machine_learning_2

March 29, 2020

1 Part of ML experiments Group A

Data reflective of those of rail operations. LDPRF 2097 is used for training while LDPRF 2098 is used for testing. The training dataset was at full size, without 10% reductions.

1.0.1 Import necessary libraries

[1]:
```
import numpy as np
import pandas as pd
import copy
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# import codebase
import thermalModel_main as tmm
import thermalModel_groupB as tm_gb

import importlib
importlib.reload(tmm)
importlib.reload(tm_gb)
```

```
Using TensorFlow backend.
```

[1]:
```
<module 'thermalModel_groupB' from
'C:\\Users\\user\\Anaconda3\\lib\\thermalModel_groupB.py'>
```

1.1 ANN Ah Model

1.1.1 Data loading and cleaning

[2]:
```
df = tm_gb.load_csv(filename = 'LDPRF_2097.csv',
#                      data_list = ['Program time','AhCha','AhDch','Temp'],
                    features_list = ['runtime_s','AhCha','AhDch','Amb','Temp'],
                    mode = 0)
```

```
df1 = tm_gb.load_csv(filename = 'LDPRF_2098.csv',
#                        data_list = ['Program time','AhCha','AhDch','Temp'],
                     features_list = ['runtime_s','AhCha','AhDch','Amb','Temp'],
                     mode = 0)
```

```
C:\Users\user\Anaconda3\lib\thermalModel_groupB.py:47: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  df['second'][set_index[index]:set_index[index+1]] =
df['second'][set_index[index]:set_index[index+1]] + second_increment[index]
C:\Users\user\Anaconda3\lib\thermalModel_groupB.py:49: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  df['second'][set_index[index]:] = df['second'][set_index[index]:] +
second_increment[index]
C:\Users\user\Anaconda3\lib\thermalModel_groupB.py:56: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  df['second'][set_index[index]:] = df['second'][set_index[index]:] +
seconds_summation[index]
```

[3]:
```
ANN_Ah_models_2097 = {}
ANN_Ah_me_2097 = {}

df_train = df.copy(deep=True)
df_train.drop(columns = ['runtime_s'], inplace = True)
try:
    df1.drop(columns = ['runtime_s'], inplace = True)
except:
    pass

print(df_train.describe())
print(df1.describe())

Ah_models_2097, Ah_me_2097 = tmm.loop_run_instances(identifier = "ANN" + '_' +␣
 ↪"full_size",
                                                      loop_name = "Ah_model",
                                                      num_layers = 1,
                                                      train_dataframe =␣
 ↪df_train,
                                                      test_dataframe = df1,
                                                      num_inputs = 3,
                                                      start_window_size = 1,
                                                      end_window_size = 1,
                                                      window_size_step = 1,
                                                      test_size = 0,
                                                      num_epochs = 1000)

ANN_Ah_models_2097["ANN" + '_' + "full_size"] = Ah_models_2097
ANN_Ah_me_2097["ANN" + '_' + "full_size"] = copy.deepcopy(Ah_me_2097)
```

```
                AhCha           AhDch           Amb           Temp
count   435839.000000   435839.000000  4.358390e+05  435839.000000
mean       126.363856      144.644944  2.579465e+01      34.312581
std         72.924632       74.703530  2.402277e-10       2.060416
min          0.000000        0.000000  2.579465e+01      25.794650
25%         64.452000       81.299000  2.579465e+01      33.008410
50%        126.039000      145.061000  2.579465e+01      35.085100
75%        187.997000      208.479000  2.579465e+01      35.850190
max        252.040000      272.253000  2.579465e+01      36.724590
                AhCha           AhDch           Amb           Temp
count   435839.000000   435839.000000  4.358390e+05  435839.000000
mean       126.437695      143.968215  2.626750e+01      34.934164
std         72.927347       74.268447  6.600594e-11       1.938317
min          0.000000        0.000000  2.626750e+01      26.267500
25%         64.531000       80.996500  2.626750e+01      33.803260
50%        126.152000      144.421000  2.626750e+01      35.741030
75%        188.089000      207.443000  2.626750e+01      36.386950
max        252.044000      270.765000  2.626750e+01      37.032870
Run parameters: 1_[3]_relu_earlyStop
Restoring model weights from the end of the best epoch
Epoch 00006: early stopping
Time to train model: 86.9372091293335 seconds
```

