machine_learning

March 29, 2020

1 Machine Learning using datasets in Group B

Data from experimental set-up. Data from battery 1 is used to train the model, while data from battery 2 is used to test the model on the ability to interpolate. Training: 1C, 2C, 3C; Testing: 1.5C, 2.5C

1.0.1 Import necessary libraries

[1]:
```
import pandas as pd
import numpy as np
import copy
import matplotlib.pyplot as plot
import time

import thermalModel_groupC as tm_gc
import importlib
importlib.reload(tm_gc)
```

```
Using TensorFlow backend.
```

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

2 Load training datasets

2.0.1 Load 'AhCha','AhDch','Amb','Temp' data for datasets with 1 C-rate and Load 'Current','Voltage','Amb','Temp' data for datasets with 1 C-rate (from first round of experiments)

[2]:
```
b1c_1 = tm_gc.load_preprocess_csv(filename = 'battery_1_1C.csv', to_plot =␣
↪False)
```

```
Data loaded!
```

2.0.2 Load 'AhCha','AhDch','Amb','Temp' data for datasets with 1 C-rate and Load 'Current','Voltage','Amb','Temp' data for datasets with 1 C-rate (from second round of experiments)

[3]:
```
b1c_2 = tm_gc.load_preprocess_csv(filename = 'battery_1_1C_day2.csv', to_plot =␣
↪False)
```

```
Data loaded!
```

2.0.3 Load 'AhCha','AhDch','Amb','Temp' data for datasets with 2 C-rate and Load 'Current','Voltage','Amb','Temp' data for datasets with 2 C-rate (from first round of experiments)

[4]:
```
b2c_1 = tm_gc.load_preprocess_csv(filename = 'battery_1_2C.csv', to_plot =␣
↪False)
```

```
Data loaded!
```

2.0.4 Load 'AhCha','AhDch','Amb','Temp' data for datasets with 2 C-rate and Load 'Current','Voltage','Amb','Temp' data for datasets with 2 C-rate (from second round of experiments)

[5]:
```
b2c_2 = tm_gc.load_preprocess_csv(filename = 'battery_1_2C_day2.csv', to_plot =␣
↪False)
```

```
Data loaded!
```

2.0.5 Load 'AhCha','AhDch','Amb','Temp' data for datasets with 3 C-rate and Load 'Current','Voltage','Amb','Temp' data for datasets with 3 C-rate (from first round of experiments)

[6]:
```
b3c_1 = tm_gc.load_preprocess_csv(filename = 'battery_1_3C.csv', to_plot =␣
↪False)
```

```
Data loaded!
```

2.0.6 Load 'AhCha','AhDch','Amb','Temp' data for datasets with 3 C-rate and Load 'Current','Voltage','Amb','Temp' data for datasets with 3 C-rate (from second round of experiments)

[7]:
```
b3c_2 = tm_gc.load_preprocess_csv(filename = 'battery_1_3C_test2.csv', to_plot␣
↪= False)
```

```
Data loaded!
```

2.1 Load test datasets

2.1.1 Load 'AhCha','AhDch','Amb','Temp' data for datasets with 1.5 C-rate and Load 'Current','Voltage','Amb','Temp' data for datasets with 1.5 C-rate

[8]:
```
b1p5c_1 = tm_gc.load_preprocess_csv(filename = 'battery_2_1point5_C.csv',␣
↪to_plot = False)
```

```
Data loaded!
```

2.1.2 Load 'AhCha','AhDch','Amb','Temp' data for datasets with 2.5 C-rate and Load 'Current','Voltage','Amb','Temp' data for datasets with 2.5 C-rate

[9]:
```
b2p5c_1 = tm_gc.load_preprocess_csv(filename = 'battery_2_2point5_C.csv',␣
↪to_plot = False)
```

```
Data loaded!
```

2.2 KIV - 3C with varying rest periods between charge-discharge cycles

[10]:
```
b_c_v_1 = tm_gc.load_preprocess_csv(filename = 'battery_2_varying_rest.csv',␣
↪to_plot = False)
```

```
Data loaded!
```

3 Machine Learning - Interpolate C-rates

3.0.1 ANN Ah model

[11]:
```
import importlib
importlib.reload(tm_gc)
```

[11]:
```
<module 'thermalModel_groupC' from
'C:\\Users\\user\\Anaconda3\\lib\\thermalModel_groupC.py'>
```

