

LookAlike trainer2 deployment document

1. Codes structure

name	description
Train_tf2.py	Main training code for training & testing LookAlike model
Model_tf2.py	Code for defining DIN model in tf2
Utils_tf2.py	Utility functions mainly to transfer input data format from tf1 to tf2
__init__.py	Required to be defined as a python package

Train_tf2.py is the program to start with training and testing

2. Inputs

The format of input data file is pickle.

There are 2 pickles with exactly the same format as tf1

- training and testing data file, such as: ad_dataset_lookalike_xxxxxx.pkl
- testing label file, such as: label_lookalike_xxxxxx.pkl

These 2 files have the same format as tf1 trainer as shown below.



Result:

```
> result = {list} <class 'list':> <Too big to print. Len: 6185466>  
▼ 0000000 = {tuple} <class 'tuple':> (1, [4, 6, 10, 11, 12, 4, 6, 10, 11, 4, 6, 10, 11, 15, 4, 10, 11,  
    ► 0 = {int64} 1  
    ► 1 = {list} <class 'list':> [4, 6, 10, 11, 12, 4, 6, 10, 11, 4, 6, 10, 11, 15, 4, 10, 11, 4, 6, 10, 11,  
      2 = {int} 11  
      3 = {int} 0  
      __len__ = {int} 4  
► 0000001 = {tuple} <class 'tuple':> (1, [11, 6, 10, 11, 14, 10, 11, 4, 10, 4, 10, 11, 14, 15, 4, 6,  
► 0000002 = {tuple} <class 'tuple':> (1, [11, 6, 10, 11, 14, 10, 11, 4, 10, 4, 10, 11, 14, 15, 4, 6,  
► 0000003 = {tuple} <class 'tuple':> (1, [10, 11, 4, 10, 4, 10, 11, 14, 15, 4, 6, 10, 11, 4, 6, 11,  
► 0000004 = {tuple} <class 'tuple':> (1, [4, 6, 10, 11, 4, 6, 11, 15, 6, 10, 11, 14, 4, 6, 10, 11, 1!  
► 0000005 = {tuple} <class 'tuple':> (1, [4, 6, 10, 11, 4, 6, 11, 15, 6, 10, 11, 14, 4, 6, 10, 11, 1!  
► 0000006 = {tuple} <class 'tuple':> (1, [4, 6, 10, 11, 4, 6, 11, 15, 6, 10, 11, 14, 4, 6, 10, 11, 1!  
► 0000007 = {tuple} <class 'tuple':> (1, [4, 10, 11, 14, 4, 6, 11, 15, 10, 11, 14, 11, 10, 11, 10,  
► 0000008 = {tuple} <class 'tuple':> (1, [4, 10, 11, 14, 4, 6, 11, 15, 10, 11, 14, 11, 10, 11, 10,  
► 0000009 = {tuple} <class 'tuple':> (1, [10, 11, 12, 4, 10, 12, 6, 11], 10, 0)  
► 0000010 = {tuple} <class 'tuple':> (2, [4, 11, 6, 14, 11, 10, 11, 10, 11, 7, 10, 11, 10, 11, 11,  
► 0000011 = {tuple} <class 'tuple':> (2, [6, 14, 11, 10, 11, 10, 11, 7, 10, 11, 10, 11, 11, 10, 7,  
► 0000012 = {tuple} <class 'tuple':> (2, [10, 11, 7, 10, 11, 10, 11, 11, 10, 7, 10, 11, 6, 1, 4, 10,  
► 0000013 = {tuple} <class 'tuple':> (2, [7, 10, 11, 10, 11, 11, 10, 7, 10, 11, 6, 1, 4, 10, 11, 1,  
► 0000014 = {tuple} <class 'tuple':> (2, [6, 1, 4, 10, 11, 1, 10, 11, 1, 10, 11, 1, 7, 10, 11, 1, 4, 7,  
► 0000015 = {tuple} <class 'tuple':> (2, [6, 1, 4, 10, 11, 1, 10, 11, 1, 10, 11, 1, 7, 10, 11, 1, 4, 7,  
► 0000016 = {tuple} <class 'tuple':> (2, [1, 4, 7, 10, 11, 6, 14, 1, 10, 1, 10, 7, 11], 1, 0)  
► 0000017 = {tuple} <class 'tuple':> (2, [1, 4, 7, 10, 11, 6, 14, 1, 10, 1, 10, 7, 11], 10, 0)  
► 0000018 = {tuple} <class 'tuple':> (2, [1, 10, 1, 10, 7, 11], 6, 1)  
► 0000019 = {tuple} <class 'tuple':> (5, [7, 11, 7, 11, 7, 11, 11, 11, 7, 11, 7, 11, 7, 11, 4, 12, 4,
```

