Passion_v4.1

July 16, 2020

1 Welcome to Passion!

Passion is a model that can detection anomaly using different methods (Both supervised and unsupervised)

1. The goal for this project is to study the difference between different anomnaly detection model, and to find the state of art method for detecting anomaly in real world data
2. Evaluate the results based on this :real server data+ https://www.kaggle.com/sohier/30-years-of-european-wind-generation + https://github.com/numenta/NAB
3. Also use real data generated from server.
4. The model has the following fuctions:

    a. Visualize the input data. Help the user to find critical features within the inputs.
    b. Give user options to choose different models that are suitable for different circumstances.
    c. Evaluate the performance based on the rules in this link https://github.com/numenta/NAB
    d. Save model. Easy to be appplied to other dataset.

5. This is the very beginning of the process. Still need to do a lot of works!

2 What's new in version 4.1

1. Add LSTM based model (Also include HMM features)
2. Add more real server data
3. Add 1d CNN
4. Add GRU
5. Add a new fusion model
6. Add MLSTM-FCN: https://arxiv.org/pdf/1709.05206.pdf

```
In [1]: # import packages


        from matplotlib.pylab import rc
        import torch
        from scipy.stats import chisquare
        from scipy.stats import pearsonr
        import pickle
```

```python
import pandas as pd
import datetime
import matplotlib
import tensorflow as tf
import sklearn
import math
import matplotlib.pyplot as plt
import xgboost
from xgboost import XGBClassifier
from xgboost import plot_importance
import numpy as np
from sklearn.model_selection import train_test_split
import sklearn
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
import copy
import scipy
import datetime
import time
import os
from sklearn.model_selection import KFold
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.covariance import EllipticEnvelope
from sklearn.ensemble import IsolationForest
from sklearn.svm import OneClassSVM
import gc

plot_path = "plots/"
```

In [2]:
```python
# Real server data

root_path = "Data/Ant_202007/"


cif = pd.read_json(root_path+'cif.json', orient='index')
paycore = pd.read_json(root_path+'paycore.json', orient='index')
paydecision = pd.read_json(root_path+'paydecision.json', orient='index')
paydecision2 = pd.read_json(root_path+'paydecision2.json', orient='index')
paydecision3 = pd.read_json(root_path+'paydecision3.json', orient='index')

df = pd.DataFrame()
df["time_stamp"] = cif.index
df["cif"] = cif[0].values
df["paycore"] = paycore[0].values
df["paydecision"] = paydecision[0].values
```

```python
df["paydecision2"] = paydecision2[0].values
df["paydecision3"] = paydecision3[0].values

# Optional
if False:
    df.to_csv(root_path+"fusion.csv")


# convert time stamp
df['time_stamp'] = pd.to_datetime(df['time_stamp'])
names_array = np.array(df.keys()[1:],dtype="str")
os.listdir(root_path)
```

Out[2]:
```
['.ipynb_checkpoints',
 'cif.json',
 'fusion.csv',
 'paycore.json',
 'paydecision.json',
 'paydecision2.json',
 'paydecision3.json']
```

In [3]:
```python
# calculate previous hour high low:
# convert to seconds
temp = df['time_stamp'] - min(df['time_stamp'])
temp = temp.dt.total_seconds().astype(int)
df["hours"] = temp//3600

h_max = max(df["hours"])+1

for n in range(len(names_array)):
    df[names_array[n]+"_open"] = df[names_array[n]]
    df[names_array[n]+"_close"] = df[names_array[n]]
    df[names_array[n]+"_max"] = df[names_array[n]]
    df[names_array[n]+"_min"] = df[names_array[n]]

for j in range(1,h_max):
    mask_j = df["hours"]==j-1
    max_val = df[mask_j][names_array].max(axis=0).values
    min_val = df[mask_j][names_array].max(axis=0).values
    open_val = df[mask_j][names_array].values[0,:]
    close_val = df[mask_j][names_array].values[-1,:]
    mask_i = df["hours"]==j
    r = df[mask_i][names_array].shape[0]
    df.loc[mask_i,[r+"_open" for r in names_array]] = np.tile(open_val,(r,1))
    df.loc[mask_i,[r+"_close" for r in names_array]] = np.tile(close_val,(r,1))

    df.loc[mask_i,[r+"_max" for r in names_array]] = np.tile(max_val,(r,1))
    df.loc[mask_i,[r+"_min" for r in names_array]] = np.tile(min_val,(r,1))
```

```
In [4]: df
```

```
Out[4]:               time_stamp        cif  paycore  paydecision  paydecision2  \
0      2020-06-03 16:00:00  5230362.0  1742333       810511        894642
1      2020-06-03 16:01:00  5430718.0  1250771       732380        720773
2      2020-06-03 16:02:00  5352478.0   998340       715939        691644
3      2020-06-03 16:03:00  5247694.0   971876       701533        669921
4      2020-06-03 16:04:00  5197260.0   926380       685236        649162
...                    ...        ...      ...          ...           ...
41756  2020-07-02 15:56:00  4573918.0   681739       549321        490459
41757  2020-07-02 15:57:00  4562205.0   675371       546406        489352
41758  2020-07-02 15:58:00  4546905.0   668348       539951        486135
41759  2020-07-02 15:59:00  4544560.0   663263       535808        481801
41760  2020-07-02 16:00:00  4906181.0   701978       557651        509909

       paydecision3  hours   cif_open  cif_close    cif_max  ...  \
0            254995      0  5230362.0  5230362.0  5230362.0  ...
1            213345      0  5430718.0  5430718.0  5430718.0  ...
2            163959      0  5352478.0  5352478.0  5352478.0  ...
3            165899      0  5247694.0  5247694.0  5247694.0  ...
4            167605      0  5197260.0  5197260.0  5197260.0  ...
...             ...    ...        ...        ...        ...  ...
41756        142306    695  8358653.0  6482250.0  8439483.0  ...
41757        137810    695  8358653.0  6482250.0  8439483.0  ...
41758        128817    695  8358653.0  6482250.0  8439483.0  ...
41759        131527    695  8358653.0  6482250.0  8439483.0  ...
41760        142412    696  6484087.0  4544560.0  6484087.0  ...

       paydecision_max  paydecision_min  paydecision2_open  \
0             810511.0         810511.0           894642.0
1             732380.0         732380.0           720773.0
2             715939.0         715939.0           691644.0
3             701533.0         701533.0           669921.0
4             685236.0         685236.0           649162.0
...                ...              ...                ...
41756        1072625.0        1072625.0           987553.0
41757        1072625.0        1072625.0           987553.0
41758        1072625.0        1072625.0           987553.0
41759        1072625.0        1072625.0           987553.0
41760         808131.0         808131.0           760330.0

       paydecision2_close  paydecision2_max  paydecision2_min  \
0                894642.0          894642.0          894642.0
1                720773.0          720773.0          720773.0
2                691644.0          691644.0          691644.0
```

```
3               669921.0          669921.0          669921.0
4               649162.0          649162.0          649162.0
...                  ...               ...               ...
41756           771209.0          990662.0          990662.0
41757           771209.0          990662.0          990662.0
41758           771209.0          990662.0          990662.0
41759           771209.0          990662.0          990662.0
41760           481801.0          760330.0          760330.0

       paydecision3_open  paydecision3_close  paydecision3_max  \
0               254995.0            254995.0          254995.0
1               213345.0            213345.0          213345.0
2               163959.0            163959.0          163959.0
3               165899.0            165899.0          165899.0
4               167605.0            167605.0          167605.0
...                  ...                 ...               ...
41756           261211.0            218522.0          293880.0
41757           261211.0            218522.0          293880.0
41758           261211.0            218522.0          293880.0
41759           261211.0            218522.0          293880.0
41760           224933.0            131527.0          226279.0

       paydecision3_min
0              254995.0
1              213345.0
2              163959.0
3              165899.0
4              167605.0
...                 ...
41756          293880.0
41757          293880.0
41758          293880.0
41759          293880.0
41760          226279.0

[41761 rows x 27 columns]
```

3 LSTM based model

This time with high+low+on+off for 1d data

In [28]:
```
df["minutes"]=df["time_stamp"].dt.hour*1440+df["time_stamp"].dt.hour*60+df["time_stam
# hyper-parameters:
# delta_t in minute,try a day first,output 5 dimensions
delta_t = 1440
n_epoch=10
n_cell = 50
```

```python
# predict 1 minute for now
N_output=1
index_name= 0

checkpoint_path = "LSTM/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)

min_max_scaler = preprocessing.StandardScaler()

# min-max scaler
np_scaled = min_max_scaler.fit_transform(df[names_array])

df_scaled = pd.DataFrame(np_scaled,columns=names_array)


X = np.zeros((df_scaled.shape[0]-delta_t,delta_t,1),dtype=float)
y = df_scaled[names_array[index_name]][delta_t:]

for i in range(len(y)):
    if i%10000==0:
        print("Prepare data %.2f percent"%(100*i/len(y)))
    X[i,:,:] = np.atleast_2d(df_scaled[i:i+delta_t][names_array[index_name]].values).

# split train test:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=True)

model = tf.keras.Sequential([
  tf.keras.layers.LSTM(n_cell,input_shape=(X_train.shape[1],X_train.shape[2])),  # mu
  tf.keras.layers.Dense(1)
])

model.compile(loss='mae', optimizer='adam')
#model.summary()

callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_path,
                                              save_weights_only=True,
                                              verbose=1)
```

```
Prepare data 0.00 percent
Prepare data 24.80 percent
Prepare data 49.60 percent
Prepare data 74.40 percent
Prepare data 99.20 percent
```

In [29]:
```python
history = model.fit(X_train, y_train, epochs=n_epoch, batch_size=64, validation_data=
```