[12]:
```
temp_features = ['AhCha','AhDch','Tamb','T1','T2','T3','T4','T5','T6','T7'] #␣
↪[inputs, outputs]

# Prepare training datasets
training_1 = copy.deepcopy(b1c_1[temp_features])
training_2 = copy.deepcopy(b1c_2[temp_features])
training_3 = copy.deepcopy(b2c_1[temp_features])
training_4 = copy.deepcopy(b2c_2[temp_features])
training_5 = copy.deepcopy(b3c_1[temp_features])
training_6 = copy.deepcopy(b3c_2[temp_features])
temp_frames = [training_1, training_2, training_3, training_4, training_5,␣
↪training_6]
training_df = pd.concat(temp_frames)
```

```
# Train model
ANN_Ah_model = tm_gc.run_instance(model_name = 'ANN_Ah_model',
                                  num_layers = 1,
                                  dataframe_entry = training_df,
                                  num_inputs = 3,
                                  num_outputs = 7,
                                  window_size = 1,
                                  test_size = 0,
                                  num_epochs = 1000)

# Prepare test datasets
test_1 = copy.deepcopy(b1p5c_1[temp_features])
test_2 = copy.deepcopy(b2p5c_1[temp_features])
test_dictionary = {
    "1.5C_test":test_1,
    "2.5C_test":test_2
}

for key, value in test_dictionary.items():
    _ = tm_gc.make_estimations(model_name = 'ANN_Ah_model',
                                    dataframe_name = key,
                                    dataframe_entry = value,
                                    model = ANN_Ah_model,
                                    window_size = 1,
                                    num_inputs = 3,
                                    num_outputs = 7)
```

```
Run parameters: 1_[7]_relu_earlyStop
Time to train model: 1051.8232159614563 seconds
MeanError of temperature estimations on dataset 1.5C_test using model
ANN_Ah_model:
[-3.642639673595487, -4.023297451025228, -3.470258266056809,
-3.5035554421207986, -3.678340072780383, -3.641224704070997,
-3.807387071175739], Average MeanError: -3.680957525832206


MeanError of temperature estimations on dataset 2.5C_test using model
ANN_Ah_model:
[0.5004436288622238, 0.3017081150345742, 0.7952897065362439,
-0.41764303985732526, 0.6963115422837179, 0.0763705902783864,
-0.00030654819251428985], Average MeanError: 0.27888199927790097
```

3.0.2 ANN IV model

[13]:
```
temp_features = ['Current','Voltage','Tamb','T1','T2','T3','T4','T5','T6','T7']␣
 ↪# [inputs, outputs]

# Prepare training datasets
training_1 = copy.deepcopy(b1c_1[temp_features])
training_2 = copy.deepcopy(b1c_2[temp_features])
training_3 = copy.deepcopy(b2c_1[temp_features])
training_4 = copy.deepcopy(b2c_2[temp_features])
training_5 = copy.deepcopy(b3c_1[temp_features])
training_6 = copy.deepcopy(b3c_2[temp_features])
temp_frames = [training_1, training_2, training_3, training_4, training_5,␣
 ↪training_6]
training_df = pd.concat(temp_frames)

# Train model
ANN_IV_model = tm_gc.run_instance(model_name = 'ANN_IV_model',
                                  num_layers = 1,
                                  dataframe_entry = training_df,
                                  num_inputs = 3,
                                  num_outputs = 7,
                                  window_size = 1,
                                  test_size = 0,
                                  num_epochs = 1000)

# Prepare test datasets
test_1 = copy.deepcopy(b1p5c_1[temp_features])
test_2 = copy.deepcopy(b2p5c_1[temp_features])
test_dictionary = {
    "1.5C_test":test_1,
    "2.5C_test":test_2
}

for key, value in test_dictionary.items():
    _ = tm_gc.make_estimations(model_name = 'ANN_IV_model',
                                    dataframe_name = key,
                                    dataframe_entry = value,
                                    model = ANN_IV_model,
                                    window_size = 1,
                                    num_inputs = 3,
                                    num_outputs = 7)
```

```
Run parameters: 1_[7]_relu_earlyStop
Time to train model: 1117.5559759140015 seconds
MeanError of temperature estimations on dataset 1.5C_test using model
ANN_IV_model:
[-3.9078419448282107, -4.385053007060493, -3.7879155941544287,
```

```
-3.690717234596656, -4.007735924765329, -3.8689148637884507,
-4.118515350306145], Average MeanError: -3.9666705599285303


MeanError of temperature estimations on dataset 2.5C_test using model
ANN_IV_model:
[-1.1372593044791746, -1.168856100978032, -0.8079117961055107,
-1.7628218553678714, -0.8899467991713448, -1.5107741001185355,
-1.1555688715072714], Average MeanError: -1.20473411824682
```

