비율로 훈련:평가 데이터셋을 생성한다.

X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2,random_state=0)

모델 훈련에 사용할 수 있는 형태로 변경

X의 값을 [samples][pixels][width][height] 형태로 reshape한다.

X_train_cnn = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')

X_test_cnn = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')

reshape된 결과 확인 및 원래 배열의 형태와 비교한다.

print("X_train: "+str(X_train.shape))

```
print("X_train_cnn: "+str(X_train_cnn.shape))
```

```
# Y의 배열에 one-hot-encoding 진행
Y_train_cnn = np_utils.to_categorical(Y_train)
Y_test_cnn = np_utils.to_categorical(Y_test)
num_classes = Y_test_cnn.shape[1] # class는 총 10개이다.
```

```
# encoding된 결과 확인 및 원래 배열의 형태와 비교
print("Y_train: "+str(Y_train.shape))
print("Y_train_cnn: "+str(Y_train_cnn.shape))
print("class 개수: "+str(num_classes))
```

3. 학습 모델 훈련(Train Model)

이미지 분류를 위해 아주 간단한 CNN 모델을 Keras를 이용하여 구축하고자 한다.

```
#모델 구축
model = keras.Sequential(
    [
        layers.Input(shape=(28,28,1)),
        layers.Conv2D(32, kernel_size=(3, 3), padding='same', activation="relu"),
        layers.Conv2D(64, kernel_size=(3, 3), padding='same', activation="relu"),
        layers.Dropout(0.25),
        layers.Conv2D(64, kernel_size=(3, 3), padding='same', activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dense(32, activation="relu"),
        layers.Dropout(0.25),
        layers.Dense(num_classes, activation="softmax")
    ]
)
```

```
model.summary() #모델층 (layer) 정보 확인
```

모델 컴파일 및 학습

모델 컴파일(Compile Model)

- keras의 compile 함수를 통해 모델의 optimizer, loss 그리고 metrics를 선택할 수 있다.
- Loss: 우리는 레이블로 one-hot-encoding을 사용했으므로 loss로 categorical_crossentropy를 사용한다.

모델 학습(Train Model)

- 앞서 구축한 모델을 준비해준 입력 데이터와 레이블 데이터로 학습시킨다.

모델 컴파일(Compile Model)

```
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
```

모델 학습(Train Model)

```
batch_size = 50
epochs = 20
```

```
model.fit(X_train_cnn, Y_train_cnn, batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

모델 평가(Evaluate Model)

```
loss, acc = model.evaluate(X_test_cnn, Y_test_cnn, verbose=0)
print("loss:" + str(loss)+ " accuracy: " + str(acc))
```

4. 추론 (Inference)

훈련시킨 모델을 직접 사용해보고자 한다. 잘 훈련된 모델이라면 우리가 직접 그린 낙서를 입력 데이터로 주었을 때, 그 낙서의 카테고리를 결과값으로 모델이 내보낼 것이다. 그 과정을 진행해보고자 한다.

- 이미지 생성
- 이미지를 모델의 입력 데이터 형태로 변환
- 결과 확인

이미지 생성

예제에서는 미리 준비해둔 이미지 파일을 대신 받아 추론을 진행해보고자 한다.

실제 사용자가 자신의 이미지를 만들어 실행해볼 수도 있다.

- 이미지 생성법
 1. 그림판 - 파일 - 이미지 속성 에서 너비(W)/높이(H)의 값을 각각 28로 설정한다.
 2. 28 * 28 크기의 캔버스에 도구 - 연필 (1px) 을 이용하여 카테고리에 맞는 그림을 그린다. 카테고리로는 다음과 같은 것들이 있다.
 - ant (개미)
 - apple (사과)
 - bus (버스)
 - butterfly (나비)
 - cup (컵)
 - envelope (봉투)
 - fish (물고기)
 - giraffe (기린)
 - lightbulb (전구)
 - pig (돼지)

dataset.zip 파일을 dataset 폴더에 압축을 풀어준다.

```
zip_source_path = './test_dataset.zip'
```

```
zip_target_path = './meta_data'
```

```
extract_zip_file = zipfile.ZipFile(zip_source_path)
```

```
extract_zip_file.extractall(zip_target_path)
```

```
extract_zip_file.close()
```

이미지를 모델의 입력 데이터 형태로 변환

- 이미지 데이터를 읽어온다.
- numpy array 형태로 변형시켜준 뒤, 정규화를 실시한다.

```
my_path='./meta_data/test_dataset/'
```

```
image_name = 'my_ant.png'
```

```
img = cv2.imread(my_path + image_name, cv2.IMREAD_GRAYSCALE)
```

```
ret, binary = cv2.threshold(img, 200, 255, cv2.THRESH_BINARY_INV)
```

```
my_image = np.asarray(cv2.resize(binary, dsize=(28, 28), interpolation=cv2.INTER_AREA))/255
```

```
#입력 이미지 확인
```

```
plt.axis('off')
```

```
plt.title('I draw')
```

```
plt.imshow(my_image, cmap='gray_r')
```


결과 확인

입력 데이터를 모델에 넣었을 때, 결과값이 하나의 숫자로 나오는 것이 아닌 총 10개의 카테고리에 대한 신뢰도 (confidence)를 전부 보여준다. 우리는 이 중에서 가장 큰 신뢰도를 보이는 레이블, 즉 모델이 가장 정답으로 확신하는 답을 print해서 보여주하고자 한다.

```
# 모델 입력 데이터로 사용하기 위해 reshape
input_image_data = (my_image.reshape(-1, 28, 28, 1))
```

```
# 추론
predict_result = model.predict(input_image_data)
```

```
# 결과 확인
# np.argmax는 배열 중 가장 큰 값의 인덱스를 반환해준다.
print(dataset[np.argmax(predict_result[0])])
```

다른 이미지들에 대해서도 확인해보자.