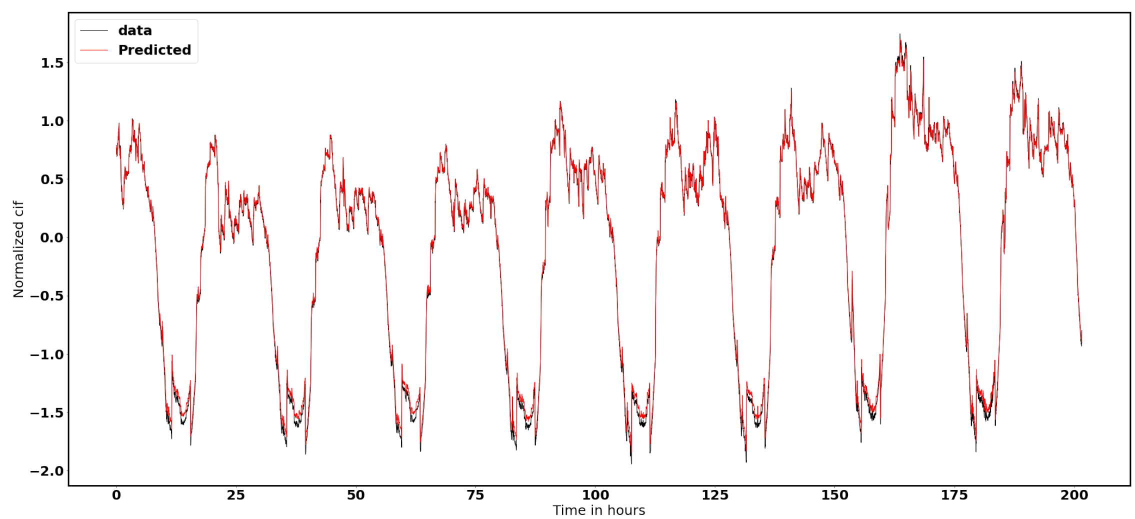
```
Epoch 1/10
441/441 [==============================] - ETA: 0s - loss: 0.1203
```

```
Epoch 00001: saving model to LSTM/cp.ckpt
441/441 [==============================] - 47s 106ms/step - loss: 0.1203 - val_loss: 0.0401
Epoch 2/10
441/441 [==============================] - ETA: 0s - loss: 0.0661
Epoch 00002: saving model to LSTM/cp.ckpt
441/441 [==============================] - 46s 105ms/step - loss: 0.0661 - val_loss: 0.0622
Epoch 3/10
441/441 [==============================] - ETA: 0s - loss: 0.0537
Epoch 00003: saving model to LSTM/cp.ckpt
441/441 [==============================] - 46s 105ms/step - loss: 0.0537 - val_loss: 0.0369
Epoch 4/10
441/441 [==============================] - ETA: 0s - loss: 0.0422
Epoch 00004: saving model to LSTM/cp.ckpt
441/441 [==============================] - 46s 105ms/step - loss: 0.0422 - val_loss: 0.0269
Epoch 5/10
441/441 [==============================] - ETA: 0s - loss: 0.0365
Epoch 00005: saving model to LSTM/cp.ckpt
441/441 [==============================] - 46s 104ms/step - loss: 0.0365 - val_loss: 0.0258
Epoch 6/10
441/441 [==============================] - ETA: 0s - loss: 0.0454
Epoch 00006: saving model to LSTM/cp.ckpt
441/441 [==============================] - 47s 106ms/step - loss: 0.0454 - val_loss: 0.0435
Epoch 7/10
441/441 [==============================] - ETA: 0s - loss: 0.0441
Epoch 00007: saving model to LSTM/cp.ckpt
441/441 [==============================] - 46s 105ms/step - loss: 0.0441 - val_loss: 0.0395
Epoch 8/10
441/441 [==============================] - ETA: 0s - loss: 0.0397
Epoch 00008: saving model to LSTM/cp.ckpt
441/441 [==============================] - 47s 105ms/step - loss: 0.0397 - val_loss: 0.0390
Epoch 9/10
441/441 [==============================] - ETA: 0s - loss: 0.0301
Epoch 00009: saving model to LSTM/cp.ckpt
441/441 [==============================] - 46s 105ms/step - loss: 0.0301 - val_loss: 0.0290
Epoch 10/10
441/441 [==============================] - ETA: 0s - loss: 0.0373
Epoch 00010: saving model to LSTM/cp.ckpt
441/441 [==============================] - 45s 101ms/step - loss: 0.0373 - val_loss: 0.0353
```

In [30]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=False
y_pre = model.predict(X_test)
font = {'family': 'normal','weight': 'bold',
        'size': 25}

matplotlib.rc('font', **font)
rc('axes', linewidth=3)
```

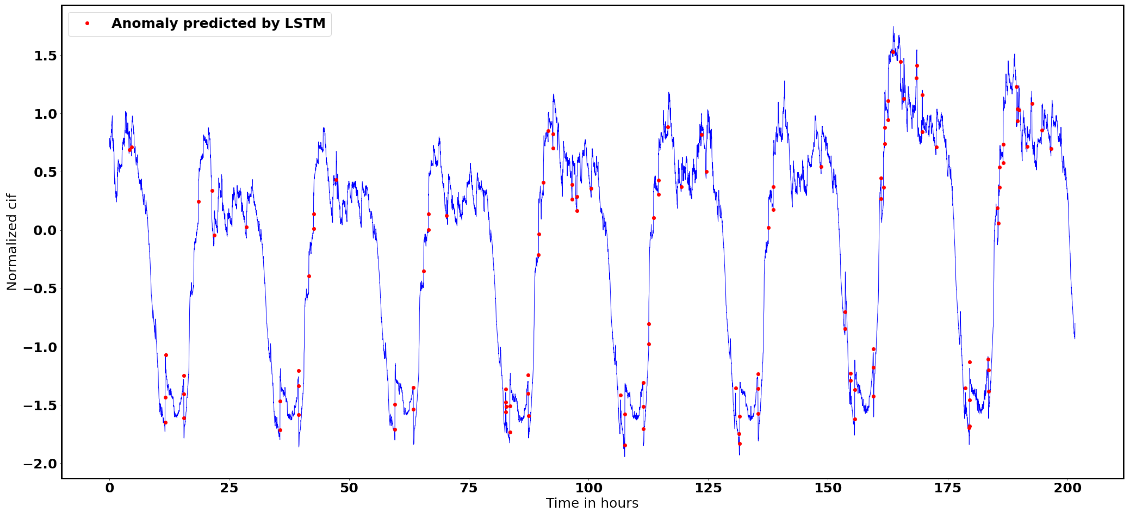
```
timeline = np.arange(0,len(y_test),1)

plt.plot(timeline/60,y_test,"k",label="data",alpha=1,linewidth=1)
plt.plot(timeline/60,y_pre[:,0],"r",label="Predicted",alpha=1,linewidth=1)

plt.xlabel("Time in hours")
plt.ylabel("Normalized %s"%names_array[index_name])

plt.legend()

fig = matplotlib.pyplot.gcf()


fig.set_size_inches(35,16)
save_path = plot_path + "LSTM_results_1D" + ".png"

fig.savefig(save_path, dpi=150)
```

```
findfont: Font family ['normal'] not found. Falling back to DejaVu Sans.
findfont: Font family ['normal'] not found. Falling back to DejaVu Sans.
```




```
In [33]: # Predict Anomaly using this rule: Bigger difference between data and prediction mean

         diff = y_test-y_pre[:,0]
         anomaly_ratio = 0.01
         mask = abs(diff)>np.nanpercentile(abs(diff),100-100*anomaly_ratio)
```

```
font = {'family': 'normal','weight': 'bold',
        'size': 25}

matplotlib.rc('font', **font)
rc('axes', linewidth=3)


timeline = np.arange(0,len(y_test),1)

plt.plot(timeline/60,y_test,"b",alpha=1,linewidth=1)
plt.plot(timeline[mask]/60,y_test[mask],"ro",label="Anomaly predicted by LSTM",alpha=

plt.xlabel("Time in hours")
plt.ylabel("Normalized %s"%names_array[index_name])

plt.legend()

fig = matplotlib.pyplot.gcf()


fig.set_size_inches(35,16)
save_path = plot_path + "LSTM_anomaly_prediction_1D" + ".png"

fig.savefig(save_path, dpi=150)
```



4 1D CNN:

Baseline

In [49]:
```
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Dropout
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D

df["minutes"]=df["time_stamp"].dt.hour*1440+df["time_stamp"].dt.hour*60+df["time_stam
# hyper-parameters:
# delta_t in minute,try a day first,output 5 dimensions
delta_t = 1440
n_epoch=50
index_name= 0
rate_dropout=0.3
# predict 1 minute for now
N_output=1

checkpoint_path = "CNN_1D/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)
```

In [50]:
```
min_max_scaler = preprocessing.StandardScaler()

# min-max scaler
np_scaled = min_max_scaler.fit_transform(df[names_array])

df_scaled = pd.DataFrame(np_scaled,columns=names_array)


X = np.zeros((df_scaled.shape[0]-delta_t,delta_t,1),dtype=float)
y = df_scaled[names_array[index_name]][delta_t:]

for i in range(len(y)):
    if i%10000==0:
        print("Prepare data %.2f percent"%(100*i/len(y)))
    X[i,:,:] = np.atleast_2d(df_scaled[i:i+delta_t][names_array[index_name]].values).
```

```
Prepare data 0.00 percent
Prepare data 24.80 percent
Prepare data 49.60 percent
Prepare data 74.40 percent
Prepare data 99.20 percent
```

In [51]:
```
# split train test:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=True

# design network: A toy 1D CNN model with a few FC layers
model = tf.keras.Sequential()
model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(X_train.s
```