3.0.3 ANN Hybrid model

[14]:
```
temp_features =␣
 ↪['Current','Voltage','AhCha','AhDch','Tamb','T1','T2','T3','T4','T5','T6','T7']␣
 ↪# [inputs, outputs]

# Prepare training datasets
training_1 = copy.deepcopy(b1c_1[temp_features])
training_2 = copy.deepcopy(b1c_2[temp_features])
training_3 = copy.deepcopy(b2c_1[temp_features])
training_4 = copy.deepcopy(b2c_2[temp_features])
training_5 = copy.deepcopy(b3c_1[temp_features])
training_6 = copy.deepcopy(b3c_2[temp_features])
temp_frames = [training_1, training_2, training_3, training_4, training_5,␣
 ↪training_6]
training_df = pd.concat(temp_frames)

# Train model
ANN_hybrid_model = tm_gc.run_instance(model_name = 'ANN_hybrid_model',
                                      num_layers = 1,
                                      dataframe_entry = training_df,
                                      num_inputs = 5,
                                      num_outputs = 7,
                                      window_size = 1,
                                      test_size = 0,
                                      num_epochs = 1000)

# Prepare test datasets
test_1 = copy.deepcopy(b1p5c_1[temp_features])
test_2 = copy.deepcopy(b2p5c_1[temp_features])
test_dictionary = {
    "1.5C_test":test_1,
    "2.5C_test":test_2
}
```

```
for key, value in test_dictionary.items():
    _ = tm_gc.make_estimations(model_name = 'ANN_hybrid_model',
                                    dataframe_name = key,
                                    dataframe_entry = value,
                                    model = ANN_hybrid_model,
                                    window_size = 1,
                                    num_inputs = 5,
                                    num_outputs = 7)
```

```
Run parameters: 1_[7]_relu_earlyStop
Restoring model weights from the end of the best epoch
Epoch 00136: early stopping
Time to train model: 145.9197075366974 seconds
MeanError of temperature estimations on dataset 1.5C_test using model
ANN_hybrid_model:
[-3.501684341311825, -3.773927385591643, -3.3272544279722407,
-3.4262912453446432, -3.478738928346247, -3.504505876826349,
-3.553500811199533], Average MeanError: -3.5094147166560687


MeanError of temperature estimations on dataset 2.5C_test using model
ANN_hybrid_model:
[1.8064548361446524, 1.5974886011664955, 2.087325281635605, 0.6874570664498078,
1.9922631326170783, 1.3560311066103932, 1.1951438832939785], Average MeanError:
1.5317377011311444
```

3.0.4 DNN Ah model

[15]:
```
import importlib
importlib.reload(tm_gc)
```

[15]:
```
<module 'thermalModel_groupC' from
'C:\\Users\\user\\Anaconda3\\lib\\thermalModel_groupC.py'>
```

[16]:
```
temp_features = ['AhCha','AhDch','Tamb','T1','T2','T3','T4','T5','T6','T7'] #␣
 ↪[inputs, outputs]

# Prepare training datasets
training_1 = copy.deepcopy(b1c_1[temp_features])
training_2 = copy.deepcopy(b1c_2[temp_features])
training_3 = copy.deepcopy(b2c_1[temp_features])
training_4 = copy.deepcopy(b2c_2[temp_features])
training_5 = copy.deepcopy(b3c_1[temp_features])
training_6 = copy.deepcopy(b3c_2[temp_features])
temp_frames = [training_1, training_2, training_3, training_4, training_5,␣
 ↪training_6]
```

```
training_df = pd.concat(temp_frames)

# Train model
DNN_Ah_model = tm_gc.run_instance(model_name = 'DNN_Ah_model',
                                  num_layers = 2,
                                  dataframe_entry = training_df,
                                  num_inputs = 3,
                                  num_outputs = 7,
                                  window_size = 1,
                                  test_size = 0,
                                  num_epochs = 1000)

# Prepare test datasets
test_1 = copy.deepcopy(b1p5c_1[temp_features])
test_2 = copy.deepcopy(b2p5c_1[temp_features])
test_dictionary = {
    "1.5C_test":test_1,
    "2.5C_test":test_2
}

for key, value in test_dictionary.items():
    _ = tm_gc.make_estimations(model_name = 'DNN_Ah_model',
                                    dataframe_name = key,
                                    dataframe_entry = value,
                                    model = DNN_Ah_model,
                                    window_size = 1,
                                    num_inputs = 3,
                                    num_outputs = 7)
```

```
Run parameters: 1_[7, 7]_relu_earlyStop
Restoring model weights from the end of the best epoch
Epoch 00221: early stopping
Time to train model: 238.81847882270813 seconds
MeanError of temperature estimations on dataset 1.5C_test using model
DNN_Ah_model:
[-3.1673730023404887, -3.4497051907774074, -2.9296408112249637,
-3.039169368892444, -3.120208018501971, -3.2088737744185774,
-3.277794717568846], Average MeanError: -3.1703949833892424


MeanError of temperature estimations on dataset 2.5C_test using model
DNN_Ah_model:
[1.0717819476326185, 1.0050366808190554, 1.4376687451496408,
0.08265571863051058, 1.3292687906171181, 0.5939504014674404,
0.6282091755350766], Average MeanError: 0.8783673514073514
```

3.0.5 DNN IV model

[17]:
```
temp_features = ['Current','Voltage','Tamb','T1','T2','T3','T4','T5','T6','T7']␣
 ↪# [inputs, outputs]

# Prepare training datasets
training_1 = copy.deepcopy(b1c_1[temp_features])
training_2 = copy.deepcopy(b1c_2[temp_features])
training_3 = copy.deepcopy(b2c_1[temp_features])
training_4 = copy.deepcopy(b2c_2[temp_features])
training_5 = copy.deepcopy(b3c_1[temp_features])
training_6 = copy.deepcopy(b3c_2[temp_features])
temp_frames = [training_1, training_2, training_3, training_4, training_5,␣
 ↪training_6]
training_df = pd.concat(temp_frames)

# Train model
DNN_IV_model = tm_gc.run_instance(model_name = 'DNN_IV_model',
                                  num_layers = 2,
                                  dataframe_entry = training_df,
                                  num_inputs = 3,
                                  num_outputs = 7,
                                  window_size = 1,
                                  test_size = 0,
                                  num_epochs = 1000)

# Prepare test datasets
test_1 = copy.deepcopy(b1p5c_1[temp_features])
test_2 = copy.deepcopy(b2p5c_1[temp_features])
test_dictionary = {
    "1.5C_test":test_1,
    "2.5C_test":test_2
}

for key, value in test_dictionary.items():
    _ = tm_gc.make_estimations(model_name = 'DNN_IV_model',
                                    dataframe_name = key,
                                    dataframe_entry = value,
                                    model = DNN_IV_model,
                                    window_size = 1,
                                    num_inputs = 3,
                                    num_outputs = 7)
```

```
Run parameters: 1_[7, 7]_relu_earlyStop
Restoring model weights from the end of the best epoch
Epoch 00925: early stopping
Time to train model: 1043.0687448978424 seconds
MeanError of temperature estimations on dataset 1.5C_test using model
```

```
DNN_IV_model:
[-2.8203676372896465, -3.0632351819154255, -2.6361363413623558,
-2.8432427471968986, -2.800955513392653, -2.854018984197456,
-2.7874252810433644], Average MeanError: -2.829340240913971


MeanError of temperature estimations on dataset 2.5C_test using model
DNN_IV_model:
[-0.9275606158083631, -0.9627183100170689, -0.6199607356460634,
-1.6575704832400822, -0.6980583125425406, -1.3332901144180438,
-0.9698052723387884], Average MeanError: -1.0241376920015643
```

3.0.6 DNN Hybrid model

[18]:
```
temp_features =␣
 ↪['Current','Voltage','AhCha','AhDch','Tamb','T1','T2','T3','T4','T5','T6','T7']␣
 ↪# [inputs, outputs]

# Prepare training datasets
training_1 = copy.deepcopy(b1c_1[temp_features])
training_2 = copy.deepcopy(b1c_2[temp_features])
training_3 = copy.deepcopy(b2c_1[temp_features])
training_4 = copy.deepcopy(b2c_2[temp_features])
training_5 = copy.deepcopy(b3c_1[temp_features])
training_6 = copy.deepcopy(b3c_2[temp_features])
temp_frames = [training_1, training_2, training_3, training_4, training_5,␣
 ↪training_6]
training_df = pd.concat(temp_frames)

# Train model
DNN_hybrid_model = tm_gc.run_instance(model_name = 'DNN_hybrid_model',
                                      num_layers = 2,
                                      dataframe_entry = training_df,
                                      num_inputs = 5,
                                      num_outputs = 7,
                                      window_size = 1,
                                      test_size = 0,
                                      num_epochs = 1000)

# Prepare test datasets
test_1 = copy.deepcopy(b1p5c_1[temp_features])
test_2 = copy.deepcopy(b2p5c_1[temp_features])
test_dictionary = {
    "1.5C_test":test_1,
    "2.5C_test":test_2
```

```
}

for key, value in test_dictionary.items():
    _ = tm_gc.make_estimations(model_name = 'DNN_hybrid_model',
                                    dataframe_name = key,
                                    dataframe_entry = value,
                                    model = DNN_hybrid_model,
                                    window_size = 1,
                                    num_inputs = 5,
                                    num_outputs = 7)
```

```
Run parameters: 1_[7, 7]_relu_earlyStop
Restoring model weights from the end of the best epoch
Epoch 00158: early stopping
Time to train model: 174.2464497089386 seconds
MeanError of temperature estimations on dataset 1.5C_test using model
DNN_hybrid_model:
[-4.482774450979499, -4.525162921917401, -4.237661530711569,
-4.1957275624141515, -4.420464131898969, -4.475947542368809,
-4.129567204918074], Average MeanError: -4.352472192172639


MeanError of temperature estimations on dataset 2.5C_test using model
DNN_hybrid_model:
[1.9178519482707312, 1.821453719985249, 2.2559673709124044, 0.7028843253488194,
2.155672907612402, 1.4189459262239408, 1.4517434036479713], Average MeanError:
1.6749313717145027
```

4 Machine Learning - Interpolate durations

[19]:
```
b_durations = tm_gc.load_preprocess_csv(filename = 'battery_2_varying_rest.
↪csv', to_plot = False)
```

```
Data loaded!
```

4.0.1 ANN Ah model

[20]:
```
import importlib
importlib.reload(tm_gc)
```

[20]:
```
<module 'thermalModel_groupC' from
'C:\\Users\\user\\Anaconda3\\lib\\thermalModel_groupC.py'>
```

[21]:
```
temp_features = ['AhCha','AhDch','Tamb','T1','T2','T3','T4','T5','T6','T7'] #␣
↪[inputs, outputs]
```

```
# Prepare training datasets
training_1 = copy.deepcopy(b_durations[temp_features])
training_2 = training_1.drop(training_1.index[[3490, 4690]]) # drop those at␣
 ↪back first
training_df = training_2.drop(training_2.index[[760, 1979]])

# Train model
ANN_Ah_model_1 = tm_gc.run_instance(model_name = 'ANN_Ah_model_1',
                                    num_layers = 1,
                                    dataframe_entry = training_df,
                                    num_inputs = 3,
                                    num_outputs = 7,
                                    window_size = 1,
                                    test_size = 0,
                                    num_epochs = 1000)

# Prepare test datasets
test_1a = copy.deepcopy(b_durations[temp_features]).iloc[760:1309, :]
test_1b = copy.deepcopy(b_durations[temp_features]).iloc[3489:4038, :]
test_2a = copy.deepcopy(b_durations[temp_features]).iloc[1310:1979, :]
test_2b = copy.deepcopy(b_durations[temp_features]).iloc[4039:4690, :]
temp_frames_1 = [test_1a, test_1b]
temp_frames_2 = [test_2a, test_2b]
test_1 = pd.concat(temp_frames_1)
test_2 = pd.concat(temp_frames_2)
test_dictionary = {
    "shorter_duration":test_1,
    "longer_duration":test_2
}

for key, value in test_dictionary.items():
    _ = tm_gc.make_estimations(model_name = 'ANN_Ah_model_1',
                                      dataframe_name = key,
                                      dataframe_entry = value,
                                      model = ANN_Ah_model_1,
                                      window_size = 1,
                                      num_inputs = 3,
                                      num_outputs = 7)
```

```
Run parameters: 1_[7]_relu_earlyStop
Time to train model: 190.14958357810974 seconds
MeanError of temperature estimations on dataset shorter_duration using model
ANN_Ah_model_1:
[0.18543688903641734, -0.0010181109863943458, 0.19255739252026177,
0.10107266616473601, 0.13522941512852923, 0.11932505980555143,
-0.1134138456082057], Average MeanError: 0.08845563800869939
```

```
MeanError of temperature estimations on dataset longer_duration using model
ANN_Ah_model_1:
[-0.5390291941934138, -0.6151353252141919, -0.5599172658392474,
-0.4196428314278391, -0.5978687533363412, -0.5031806222772485,
-0.5963320996311234], Average MeanError: -0.5473008702742007
```

4.0.2 ANN IV model

[22]:
```
temp_features = ['Current','Voltage','Tamb','T1','T2','T3','T4','T5','T6','T7']␣
 ↪# [inputs, outputs]

# Prepare training datasets
training_1 = copy.deepcopy(b_durations[temp_features])
training_2 = training_1.drop(training_1.index[[3490, 4690]]) # drop those at␣
 ↪back first
training_df = training_2.drop(training_2.index[[760, 1979]])

# Train model
ANN_IV_model_1 = tm_gc.run_instance(model_name = 'ANN_IV_model_1',
                                    num_layers = 1,
                                    dataframe_entry = training_df,
                                    num_inputs = 3,
                                    num_outputs = 7,
                                    window_size = 1,
                                    test_size = 0,
                                    num_epochs = 1000)

# Prepare test datasets
test_1a = copy.deepcopy(b_durations[temp_features]).iloc[760:1309, :]
test_1b = copy.deepcopy(b_durations[temp_features]).iloc[3489:4038, :]
test_2a = copy.deepcopy(b_durations[temp_features]).iloc[1310:1979, :]
test_2b = copy.deepcopy(b_durations[temp_features]).iloc[4039:4690, :]
temp_frames_1 = [test_1a, test_1b]
temp_frames_2 = [test_2a, test_2b]
test_1 = pd.concat(temp_frames_1)
test_2 = pd.concat(temp_frames_2)
test_dictionary = {
    "shorter_duration":test_1,
    "longer_duration":test_2
}

for key, value in test_dictionary.items():
    _ = tm_gc.make_estimations(model_name = 'ANN_IV_model_1',
```

```
                                        dataframe_name = key,
                                        dataframe_entry = value,
                                        model = ANN_IV_model_1,
                                        window_size = 1,
                                        num_inputs = 3,
                                        num_outputs = 7)
```

```
Run parameters: 1_[7]_relu_earlyStop
Restoring model weights from the end of the best epoch
Epoch 00415: early stopping
Time to train model: 81.19367170333862 seconds
MeanError of temperature estimations on dataset shorter_duration using model
ANN_IV_model_1:
[0.6056559244328019, 0.505096276699682, 0.6586415867905461, 0.42039144795921196,
0.585934985985839, 0.514562948569452, 0.3118225355678622], Average MeanError:
0.5145865294293421


MeanError of temperature estimations on dataset longer_duration using model
ANN_IV_model_1:
[-0.38048332486214065, -0.454374494863514, -0.40107994524975227,
-0.26713495978990903, -0.41860471799212695, -0.3462853658009511,
-0.44078592862931015], Average MeanError: -0.38696410531252917
```

4.0.3 ANN Hybrid model

[23]:
```
temp_features =␣
 ↪['Current','Voltage','AhCha','AhDch','Tamb','T1','T2','T3','T4','T5','T6','T7']␣
 ↪# [inputs, outputs]

# Prepare training datasets
training_1 = copy.deepcopy(b_durations[temp_features])
training_2 = training_1.drop(training_1.index[[3490, 4690]]) # drop those at␣
 ↪back first
training_df = training_2.drop(training_2.index[[760, 1979]])

# Train model
ANN_hybrid_model_1 = tm_gc.run_instance(model_name = 'ANN_hybrid_model_1',
                                        num_layers = 1,
                                        dataframe_entry = training_df,
                                        num_inputs = 5,
                                        num_outputs = 7,
                                        window_size = 1,
                                        test_size = 0,
                                        num_epochs = 1000)
```

```
# Prepare test datasets
test_1a = copy.deepcopy(b_durations[temp_features]).iloc[760:1309, :]
test_1b = copy.deepcopy(b_durations[temp_features]).iloc[3489:4038, :]
test_2a = copy.deepcopy(b_durations[temp_features]).iloc[1310:1979, :]
test_2b = copy.deepcopy(b_durations[temp_features]).iloc[4039:4690, :]
temp_frames_1 = [test_1a, test_1b]
temp_frames_2 = [test_2a, test_2b]
test_1 = pd.concat(temp_frames_1)
test_2 = pd.concat(temp_frames_2)
test_dictionary = {
    "shorter_duration":test_1,
    "longer_duration":test_2
}

for key, value in test_dictionary.items():
    _ = tm_gc.make_estimations(model_name = 'ANN_hybrid_model_1',
                                    dataframe_name = key,
                                    dataframe_entry = value,
                                    model = ANN_hybrid_model_1,
                                    window_size = 1,
                                    num_inputs = 5,
                                    num_outputs = 7)
```

```
Run parameters: 1_[7]_relu_earlyStop
Time to train model: 201.4973065853119 seconds
MeanError of temperature estimations on dataset shorter_duration using model
ANN_hybrid_model_1:
[0.21563428817929897, 0.04081296380042215, 0.2300727598039046,
0.12222741237421356, 0.168905486086878, 0.15272694896325958,
-0.0571813584417236], Average MeanError: 0.12474264296660761


MeanError of temperature estimations on dataset longer_duration using model
ANN_hybrid_model_1:
[-0.5249038112660621, -0.5820166224435567, -0.5365717798656723,
-0.376472914712369, -0.5568633447260992, -0.4954395791632914,
-0.5538866702781831], Average MeanError: -0.5180221032078905
```