[4]:
```
importlib.reload(tmm)
importlib.reload(tm_gb)
ANN_Ah_models_2097_df = tm_gb.extract_complexity(nested_model_dictionary =␣
 ↪ANN_Ah_models_2097,
                                                  nested_errors_dictionary =␣
 ↪ANN_Ah_me_2097)
```

[5]:
```
ANN_Ah_models_2097_df
```

[5]:
```
   Percentage_reduced  NN_size  mean_error
0                  10       16     0.31036
```

1.2 ANN IV Model

1.2.1 Data loading and cleaning

[6]:
```
df = tm_gb.load_csv(filename = 'LDPRF_2097.csv',
#                      data_list = ['Program time','AhCha','AhDch','Temp'],
                    features_list =␣
 ↪['runtime_s','Current','Voltage','Amb','Temp'],
                    mode = 1)

df1 = tm_gb.load_csv(filename = 'LDPRF_2098.csv',
#                      data_list = ['Program time','AhCha','AhDch','Temp'],
                     features_list =␣
 ↪['runtime_s','Current','Voltage','Amb','Temp'],
                     mode = 1)
```

[7]:
```
ANN_IV_models_2097 = {}
ANN_IV_me_2097 = {}

df_train = df.copy(deep=True)
df_train.drop(columns = ['runtime_s'], inplace = True)
try:
    df1.drop(columns = ['runtime_s'], inplace = True)
except:
    pass

print(df_train.describe())
print(df1.describe())

IV_models_2097, IV_me_2097 = tmm.loop_run_instances(identifier = "ANN" + '_' +␣
 ↪"full_size",
                                                 loop_name = "IV_model",
                                                 num_layers = 1,
                                                 train_dataframe =␣
 ↪df_train,
                                                 test_dataframe = df1,
                                                 num_inputs = 3,
                                                 start_window_size = 1,
                                                 end_window_size = 1,
                                                 window_size_step = 1,
                                                 test_size = 0,
                                                 num_epochs = 1000)

ANN_IV_models_2097["ANN" + '_' + "full_size"] = IV_models_2097
ANN_IV_me_2097["ANN" + '_' + "full_size"] = copy.deepcopy(IV_me_2097)
```

```
             Current        Voltage           Amb           Temp
count  435839.000000  435839.000000  4.358390e+05  435839.000000
```

```
mean        -0.595961      3.775370  2.579465e+01      34.312581
std         85.854861      0.091213  2.402277e-10       2.060416
min       -177.639340      3.536830  2.579465e+01      25.794650
25%          0.009580      3.730960  2.579465e+01      33.008410
50%          0.009580      3.766810  2.579465e+01      35.085100
75%          0.019150      3.807290  2.579465e+01      35.850190
max        223.268950      4.160100  2.579465e+01      36.724590
              Current       Voltage           Amb           Temp
count   435839.000000  435839.000000  4.358390e+05  435839.000000
mean        -0.497548      3.782469  2.626750e+01      34.934164
std         85.732075      0.086605  6.600594e-11       1.938317
min       -176.603480      3.557440  2.626750e+01      26.267500
25%          0.009560      3.741410  2.626750e+01      33.803260
50%          0.009560      3.773560  2.626750e+01      35.741030
75%          0.009560      3.813390  2.626750e+01      36.386950
max        222.893370      4.161120  2.626750e+01      37.032870
Run parameters: 1_[3]_relu_earlyStop
Restoring model weights from the end of the best epoch
Epoch 00257: early stopping
Time to train model: 5597.285449266434 seconds
```