```
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
    model.add(Dropout(rate_dropout))
    model.add(MaxPooling1D(pool_size=2))

    model.add(Flatten())
    # Add a few FC layer
    model.add(Dense(100, activation='relu'))
    model.add(Dense(N_output))

    model.compile(loss='mae', optimizer='adam')

    model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_6 (Conv1D)            (None, 1438, 64)          256
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 1436, 64)          12352
_________________________________________________________________
dropout_3 (Dropout)          (None, 1436, 64)          0
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 718, 64)           0
_________________________________________________________________
flatten_3 (Flatten)          (None, 45952)             0
_________________________________________________________________
dense_7 (Dense)              (None, 100)               4595300
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 101
=================================================================
Total params: 4,608,009
Trainable params: 4,608,009
Non-trainable params: 0
_________________________________________________________________
```

```
In [52]: callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_path,
                                                          save_weights_only=True,
                                                          verbose=1)

         history = model.fit(X_train, y_train, epochs=n_epoch, batch_size=32, validation_data=
```

```
Epoch 1/50
881/882 [============================>.] - ETA: 0s - loss: 0.1248
Epoch 00001: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.1247 - val_loss: 0.1891
Epoch 2/50
```

```
877/882 [============================>.] - ETA: 0s - loss: 0.0713
Epoch 00002: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 10s 12ms/step - loss: 0.0712 - val_loss: 0.1999
Epoch 3/50
877/882 [============================>.] - ETA: 0s - loss: 0.0617
Epoch 00003: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 10s 12ms/step - loss: 0.0617 - val_loss: 0.1613
Epoch 4/50
877/882 [============================>.] - ETA: 0s - loss: 0.0583
Epoch 00004: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 10s 12ms/step - loss: 0.0583 - val_loss: 0.1879
Epoch 5/50
877/882 [============================>.] - ETA: 0s - loss: 0.0539
Epoch 00005: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0538 - val_loss: 0.1847
Epoch 6/50
877/882 [============================>.] - ETA: 0s - loss: 0.0505
Epoch 00006: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 10s 12ms/step - loss: 0.0504 - val_loss: 0.1756
Epoch 7/50
877/882 [============================>.] - ETA: 0s - loss: 0.0482
Epoch 00007: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0482 - val_loss: 0.1688
Epoch 8/50
877/882 [============================>.] - ETA: 0s - loss: 0.0455
Epoch 00008: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0454 - val_loss: 0.1998
Epoch 9/50
877/882 [============================>.] - ETA: 0s - loss: 0.0438
Epoch 00009: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0438 - val_loss: 0.2012
Epoch 10/50
877/882 [============================>.] - ETA: 0s - loss: 0.0437
Epoch 00010: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0438 - val_loss: 0.1470
Epoch 11/50
877/882 [============================>.] - ETA: 0s - loss: 0.0415
Epoch 00011: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0415 - val_loss: 0.1701
Epoch 12/50
877/882 [============================>.] - ETA: 0s - loss: 0.0399
Epoch 00012: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0399 - val_loss: 0.1870
Epoch 13/50
877/882 [============================>.] - ETA: 0s - loss: 0.0400
Epoch 00013: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0400 - val_loss: 0.1553
Epoch 14/50
```

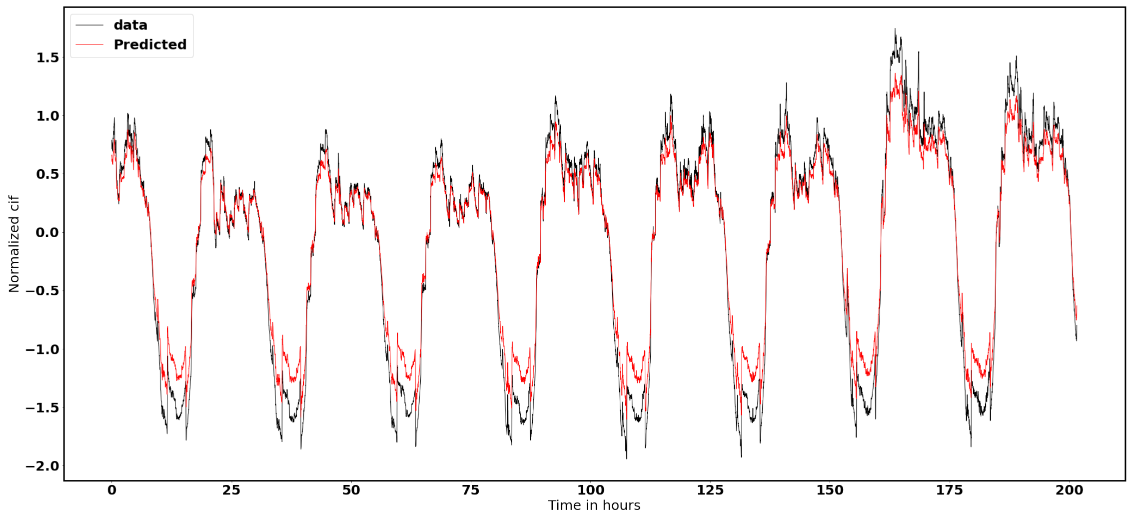
```
877/882 [============================>.] - ETA: 0s - loss: 0.0388
Epoch 00014: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0388 - val_loss: 0.1975
Epoch 15/50
877/882 [============================>.] - ETA: 0s - loss: 0.0393
Epoch 00015: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0392 - val_loss: 0.1818
Epoch 16/50
882/882 [==============================] - ETA: 0s - loss: 0.0379
Epoch 00016: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0379 - val_loss: 0.1900
Epoch 17/50
877/882 [============================>.] - ETA: 0s - loss: 0.0363
Epoch 00017: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0363 - val_loss: 0.1917
Epoch 18/50
877/882 [============================>.] - ETA: 0s - loss: 0.0372
Epoch 00018: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0371 - val_loss: 0.1821
Epoch 19/50
877/882 [============================>.] - ETA: 0s - loss: 0.0361
Epoch 00019: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0361 - val_loss: 0.1890
Epoch 20/50
877/882 [============================>.] - ETA: 0s - loss: 0.0353
Epoch 00020: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 10s 12ms/step - loss: 0.0353 - val_loss: 0.1731
Epoch 21/50
877/882 [============================>.] - ETA: 0s - loss: 0.0359
Epoch 00021: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 10s 12ms/step - loss: 0.0358 - val_loss: 0.1775
Epoch 22/50
877/882 [============================>.] - ETA: 0s - loss: 0.0354
Epoch 00022: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0354 - val_loss: 0.1838
Epoch 23/50
877/882 [============================>.] - ETA: 0s - loss: 0.0343
Epoch 00023: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0343 - val_loss: 0.1755
Epoch 24/50
877/882 [============================>.] - ETA: 0s - loss: 0.0339
Epoch 00024: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0338 - val_loss: 0.1695
Epoch 25/50
877/882 [============================>.] - ETA: 0s - loss: 0.0352
Epoch 00025: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0351 - val_loss: 0.1820
Epoch 26/50
```

```
877/882 [============================>.] - ETA: 0s - loss: 0.0346
Epoch 00026: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 10s 12ms/step - loss: 0.0346 - val_loss: 0.1789
Epoch 27/50
877/882 [============================>.] - ETA: 0s - loss: 0.0342
Epoch 00027: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0342 - val_loss: 0.1736
Epoch 28/50
877/882 [============================>.] - ETA: 0s - loss: 0.0335
Epoch 00028: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0335 - val_loss: 0.1778
Epoch 29/50
877/882 [============================>.] - ETA: 0s - loss: 0.0331
Epoch 00029: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0331 - val_loss: 0.1952
Epoch 30/50
877/882 [============================>.] - ETA: 0s - loss: 0.0330
Epoch 00030: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0330 - val_loss: 0.1972
Epoch 31/50
877/882 [============================>.] - ETA: 0s - loss: 0.0326
Epoch 00031: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0326 - val_loss: 0.1786
Epoch 32/50
877/882 [============================>.] - ETA: 0s - loss: 0.0342
Epoch 00032: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0342 - val_loss: 0.2061
Epoch 33/50
877/882 [============================>.] - ETA: 0s - loss: 0.0326
Epoch 00033: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0326 - val_loss: 0.1992
Epoch 34/50
877/882 [============================>.] - ETA: 0s - loss: 0.0322
Epoch 00034: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0322 - val_loss: 0.1856
Epoch 35/50
877/882 [============================>.] - ETA: 0s - loss: 0.0316
Epoch 00035: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0316 - val_loss: 0.1727
Epoch 36/50
877/882 [============================>.] - ETA: 0s - loss: 0.0307
Epoch 00036: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0307 - val_loss: 0.1884
Epoch 37/50
877/882 [============================>.] - ETA: 0s - loss: 0.0312
Epoch 00037: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0313 - val_loss: 0.1875
Epoch 38/50
```

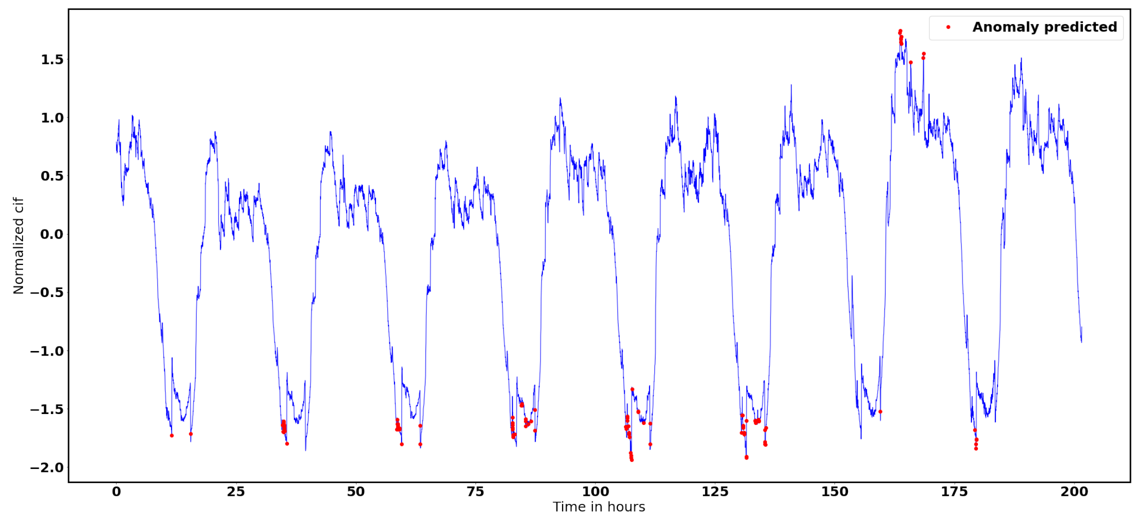
```
877/882 [============================>.] - ETA: 0s - loss: 0.0317
Epoch 00038: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 10s 12ms/step - loss: 0.0317 - val_loss: 0.1787
Epoch 39/50
877/882 [============================>.] - ETA: 0s - loss: 0.0308
Epoch 00039: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0308 - val_loss: 0.1928
Epoch 40/50
877/882 [============================>.] - ETA: 0s - loss: 0.0313
Epoch 00040: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 10s 12ms/step - loss: 0.0313 - val_loss: 0.2028
Epoch 41/50
877/882 [============================>.] - ETA: 0s - loss: 0.0318
Epoch 00041: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0318 - val_loss: 0.1855
Epoch 42/50
877/882 [============================>.] - ETA: 0s - loss: 0.0304
Epoch 00042: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0304 - val_loss: 0.1840
Epoch 43/50
877/882 [============================>.] - ETA: 0s - loss: 0.0308
Epoch 00043: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 10s 12ms/step - loss: 0.0308 - val_loss: 0.1885
Epoch 44/50
877/882 [============================>.] - ETA: 0s - loss: 0.0300
Epoch 00044: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 10s 12ms/step - loss: 0.0300 - val_loss: 0.1872
Epoch 45/50
877/882 [============================>.] - ETA: 0s - loss: 0.0292
Epoch 00045: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0292 - val_loss: 0.1822
Epoch 46/50
877/882 [============================>.] - ETA: 0s - loss: 0.0297
Epoch 00046: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 10s 12ms/step - loss: 0.0297 - val_loss: 0.1797
Epoch 47/50
877/882 [============================>.] - ETA: 0s - loss: 0.0293
Epoch 00047: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 10s 12ms/step - loss: 0.0293 - val_loss: 0.1903
Epoch 48/50
877/882 [============================>.] - ETA: 0s - loss: 0.0301
Epoch 00048: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0301 - val_loss: 0.1777
Epoch 49/50
877/882 [============================>.] - ETA: 0s - loss: 0.0300
Epoch 00049: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0300 - val_loss: 0.1889
Epoch 50/50
```

