모델 개발 매뉴얼

Team Archits

2023.10.20

Contents

I. 학습 및 배포를 위한 SW 및 HW

구분	상세	비고
OS	Windows 11 / Linux	
언어	Python 3.9	
Framework	Tensorflow	Window = 2.10.0 Linux = 2.13.0
그래픽카드	NVIDIA GeForce RTX 3070	
CUDA	cuDNN 11.2	
그 외 Python 패키지	pandas numpy scipy sklearn tqdm pyarrow plotly Pytictoc	

II. 파일 저장 구조

데이터 셋

- ✓ Apache Parquet

※ 효율적인 데이터 스토리지와 검색을 지원하도록 설계되었으며, 컬럼 중심의 오픈 소스 데이터 파일 형식

학습 모델

- ✓ HDF

III. 모델 실행 방법 ▶ Library 파트

코드

```
import os
import pandas as pd
import numpy as np

from IPython.display import display, clear_output

## Parquet 파일 로드
try:
    import pyarrow.parquet as pq
except:
    os.system("pip install pyarrow")
    import pyarrow.parquet as pq


## Tensorflow 로드
try:
    import tensorflow as tf
    print("Tensorflow Version: {}".format(tf.__version__))

    if tf.__version__ != '2.10.0':
        os.system("pip install tensorflow==2.10.0")
        import tensorflow as tf
        print("Tensorflow Version: {}".format(tf.__version__))
except:
    os.system("pip install tensorflow==2.10.0")
    import tensorflow as tf
    print("Tensorflow Version: {}".format(tf.__version__))
    print("Please install tensorflow")


## Tensorflow model 로드
from tensorflow.keras.models import load_model

## Evaluation
from sklearn.metrics import precision_score, recall_score, f1_score

try:
    from pytictoc import TicToc
except:
    os.system("pip install pytictoc")
    from pytictoc import TicToc
```

코드 수행 결과

```
Tensorflow Version: 2.10.0
```

III. 모델 실행 방법 ▶ 모델 Load

코드

Model Load

```
[ ]: ## Model Load
     MIMO_transformer = load_model(os.path.join(base_path, "Model", "MIMO_Transformer_ts_class_FIN.h5"), compile = False)

     print("Model was loaded successfully")

[ ]: MIMO_transformer.summary()
```

코드 수행 결과

Model Load

```
[5]: ## Model Load
     MIMO_transformer = load_model(os.path.join(base_path, "Model", "MIMO_Transformer_ts_class_FIN.h5"), compile = False)

     print("Model was loaded successfully")

     Model was loaded successfully

[6]: MIMO_transformer.summary()

     Model: "model"
     __________________________________________________________________________________________________
      Layer (type)                    Output Shape         Param #     Connected to
     ==================================================================================================
      input_1 (InputLayer)            [(None, 70000, 1)]   0           []

      conv1d (Conv1D)                 (None, 6901, 128)    128128      ['input_1[0][0]']

      conv1d_27 (Conv1D)              (None, 6901, 128)    128128      ['input_1[0][0]']

      global_average_pooling1d (Glob  (None, 6901, 1)      0           ['conv1d[0][0]']
      alAveragePooling1D)

      global_average_pooling1d_4 (Gl  (None, 6901, 1)      0           ['conv1d_27[0][0]']
      obalAveragePooling1D)
```

III. 모델 실행 방법 ▶ 데이터 Load

코드

Data Set Load

```
[ ]: with tf.device("/CPU:0"):

         df_import = pq.read_table( os.path.join(base_path, "Data_Set", "Train_Data_Time_Norm_with_Label_70000_pt_severity_mod.parquet")).to_pandas()

         df_data_all = df_import.loc[:, ~df_import.columns.isin(['LABEL','SEVERITY', 'ANOMAL_POS_METER'])].copy().reset_index(drop=True)
         df_y_all = df_import.loc[:, df_import.columns.isin( ['LABEL', 'SEVERITY', 'ANOMAL_POS_METER']) ].copy().reset_index(drop=True)

[ ]: with tf.device("/CPU:0"):
         ## X Train / Test :: Category
         X_all_1 = df_data_all.loc[:, ~df_data_all.columns.isin(['TOT_TIME_SEC','DISTANCE_METER', 'ANOM_TIME_SEC'])].copy().values
         X_all_2 = df_data_all.loc[:, df_data_all.columns.isin(['TOT_TIME_SEC','DISTANCE_METER', 'ANOM_TIME_SEC'])].copy().values

         # Y Category Label
         y_all_1 = df_y_all.LABEL.values

         # Y Severity Label
         y_all_2 = df_y_all.SEVERITY.values

         # Y anomal position (label)
         y_all_3 = df_y_all.ANOMAL_POS_METER.values

[ ]: print("Category Class :: Shape of X train and test: {} / {}".format(X_all_1.shape, y_all_1.shape))
     print("Severity Class :: Shape of X train and test: {} / {}".format(X_all_1.shape, y_all_2.shape))
     print("Anom. Pos Reg  :: Shape of X train and test: {} / {}".format(X_all_2.shape, y_all_3.shape))
```

코드 수행 결과

```
[10]: print("Category Class :: Shape of X train and test: {} / {}".format(X_all_1.shape, y_all_1.shape))
      print("Severity Class :: Shape of X train and test: {} / {}".format(X_all_1.shape, y_all_2.shape))
      print("Anom. Pos Reg  :: Shape of X train and test: {} / {}".format(X_all_2.shape, y_all_3.shape))

      Category Class :: Shape of X train and test: (3285, 70000) / (3285,)
      Severity Class :: Shape of X train and test: (3285, 70000) / (3285,)
      Anom. Pos Reg  :: Shape of X train and test: (3285, 3) / (3285,)
```

III. 모델 실행 방법 ▶ 모델 평가

코드

Evaluation

```
t = TicToc()

df_res_all = pd.DataFrame([])
elap_times = []

y_pred_cat = []
y_pred_sev = []
y_pred_pos = []

for i in range(len(X_all_1)):
    clear_output(wait=True)
    t.tic()

    pred_cat, pred_sev, pred_pos = MIMO_transformer.predict(
                                                [ np.expand_dims(np.expand_dims(X_all_1[i], axis = 1), axis=0), np.expand_dims(X_all_2[i], axis=0)]
                                            )

    y_pred_cat.append(pred_cat.argmax())
    y_pred_sev.append(pred_sev.argmax())
    y_pred_pos.append(pred_pos)

    t.toc()

    elap_times.append(t.tocvalue())
print("Average Elapsed Time [s]: {} s".format( np.mean(elap_times) ) )
```

코드 수행 결과

```
print("Average Elapsed Time [s]: {} s".format( np.mean(elap_times) ) )
Average Elapsed Time [s]: 0.08376279232876796 s
```

III. 모델 실행 방법 ▶ 모델 평가

코드

```
[ ]: ## Category
p = precision_score(y_all_1, y_pred_cat, average='weighted')
r = recall_score(y_all_1, y_pred_cat, average = 'weighted')
f1 = f1_score(y_all_1, y_pred_cat, average = 'weighted')

print("Category Class")
print("Precision : {} % (정확도)".format(round(p*100, 2) ) )
print("Recall    : {} % (재현율)".format(round(r*100, 2) ) )
print("F1 Score  : {} ".format(round(f1, 3) ) )

p = precision_score(y_all_2, y_pred_sev, average='weighted')
r = recall_score(y_all_2, y_pred_sev, average = 'weighted')
f1 = f1_score(y_all_2, y_pred_sev, average = 'weighted')

print("================================================================")
print("Severity Class")
print("Precision : {} % (정확도)".format(round(p*100, 2) ) )
print("Recall    : {} % (재현율)".format(round(r*100, 2) ) )
print("F1 Score  : {} ".format(round(f1, 3) ) )

MAPE = np.mean(abs(y_all_3 - y_pred_pos)/y_all_3)
MAE = np.mean(abs(y_all_3 - y_pred_pos))
RMSE = np.sqrt(np.sum(np.power(y_all_3 - y_pred_pos, 2))/(len(y_pred_pos)-1))

print("================================================================")
print("Position [m]")
print("MAPE  : {} % (Mean Absolute Percentage Error)".format(round(MAPE*100, 2) ) )
print("MAE   : {} [m] (Mean Abs Error)".format(round(MAE, 2) ) )
print("RMSE  : {} [m] (Mean Squared Error)".format(round(RMSE, 2) ) )
```

코드 수행 결과

```
Category Class
Precision : 99.91 % (정확도)
Recall    : 99.91 % (재현율)
F1 Score  : 0.999
================================================================
Severity Class
Precision : 95.94 % (정확도)
Recall    : 95.56 % (재현율)
F1 Score  : 0.955
[redacted]: RuntimeWarning: invalid value encountered in divide
  MAPE = np.mean(abs(y_all_3 - y_pred_pos)/y_all_3)
================================================================
Position [m]
MAPE  : nan % (Mean Absolute Percentage Error)
MAE   : 32.47 [m] (Mean Abs Error)
RMSE  : 2402.88 [m] (Mean Squared Error)
```

Fin