4.0.4 DNN Ah model

[24]:
```
import importlib
importlib.reload(tm_gc)
```

[24]:
```
<module 'thermalModel_groupC' from
'C:\\Users\\user\\Anaconda3\\lib\\thermalModel_groupC.py'>
```

[25]:
```
temp_features = ['AhCha','AhDch','Tamb','T1','T2','T3','T4','T5','T6','T7'] #␣
 ↪[inputs, outputs]

# Prepare training datasets
training_1 = copy.deepcopy(b_durations[temp_features])
training_2 = training_1.drop(training_1.index[[3490, 4690]]) # drop those at␣
 ↪back first
training_df = training_2.drop(training_2.index[[760, 1979]])

# Train model
DNN_Ah_model_1 = tm_gc.run_instance(model_name = 'DNN_Ah_model_1',
                                    num_layers = 2,
                                    dataframe_entry = training_df,
                                    num_inputs = 3,
                                    num_outputs = 7,
                                    window_size = 1,
                                    test_size = 0,
                                    num_epochs = 1000)

# Prepare test datasets
test_1a = copy.deepcopy(b_durations[temp_features]).iloc[760:1309, :]
test_1b = copy.deepcopy(b_durations[temp_features]).iloc[3489:4038, :]
test_2a = copy.deepcopy(b_durations[temp_features]).iloc[1310:1979, :]
test_2b = copy.deepcopy(b_durations[temp_features]).iloc[4039:4690, :]
temp_frames_1 = [test_1a, test_1b]
temp_frames_2 = [test_2a, test_2b]
test_1 = pd.concat(temp_frames_1)
test_2 = pd.concat(temp_frames_2)
test_dictionary = {
    "shorter_duration":test_1,
    "longer_duration":test_2
}

for key, value in test_dictionary.items():
    _ = tm_gc.make_estimations(model_name = 'DNN_Ah_model_1',
                                       dataframe_name = key,
                                       dataframe_entry = value,
                                       model = DNN_Ah_model_1,
                                       window_size = 1,
                                       num_inputs = 3,
                                       num_outputs = 7)
```

```
Run parameters: 1_[7, 7]_relu_earlyStop
Restoring model weights from the end of the best epoch
Epoch 00659: early stopping
Time to train model: 123.87034058570862 seconds
MeanError of temperature estimations on dataset shorter_duration using model
```

```
DNN_Ah_model_1:
[0.2729766696870821, 0.04362249007525016, 0.313359796320621,
0.23091212082256968, 0.20855542686274667, 0.23553715179916326,
-0.02421519499424877], Average MeanError: 0.18296406579616917


MeanError of temperature estimations on dataset longer_duration using model
DNN_Ah_model_1:
[-0.46731499395016163, -0.5672362481869917, -0.44519097294203663,
-0.28292788311059547, -0.5082040786453946, -0.39136673939598565,
-0.5208331224790025], Average MeanError: -0.45472486267288115
```

4.0.5 DNN IV model

[26]:
```
temp_features = ['Current','Voltage','Tamb','T1','T2','T3','T4','T5','T6','T7']␣
 ↪# [inputs, outputs]

# Prepare training datasets
training_1 = copy.deepcopy(b_durations[temp_features])
training_2 = training_1.drop(training_1.index[[3490, 4690]]) # drop those at␣
 ↪back first
training_df = training_2.drop(training_2.index[[760, 1979]])

# Train model
DNN_IV_model_1 = tm_gc.run_instance(model_name = 'DNN_IV_model_1',
                                    num_layers = 2,
                                    dataframe_entry = training_df,
                                    num_inputs = 3,
                                    num_outputs = 7,
                                    window_size = 1,
                                    test_size = 0,
                                    num_epochs = 1000)

# Prepare test datasets
test_1a = copy.deepcopy(b_durations[temp_features]).iloc[760:1309, :]
test_1b = copy.deepcopy(b_durations[temp_features]).iloc[3489:4038, :]
test_2a = copy.deepcopy(b_durations[temp_features]).iloc[1310:1979, :]
test_2b = copy.deepcopy(b_durations[temp_features]).iloc[4039:4690, :]
temp_frames_1 = [test_1a, test_1b]
temp_frames_2 = [test_2a, test_2b]
test_1 = pd.concat(temp_frames_1)
test_2 = pd.concat(temp_frames_2)
test_dictionary = {
    "shorter_duration":test_1,
    "longer_duration":test_2
```

```
}

for key, value in test_dictionary.items():
    _ = tm_gc.make_estimations(model_name = 'DNN_IV_model_1',
                                    dataframe_name = key,
                                    dataframe_entry = value,
                                    model = DNN_IV_model_1,
                                    window_size = 1,
                                    num_inputs = 3,
                                    num_outputs = 7)
```

```
Run parameters: 1_[7, 7]_relu_earlyStop
Time to train model: 198.4151270389557 seconds
MeanError of temperature estimations on dataset shorter_duration using model
DNN_IV_model_1:
[0.4570000164967406, 0.2958726624564468, 0.4889296870548943,
0.32364612471546655, 0.4253940131605599, 0.37119517428068693,
0.1594393154481598], Average MeanError: 0.360210999087565


MeanError of temperature estimations on dataset longer_duration using model
DNN_IV_model_1:
[-0.5114315484707056, -0.6152092577491044, -0.5384288520032113,
-0.3302034480143348, -0.5736387955301899, -0.46573260830061414,
-0.5980529272095373], Average MeanError: -0.5189567767539568
```

4.0.6 DNN Hybrid model

[27]:
```
temp_features =␣
 ↪['Current','Voltage','AhCha','AhDch','Tamb','T1','T2','T3','T4','T5','T6','T7']␣
 ↪# [inputs, outputs]

# Prepare training datasets
training_1 = copy.deepcopy(b_durations[temp_features])
training_2 = training_1.drop(training_1.index[[3490, 4690]]) # drop those at␣
 ↪back first
training_df = training_2.drop(training_2.index[[760, 1979]])

# Train model
DNN_hybrid_model_1 = tm_gc.run_instance(model_name = 'DNN_hybrid_model_1',
                                        num_layers = 2,
                                        dataframe_entry = training_df,
                                        num_inputs = 5,
                                        num_outputs = 7,
                                        window_size = 1,
```

```
                                        test_size = 0,
                                        num_epochs = 1000)

# Prepare test datasets
test_1a = copy.deepcopy(b_durations[temp_features]).iloc[760:1309, :]
test_1b = copy.deepcopy(b_durations[temp_features]).iloc[3489:4038, :]
test_2a = copy.deepcopy(b_durations[temp_features]).iloc[1310:1979, :]
test_2b = copy.deepcopy(b_durations[temp_features]).iloc[4039:4690, :]
temp_frames_1 = [test_1a, test_1b]
temp_frames_2 = [test_2a, test_2b]
test_1 = pd.concat(temp_frames_1)
test_2 = pd.concat(temp_frames_2)
test_dictionary = {
    "shorter_duration":test_1,
    "longer_duration":test_2
}

for key, value in test_dictionary.items():
    _ = tm_gc.make_estimations(model_name = 'DNN_hybrid_model_1',
                                       dataframe_name = key,
                                       dataframe_entry = value,
                                       model = DNN_hybrid_model_1,
                                       window_size = 1,
                                       num_inputs = 5,
                                       num_outputs = 7)
```

```
Run parameters: 1_[7, 7]_relu_earlyStop
Time to train model: 208.51975178718567 seconds
MeanError of temperature estimations on dataset shorter_duration using model
DNN_hybrid_model_1:
[0.2691566537528447, 0.0951481276506066, 0.2783850069837131,
0.18706215315117272, 0.21591510282393075, 0.19600967601960467,
-0.020250196222184866], Average MeanError: 0.17448950345138395


MeanError of temperature estimations on dataset longer_duration using model
DNN_hybrid_model_1:
[-0.46090200341769066, -0.5576736827905289, -0.48625087908671744,
-0.29995131381991375, -0.5170196430243891, -0.4154873579717925,
-0.5366151916989779], Average MeanError: -0.4677000102585729
```