```
877/882 [============================>.] - ETA: 0s - loss: 0.0299
Epoch 00050: saving model to CNN_1D/cp.ckpt
882/882 [==============================] - 11s 12ms/step - loss: 0.0299 - val_loss: 0.1682
```

```
In [53]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=False
         y_pre = model.predict(X_test)

In [54]: font = {'family': 'normal','weight': 'bold',
                 'size': 25}

         matplotlib.rc('font', **font)
         rc('axes', linewidth=3)


         timeline = np.arange(0,len(y_test),1)

         plt.plot(timeline/60,y_test,"k",label="data",alpha=1,linewidth=1)
         plt.plot(timeline/60,y_pre[:,0],"r",label="Predicted",alpha=1,linewidth=1)

         plt.xlabel("Time in hours")
         plt.ylabel("Normalized %s"%names_array[index_name])

         plt.legend()

         fig = matplotlib.pyplot.gcf()


         fig.set_size_inches(35,16)
         save_path = plot_path + "CNN_1D_results" + ".png"

         fig.savefig(save_path, dpi=150)
```



In [56]:
```
# Predict Anomaly using this rule: Bigger difference between data and prediction mean

diff = y_test-y_pre[:,0]
anomaly_ratio = 0.01
mask = abs(diff)>np.nanpercentile(abs(diff),100-100*anomaly_ratio)



font = {'family': 'normal','weight': 'bold',
        'size': 25}

matplotlib.rc('font', **font)
rc('axes', linewidth=3)


timeline = np.arange(0,len(y_test),1)

plt.plot(timeline/60,y_test,"b",alpha=1,linewidth=1)
plt.plot(timeline[mask]/60,y_test[mask],"ro",label="Anomaly predicted CNN",alpha=1,lin

plt.xlabel("Time in hours")
plt.ylabel("Normalized %s"%names_array[index_name])

plt.legend()

fig = matplotlib.pyplot.gcf()


fig.set_size_inches(35,16)
save_path = plot_path + "CNN_1D_anomaly_prediction" + ".png"

fig.savefig(save_path, dpi=150)
```



4.1 LSTM with high low open close:

Idea from this: Stock Market Forecasting Using Hidden Markov Model: A New Approach

In [ ]:

In [91]:
```
df["minutes"]=df["time_stamp"].dt.hour*1440+df["time_stamp"].dt.hour*60+df["time_stam
# hyper-parameters:
# delta_t in minute,try a day first,output 5 dimensions
delta_t = 1440
n_epoch=10
n_cell = 50
# predict 1 minute for now
N_output=1
index_name= 0

checkpoint_path = "LSTM/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)

min_max_scaler = preprocessing.StandardScaler()

name_mod = [names_array[index_name],names_array[index_name]+"_open",names_array[index_

np_scaled = min_max_scaler.fit_transform(df[name_mod])

df_scaled = pd.DataFrame(np_scaled,columns=name_mod)


X = np.zeros((df_scaled.shape[0]-delta_t,delta_t,5),dtype=float)
y = df_scaled[names_array[index_name]][delta_t:]
```

```
for i in range(len(y)):
    if i%10000==0:
        print("Prepare data %.2f percent"%(100*i/len(y)))
    X[i,:,:] = df_scaled[i:i+delta_t][name_mod].values

# split train test:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=True

model = tf.keras.Sequential([
  tf.keras.layers.LSTM(n_cell,input_shape=(X_train.shape[1],X_train.shape[2])),  # mu
  tf.keras.layers.Dense(1)
])

model.compile(loss='mae', optimizer='adam')
#model.summary()

callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_path,
                                              save_weights_only=True,
                                              verbose=1)
```

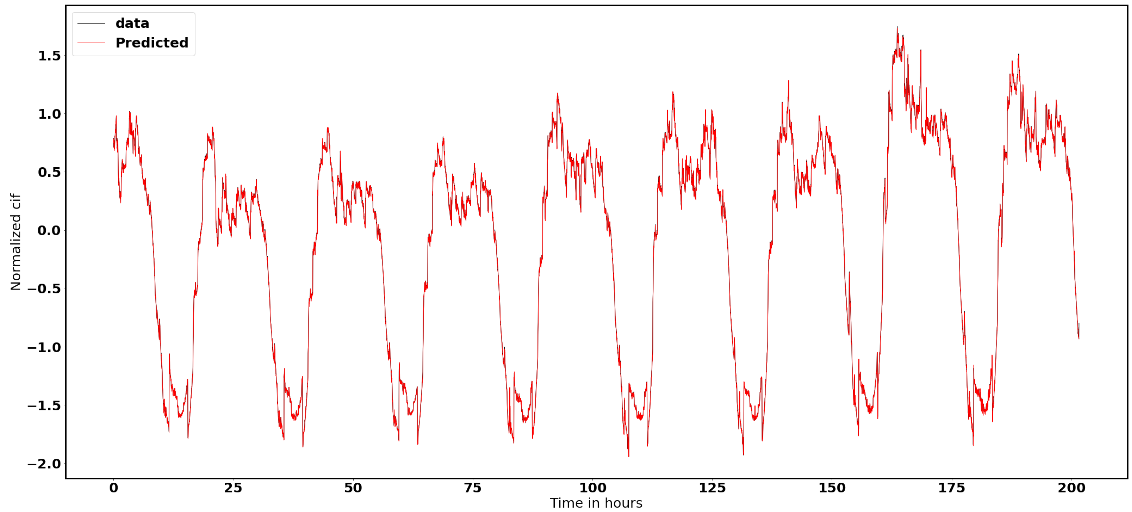
In [93]:
```
history = model.fit(X_train, y_train, epochs=n_epoch, batch_size=64, validation_data=
```

```
Epoch 1/10
441/441 [==============================] - ETA: 0s - loss: 0.0599
Epoch 00001: saving model to LSTM/cp.ckpt
441/441 [==============================] - 48s 109ms/step - loss: 0.0599 - val_loss: 0.0329
Epoch 2/10
441/441 [==============================] - ETA: 0s - loss: 0.0298
Epoch 00002: saving model to LSTM/cp.ckpt
441/441 [==============================] - 49s 110ms/step - loss: 0.0298 - val_loss: 0.0263
Epoch 3/10
441/441 [==============================] - ETA: 0s - loss: 0.0257
Epoch 00003: saving model to LSTM/cp.ckpt
441/441 [==============================] - 46s 105ms/step - loss: 0.0257 - val_loss: 0.0242
Epoch 4/10
441/441 [==============================] - ETA: 0s - loss: 0.0232
Epoch 00004: saving model to LSTM/cp.ckpt
441/441 [==============================] - 47s 106ms/step - loss: 0.0232 - val_loss: 0.0230
Epoch 5/10
441/441 [==============================] - ETA: 0s - loss: 0.0209
Epoch 00005: saving model to LSTM/cp.ckpt
441/441 [==============================] - 47s 107ms/step - loss: 0.0209 - val_loss: 0.0205
Epoch 6/10
441/441 [==============================] - ETA: 0s - loss: 0.0195
Epoch 00006: saving model to LSTM/cp.ckpt
441/441 [==============================] - 48s 108ms/step - loss: 0.0195 - val_loss: 0.0218
Epoch 7/10
```