1.3 ANN Hybrid Model

1.3.1 Data loading and cleaning

[8]:
```
df = tm_gb.load_csv(filename = 'LDPRF_2097.csv',
#                      data_list = ['Program␣
↪time','Current','Voltage','AhCha','AhDch','Temp'],
                    features_list =␣
↪['runtime_s','Current','Voltage','AhCha','AhDch','Amb','Temp'],
                    mode = 2)

df1 = tm_gb.load_csv(filename = 'LDPRF_2098.csv',
#                      data_list = ['Program␣
↪time','Current','Voltage','AhCha','AhDch','Temp'],
                     features_list =␣
↪['runtime_s','Current','Voltage','AhCha','AhDch','Amb','Temp'],
                     mode = 2)
```

[9]:
```
ANN_hybrid_models_2097 = {}
ANN_hybrid_me_2097 = {}

df_train = df.copy(deep=True)
df_train.drop(columns = ['runtime_s'], inplace = True)
try:
    df1.drop(columns = ['runtime_s'], inplace = True)
except:
    pass
```

```
print(df_train.describe())
print(df1.describe())

hybrid_models_2097, hybrid_me_2097 = tmm.loop_run_instances(identifier = "ANN"␣
 ↪+ '_' + "full_size",
                                                             loop_name =␣
 ↪"hybrid_model",
                                                             num_layers = 1,
                                                             train_dataframe␣
 ↪= df_train,
                                                             test_dataframe =␣
 ↪df1,
                                                             num_inputs = 5,
                                                            ␣
 ↪start_window_size = 1,
                                                             end_window_size␣
 ↪= 1,
                                                             window_size_step␣
 ↪= 1,
                                                             test_size = 0,
                                                             num_epochs =␣
 ↪1000)

ANN_hybrid_models_2097["ANN" + '_' + "full_size"] = hybrid_models_2097
ANN_hybrid_me_2097["ANN" + '_' + "full_size"] = copy.deepcopy(hybrid_me_2097)
```

```
             Current         Voltage           AhCha           AhDch  \
count  435839.000000  435839.000000  435839.000000  435839.000000
mean       -0.595961        3.775370     126.363856     144.644944
std        85.854861        0.091213      72.924632      74.703530
min      -177.639340        3.536830       0.000000       0.000000
25%         0.009580        3.730960      64.452000      81.299000
50%         0.009580        3.766810     126.039000     145.061000
75%         0.019150        3.807290     187.997000     208.479000
max       223.268950        4.160100     252.040000     272.253000

                Amb           Temp
count  4.358390e+05  435839.000000
mean   2.579465e+01      34.312581
std    2.402277e-10       2.060416
min    2.579465e+01      25.794650
25%    2.579465e+01      33.008410
50%    2.579465e+01      35.085100
75%    2.579465e+01      35.850190
max    2.579465e+01      36.724590
             Current         Voltage           AhCha           AhDch  \
count  435839.000000  435839.000000  435839.000000  435839.000000
mean       -0.497548       3.782469     126.437695     143.968215
std        85.732075       0.086605      72.927347      74.268447
min      -176.603480       3.557440       0.000000       0.000000
25%         0.009560       3.741410      64.531000      80.996500
50%         0.009560       3.773560     126.152000     144.421000
75%         0.009560       3.813390     188.089000     207.443000
max       222.893370       4.161120     252.044000     270.765000

                Amb           Temp
count  4.358390e+05  435839.000000
mean   2.626750e+01      34.934164
std    6.600594e-11       1.938317
min    2.626750e+01      26.267500
25%    2.626750e+01      33.803260
50%    2.626750e+01      35.741030
75%    2.626750e+01      36.386950
max    2.626750e+01      37.032870
Run parameters: 1_[5]_relu_earlyStop
Restoring model weights from the end of the best epoch
Epoch 00056: early stopping
Time to train model: 791.5689101219177 seconds
```

1.4 ANN errors

[10]:
```
importlib.reload(tmm)
importlib.reload(tm_gb)

ANN_Ah_models_2097_df = tm_gb.extract_complexity(nested_model_dictionary =␣
 ↪ANN_Ah_models_2097,
                                                  nested_errors_dictionary =␣
 ↪ANN_Ah_me_2097)

ANN_IV_models_2097_df = tm_gb.extract_complexity(nested_model_dictionary =␣
 ↪ANN_IV_models_2097,
                                                  nested_errors_dictionary =␣
 ↪ANN_IV_me_2097)

ANN_hybrid_models_2097_df = tm_gb.extract_complexity(nested_model_dictionary =␣
 ↪ANN_hybrid_models_2097,
                                                      nested_errors_dictionary =␣
 ↪ANN_hybrid_me_2097)
```

[11]:
```
ANN_reductions_dict = {
    'Ah_model':ANN_Ah_models_2097_df,
    'IV_model':ANN_IV_models_2097_df,
    'hybrid_model':ANN_hybrid_models_2097_df
}
```

[12]:
```
import pickle

with open('ANN_%reductions_dict.pickle', 'wb') as handle:
    pickle.dump(ANN_reductions_dict, handle)

for key in ANN_reductions_dict.keys():
    ANN_reductions_dict[key].to_csv('ANN_%reductions_' + key + '.csv',␣
 ↪index=False)
```

1.5 DNN Ah Model

1.5.1 Data loading and cleaning

[13]:
```
df = tm_gb.load_csv(filename = 'LDPRF_2097.csv',
#                      data_list = ['Program time','AhCha','AhDch','Temp'],
                    features_list = ['runtime_s','AhCha','AhDch','Amb','Temp'],
                    mode = 0)

df1 = tm_gb.load_csv(filename = 'LDPRF_2098.csv',
#                       data_list = ['Program time','AhCha','AhDch','Temp'],
                     features_list = ['runtime_s','AhCha','AhDch','Amb','Temp'],
                     mode = 0)
```

[14]:
```
DNN_Ah_models_2097 = {}
DNN_Ah_me_2097 = {}

df_train = df.copy(deep=True)
df_train.drop(columns = ['runtime_s'], inplace = True)
try:
    df1.drop(columns = ['runtime_s'], inplace = True)
except:
    pass

print(df_train.describe())
print(df1.describe())

Ah_models_2097, Ah_me_2097 = tmm.loop_run_instances(identifier = "DNN" + '_' +␣
 ↪"full_size",
                                                       loop_name = "Ah_model",
                                                       num_layers = 2,
                                                       train_dataframe =␣
 ↪df_train,
                                                       test_dataframe = df1,
                                                       num_inputs = 3,
                                                       start_window_size = 1,
                                                       end_window_size = 1,
                                                       window_size_step = 1,
                                                       test_size = 0,
                                                       num_epochs = 1000)

DNN_Ah_models_2097["DNN" + '_' + "full_size"] = Ah_models_2097
DNN_Ah_me_2097["DNN" + '_' + "full_size"] = copy.deepcopy(Ah_me_2097)
```

```
               AhCha          AhDch           Amb           Temp
count  435839.000000  435839.000000  4.358390e+05  435839.000000
mean      126.363856     144.644944  2.579465e+01      34.312581
std        72.924632      74.703530  2.402277e-10       2.060416
min         0.000000       0.000000  2.579465e+01      25.794650
25%        64.452000      81.299000  2.579465e+01      33.008410
50%       126.039000     145.061000  2.579465e+01      35.085100
75%       187.997000     208.479000  2.579465e+01      35.850190
max       252.040000     272.253000  2.579465e+01      36.724590
               AhCha          AhDch           Amb           Temp
count  435839.000000  435839.000000  4.358390e+05  435839.000000
mean      126.437695     143.968215  2.626750e+01      34.934164
std        72.927347      74.268447  6.600594e-11       1.938317
min         0.000000       0.000000  2.626750e+01      26.267500
25%        64.531000      80.996500  2.626750e+01      33.803260
50%       126.152000     144.421000  2.626750e+01      35.741030
75%       188.089000     207.443000  2.626750e+01      36.386950
max       252.044000     270.765000  2.626750e+01      37.032870
Run parameters: 1_[3, 3]_relu_earlyStop
Restoring model weights from the end of the best epoch
Epoch 00163: early stopping
Time to train model: 2339.326777935028 seconds
```

1.6 DNN IV Model

1.6.1 Data loading and cleaning

[15]:
```
df = tm_gb.load_csv(filename = 'LDPRF_2097.csv',
#                      data_list = ['Program time','AhCha','AhDch','Temp'],
                    features_list =␣
 ↪['runtime_s','Current','Voltage','Amb','Temp'],
                    mode = 1)

df1 = tm_gb.load_csv(filename = 'LDPRF_2098.csv',
#                      data_list = ['Program time','AhCha','AhDch','Temp'],
                     features_list =␣
 ↪['runtime_s','Current','Voltage','Amb','Temp'],
                     mode = 1)
```

[16]:
```
DNN_IV_models_2097 = {}
DNN_IV_me_2097 = {}

df_train = df.copy(deep=True)
df_train.drop(columns = ['runtime_s'], inplace = True)
try:
    df1.drop(columns = ['runtime_s'], inplace = True)
except:
    pass

print(df_train.describe())
print(df1.describe())

IV_models_2097, IV_me_2097 = tmm.loop_run_instances(identifier = "DNN" + '_' +␣
 ↪"full_size",
                                                        loop_name = "IV_model",
                                                        num_layers = 2,
                                                        train_dataframe =␣
 ↪df_train,
                                                        test_dataframe = df1,
                                                        num_inputs = 3,
                                                        start_window_size = 1,
                                                        end_window_size = 1,
                                                        window_size_step = 1,
                                                        test_size = 0,
                                                        num_epochs = 1000)

DNN_IV_models_2097["DNN" + '_' + "full_size"] = IV_models_2097
DNN_IV_me_2097["DNN" + '_' + "full_size"] = copy.deepcopy(IV_me_2097)
```

```
             Current        Voltage           Amb           Temp
count  435839.000000  435839.000000  4.358390e+05  435839.000000
mean       -0.595961       3.775370  2.579465e+01      34.312581
std        85.854861       0.091213  2.402277e-10       2.060416
min      -177.639340       3.536830  2.579465e+01      25.794650
25%         0.009580       3.730960  2.579465e+01      33.008410
50%         0.009580       3.766810  2.579465e+01      35.085100
75%         0.019150       3.807290  2.579465e+01      35.850190
max       223.268950       4.160100  2.579465e+01      36.724590
             Current        Voltage           Amb           Temp
count  435839.000000  435839.000000  4.358390e+05  435839.000000
mean       -0.497548       3.782469  2.626750e+01      34.934164
std        85.732075       0.086605  6.600594e-11       1.938317
min      -176.603480       3.557440  2.626750e+01      26.267500
25%         0.009560       3.741410  2.626750e+01      33.803260
50%         0.009560       3.773560  2.626750e+01      35.741030
75%         0.009560       3.813390  2.626750e+01      36.386950
```

```
max       222.893370       4.161120  2.626750e+01      37.032870
Run parameters: 1_[3, 3]_relu_earlyStop
Restoring model weights from the end of the best epoch
Epoch 00154: early stopping
Time to train model: 2160.6920866966248 seconds
```

1.7 DNN Hybrid Model

1.7.1 Data loading and cleaning

[17]:
```
df = tm_gb.load_csv(filename = 'LDPRF_2097.csv',
#                     data_list = ['Program␣
↪time','Current','Voltage','AhCha','AhDch','Temp'],
                    features_list =␣
↪['runtime_s','Current','Voltage','AhCha','AhDch','Amb','Temp'],
                    mode = 2)

df1 = tm_gb.load_csv(filename = 'LDPRF_2098.csv',
#                     data_list = ['Program␣
↪time','Current','Voltage','AhCha','AhDch','Temp'],
                     features_list =␣
↪['runtime_s','Current','Voltage','AhCha','AhDch','Amb','Temp'],
                     mode = 2)
```

[18]:
```
DNN_hybrid_models_2097 = {}
DNN_hybrid_me_2097 = {}

df_train = df.copy(deep=True)
df_train.drop(columns = ['runtime_s'], inplace = True)
try:
    df1.drop(columns = ['runtime_s'], inplace = True)
except:
    pass

print(df_train.describe())
print(df1.describe())

hybrid_models_2097, hybrid_me_2097 = tmm.loop_run_instances(identifier = "DNN"␣
↪+ '_' + "full_size",
                                                              loop_name =␣
↪"hybrid_model",
                                                              num_layers = 2,
                                                              train_dataframe␣
↪= df_train,
                                                              test_dataframe =␣
↪df1,
                                                              num_inputs = 5,
                                                                    ␣
↪start_window_size = 1,
                                                                     end_window_size␣
↪= 1,
                                                                     window_size_step␣
↪= 1,
                                                                     test_size = 0,
                                                                     num_epochs =␣
↪1000)

DNN_hybrid_models_2097["DNN" + '_' + "full_size"] = hybrid_models_2097
DNN_hybrid_me_2097["DNN" + '_' + "full_size"] = copy.deepcopy(hybrid_me_2097)
```

```
             Current         Voltage           AhCha           AhDch  \
count  435839.000000   435839.000000   435839.000000   435839.000000
mean       -0.595961        3.775370      126.363856      144.644944
std        85.854861        0.091213       72.924632       74.703530
min      -177.639340        3.536830        0.000000        0.000000
25%         0.009580        3.730960       64.452000       81.299000
50%         0.009580        3.766810      126.039000      145.061000
75%         0.019150        3.807290      187.997000      208.479000
max       223.268950        4.160100      252.040000      272.253000

                Amb           Temp
count  4.358390e+05  435839.000000
mean   2.579465e+01      34.312581
std    2.402277e-10       2.060416
min    2.579465e+01      25.794650
25%    2.579465e+01      33.008410
50%    2.579465e+01      35.085100
75%    2.579465e+01      35.850190
max    2.579465e+01      36.724590
             Current         Voltage           AhCha           AhDch  \
count  435839.000000   435839.000000   435839.000000   435839.000000
mean       -0.497548        3.782469      126.437695      143.968215
std        85.732075        0.086605       72.927347       74.268447
min      -176.603480        3.557440        0.000000        0.000000
25%         0.009560        3.741410       64.531000       80.996500
50%         0.009560        3.773560      126.152000      144.421000
75%         0.009560        3.813390      188.089000      207.443000
max       222.893370        4.161120      252.044000      270.765000

                Amb           Temp
count  4.358390e+05  435839.000000
mean   2.626750e+01      34.934164
std    6.600594e-11       1.938317
min    2.626750e+01      26.267500
```

```
25%    2.626750e+01      33.803260
50%    2.626750e+01      35.741030
75%    2.626750e+01      36.386950
max    2.626750e+01      37.032870
Run parameters: 1_[5, 5]_relu_earlyStop
Restoring model weights from the end of the best epoch
Epoch 00037: early stopping
Time to train model: 513.9118633270264 seconds
```

1.8 DNN errors

[19]:
```
importlib.reload(tmm)
importlib.reload(tm_gb)

DNN_Ah_models_2097_df = tm_gb.extract_complexity(nested_model_dictionary =␣
 ↪DNN_Ah_models_2097,
                                                  nested_errors_dictionary =␣
 ↪DNN_Ah_me_2097)

DNN_IV_models_2097_df = tm_gb.extract_complexity(nested_model_dictionary =␣
 ↪DNN_IV_models_2097,
                                                  nested_errors_dictionary =␣
 ↪DNN_IV_me_2097)

DNN_hybrid_models_2097_df = tm_gb.extract_complexity(nested_model_dictionary =␣
 ↪DNN_hybrid_models_2097,
                                                      nested_errors_dictionary =␣
 ↪DNN_hybrid_me_2097)
```

[20]:
```
DNN_reductions_dict = {
    'Ah_model':DNN_Ah_models_2097_df,
    'IV_model':DNN_IV_models_2097_df,
    'hybrid_model':DNN_hybrid_models_2097_df
}
```

[21]:
```
import pickle

with open('DNN_%reductions_dict.pickle', 'wb') as handle:
    pickle.dump(DNN_reductions_dict, handle)

for key in DNN_reductions_dict.keys():
    DNN_reductions_dict[key].to_csv('DNN_%reductions_' + key + '.csv',␣
 ↪index=False)
```