```
image_name = 'my_ant.png'
img = cv2.imread(my_path + image_name, cv2.IMREAD_GRAYSCALE)
ret, binary = cv2.threshold(img, 200, 255, cv2.THRESH_BINARY_INV)
my_image = np.asarray(cv2.resize(binary, dsize=(28, 28), interpolation=cv2.INTER_AREA)).reshape(-1, 784)
```

```
for i in range(1, len(dataset)):
    image_name = 'my_' + str(dataset[i]) + '.png'
    img = cv2.imread(my_path + image_name, cv2.IMREAD_GRAYSCALE)
    ret, binary = cv2.threshold(img, 200, 255, cv2.THRESH_BINARY_INV)
    my_image_2 = np.asarray(cv2.resize(binary, dsize=(28, 28),
    interpolation=cv2.INTER_AREA)).reshape(-1, 784)
    my_image = np.concatenate((my_image, my_image_2), axis = 0)
```

```
plot_samples(my_image, rows = 2, cols = 5, title = 'input')
```

```
# 모델 입력 데이터로 사용하기 위해 reshape
my_image_norm = (my_image/255).astype('float32')
input_image_data = (my_image_norm.reshape(my_image_norm.shape[0], 28, 28, 1))
```

```
# 추론
predict_result = model.predict(input_image_data)
```

```
#결과 확인
# np.argmax는 배열 중 가장 큰 값의 인덱스를 반환해준다.
print('WtWtWtLABELS :', dataset)
for i in range(len(predict_result)):
    print('prediction :', dataset[np.argmax(predict_result[i])], end='Wt')
    print('possibility :', [f'{result:.3f}' for result in predict_result[i].tolist()])
```

1_local_platform_image_classification.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_classification.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.

◆ 로컬 개발 워크플로우(workflow)

- ✓ 로컬 개발 워크플로우를 다음의 4단계로 분리한다.

1. 데이터셋 준비(Data Setup)

- ✓ 로컬 저장소에서 전처리 및 학습에 필요한 학습 데이터셋을 준비한다.

2. 데이터 전처리(Data Preprocessing)

- ✓ 데이터셋의 분석 및 정규화(Normalization) 등의 전처리를 수행한다.
- ✓ 데이터를 모델 학습에 사용할 수 있도록 가공한다.
- ✓ 추론과정에서 필요한 경우, 데이터 전처리에 사용된 객체를 meta_data 폴더 아래에 저장한다.

3. 학습 모델 훈련(Train Model)

- ✓ 데이터를 훈련에 사용할 수 있도록 가공한 뒤에 학습 모델을 구성한다.
- ✓ 학습 모델을 준비된 데이터셋으로 훈련시킨다.
- ✓ 정확도(Accuracy)나 손실(Loss) 등 학습 모델의 성능을 검증한다.
- ✓ 학습 모델의 성능 검증 후, 학습 모델을 배포한다.
- ✓ 배포할 학습 모델을 meta_data 폴더 아래에 저장한다.

4. 추론(Inference)

- ✓ 저장된 전처리 객체나 학습 모델 객체를 준비한다.
- ✓ 추론에 필요한 테스트 데이터셋을 준비한다.
- ✓ 배포된 학습 모델을 통해 테스트 데이터에 대한 추론을 진행한다.

파일명: image_classification_preprocess.py

'''

from image_classification_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_classification_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)

    # 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_classification_train.py

'''
from image_classification_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_classification_train_sub.py

# Imports
import os
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import utils
from tensorflow.keras import layers
from tensorflow.keras.models import load_model
import logging
import base64
import io
from PIL import Image

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

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

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)

def exec_train(tm):

    logging.info('[hunmin log] the start line of the function [exec_train]')

    logging.info('[hunmin log] tm.train_data_path : {}'.format(tm.train_data_path))

    # 저장 파일 확인
    list_files_directories(tm.train_data_path)

    #####
    ## 1. 데이터셋 준비(Data Setup)
    #####

    my_path = os.path.join(tm.train_data_path, 'dataset') + '/'

    # 카테고리
    dataset=['ant','apple', 'bus', 'butterfly', 'cup', 'envelope','fish', 'giraffe', 'lightbulb','pig']
    dataset_num= len(dataset) #10

    # 경로에 있는 numpy를 load하고 dataset_numpy list에 추가한다.
    dataset_numpy = []
    for i in range (dataset_num):
        ad = my_path + str(dataset[i]) + '.npy'
        dataset_numpy.append(np.load(ad))

```



```

logging.info('[hunmin log] : (image_number, image_size)')

for i in range (dataset_num):
    logging.info('[hunmin log] : {}'.format(dataset_numpy[i].shape))

np.set_printoptions(linewidth=116)
# dataset_numpy[5] 가 envelope numpy 이다.
logging.info('[hunmin log] envelope : ')
logging.info('{}'.format(dataset_numpy[5][0]))

#####
## 2. 데이터 전처리(Data Preprocessing)
#####

# 카테고리별로 같은 수의 이미지를 훈련시키기 위해 훈련시키고자 하는 이미지의 개수를
정해준다.
idx = 1000

# 데이터 정규화 (Normalization) & 데이터 합치기 & 레이블 생성
# X: 입력 이미지 배열 데이터
# Y: 정답 레이블 데이터
# 정규화 및 정답 레이블 생성
X = np.array([data_numpy[:idx, :]/255. for data_numpy in
dataset_numpy]).astype('float32')
X = X.reshape(-1, 28*28)

```

```

Y = np.array([i for i in range(10) for j in range(idx)]).astype('float32')

# 훈련 & 평가 데이터셋 생성
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2,random_state=0)

# 모델 훈련에 사용할 수 있는 형태로 변경
# X의 값을 [samples][pixels][width][height] 형태로 reshape한다.
X_train_cnn = X_train.reshape(X_train.shape[0], 28, 28 , 1).astype('float32')
X_test_cnn = X_test.reshape(X_test.shape[0], 28, 28 , 1).astype('float32')

# reshape된 결과 확인 및 원래 배열의 형태와 비교
logging.info('[hunmin log] X_train : {}'.format(X_train.shape))
logging.info('[hunmin log] X_train_cnn : {}'.format(X_train_cnn.shape))

# Y의 배열에 one-hot-encoding 진행
Y_train_cnn = utils.to_categorical(Y_train)
Y_test_cnn = utils.to_categorical(Y_test)
num_classes = Y_test_cnn.shape[1] # class는 총 10개이다.

# encoding된 결과 확인 및 원래 배열의 형태와 비교
logging.info('[hunmin log] Y_train : {}'.format(Y_train.shape))
logging.info('[hunmin log] Y_train_cnn : {}'.format(Y_train_cnn.shape))
logging.info('[hunmin log] class number : {}'.format(num_classes))