```
441/441 [==============================] - ETA: 0s - loss: 0.0186
Epoch 00007: saving model to LSTM/cp.ckpt
441/441 [==============================] - 47s 106ms/step - loss: 0.0186 - val_loss: 0.0172
Epoch 8/10
441/441 [==============================] - ETA: 0s - loss: 0.0181
Epoch 00008: saving model to LSTM/cp.ckpt
441/441 [==============================] - 47s 107ms/step - loss: 0.0181 - val_loss: 0.0175
Epoch 9/10
441/441 [==============================] - ETA: 0s - loss: 0.0176
Epoch 00009: saving model to LSTM/cp.ckpt
441/441 [==============================] - 47s 107ms/step - loss: 0.0176 - val_loss: 0.0170
Epoch 10/10
441/441 [==============================] - ETA: 0s - loss: 0.0175
Epoch 00010: saving model to LSTM/cp.ckpt
441/441 [==============================] - 46s 105ms/step - loss: 0.0175 - val_loss: 0.0169
```

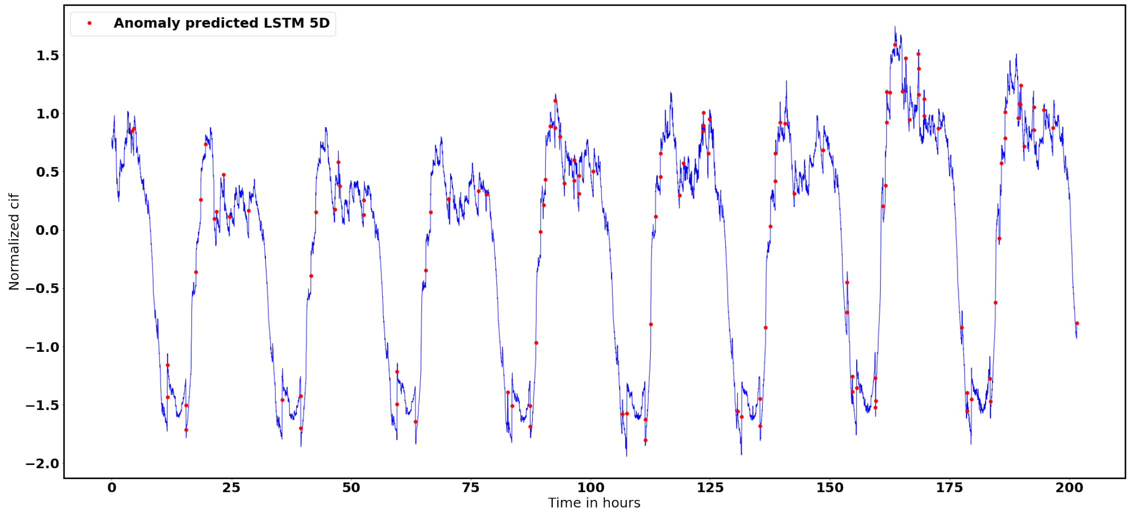
```
In [94]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=False
         y_pre = model.predict(X_test)
         font = {'family': 'normal','weight': 'bold',
                 'size': 25}

         matplotlib.rc('font', **font)
         rc('axes', linewidth=3)


         timeline = np.arange(0,len(y_test),1)

         plt.plot(timeline/60,y_test,"k",label="data",alpha=1,linewidth=1)
         plt.plot(timeline/60,y_pre[:,0],"r",label="Predicted",alpha=1,linewidth=1)

         plt.xlabel("Time in hours")
         plt.ylabel("Normalized %s"%names_array[index_name])

         plt.legend()

         fig = matplotlib.pyplot.gcf()


         fig.set_size_inches(35,16)
         save_path = plot_path + "LSTM_results_5D" + ".png"

         fig.savefig(save_path, dpi=150)
```

```
findfont: Font family ['normal'] not found. Falling back to DejaVu Sans.
findfont: Font family ['normal'] not found. Falling back to DejaVu Sans.
```




```
In [95]: # Predict Anomaly using this rule: Bigger difference between data and prediction mean

         diff = y_test-y_pre[:,0]
         anomaly_ratio = 0.01
         mask = abs(diff)>np.nanpercentile(abs(diff),100-100*anomaly_ratio)



         font = {'family': 'normal','weight': 'bold',
                 'size': 25}

         matplotlib.rc('font', **font)
         rc('axes', linewidth=3)


         timeline = np.arange(0,len(y_test),1)

         plt.plot(timeline/60,y_test,"b",alpha=1,linewidth=1)
         plt.plot(timeline[mask]/60,y_test[mask],"ro",label="Anomaly predicted LSTM 5D",alpha=

         plt.xlabel("Time in hours")
         plt.ylabel("Normalized %s"%names_array[index_name])

         plt.legend()

         fig = matplotlib.pyplot.gcf()


         fig.set_size_inches(35,16)
         save_path = plot_path + "LSTM_anomaly_prediction_5D" + ".png"
```

```
fig.savefig(save_path, dpi=150)
```




```
In [57]: # GRU simple
         df["minutes"]=df["time_stamp"].dt.hour*1440+df["time_stamp"].dt.hour*60+df["time_stam
         # hyper-parameters:
         # delta_t in minute,try a day first,output 5 dimensions
         delta_t = 1440
         n_epoch=10
         n_cell = 50
         # predict 1 minute for now
         N_output=1
         index_name= 0

         checkpoint_path = "LSTM/cp.ckpt"
         checkpoint_dir = os.path.dirname(checkpoint_path)

         min_max_scaler = preprocessing.StandardScaler()

         # min-max scaler
         np_scaled = min_max_scaler.fit_transform(df[names_array])

         df_scaled = pd.DataFrame(np_scaled,columns=names_array)


         X = np.zeros((df_scaled.shape[0]-delta_t,delta_t,1),dtype=float)
         y = df_scaled[names_array[index_name]][delta_t:]

         for i in range(len(y)):
             if i%10000==0:
```

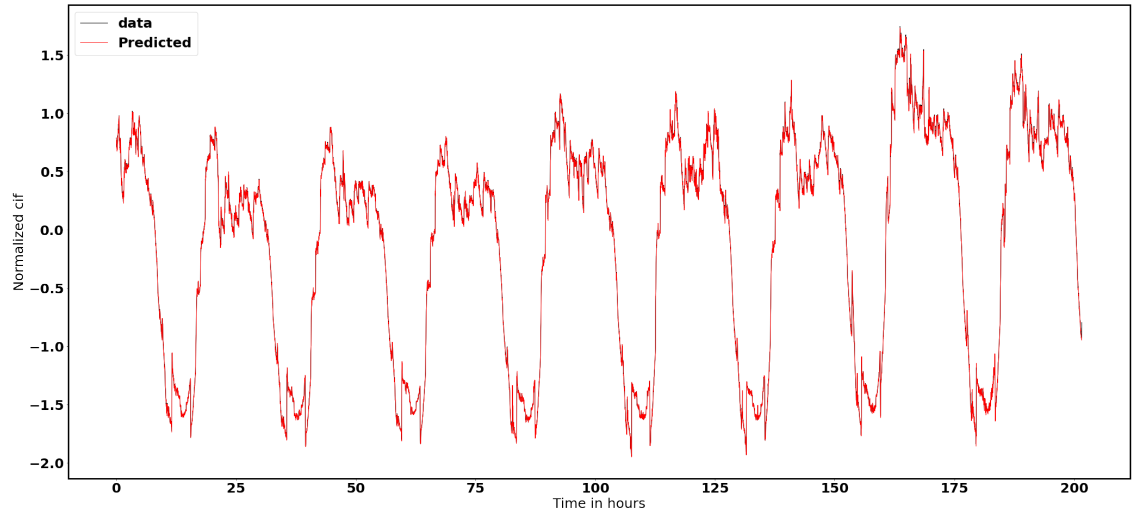
```
                print("Prepare data %.2f percent"%(100*i/len(y)))
            X[i,:,:] = np.atleast_2d(df_scaled[i:i+delta_t][names_array[index_name]].values).T

        # split train test:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=True)

        model = tf.keras.Sequential([
          tf.keras.layers.GRU(n_cell,input_shape=(X_train.shape[1],X_train.shape[2])),  # mus
          tf.keras.layers.Dense(1)
        ])

        model.compile(loss='mae', optimizer='adam')
        #model.summary()

        callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_path,
                                                      save_weights_only=True,
                                                      verbose=1)



        history = model.fit(X_train, y_train, epochs=n_epoch, batch_size=64, validation_data=
```

```
Prepare data 0.00 percent
Prepare data 24.80 percent
Prepare data 49.60 percent
Prepare data 74.40 percent
Prepare data 99.20 percent
Epoch 1/10
441/441 [==============================] - ETA: 0s - loss: 0.0762
Epoch 00001: saving model to LSTM/cp.ckpt
441/441 [==============================] - 48s 109ms/step - loss: 0.0762 - val_loss: 0.0258
Epoch 2/10
441/441 [==============================] - ETA: 0s - loss: 0.0231
Epoch 00002: saving model to LSTM/cp.ckpt
441/441 [==============================] - 49s 110ms/step - loss: 0.0231 - val_loss: 0.0210
Epoch 3/10
441/441 [==============================] - ETA: 0s - loss: 0.0202
Epoch 00003: saving model to LSTM/cp.ckpt
441/441 [==============================] - 49s 111ms/step - loss: 0.0202 - val_loss: 0.0187
Epoch 4/10
441/441 [==============================] - ETA: 0s - loss: 0.0189
Epoch 00004: saving model to LSTM/cp.ckpt
441/441 [==============================] - 48s 108ms/step - loss: 0.0189 - val_loss: 0.0185
Epoch 5/10
441/441 [==============================] - ETA: 0s - loss: 0.0181
Epoch 00005: saving model to LSTM/cp.ckpt
441/441 [==============================] - 48s 110ms/step - loss: 0.0181 - val_loss: 0.0176
Epoch 6/10
```