test_set_tf1

Use Ctrl+Shift+Enter to add to Watches

Result:

```
▼ result = {list} <class 'list'>: <Too big to print. Len: 936289>
  ▼ 000000 = {tuple} <class 'tuple'>: (1, [4, 10, 10, 11, 4, 6, 10, 11, 12, 4, 6, 10, 11, 4, 6, 10, 11,
    ► 0 = {int64} 1
    ► 1 = {list} <class 'list'>: [4, 10, 10, 11, 4, 6, 10, 11, 12, 4, 6, 10, 11, 4, 6, 10, 11, 15, 4, 10, 1
    ► 2 = {tuple} <class 'tuple'>: (4, 4)
    __len__ = {int} 3
  ► 000001 = {tuple} <class 'tuple'>: (1, [4, 10, 10, 11, 4, 6, 10, 11, 12, 4, 6, 10, 11, 4, 6, 10, 11,
  ► 000002 = {tuple} <class 'tuple'>: (1, [4, 10, 10, 11, 4, 6, 10, 11, 12, 4, 6, 10, 11, 4, 6, 10, 11,
  ► 000003 = {tuple} <class 'tuple'>: (2, [4, 4, 11, 6, 14, 11, 10, 11, 10, 11, 7, 10, 11, 10, 11, 11,
  ► 000004 = {tuple} <class 'tuple'>: (2, [4, 4, 11, 6, 14, 11, 10, 11, 10, 11, 7, 10, 11, 10, 11, 11,
  ► 000005 = {tuple} <class 'tuple'>: (5, [7, 11, 7, 11, 11, 7, 11, 7, 11, 7, 11, 11, 11, 7, 11, 7, 11])
  ► 000006 = {tuple} <class 'tuple'>: (5, [7, 11, 7, 11, 11, 7, 11, 7, 11, 7, 11, 11, 11, 7, 11, 7, 11])
  ► 000007 = {tuple} <class 'tuple'>: (8, [7, 7, 12, 7, 12, 7, 11, 12, 11, 7, 12, 11, 12, 11, 11, 12,
  ► 000008 = {tuple} <class 'tuple'>: (12, [3, 4, 11, 14, 11, 14, 11, 11, 10, 11, 3, 7, 11, 12, 11, 4,
  ► 000009 = {tuple} <class 'tuple'>: (14, [7, 12, 7, 11, 12, 7, 11, 7, 11, 4, 7, 11, 11, 4, 7, 10, 4, 7
  ► 000010 = {tuple} <class 'tuple'>: (14, [7, 12, 7, 11, 12, 7, 11, 7, 11, 4, 7, 11, 11, 4, 7, 10, 4, 7
  ► 000011 = {tuple} <class 'tuple'>: (15, [6, 10, 11, 6, 10, 11, 6, 10, 10, 6, 10, 11, 2, 4, 6, 10, 1
  ► 000012 = {tuple} <class 'tuple'>: (16, [11, 11, 10, 11, 1, 11, 11, 1, 6, 11, 12, 11, 11, 10, 11,
  ► 000013 = {tuple} <class 'tuple'>: (16, [11, 11, 10, 11, 1, 11, 11, 1, 6, 11, 12, 11, 11, 10, 11,
  ► 000014 = {tuple} <class 'tuple'>: (17, [11, 7, 7, 11, 4, 4, 10, 7, 4, 7, 10, 4, 4, 11], (11, 11))
  ► 000015 = {tuple} <class 'tuple'>: (19, [10, 10, 10, 10, 10, 11, 4, 10, 14, 10, 11, 14, 10, 4, 10
  ► 000016 = {tuple} <class 'tuple'>: (19, [10, 10, 10, 10, 10, 11, 4, 10, 14, 10, 11, 14, 10, 4, 10
  ► 000017 = {tuple} <class 'tuple'>: (24, [1, 4, 7, 12, 1, 5, 12, 7, 1, 12, 1, 7, 12, 6, 1, 3, 4, 7, 10,
  ► 000018 = {tuple} <class 'tuple'>: (27, [4, 4, 11, 4, 4, 11, 11, 4, 6, 7, 11, 4, 11, 11, 4, 11, 4], (
  ► 000019 = {tuple} <class 'tuple'>: (27, [4, 4, 11, 4, 4, 11, 11, 4, 6, 7, 11, 4, 11, 11, 4, 11, 4], (
  ► 000020 = {tuple} <class 'tuple'>: (27, [4, 4, 11, 4, 4, 11, 11, 4, 6, 7, 11, 4, 11, 11, 4, 11, 4], (
```

```
test_lb|
Use Ctrl+Shift+Enter to add to Watches

Result:
▼ result = {list} <class 'list'>: <Too big to print. Len: 936289>
  000000 = {int} 0
  000001 = {int} 0
  000002 = {int} 0
  000003 = {int} 0
  000004 = {int} 0
  000005 = {int} 0
  000006 = {int} 0
  000007 = {int} 0
  000008 = {int} 0
  000009 = {int} 0
  000010 = {int} 0
  000011 = {int} 0
  000012 = {int} 0
  000013 = {int} 0
  000014 = {int} 1
  000015 = {int} 0
  000016 = {int} 1
  000017 = {int} 0
  000018 = {int} 0
  000019 = {int} 0
  000020 = {int} 0
  000021 = {int} 0
  000022 = {int} 0
  000023 = {int} 1
  000024 = {int} 1
```

The function named “tf1_to_tf2_data_conversion” defined in utils_tf2.py converts the above mentions tf1 data format to tf2 data format and then saved in a pickle file named “train_test_lookalike.pkl”.

If “train_test_lookalike.pkl” already exists, train_tf2.py will skip the conversion step and read tf2 input data from “train_test_lookalike.pkl” directly.

Input data for trainer2 are stored in 2 tuples of variables, i.e. (train_X, train_y) and (test_x, test_y), they will be feed into function model.fit to train the tf2 DIN model

3. output

```
scores = model.predict(test_X, batch_size=batch_size)
```

will return scores for all testing samples as shown below

scores

Use Ctrl+Shift+Enter to add to Watches

Result:

▼ result = {ndarray} [[0.43866727]\n [0.19806415]\n [0.3554388]\n ... \n [0.0...View as Array

▶ __internals__ = {dict} {'T': array([[0.43866727, 0.19806415, 0.3554388 , ..., 0.041... View

▶ min = {float32} 0.0247937

▶ max = {float32} 0.856926

▶ shape = {tuple} <class 'tuple'>: (936168, 1)

▶ dtype = {dtype} float32

▶ size = {int} 936168

▼ [0:936168] = {list} <pydevd_plugins.extensions.types.pydevd_plugin_numpy_types.N

▶ 000000 = {ndarray} [0.43866727] ...View as Array

▶ 000001 = {ndarray} [0.19806415] ...View as Array

▶ 000002 = {ndarray} [0.3554388] ...View as Array

▶ 000003 = {ndarray} [0.42551658] ...View as Array

▶ 000004 = {ndarray} [0.5490022] ...View as Array

▶ 000005 = {ndarray} [0.5972728] ...View as Array

▶ 000006 = {ndarray} [0.4002043] ...View as Array

▶ 000007 = {ndarray} [0.18481195] ...View as Array

▶ 000008 = {ndarray} [0.34747666] ...View as Array

▶ 000009 = {ndarray} [0.525384] ...View as Array

▶ 000010 = {ndarray} [0.23177826] ...View as Array

▶ 000011 = {ndarray} [0.516163] ...View as Array

▶ 000012 = {ndarray} [0.3565858] ...View as Array

▶ 000013 = {ndarray} [0.47983018] ...View as Array

▶ 000014 = {ndarray} [0.15843396] ...View as Array

▶ 000015 = {ndarray} [0.28008723] ...View as Array

▶ 000016 = {ndarray} [0.15173629] ...View as Array

▶ 000017 = {ndarray} [0.37323993] ...View as Array

```
auc_test = model.evaluate(test_X, test_y, batch_size=batch_size)
```

will return AUC (area under curve – accuracy) for all testing samples, for example:

test AUC: 0.690566