```

```
#####
## 3. 학습 모델 훈련(Train Model)
#####

# 모델 구축 (Build Model)
# 이미지 분류를 위해 아주 간단한 CNN 모델을 Keras를 이용하여 구축하고자 한다.

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

# 사용자 입력 파라미터
batch_size = int(tm.param_info['batch_size'])
epochs = int(tm.param_info['epoch'])

# gpu에 따른 batch_size 설정
batch_size = batch_size * len(gpus) if len(gpus) > 0 else batch_size

# 모델 학습 (Train Model)
history = model.fit(X_train_cnn, Y_train_cnn,
```

```

        batch_size=batch_size,
        epochs=epochs,
        validation_split=0.1,
        verbose=0,
        callbacks=[LossAndErrorPrintingCallback()]
    )

    # 모델 평가 (Evaluate Model)
    loss, acc = model.evaluate(X_test_cnn, Y_test_cnn, verbose=0,
    callbacks=[LossAndErrorPrintingCallback()])

    logging.info('[hunmin log] loss : {}'.format(loss))
    logging.info('[hunmin log] acc : {}'.format(acc))

    #####
    ## 플랫폼 시각화
    #####
    '''
    plot_metrics(tm, history, model, X_test_cnn, Y_test_cnn)
    '''

```

```
#####  
## 학습 모델 저장  
#####
```

```
logging.info('[hunmin log] tm.model_path : {}'.format(tm.model_path))  
model.save(os.path.join(tm.model_path, 'cnn_model.h5'))
```

```
# 저장 파일 확인  
list_files_directories(tm.model_path)
```

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

```
def exec_init_svc(im):
```

```
    logging.info('[hunmin log] im.model_path : {}'.format(im.model_path))
```

```
    # 저장 파일 확인  
    list_files_directories(im.model_path)
```

```
#####  
## 학습 모델 준비  
#####
```

```
# load the model
```

```

model = load_model(os.path.join(im.model_path, 'cnn_model.h5'))

return {'model' : model}

```

```

def exec_inference(df, params, batch_id):

```

```

#####
## 4. 추론(Inference)
#####

```

```

logging.info('[hunmin log] the start line of the function [exec_inference]')

```

```

## 학습 모델 준비
model = params['model']
logging.info('[hunmin log] model.summary() :')
model.summary(print_fn=logging.info)

```

```

dataset=['ant','apple', 'bus', 'butterfly', 'cup', 'envelope','fish', 'giraffe', 'lightbulb','pig']

```

```

# image preprocess
img_base64 = df.iloc[0, 0]
image_bytes = io.BytesIO(base64.b64decode(img_base64))
image = Image.open(image_bytes).convert('L')
image = image.resize((28, 28))
image = np.invert(image).astype('float32')/255.
image = image.reshape(-1, 28, 28 , 1)

```



```
# data predict
y_pred = model.predict(image)
y_pred_idx=np.argmax(y_pred, axis=1)
```

```
# inverse transform
result = {'inference' : dataset[y_pred_idx[0]]}
logging.info('[hunmin log] result : {}'.format(result))
```

```
return result
```

```
# 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))
```

```
#####
## exec_train(tm) 호출 함수
#####
```

```
# for epoch, loss
```

```

class LossAndErrorPrintingCallback(keras.callbacks.Callback):
    def on_epoch_end(self, batch, logs={}):
        #logging.info("For epoch {}, loss is {:.2f}, acc is {:.2f}.".format(batch, logs.get('loss'),
        logs.get('acc')))
        logging.info('[hunmin log] For epoch {}, loss is {:.2f}.'.format(batch+1, logs['loss']))

def model_build_and_compile(num_classes):
    #모델 구축
    model = keras.Sequential(
        [
            layers.Input(shape=(28,28,1)),
            layers.Conv2D(32, kernel_size=(3, 3), padding='same', activation="relu"),
            layers.Conv2D(64, kernel_size=(3, 3), padding='same', activation="relu"),
            layers.Dropout(0.25),
            layers.Conv2D(64, kernel_size=(3, 3), padding='same', activation="relu"),
            layers.MaxPooling2D(pool_size=(2, 2)),
            layers.Flatten(),
            layers.Dense(32, activation="relu"),
            layers.Dropout(0.25),
            layers.Dense(num_classes, activation="softmax")
        ]
    )
    logging.info('[hunmin log] model.summary() :')
    model.summary(print_fn=logging.info)

    # 모델 컴파일 (Compile Model)
    model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

    return model

```

```

# 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')
```

```
# PM 클래스: pm 객체
```

```
class PM:
```

```
    def __init__(self):
```

```
        self.source_path = './'
```

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

```
# TM 클래스: tm 객체
```

```
class TM:
```

```
    param_info = {}
```

```
    def __init__(self):
```

```
        self.train_data_path = './meta_data'
```

```
        self.model_path = './meta_data'
```

```
        self.param_info['batch_size'] = 10
```

```
        self.param_info['epoch'] = 20
```

```
# 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)
print('tm.param_info[W'batch_sizeW']: ', tm.param_info['batch_size'])
print('tm.param_info[W'epochW']: ', tm.param_info['epoch'])

# 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

# base64 encoded image
data =
[[iVBORw0KGgoAAAANSUUEUgAAABwAAAAcCAIAAAD9b0jDAAAAAXNSR0IArs4c6QAAAAR
nQU1BAACxjwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAACySURBVEhL7ZLRDoAgCEW
t//9nc8EIeepStvXQeWkyPEKw1FrLbFb+TuWXzichXXb4cAokJR0tH+JFK21G8V6CSqlApMxGelB
lsVBHeOOEzZYGdRzpussfL7elazHPa0wrx0GMdBsiuMbkdINyb5og0gRL3dTbcPtNOpYZveQ2p
A1n0hTakF5+hA9Lzd87pNFYLhkvsvT2IMhorndluxkRUuCYbzcp9ROi55+up8sLK1XKBj1wbx3D

```

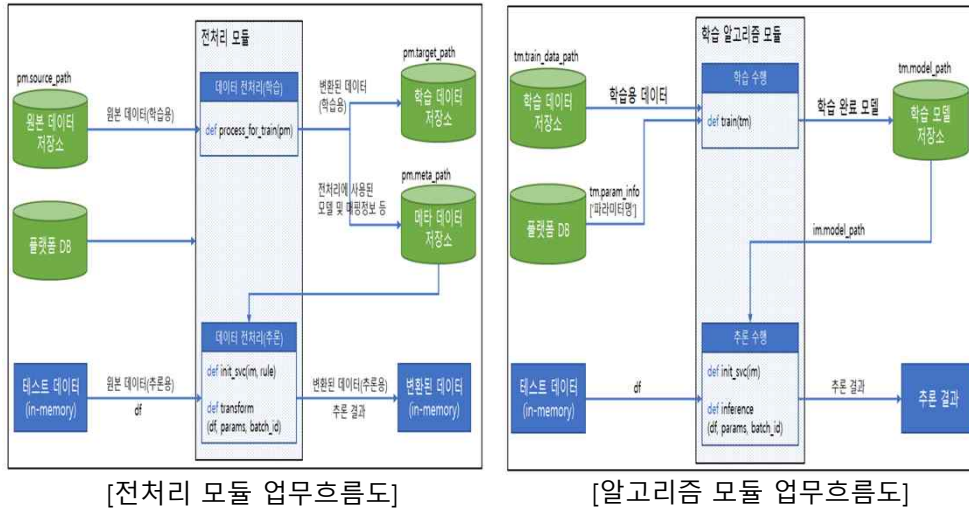
```
eIAOAAAAAEIFTkSuQmCC']]  
df = pd.DataFrame(data)  
print('df: ', df)  
print('df.dtypes:', df.dtypes)  
df.columns
```

```
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_classification_preprocess.py
 - ✓ from image_classification_preprocess_sub import exec_process
 - ✓ def process_for_train(pm):
 - ✓ def init_svc(im, rule):
 - ✓ def transform(df, params, batch_id):
- 파일명: image_classification_preprocess_sub.py
 - ✓ def exec_process(pm):
- 파일명: image_classification_train.py
 - ✓ from image_classification_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_classification_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



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}
```

```
=====
```



```
import logging
```

11 11 //

tm.model_path: 전처리 객체와 학습 모델 객체를 저장하는 경로

(3) 설명:

init_svc(im) 함수의 im.model_path를 통해 전처리 객체와 학습 모델 객체를 준비

함수의 정상적인 동작 체크를 위해 마지막 라인(the end line)에 로그 출력 수행

■■ ■■ ■■

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

|| || ||

(3) 설명:

리턴(return) 값을 inference(df, params, batch_id) 함수의 입력 params 변수로 전달

함수의 정상적인 동작 체크를 위해 마지막 라인(the end line)에 로그 출력 수행

■■ ■■ ■■

```
return {**params}
```

```

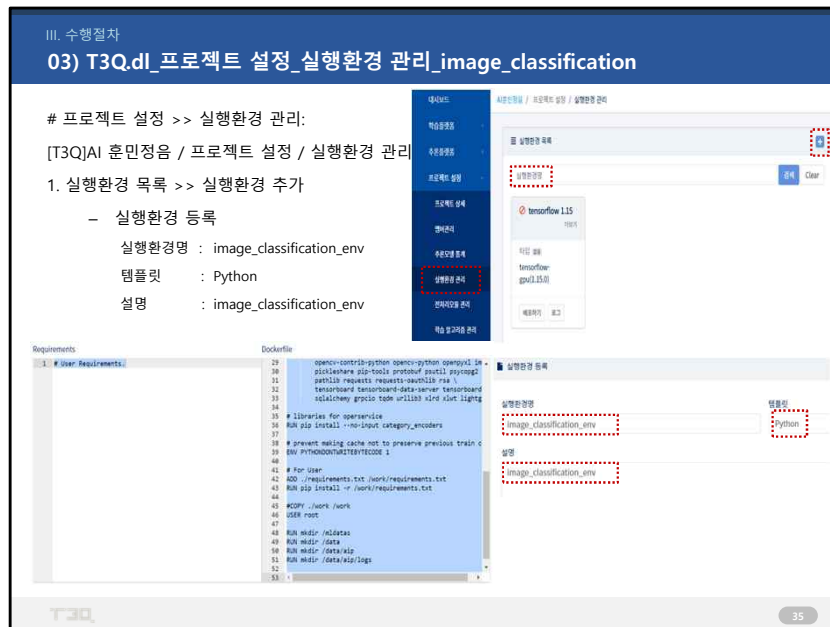
def inference(df, params, batch_id):
    """
    (1) 입력: df, params, batch_id
        # df: transform(df, params, batch_id)함수의 리턴(return) 값으로 전달된 df, 추론 입력
        데이터(dataframe 형태)
        # params: init_svc(im) 함수의 리턴(return) 값을 params 변수로 전달
        ## 학습 모델 객체 사용 예시      model=params['model']
        ## 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_classification : 해당 예제에서는 수행 절차 없음
- 02) T3Q.cep_데이터변환 파이프라인_image_classification : 해당 예제에서는 수행 절차 없음
- 03) T3Q.dl_프로젝트 설정_실행환경 관리_image_classification
- 04) T3Q.dl_프로젝트 설정_전처리 모듈 관리_image_classification
- 05) T3Q.dl_프로젝트 설정_학습 알고리즘 관리_image_classification
- 06) T3Q.dl_학습플랫폼_데이터셋 관리_image_classification
- 07) T3Q.dl_학습플랫폼_전처리 모델 설계_image_classification
- 08) T3Q.dl_학습플랫폼_전처리 모델 관리_image_classification
- 09) T3Q.dl_학습플랫폼_학습모델 설계_image_classification
- 10) T3Q.dl_학습플랫폼_학습모델 관리_image_classification
- 11) T3Q.dl_추론플랫폼_추론모델 관리_image_classification
- 12) T3Q.dl_추론플랫폼_추론API관리_image_classification
- 13) T3Q.cep_실시간 추론 파이프라인_image_classification



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 ₩
```

```
postgres ₩
```

```
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
```

```

=====
2) Dockerfile-계속
=====
# 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 pycopg2 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
=====
추가된 실행 환경 [저장] 하고, [배포하기] 시작 , 완료 후 [로그] 에서 확인

```

≡ 실행환경 목록

실행환경명

image_classification_env

더보기

타입 Python

image_classification_env

배포하기
로그

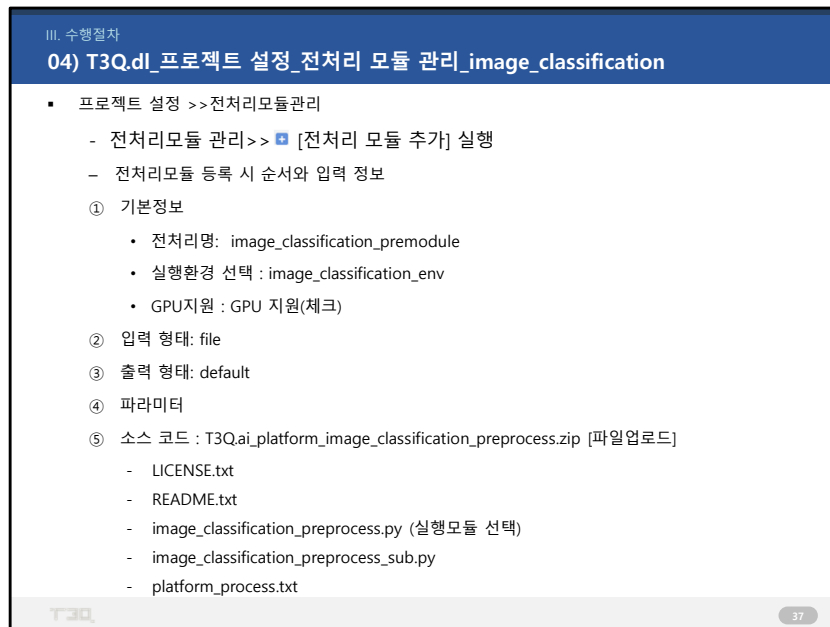
로그 확인

```

2022-08-24 04:22:42,795 [ INFO] root: {"status":"Preparing", "progressDetail": {}, "id":"d5f0eff44d91"}
2022-08-24 04:22:42,800 [ INFO] root: {"status":"Preparing", "progressDetail": {}, "id":"38482f47bc58"}
2022-08-24 04:22:42,800 [ INFO] root: {"status":"Preparing", "progressDetail": {}, "id":"daf57e1d9792"}
2022-08-24 04:22:42,800 [ INFO] root: {"status":"Preparing", "progressDetail": {}, "id":"53194dce1444"}
2022-08-24 04:22:42,800 [ INFO] root: {"status":"Preparing", "progressDetail": {}, "id":"ef8330bcc944"}
2022-08-24 04:22:42,800 [ INFO] root: {"status":"Preparing", "progressDetail": {}, "id":"964ee116c0c0"}
2022-08-24 04:22:42,801 [ INFO] root: {"status":"Preparing", "progressDetail": {}, "id":"7a694df0ad6c"}
2022-08-24 04:22:42,801 [ INFO] root: {"status":"Preparing", "progressDetail": {}, "id":"3fd9df553184"}

```

2022-09-23 10:12:13 ✕



- 전처리모듈 등록 시 순서와 입력 정보

① 기본정보

전처리명: image_classification_premodule

실행환경 선택 : image_classification_env

GPU지원 : GPU 지원(체크)

② 입력 형태: file

③ 출력 형태: default

전처리모듈 등록

기본 정보

image_classification_premodule

image_classification_env

ON GPU 지원

입력 형태

file

출력 형태

default

파라미터

이름을 입력해주세요

값을 입력해주세요

III. 수행절차

04) T3Q.ai 프로젝트 설정_전처리 모듈 관리_image_classification

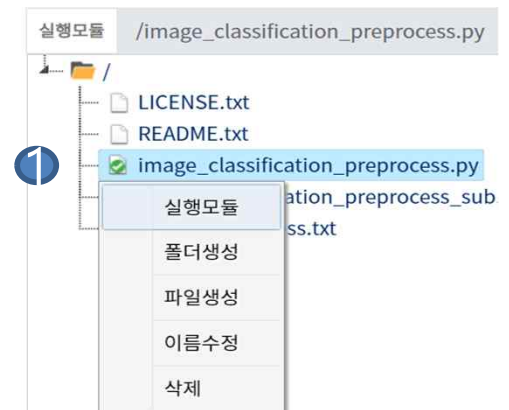
- 프로젝트 설정 >> 전처리모듈관리
 - 전처리모듈 관리 >> [전처리 모듈 추가] 실행
 - 전처리모듈 등록 시 순서와 입력 정보
 - 기본정보
 - 전처리명: image_classification_preprocess
 - 실행환경 선택 : image_classification_env
 - GPU자원 : GPU 지원(체크)
 - 입력 형태: file
 - 출력 형태: default
 - 파라미터
 - 소스 코드 : T3Q.ai_platform_image_classification_preprocess.zip [파일업로드]
 - LICENSE.txt
 - README.txt
 - image_classification_preprocess.py (실행모듈 선택)
 - image_classification_preprocess_sub.py
 - platform_process.txt

전처리모듈 등록 시 순서와 입력 정보

⑤ 소스 코드 : T3Q.ai_platform_image_classification_preprocess.zip

[파일업로드] 누름

- LICENSE.txt
- README.txt
- image_classification_preprocess.py (실행모듈 선택)
- image_classification_preprocess_sub.py
- platform_process.txt



2 소스 코드

실행모듈 /image_classification_pr 선택파일 /image_classification_preprocess.py

```

1 # 파일명 : image_classification_preprocess.py
2
3 from image_classification_preprocess_sub import exec_process
4 import logging
5
6
7 def process_for_train(pm):
8     exec_process(pm)
9     logging.info('[hunmin log] the end line of the function [process_for_train]')
10
11
12
13 def init_svc(im, rule):
14     return {}
15
16
17
18 def transform(df, params, batch_id):
19     logging.info('[hunmin log] df : {}'.format(df))
20     logging.info('[hunmin log] df.shape : {}'.format(df.shape))
21     logging.info('[hunmin log] type(df) : {}'.format(type(df)))
22     logging.info('[hunmin log] the end line of the function [transform]')
23     return df
24
25
  
```

파일업로드

05) T3Q.dl_프로젝트 설정_학습 알고리즘 관리_image_classification

- 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_classification_train
 - 설명 : image_classification_train
 - 카테고리 : Classification
 - 실행환경 : image_classification_env
 - GPU 지원 : 체크

② 공통 파라미터: 아래 2가지 항목 사용

- 학습수행횟수: 사용
 - 배치 사이즈: 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.ai 프로젝트 설정_학습 알고리즘 관리_image_classification

- 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_classification_train.zip

- LICENSE.txt
- README.txt
- image_classification_train.py
- image_classification_train_sub.py
- platform_process.txt

: 실행 모듈 설정 >> [저장]

- /image_classification_train.py 실행 모듈 지정
- [저장]을 누른다.

로 학습 알고리즘 관리

학습알고리즘명, 문제유형, 실행환경 검색

image_classification_train	새기기
image_classification_train	
Classifier	
image_classification_exe	
2022-08-24 13:24:53	

실행모듈 /image_classification_train.py

선택파일 /image_classification_train.py

```

1  #
2  #
3  from image_classification_train_sub import
4  import logging
5
6
7  def train(train):
8
9      exec_train(train)
10     logging.info([human log] the end)
11
12
13
14 def init_svc(in):
15
16     params = exec_init_svc(in)
17     logging.info([human log] the end)
18     return { "params" }
19
20
21
22 def inference(df, params, batch_id):
23
24     result = exec_inference(df, params,
25                             logging.info([human log] the end)
26     return { "result" }
27
28

```

실행모듈

III. 수행절차

06) T3Q.dl_학습플랫폼_데이터셋 관리_image_classification

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

데이터셋 등록

1

2 Drag & drop Files here

3

4

5

데이터셋 관리

번호	데이터셋명	등록자	등록일자	등록으로
1	image_classification_dataset	delkin	2022-09-24 13:28:28	

Showing 1 to 1 of 1 entries

Prev 1 Next

T3Q

41

07) T3Q.dl_학습플랫폼_전처리 모델 설계_image_classification

- 학습플랫폼 >> 전처리 모델 설계
- 전처리 모델 설계 절차
- ① 학습플랫폼 >> 데이터셋 관리
 - 데이터셋 명 : image_classification_dataset 선택
- ② image_classification_dataset 아래의
[전처리 모델 설계] 선택

데이터셋 관리

데이터셋 | 데이터셋 생성 | 데이터셋 삭제 | 데이터셋 검색

이름	데이터셋명	등록자	등록일자
1	image_classification_dataset	dekin	2022-08-24 13:28:28

Showing 1 to 1 of 1 entries

이전 1 다음

image_classification_dataset

File explorer / (1파일, 0 디렉토리)

dataset.zip 367.54 MB

dataset.zip 367.54 MB

목록으로 삭제 전처리 모델 설계 2

III. 수행절차

07) T3Q.dl_학습플랫폼_전처리 모델 설계_image_classification

■ 학습플랫폼 >> 전처리 모델 설계

② 전처리 모델 설계 절차

Step1. 기본 정보

- 모델명: image_classification_premodel

Step2. 데이터셋 등록:

- 참조데이터셋 : image_classification_dataset

Step3. ID/LABEL 지정

Step4. 전처리 규칙 정보

- [전처리 규칙 등록] 선택

③ 전처리 규칙 등록 시 절차

- 소스컬럼 : dataset(file) 선택

- 변환 함수 : image_classification_premodule
선택 후 [저장]



전처리 모델 설계

Step 1. 기본 정보
모델명 (필수) : image_classification_premodel

Step 2. 데이터셋 등록
참조 데이터셋 (필수) : image_classification_dataset

Step 3. ID/LABEL 지정
변환 함수 (필수) : image_classification_premodule

전처리 규칙 등록

사용자명	모델명	데이터셋명	파일	ID	LABEL
dataset	-	file	-	-	-

Showing 1 to 1 of 1 entries

Step 4. 전처리 규칙 정보
변환 함수 (필수) : image_classification_premodule

소스컬럼	변환규칙	변환규칙 상세정보	변환
No data available in table			

Showing 1 to 1 of 1 entries

Prev Next



■ 학습플랫폼 >> 전처리 모델 관리

■ 전처리 모델 설계 상세

Step1. 기본 정보

- 모델명: image_classification_premodel
- 참조데이터셋 : image_classification_dataset

Step2. 데이터셋 컬럼정보

Step3. 전처리 룰 정보

Step4. 전처리 실행정보

전처리 상세

- 전처리 상세 버튼 누름
- 진행상황 확인
- 0_image_classification_premodule - [로그] 아래 [보기] 누름

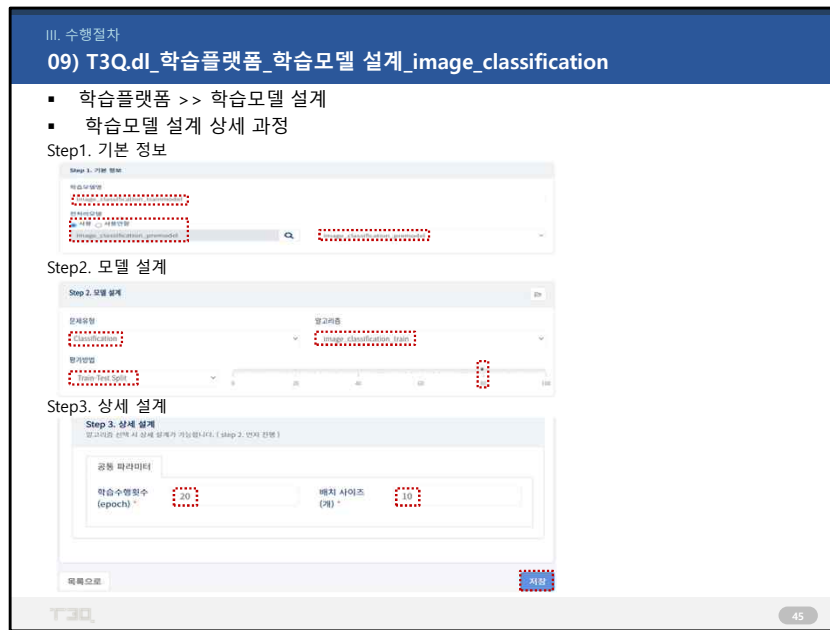
[학습 모델 설계] 선택하여 다음 단계 진행

로그 확인



마지막 로딩된 시간 : 2022-09-23 11:23:30

```
2022-08-24 04:29:37,366 [ INFO] root: ### preprocessing start ###
2022-08-24 04:29:37,366 [ INFO] root: params={'pre_dataset_id': 999, 'rule': {'source_column':
['dataset'], 'rule': 'preModel', 'rule_type': 'image_classification_premodule_v1', 'mod': 'U',
'param': {}, 'rule_no': '0', 'source_type': ['file'], 'module_info': '{"deploy_dt": "2022-08-24
13:22:32", "template": "Python", "version": "1.0", "status": "deployed", "image_name": 364,
"module_name": "image_classification_preprocess"}', 'output_type': ['default']}, 'do_fit':
True, 'test_no': None, 'test_dataset_path': None, 'log_path': '/data/aip/logs'}
2022-08-24 04:29:37,420 [ WARN] root: datasource_repo_id : 159, datasource_repo_obj :
<DataSourceRepo 159>, repo_type : path
2022-08-24 04:29:37,452 [ INFO] root:
module_path=/data/aip/logs/t3qai/premodule/premodule_733/1
2022-08-24 04:29:37,458 [ INFO] root: dp_module=<module 'image_classification_preprocess'
from
'/data/aip/logs/t3qai/premodule/premodule_733/1/image_classification_preprocess.py'>
2022-08-24 04:29:37,458 [ INFO] root: [hunmin log] the start line of the function
[exec_process]
```

학습플랫폼 >> 학습모델 설계

1. AI 훈민정음 >> 학습플랫폼 >> 학습모델 설계 상세 과정

1) Step 1. 기본 정보

학습모델명

image_classification_trainmodel

전처리모델

[사용] 체크

image_classification_premodel

image_classification_premodel

2) Step 2. 모델 설계

문제유형 Classification

알고리즘 image_classification_train

평가방법

Train-Test Split : 80

3) Step 3. 상세 설계

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

(1) 공통 파라미터

학습수행횟수 (epoch) : *20

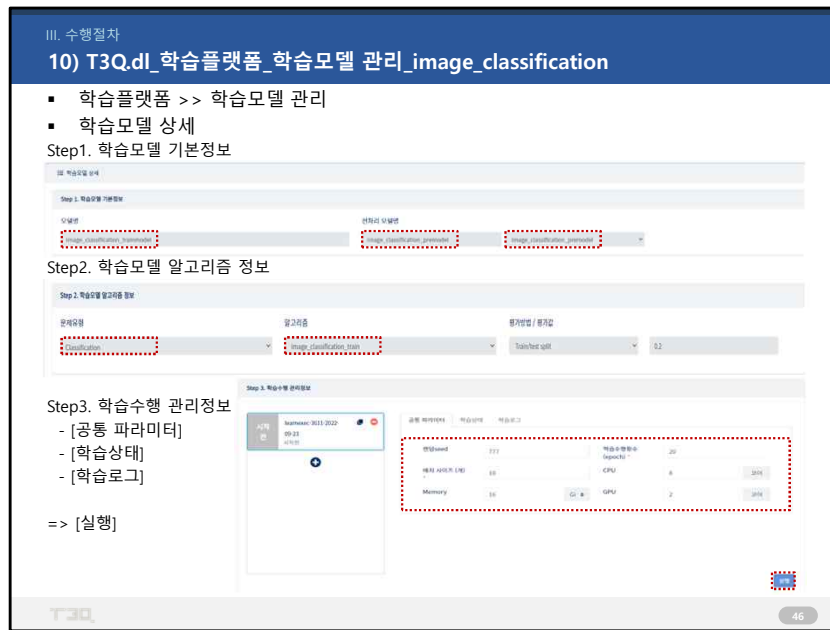
배치 사이즈 (개) : *10

4) [저장] 누름

학습 모델 관리								
카테고리 선택		알고리즘 선택		학습명 검색		등록자	조회	Clear
번호	카테고리	알고리즘	학습명	등록자	실행정보			학습상태
					수행시작일시	수행종료일시	결과(Accuracy)	
1	Classification	image_classification_train	image_classification_trainmodel	admin	-	-	-	시작전

Showing 1 to 1 of 1 entries

Prev 1 Next



학습플랫폼 >> 학습모델 관리

1. 학습플랫폼 >> 학습모델 관리

≡ 학습 모델 관리

카테고리 선택 알고리즘 선택 학습명 검색 등록자 조회 Clear

번호	카테고리	알고리즘	학습명	등록자	실행정보			학습상태
					수행시작일시	수행종료일시	결과(Accuracy)	
1	Classification	image_classification_train	image_classification_trainmodel	admin	-	-	-	시작전

Showing 1 to 1 of 1 entries

Prev 1 Next

2 학습모델 상세

1) Step 1. 학습모델 기본정보

모델명 image_classification_trainmodel

전처리 모델명 image_classification_premodel image_classification_premodel

2) Step 2. 학습모델 알고리즘 정보

문제유형 Classification

알고리즘 image_classification_train

평가방법 / 평가값 Train/test split 0.2

3) Step 3. 학습수행 관리정보

(1) 공통 파라미터

랜덤seed 777

학습수행횟수 (epoch) * 20

배치 사이즈 (개) * 10

CPU 8 코어

Memory 16 Gi

GPU 2 코어

(2) 학습상태

학습상태 시작전 [- ~ -]

Accuracy

Loss

Confusion Matrix

Precision/Recall/F1-score

[실행] 버튼 누름

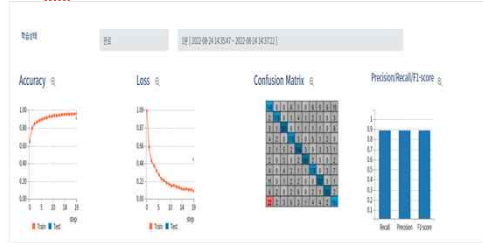
III. 수행절차

10) T3Q.dl_프로젝트 설정_학습모델 관리_image_classification

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

- [학습상태]

관리 화면 : 실행 : 정지



- [학습로그]

공통 파라미터 : 학습상태 : 학습로그

```
2022-08-24 05:35:47.665 [INFO] root: ### Train params ###
2022-08-24 05:35:47.666 [INFO] root: {learn_id: 3057,
'learn_e_id': 3611, 'workspace_id': 't3qal', 'user_id': None,
'epoch': None, 'comm_method': 'db', 'log_path':
'/data/aipl/flags', 'cpu': '8', 'memory': '16G', 'gpu': '2',
'train_profiling': True, 'ds_id': 1}
2022-08-24 05:35:47.666 [INFO] root: ### end ###
2022-08-24 05:35:47.713 [WARN] root: datasource_repo_id: 159,
datasource_repo_obj: <DataSourceRepo 159>, repo_type: path
2022-08-24 05:35:47.768 [INFO] root:
algo_path=/data/aipl/flags/t3qal/models/algo_577/1
2022-08-24 05:35:49.541 [DEBUG] tensorflow: Falling back to
TensorFlow client; we recommended you install the Cloud TPU
client directly with pip install cloud-tpu-client.
2022-08-24 05:35:50.622 [INFO] root: (thummin log) tensorflow
```

실행 : 모델 배포

- 모델 배포 : [모델 배포] 버튼 누르고 [배포] 누름

학습모델 배포

모델명을 입력하고 추론모델로 배포가 가능합니다.(50자 이내)

image_classification_trainmodel-2022-08-24

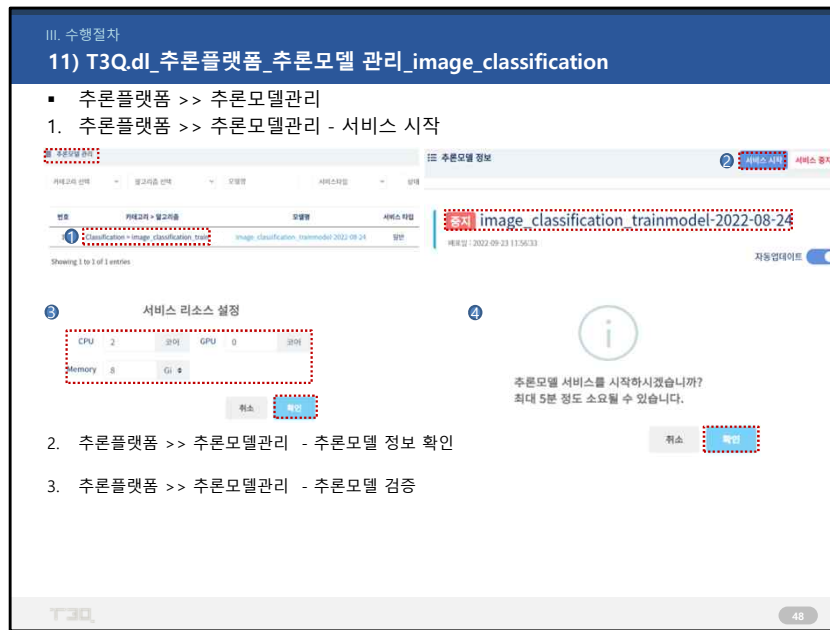
배포

모델명	모델종류	모델상태	모델이동	모델삭제	모델복제
image_classification_trainmodel-2022-08-24	Classification	image_classification_trainmodel-2022-08-24	실행	배포	복제

Showing 1 to 1 of 1 entries

T3Q

47



추론플랫폼 >> 추론모델관리

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



운영중 image_classification_trainmodel-2022-08-24



3. 추론플랫폼 >> 추론모델관리 - 추론모델 검증

[추론모델테스트]-[요청] 에 아래의 값을 넣고, [테스트] 눌러 수행

요청 : 입력 예시 : ["/data/aip/file_group/pm/pm_334/ds_441/image/1/1230.png"]

요청 :

```
[["ivBORw0KGgoAAAANSUheUgAAABWAAAAcCAIAAAD9b0jDAAAAAXNSR0IArs4c6QAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAACSSURBVEhl7Y6LCoAgDEX9/5823V0653CSBgkdCPa6x0J8gV+6nw9LQ2g8hjRdPIpZkPJMgN0M5V6mai+nk2TfnUUBGqlcuMh7FaxSFADtAHWmlo0U0P3l2x/oNn+dgllZ7gXmXE1saYEMvJW1Qs0daYJUDl861MqXJkjoPMwVMSV1OV26DnRg9R9Nful+TpHGeAEq4up46Jf19gAAAABJRu5ErkJggg=="]]
```

응답 : "{W"inferenceW":W"envelopeW"}Wn"



III. 수행절차

12) T3Q.dl_추론플랫폼_추론API관리_image_classification

- 추론플랫폼 >> 추론API관리

- 추론플랫폼 >> 추론API관리 - 신규 등록
- 추론플랫폼 >> 추론API관리 - 추론 API 상세

- 추론플랫폼 >> 추론API관리
- 추론플랫폼 >> 추론API관리 - 신규 등록
 - 추론플랫폼 >> 추론API관리 - 추론 API 상세

추론 API 조회

번호	사용 여부	API명	등록일	등록자	추론모델			
					카테고리	알고리즘	모델명	배포일자
1	운영중	image_classification_api	2022-09-23 14:16:11	hyin	Classification	image_classification_train	image_classification_trainmodel-2022-08-24	2022-09-23 10:11:54

Showing 1 to 1 of 1 entries

Prev 1 Next

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

2 추론 API 상세

테스트

API URL: http://idro3vub.dl.nhnes.net/model/api/52ee8/inference

METHOD: POST

요청: {"data": "[테스트 데이터 값 입력="]}

API 호출 초기화

응답: {"inference": {"envelope": [결과]}}

III. 수행절차

13) T3Q.cep_실시간 추론 파이프라인_image_classification

- T3Q.cep >> 실시간 추론
 - 실시간 추론 파이프라인 등록
 - 실시간 추론 파이프라인 구성 정보
 - 파이프라인 기본정보
 - Image to API(FileUpload) 선택
 - 파이프라인 이름 : image_classification_inference
 - 파이프라인 설명 : image_classification_inference
 - 기본 설정
 - DBCPConnectionPool Password: postgres
 - 사용자 설정
 - Image_API_FileUpload_API_URL : /model/api/52ee8/inference
 - Image_API_FileUpload_SourcePath : /AI_HUNMIN/image_classification/inference
 - Image_API_FileUpload_Topic : image_classification_topic
 - 파이프라인 시작 : [image_classification_inference] 우측 상단의 기능 버튼 클릭 후 [시작] 선택
 - 원본 데이터 업로드
 - T3Q.ai>>Tools>>FileViewer를 이용하여
Image_API_FileUpload_SourcePath 에서 설정한 경로
(/AI_HUNMIN/image_classification/inference)에
로컬 폴더 inference_dataset 폴더의
my_ant.png, my_apple.png, my_bus.png... 파일 업로드
 - 데이터 적재 확인
 - pgadmin 도구 이용 inference_result 테이블 조회

(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/52ee8/inference%'
order by start_time desc
```