```
441/441 [==============================] - ETA: 0s - loss: 0.0178
Epoch 00006: saving model to LSTM/cp.ckpt
441/441 [==============================] - 48s 110ms/step - loss: 0.0178 - val_loss: 0.0172
Epoch 7/10
441/441 [==============================] - ETA: 0s - loss: 0.0176
Epoch 00007: saving model to LSTM/cp.ckpt
441/441 [==============================] - 48s 109ms/step - loss: 0.0176 - val_loss: 0.0177
Epoch 8/10
441/441 [==============================] - ETA: 0s - loss: 0.0177
Epoch 00008: saving model to LSTM/cp.ckpt
441/441 [==============================] - 47s 106ms/step - loss: 0.0177 - val_loss: 0.0178
Epoch 9/10
441/441 [==============================] - ETA: 0s - loss: 0.0176
Epoch 00009: saving model to LSTM/cp.ckpt
441/441 [==============================] - 47s 107ms/step - loss: 0.0176 - val_loss: 0.0178
Epoch 10/10
441/441 [==============================] - ETA: 0s - loss: 0.0175
Epoch 00010: saving model to LSTM/cp.ckpt
441/441 [==============================] - 46s 104ms/step - loss: 0.0175 - val_loss: 0.0174
```

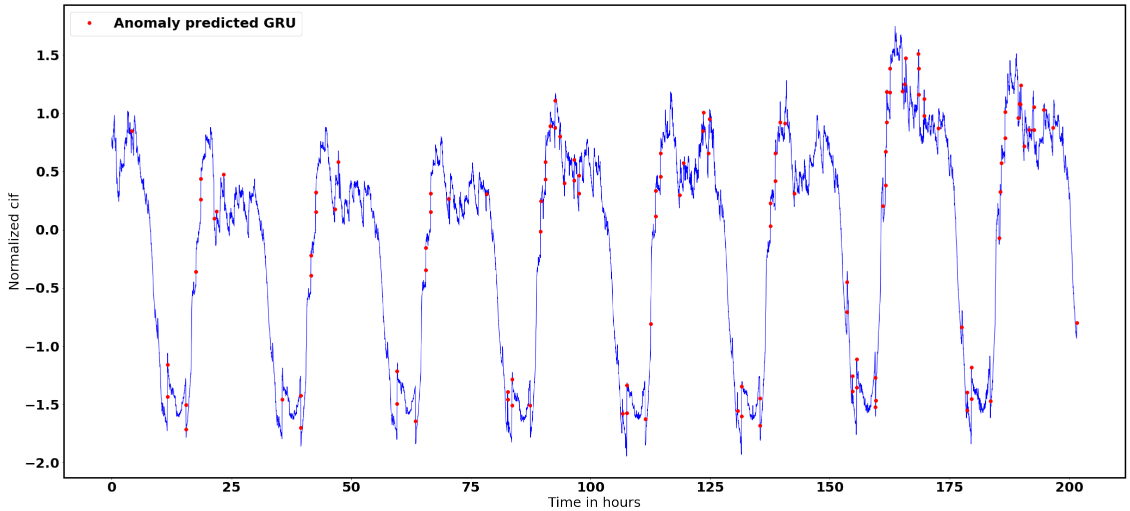
In [58]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=False
y_pre = model.predict(X_test)
font = {'family': 'normal','weight': 'bold',
        'size': 25}

matplotlib.rc('font', **font)
rc('axes', linewidth=3)


timeline = np.arange(0,len(y_test),1)

plt.plot(timeline/60,y_test,"k",label="data",alpha=1,linewidth=1)
plt.plot(timeline/60,y_pre[:,0],"r",label="Predicted",alpha=1,linewidth=1)

plt.xlabel("Time in hours")
plt.ylabel("Normalized %s"%names_array[index_name])

plt.legend()

fig = matplotlib.pyplot.gcf()


fig.set_size_inches(35,16)
save_path = plot_path + "GRU_results_1D" + ".png"

fig.savefig(save_path, dpi=150)
```



In [60]:
```
# Predict Anomaly using this rule: Bigger difference between data and prediction mean

diff = y_test-y_pre[:,0]
anomaly_ratio = 0.01
mask = abs(diff)>np.nanpercentile(abs(diff),100-100*anomaly_ratio)



font = {'family': 'normal','weight': 'bold',
        'size': 25}

matplotlib.rc('font', **font)
rc('axes', linewidth=3)


timeline = np.arange(0,len(y_test),1)

plt.plot(timeline/60,y_test,"b",alpha=1,linewidth=1)
plt.plot(timeline[mask]/60,y_test[mask],"ro",label="Anomaly predicted GRU",alpha=1,li

plt.xlabel("Time in hours")
plt.ylabel("Normalized %s"%names_array[index_name])

plt.legend()

fig = matplotlib.pyplot.gcf()


fig.set_size_inches(35,16)
save_path = plot_path + "GRU_anomaly_prediction_1D" + ".png"
```

```
fig.savefig(save_path, dpi=150)
```



In [98]:
```
#  GRU+ 1D CNN:
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Dropout
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
df["minutes"]=df["time_stamp"].dt.hour*1440+df["time_stamp"].dt.hour*60+df["time_stam
# hyper-parameters:
# delta_t in minute,try a day first,output 5 dimensions
delta_t = 1440
n_epoch=10
n_cell = 50
# predict 1 minute for now
N_output=1
index_name= 0
rate_dropout = 0.2

checkpoint_path = "LSTM/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)

min_max_scaler = preprocessing.StandardScaler()

# min-max scaler
np_scaled = min_max_scaler.fit_transform(df[names_array])

df_scaled = pd.DataFrame(np_scaled,columns=names_array)
```

```
    X = np.zeros((df_scaled.shape[0]-delta_t,delta_t,1),dtype=float)
    y = df_scaled[names_array[index_name]][delta_t:]

    for i in range(len(y)):
        if i%10000==0:
            print("Prepare data %.2f percent"%(100*i/len(y)))
        X[i,:,:] = np.atleast_2d(df_scaled[i:i+delta_t][names_array[index_name]].values).T

    # split train test:
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=True)
```

```
Prepare data 0.00 percent
Prepare data 24.80 percent
Prepare data 49.60 percent
Prepare data 74.40 percent
Prepare data 99.20 percent
```

```
In [125]: model = tf.keras.Sequential()
          model.add(tf.keras.layers.GRU(n_cell,input_shape=(X_train.shape[1],X_train.shape[2])))
          model.add(tf.keras.layers.Dense(100))
          model.add(Dropout(rate_dropout))
          model.add(tf.keras.layers.Dense(1))
          model.compile(loss='mae', optimizer='adam')
          model.summary()
```

```
Model: "sequential_26"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru_5 (GRU)                  (None, 50)                7950
_________________________________________________________________
dense_9 (Dense)              (None, 100)               5100
_________________________________________________________________
dropout_3 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 101
=================================================================
Total params: 13,151
Trainable params: 13,151
Non-trainable params: 0
_________________________________________________________________
```

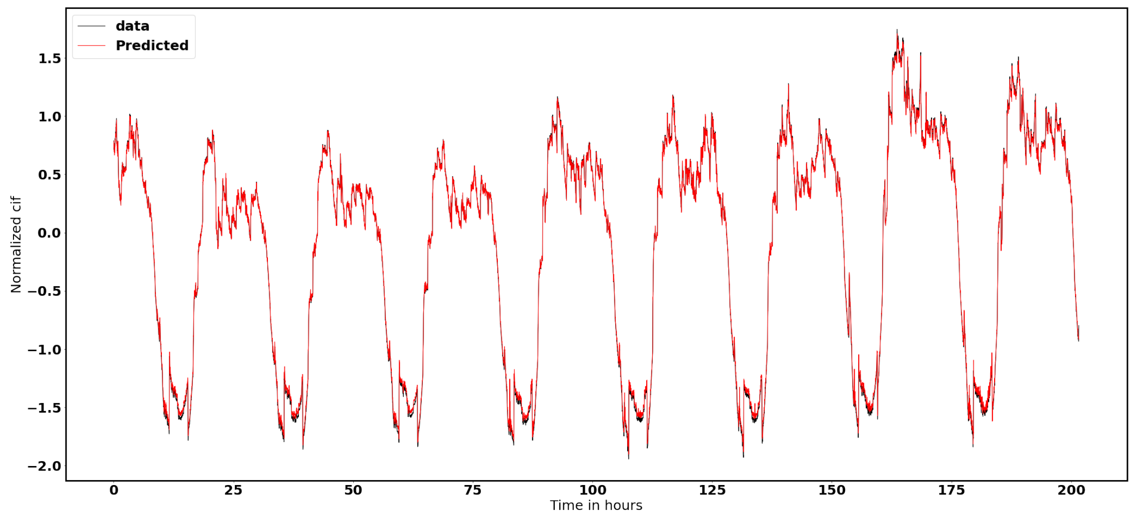
```
In [126]: callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_path,
                                                         save_weights_only=True,
                                                         verbose=1)
```

```
history = model.fit(X_train, y_train, epochs=n_epoch, batch_size=64, validation_data=
```

```
Epoch 1/10
441/441 [==============================] - ETA: 0s - loss: 0.0688
Epoch 00001: saving model to LSTM/cp.ckpt
441/441 [==============================] - 48s 108ms/step - loss: 0.0688 - val_loss: 0.0200
Epoch 2/10
441/441 [==============================] - ETA: 0s - loss: 0.0457
Epoch 00002: saving model to LSTM/cp.ckpt
441/441 [==============================] - 48s 108ms/step - loss: 0.0457 - val_loss: 0.0257
Epoch 3/10
441/441 [==============================] - ETA: 0s - loss: 0.0442
Epoch 00003: saving model to LSTM/cp.ckpt
441/441 [==============================] - 48s 109ms/step - loss: 0.0442 - val_loss: 0.0208
Epoch 4/10
441/441 [==============================] - ETA: 0s - loss: 0.0443
Epoch 00004: saving model to LSTM/cp.ckpt
441/441 [==============================] - 46s 104ms/step - loss: 0.0443 - val_loss: 0.0198
Epoch 5/10
441/441 [==============================] - ETA: 0s - loss: 0.0438
Epoch 00005: saving model to LSTM/cp.ckpt
441/441 [==============================] - 48s 109ms/step - loss: 0.0438 - val_loss: 0.0287
Epoch 6/10
441/441 [==============================] - ETA: 0s - loss: 0.0442
Epoch 00006: saving model to LSTM/cp.ckpt
441/441 [==============================] - 48s 109ms/step - loss: 0.0442 - val_loss: 0.0295
Epoch 7/10
441/441 [==============================] - ETA: 0s - loss: 0.0437
Epoch 00007: saving model to LSTM/cp.ckpt
441/441 [==============================] - 46s 105ms/step - loss: 0.0437 - val_loss: 0.0205
Epoch 8/10
441/441 [==============================] - ETA: 0s - loss: 0.0439
Epoch 00008: saving model to LSTM/cp.ckpt
441/441 [==============================] - 46s 105ms/step - loss: 0.0439 - val_loss: 0.0222
Epoch 9/10
441/441 [==============================] - ETA: 0s - loss: 0.0430
Epoch 00009: saving model to LSTM/cp.ckpt
441/441 [==============================] - 47s 107ms/step - loss: 0.0430 - val_loss: 0.0256
Epoch 10/10
441/441 [==============================] - ETA: 0s - loss: 0.0434
Epoch 00010: saving model to LSTM/cp.ckpt
441/441 [==============================] - 48s 108ms/step - loss: 0.0434 - val_loss: 0.0257
```

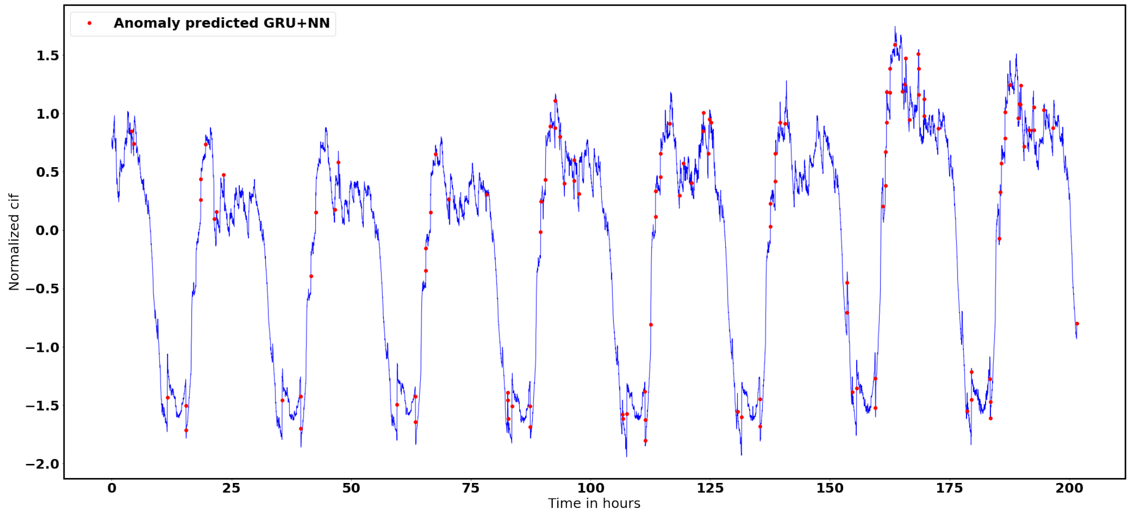
In [128]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=Fals
y_pre = model.predict(X_test)
```

```
font = {'family': 'normal','weight': 'bold',
        'size': 25}

matplotlib.rc('font', **font)
rc('axes', linewidth=3)


timeline = np.arange(0,len(y_test),1)

plt.plot(timeline/60,y_test,"k",label="data",alpha=1,linewidth=1)
plt.plot(timeline/60,y_pre[:,0],"r",label="Predicted",alpha=1,linewidth=1)

plt.xlabel("Time in hours")
plt.ylabel("Normalized %s"%names_array[index_name])

plt.legend()

fig = matplotlib.pyplot.gcf()


fig.set_size_inches(35,16)
save_path = plot_path + "GRU_NN_results_1D" + ".png"

fig.savefig(save_path, dpi=150)
```



In [129]:
```
# Predict Anomaly using this rule: Bigger difference between data and prediction mea

diff = y_test-y_pre[:,0]
anomaly_ratio = 0.01
mask = abs(diff)>np.nanpercentile(abs(diff),100-100*anomaly_ratio)
```

```
font = {'family': 'normal','weight': 'bold',
        'size': 25}

matplotlib.rc('font', **font)
rc('axes', linewidth=3)


timeline = np.arange(0,len(y_test),1)

plt.plot(timeline/60,y_test,"b",alpha=1,linewidth=1)
plt.plot(timeline[mask]/60,y_test[mask],"ro",label="Anomaly predicted GRU+NN",alpha=1

plt.xlabel("Time in hours")
plt.ylabel("Normalized %s"%names_array[index_name])

plt.legend()

fig = matplotlib.pyplot.gcf()


fig.set_size_inches(35,16)
save_path = plot_path + "GRU_NN_anomaly_prediction_1D" + ".png"

fig.savefig(save_path, dpi=150)
```



4.2 MLSTM-FCN:

https://arxiv.org/pdf/1709.05206.pdf

In [5]:
```
from keras.models import Model
from keras.layers import Input, Dense, LSTM, multiply, concatenate, Activation, Masking
from keras.layers import Conv1D, BatchNormalization, GlobalAveragePooling1D, Permute, I
```

In [ ]:
```
# hyper-parameters:
# delta_t in minute,try a day first,output 5 dimensions
delta_t = 1440
n_epoch=10
n_cell = 50
# predict 1 minute for now
N_output=1
N_input = 5
index_name= 0
```

In [41]:
```
NB_CLASS = 1

def squeeze_excite_block(input):

    filters = input._shape[-1] # channel_axis = -1 for TF

    se = GlobalAveragePooling1D()(input)
    se = Reshape((1, filters))(se)
    se = Dense(filters // 16,  activation='relu', kernel_initializer='he_normal', use_
    se = Dense(filters, activation='sigmoid', kernel_initializer='he_normal', use_bias
    se = multiply([input, se])
    return se

def generate_model(MAX_TIMESTEPS,MAX_NB_VARIABLES):
    ip = Input(shape=(MAX_TIMESTEPS,MAX_NB_VARIABLES))
    # split into x and y two channels

    x = Masking()(ip)
    x = LSTM(8)(x)
    x = Dropout(0.8)(x)

    y = Permute((2, 1))(ip)
    y = Conv1D(128, 8, padding='same', kernel_initializer='he_uniform')(y)
    y = BatchNormalization()(y)
    y = Activation('relu')(y)
    y = squeeze_excite_block(y)

    y = Conv1D(256, 5, padding='same', kernel_initializer='he_uniform')(y)
    y = BatchNormalization()(y)
    y = Activation('relu')(y)
    y = squeeze_excite_block(y)
```

```
    y = Conv1D(128, 3, padding='same', kernel_initializer='he_uniform')(y)
    y = BatchNormalization()(y)
    y = Activation('relu')(y)

    y = GlobalAveragePooling1D()(y)
    # combine
    x = concatenate([x, y])

    #out = Dense(NB_CLASS, activation='softmax')(x)
    # For regression model use MAE
    out = Dense(N_output)(x)

    model = Model(ip, out)
    model.summary()

    # add load model code here to fine-tune

    return model
```

In [42]:
```
#model = generate_model(1440,5)
```

In [ ]:

In [43]:
```
df["minutes"]=df["time_stamp"].dt.hour*1440+df["time_stamp"].dt.hour*60+df["time_stam


checkpoint_path = "LSTM/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)

min_max_scaler = preprocessing.StandardScaler()

name_mod = [names_array[index_name],names_array[index_name]+"_open",names_array[index_

np_scaled = min_max_scaler.fit_transform(df[name_mod])

df_scaled = pd.DataFrame(np_scaled,columns=name_mod)


X = np.zeros((df_scaled.shape[0]-delta_t,delta_t,5),dtype=float)
y = df_scaled[names_array[index_name]][delta_t:]

for i in range(len(y)):
    if i%10000==0:
        print("Prepare data %.2f percent"%(100*i/len(y)))
    X[i,:,:] = df_scaled[i:i+delta_t][name_mod].values

# split train test:
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=True

model = generate_model(1440,5)

model.compile(loss='mae', optimizer='adam')
#model.summary()

callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_path,
                                              save_weights_only=True,
                                              verbose=1)
```

```
Prepare data 0.00 percent
Prepare data 24.80 percent
Prepare data 49.60 percent
Prepare data 74.40 percent
Prepare data 99.20 percent
WARNING:tensorflow:Tensor._shape is private, use Tensor.shape instead. Tensor._shape will even
WARNING:tensorflow:Tensor._shape is private, use Tensor.shape instead. Tensor._shape will even
Model: "model_7"
______________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==============================================================================================
input_14 (InputLayer)           [(None, 1440, 5)]    0
______________________________________________________________________________________________
permute_13 (Permute)            (None, 5, 1440)      0           input_14[0][0]
______________________________________________________________________________________________
conv1d_29 (Conv1D)              (None, 5, 128)       1474688     permute_13[0][0]
______________________________________________________________________________________________
batch_normalization_29 (BatchNo (None, 5, 128)       512         conv1d_29[0][0]
______________________________________________________________________________________________
activation_29 (Activation)      (None, 5, 128)       0           batch_normalization_29[0][0]
______________________________________________________________________________________________
global_average_pooling1d_24 (Gl (None, 128)          0           activation_29[0][0]
______________________________________________________________________________________________
reshape_16 (Reshape)            (None, 1, 128)       0           global_average_pooling1d_24[0
______________________________________________________________________________________________
dense_39 (Dense)                (None, 1, 8)         1024        reshape_16[0][0]
______________________________________________________________________________________________
dense_40 (Dense)                (None, 1, 128)       1024        dense_39[0][0]
______________________________________________________________________________________________
multiply_16 (Multiply)          (None, 5, 128)       0           activation_29[0][0]
                                                                 dense_40[0][0]
______________________________________________________________________________________________
conv1d_30 (Conv1D)              (None, 5, 256)       164096      multiply_16[0][0]
______________________________________________________________________________________________
batch_normalization_30 (BatchNo (None, 5, 256)       1024        conv1d_30[0][0]
______________________________________________________________________________________________
activation_30 (Activation)      (None, 5, 256)       0           batch_normalization_30[0][0]
```

```
________________________________________________________________________________________________
global_average_pooling1d_25 (Gl (None, 256)          0           activation_30[0][0]
________________________________________________________________________________________________
reshape_17 (Reshape)            (None, 1, 256)       0           global_average_pooling1d_25[0
________________________________________________________________________________________________
dense_41 (Dense)                (None, 1, 16)        4096        reshape_17[0][0]
________________________________________________________________________________________________
dense_42 (Dense)                (None, 1, 256)       4096        dense_41[0][0]
________________________________________________________________________________________________
multiply_17 (Multiply)          (None, 5, 256)       0           activation_30[0][0]
                                                                 dense_42[0][0]
________________________________________________________________________________________________
conv1d_31 (Conv1D)              (None, 5, 128)       98432       multiply_17[0][0]
________________________________________________________________________________________________
masking_13 (Masking)            (None, 1440, 5)      0           input_14[0][0]
________________________________________________________________________________________________
batch_normalization_31 (BatchNo (None, 5, 128)       512         conv1d_31[0][0]
________________________________________________________________________________________________
lstm_13 (LSTM)                  (None, 8)            448         masking_13[0][0]
________________________________________________________________________________________________
activation_31 (Activation)      (None, 5, 128)       0           batch_normalization_31[0][0]
________________________________________________________________________________________________
dropout_13 (Dropout)            (None, 8)            0           lstm_13[0][0]
________________________________________________________________________________________________
global_average_pooling1d_26 (Gl (None, 128)          0           activation_31[0][0]
________________________________________________________________________________________________
concatenate_8 (Concatenate)     (None, 136)          0           dropout_13[0][0]
                                                                 global_average_pooling1d_26[0
________________________________________________________________________________________________
dense_43 (Dense)                (None, 1)            137         concatenate_8[0][0]
================================================================================================
Total params: 1,750,089
Trainable params: 1,749,065
Non-trainable params: 1,024
________________________________________________________________________________________________
```

```
In [44]: history = model.fit(X_train, y_train, epochs=n_epoch, batch_size=64, validation_data=
```

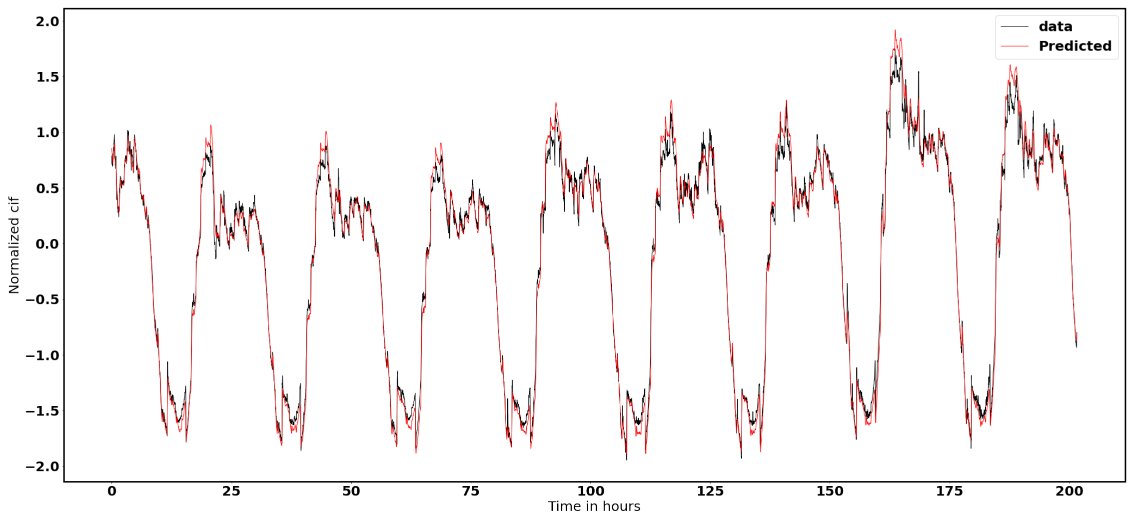
```
Epoch 1/10
441/441 [==============================] - ETA: 0s - loss: 0.1835
Epoch 00001: saving model to LSTM/cp.ckpt
441/441 [==============================] - 54s 123ms/step - loss: 0.1835 - val_loss: 0.1500
Epoch 2/10
441/441 [==============================] - ETA: 0s - loss: 0.1199
Epoch 00002: saving model to LSTM/cp.ckpt
441/441 [==============================] - 41s 94ms/step - loss: 0.1199 - val_loss: 0.1379
Epoch 3/10
```

```
441/441 [==============================] - ETA: 0s - loss: 0.1052
Epoch 00003: saving model to LSTM/cp.ckpt
441/441 [==============================] - 42s 94ms/step - loss: 0.1052 - val_loss: 0.1072
Epoch 4/10
441/441 [==============================] - ETA: 0s - loss: 0.0946
Epoch 00004: saving model to LSTM/cp.ckpt
441/441 [==============================] - 41s 93ms/step - loss: 0.0946 - val_loss: 0.0975
Epoch 5/10
441/441 [==============================] - ETA: 0s - loss: 0.0849
Epoch 00005: saving model to LSTM/cp.ckpt
441/441 [==============================] - 41s 93ms/step - loss: 0.0849 - val_loss: 0.1315
Epoch 6/10
441/441 [==============================] - ETA: 0s - loss: 0.0764
Epoch 00006: saving model to LSTM/cp.ckpt
441/441 [==============================] - 41s 93ms/step - loss: 0.0764 - val_loss: 0.0946
Epoch 7/10
441/441 [==============================] - ETA: 0s - loss: 0.0724
Epoch 00007: saving model to LSTM/cp.ckpt
441/441 [==============================] - 41s 93ms/step - loss: 0.0724 - val_loss: 0.0739
Epoch 8/10
441/441 [==============================] - ETA: 0s - loss: 0.0678
Epoch 00008: saving model to LSTM/cp.ckpt
441/441 [==============================] - 41s 93ms/step - loss: 0.0678 - val_loss: 0.0911
Epoch 9/10
441/441 [==============================] - ETA: 0s - loss: 0.0641
Epoch 00009: saving model to LSTM/cp.ckpt
441/441 [==============================] - 41s 94ms/step - loss: 0.0641 - val_loss: 0.0533
Epoch 10/10
441/441 [==============================] - ETA: 0s - loss: 0.0631
Epoch 00010: saving model to LSTM/cp.ckpt
441/441 [==============================] - 41s 94ms/step - loss: 0.0631 - val_loss: 0.0779
```

In [45]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=Fals
y_pre = model.predict(X_test)
font = {'family': 'normal','weight': 'bold',
        'size': 25}

matplotlib.rc('font', **font)
rc('axes', linewidth=3)


timeline = np.arange(0,len(y_test),1)

plt.plot(timeline/60,y_test,"k",label="data",alpha=1,linewidth=1)
plt.plot(timeline/60,y_pre[:,0],"r",label="Predicted",alpha=1,linewidth=1)

plt.xlabel("Time in hours")
```

```
    plt.ylabel("Normalized %s"%names_array[index_name])

    plt.legend()

    fig = matplotlib.pyplot.gcf()


    fig.set_size_inches(35,16)
    save_path = plot_path + "MLSTM_FCN_results_5D" + ".png"

    fig.savefig(save_path, dpi=150)
```

```
findfont: Font family ['normal'] not found. Falling back to DejaVu Sans.
findfont: Font family ['normal'] not found. Falling back to DejaVu Sans.
```




```
In [ ]: # Predict Anomaly using this rule: Bigger difference between data and prediction means

        diff = y_test-y_pre[:,0]
        anomaly_ratio = 0.01
        mask = abs(diff)>np.nanpercentile(abs(diff),100-100*anomaly_ratio)



        font = {'family': 'normal','weight': 'bold',
                'size': 25}

        matplotlib.rc('font', **font)
        rc('axes', linewidth=3)
```

```
timeline = np.arange(0,len(y_test),1)

plt.plot(timeline/60,y_test,"b",alpha=1,linewidth=1)
plt.plot(timeline[mask]/60,y_test[mask],"ro",label="Anomaly predicted MLSTM+FCN",alpha=

plt.xlabel("Time in hours")
plt.ylabel("Normalized %s"%names_array[index_name])

plt.legend()

fig = matplotlib.pyplot.gcf()


fig.set_size_inches(35,16)
save_path = plot_path + "MLSTM_FCN_anomaly_prediction_5D" + ".png"

fig.savefig(save_path, dpi=150)
```

In [ ]: