RNNs for Time Series Forecast: Energy Demand Forecast

PART I

1) INTRODUCTION:

The type of deep learning model used in this project is Recurrent Neural Network.

The Concept of Recurrent Neural Networks

Deep learning models called recurrent neural networks (RNNs) are frequently employed to address issues involving sequential input data, such as time series. RNNs are a sort of neural network that can learn from earlier iterations during training because it keeps track of the information it has already processed.

Time Series Forecasting:

A set of techniques in statistics and data science called time series forecasting are used to forecast some variables that vary and evolve over time. Temporal series forecasting's main goal is to predict how target variables will change in the future by looking at past data from a temporal perspective, identifying patterns, and making short- or long-term predictions about change while taking into account the patterns identified.

The models I developed in this project contains LSTM layer.

LSTMs, or Long Short-Term Memory Networks:

When learning a mapping function from inputs to outputs, recurrent neural networks, such as the Long Short-Term Memory network or LSTM, add the explicit handling of order between observations, which is not provided by MLPs or CNNs. These neural networks add native support for input information made up of collections of observations.

In this project we create 3 datasets that is first 50% of data is split for training and intermediate 25% for validation and last 25% for test dataset.

And we will address 2 prediction challenges in this project namely:

3 –hour horizon and 6- hour horizon.

For each of the challenges we will develop:

a) A 1-input predictor (which will USE VALUES OF JUST *eload* TO PREDICT *eload*.

b) A 2-input predictor (which will USE VALUES OF BOTH *eload* and *tempf* TO PREDICT *eload*.

2) a) What is the highest (max) value of eload in the TEST SET?

```
Max eload in the test set:  5224.0
```

b) What is the lowest (min) value of eload in the TEST SET?

```
Min eload in the test set:  1979.0
```

c) what is the difference between the max and the min that you found in parts a) and b) ? (We will call this the “Full Range” of eload, within the TEST SET).

```
Full range of eload in the test set:  3245.0
```

Part II-A: 1-Input Predictor (3 hour):

a) **sequence_length = 12**

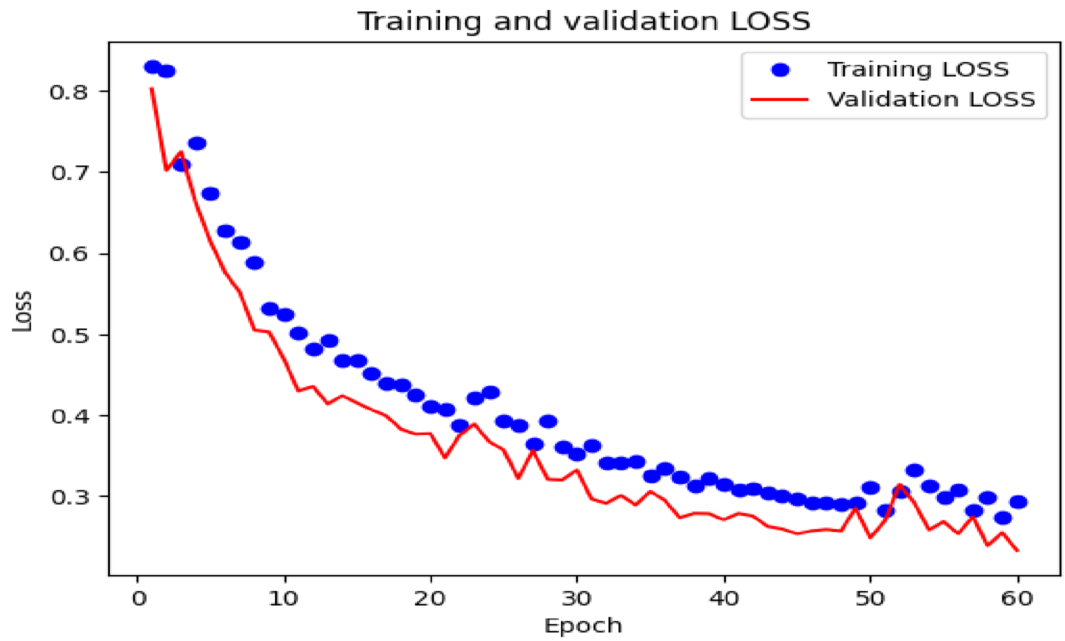
b) **COMPILE METHOD & FIT METHOD:**

```
model.compile(loss='mae' , optimizer=opt , metrics=['mae'])

# Define callbacks using the best model only
best_model_callback = ModelCheckpoint("best_model.h5", monitor='val_loss',
 save_best_only=True)
early_stop = EarlyStopping(monitor='val_loss', patience=5)

# Train model
history = model.fit(train_dataset, epochs=60, batch_size = 128, validation
_data=val_dataset, callbacks=[best_model_callback])
```

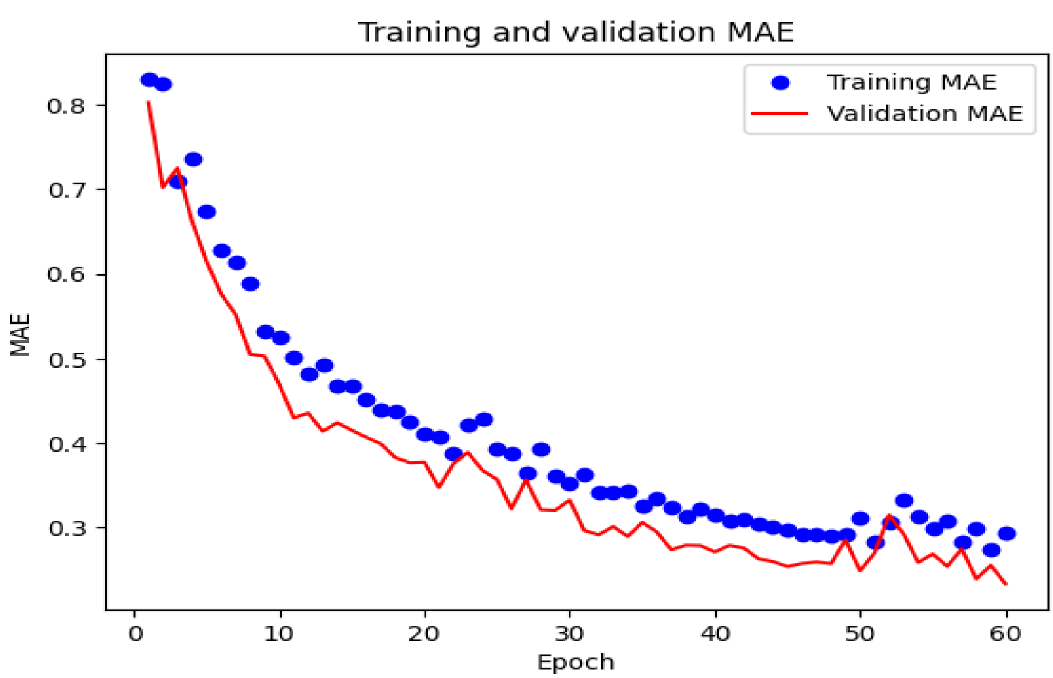
c) **`model.summary()`**

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                  (None, 64)               16896

 dropout (Dropout)            (None, 64)               0

 dense (Dense)                (None, 64)               4160

 dropout_1 (Dropout)          (None, 64)               0

 dense_1 (Dense)              (None, 32)               2080

 dense_2 (Dense)              (None, 1)                33

=================================================================
Total params: 23,169
Trainable params: 23,169
Non-trainable params: 0
_________________________________________________________________
```

d) TRAINING AND VALIDATION LOSS



e) TRAINING AND VALIDATION MAE



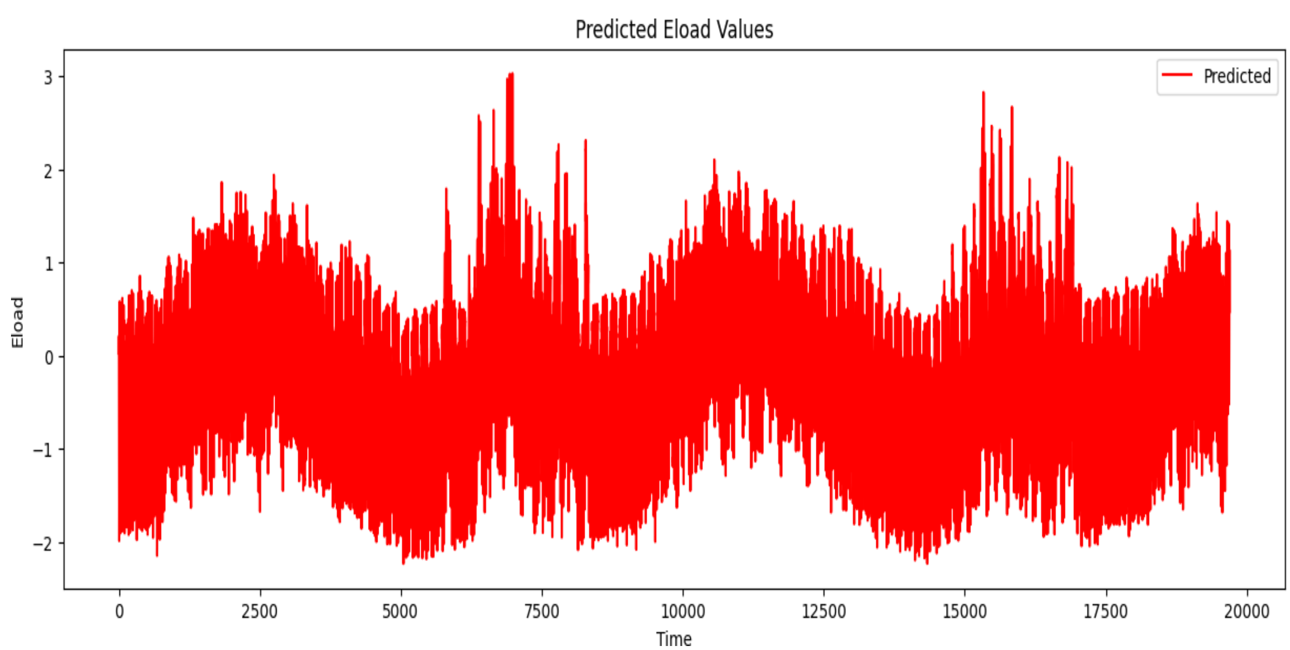
f) Final Training and Validation LOSS & MAE:

```
loss: 0.2934 - mae: 0.2934 - val_loss: 0.2335 - val_mae: 0.2335

Training Loss   : 0.2934
Training  MAE   :  0.2934
Validation LOSS : 0.2335
Validation MAE  : 0.2335
```

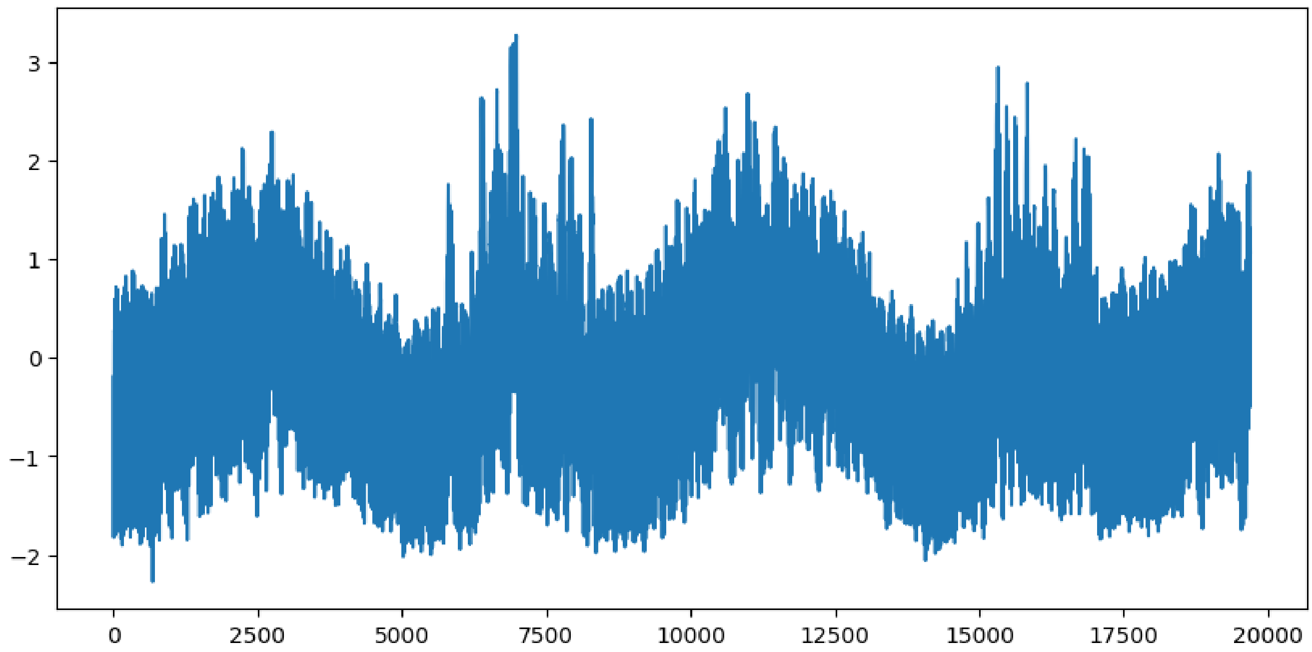
g) MAE obtained on the TEST SET:

```
Test MAE: 0.237017422914505
```

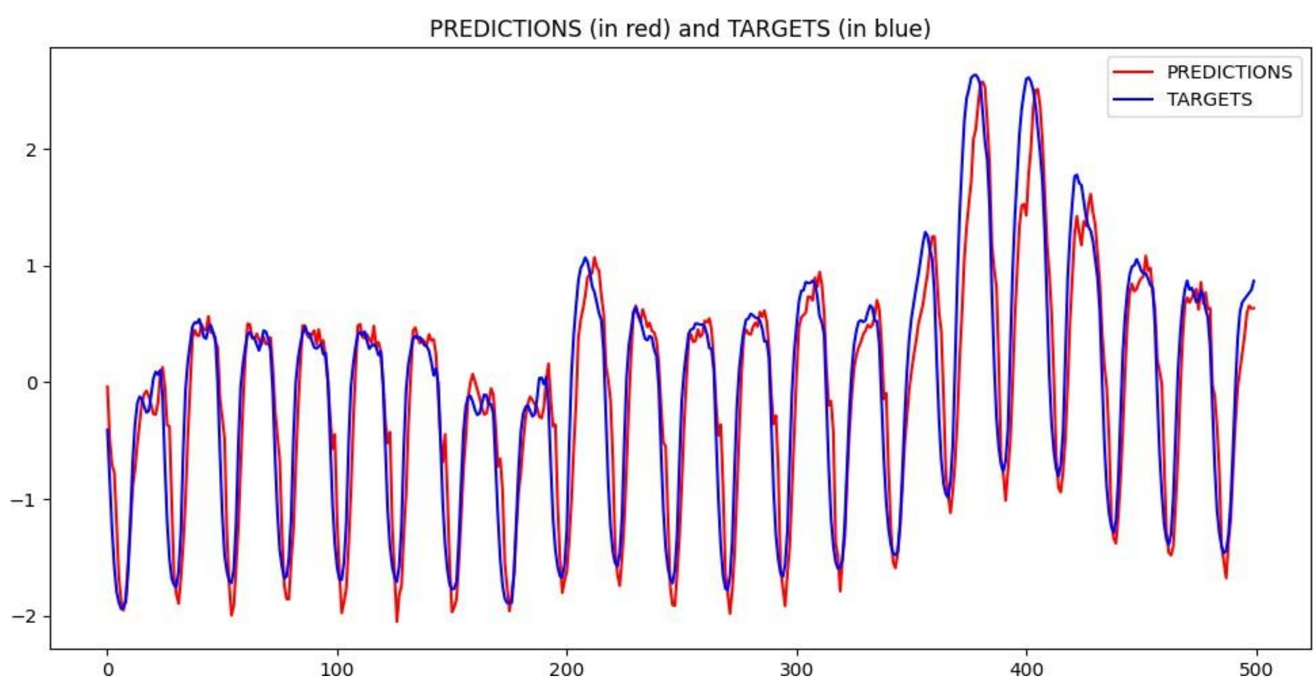
h) Time series plot of all values predicted by the model on the test set:



i) Time series plot of the corresponding targets:



j) Overlay plot of the predictions (red solid line) and the targets (blue solid line) for samples 6000 to 6500 OF THE PREDICTIONS FROM THE MODEL.

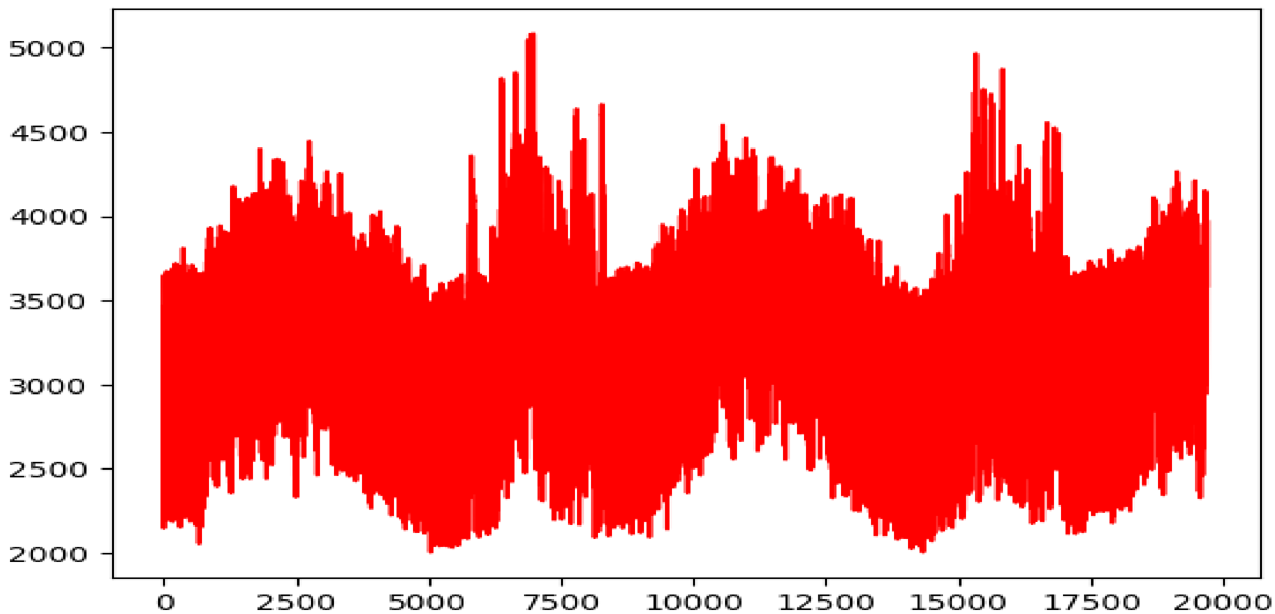


AFTER DOING DENORMALIZATION

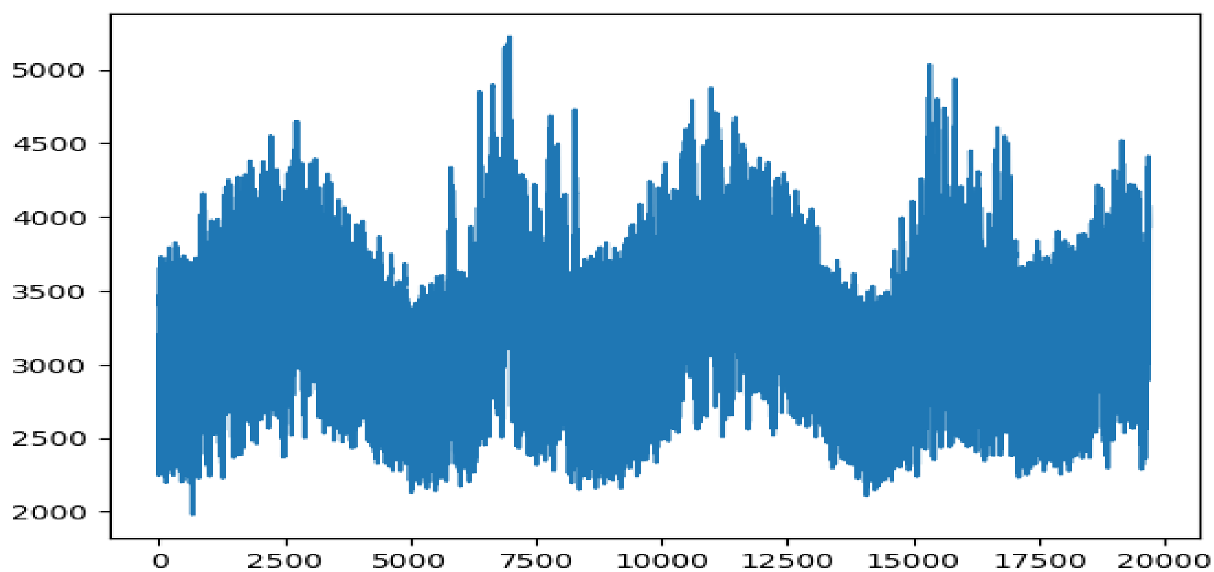
g2) True MAE obtained on the TEST SET:

```
EFFECTIVE real-scale MAE: 198.14
```

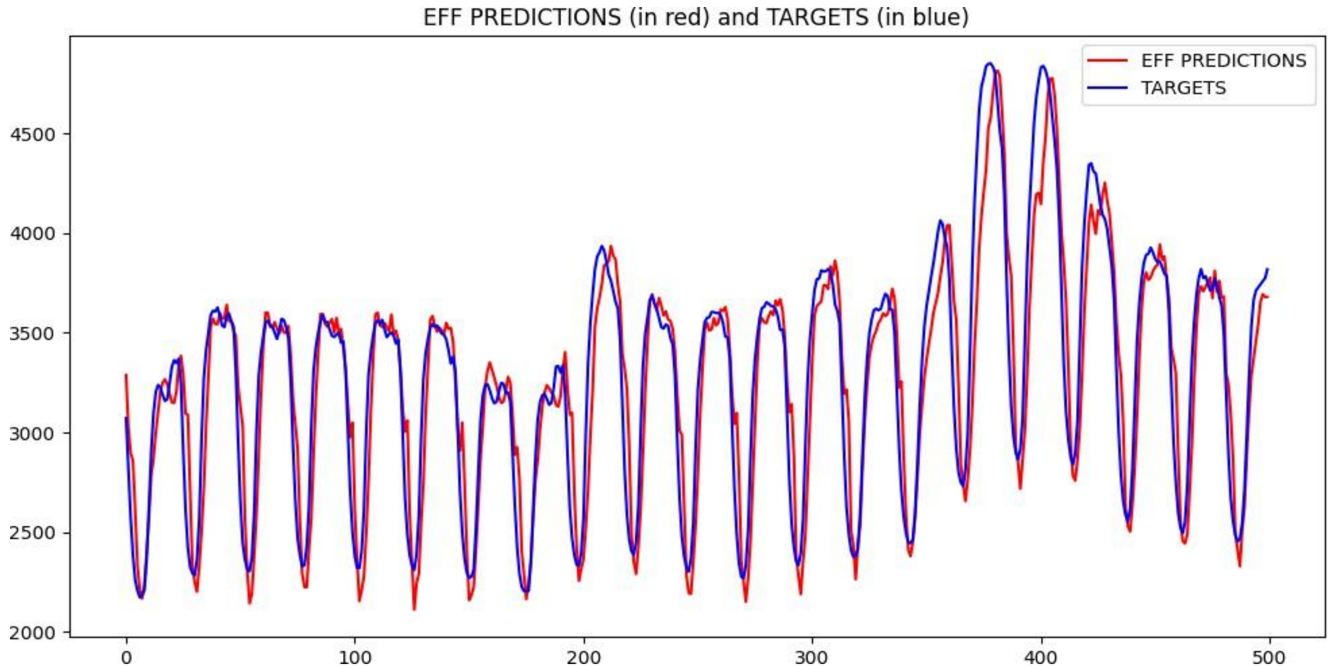
h2) Time series plot of all values predicted by the model on the test set:



i2) Time series plot of the corresponding targets:



j2) Overlay plot of the predictions (red solid line) and the targets (blue solid line) for samples 6000 to 6500 OF THE PREDICTIONS FROM THE MODEL.



k)

$$PMAE = \frac{(true)MAE\ obtained\ by\ the\ model\ on\ the\ test\ set}{Full\ range\ of\ elaod\ in\ the\ test\ set}$$

PMAE = 198.14/3245

= 0.061

= 6.1%

Part II-B: 1-Input Predictor (6 hour):

a) sequence_length = 24

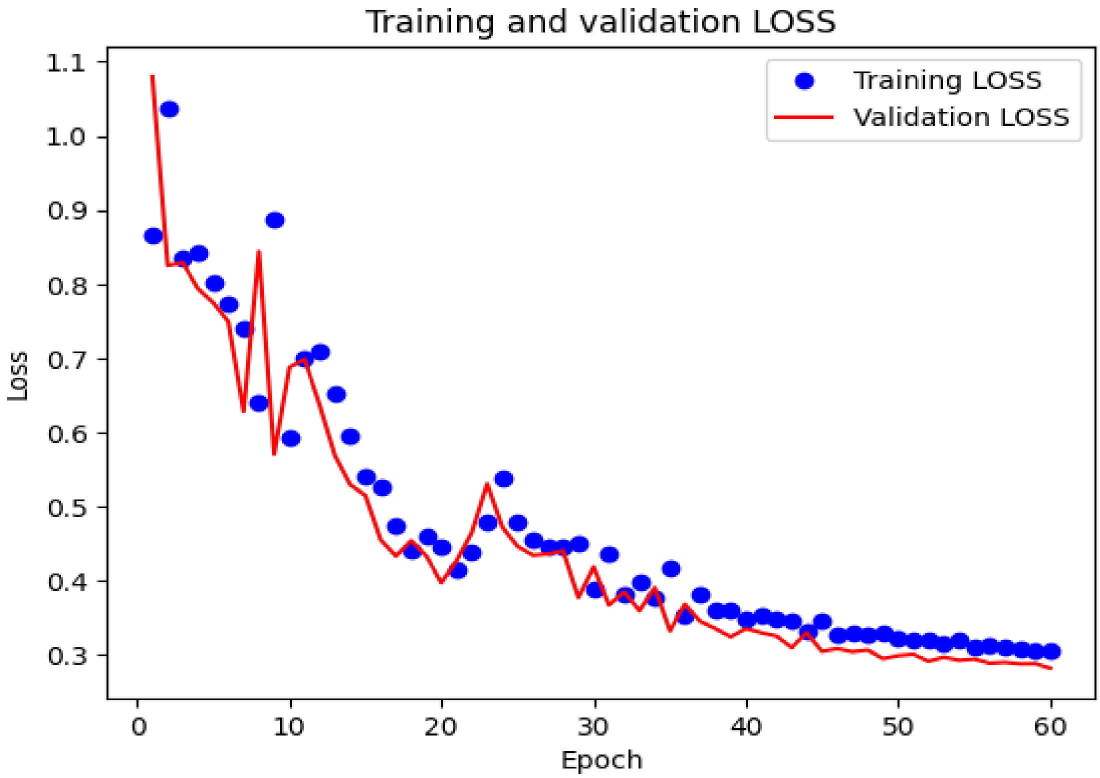
b) COMPILE METHOD & FIT METHOD:

```
model.compile(loss='mae' , optimizer=opt , metrics=['mae'])

# Define callbacks using the best model only
best_model_callback = ModelCheckpoint("best_model.h5", monitor='val_loss',
 save_best_only=True)
early_stop = EarlyStopping(monitor='val_loss', patience=5)

# Train model
history = model.fit(train_dataset, epochs=60, batch_size = 128, validation
_data=val_dataset, callbacks=[best_model_callback])
```

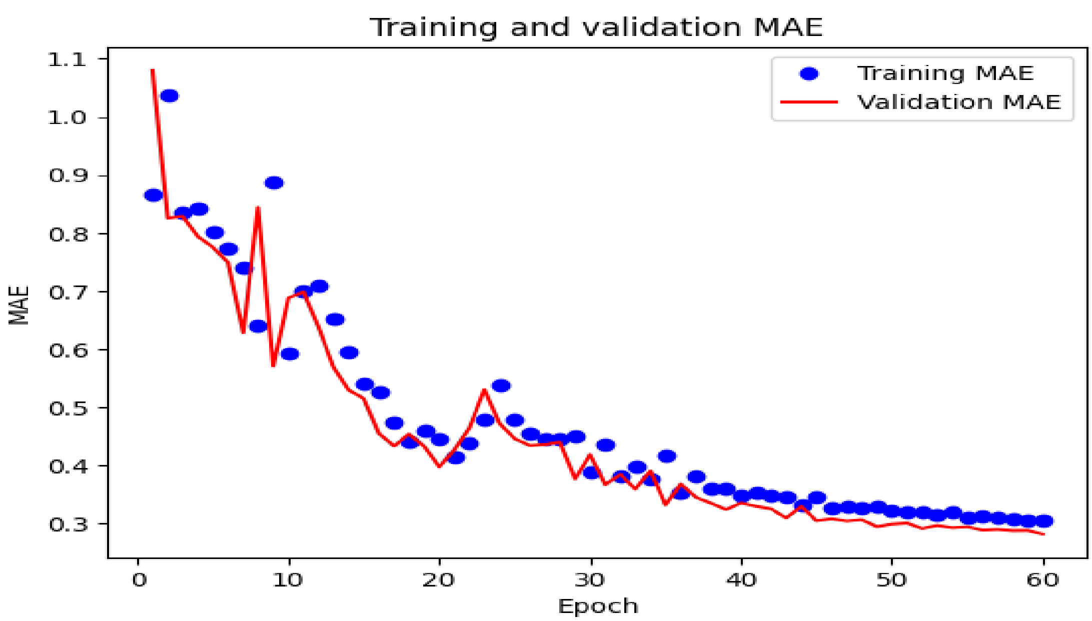
c) `model.summary()`

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                16896

 dropout (Dropout)           (None, 64)                0

 dense (Dense)               (None, 64)                4160

 dropout_1 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 1)                 33

=================================================================
Total params: 23,169
Trainable params: 23,169
Non-trainable params: 0
_________________________________________________________________
```

d) TRAINING AND VALIDATION LOSS



e) TRAINING AND VALIDATION MAE



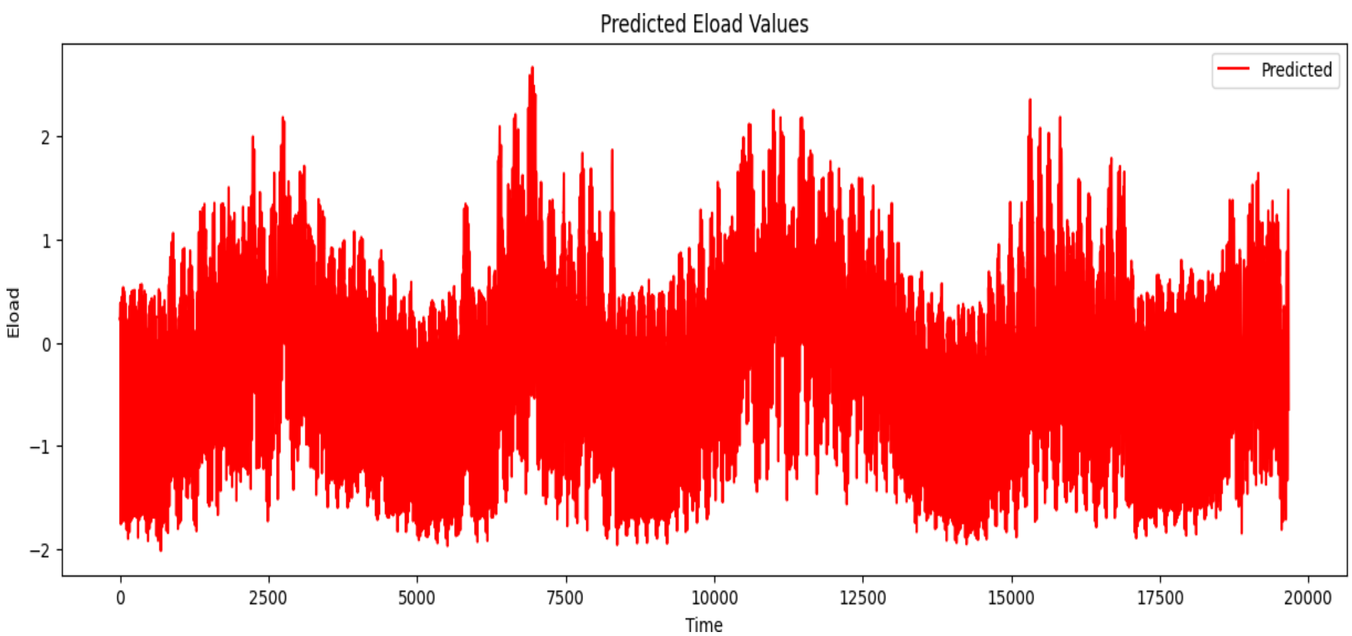
f) Final Training and Validation LOSS & MAE:

```
loss: 0.3063 - mae: 0.3063 - val_loss: 0.2820 - val_mae: 0.2820

Training Loss   : 0.3063
Training  MAE   :  0.3063
Validation LOSS : 0.2820
Validation MAE  : 0.2820
```

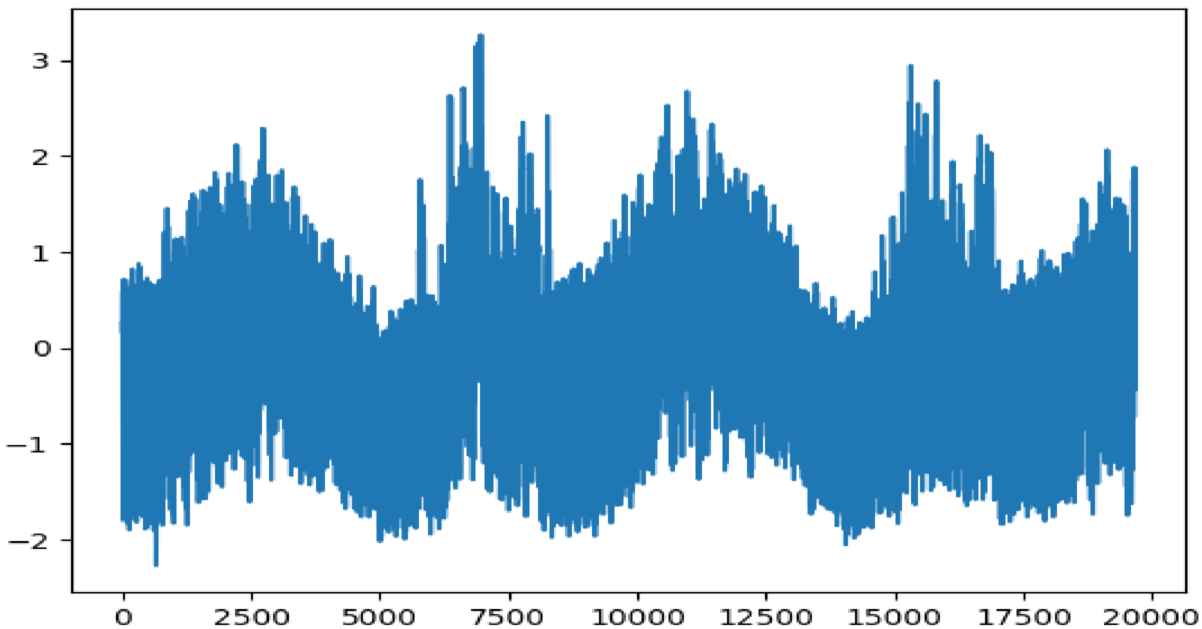
g) MAE obtained on the TEST SET:

```
Test MAE: 0.283030241727829
```

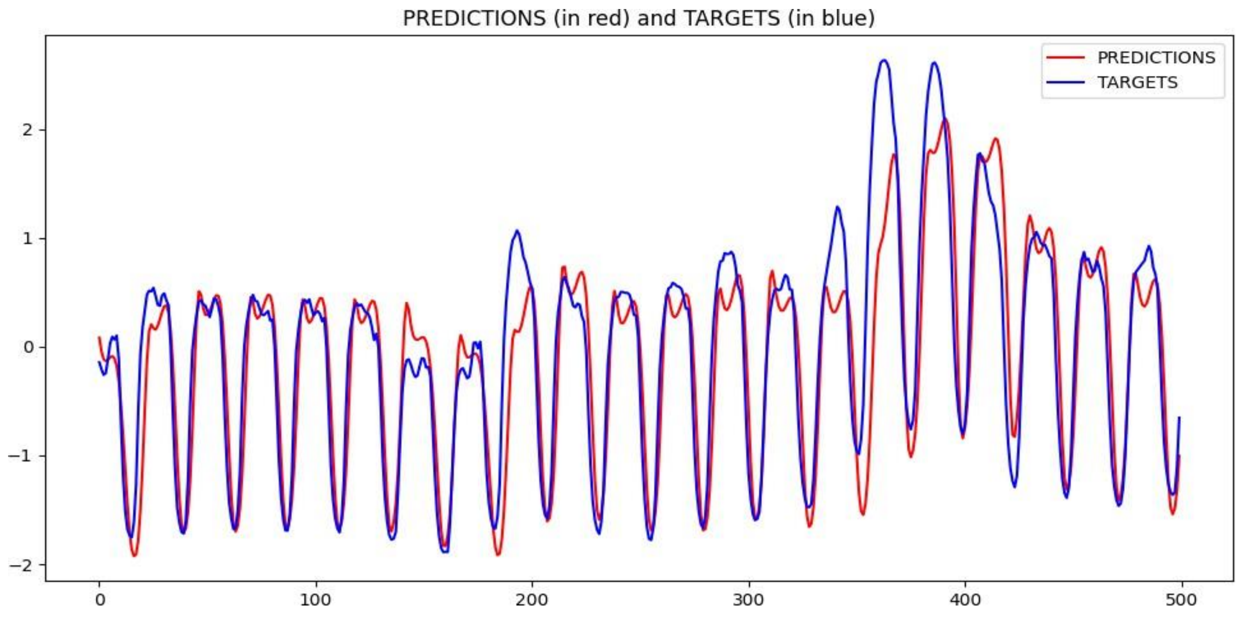
h) Time series plot of all values predicted by the model on the test set:



i) Time series plot of the corresponding targets



j) Overlay plot of the predictions (red solid line) and the targets (blue solid line) for samples 6000 to 6500 OF THE PREDICTIONS FROM THE MODEL.

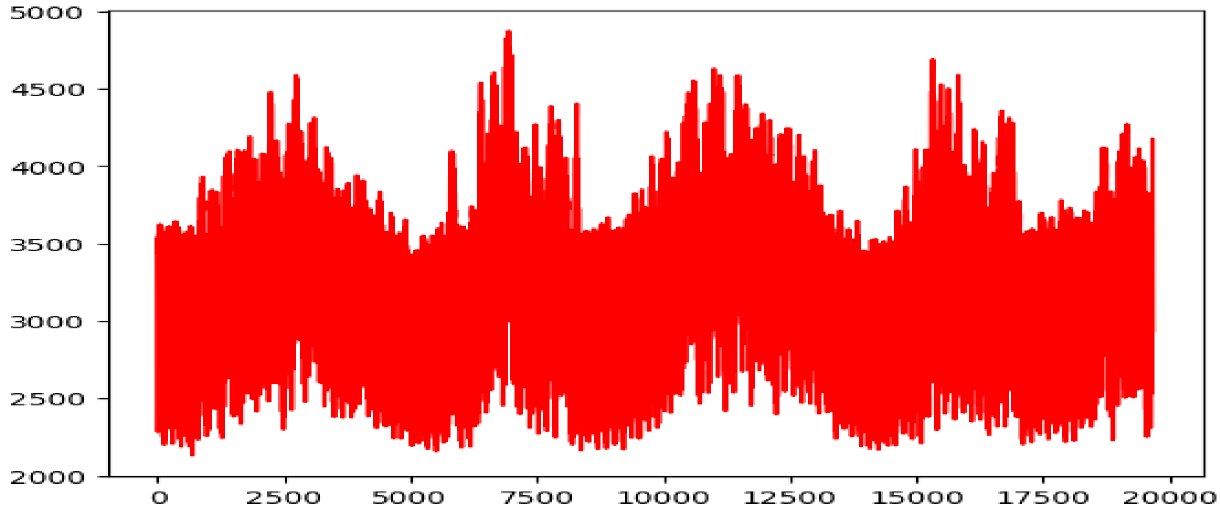


AFTER DOING DENORMALIZATION

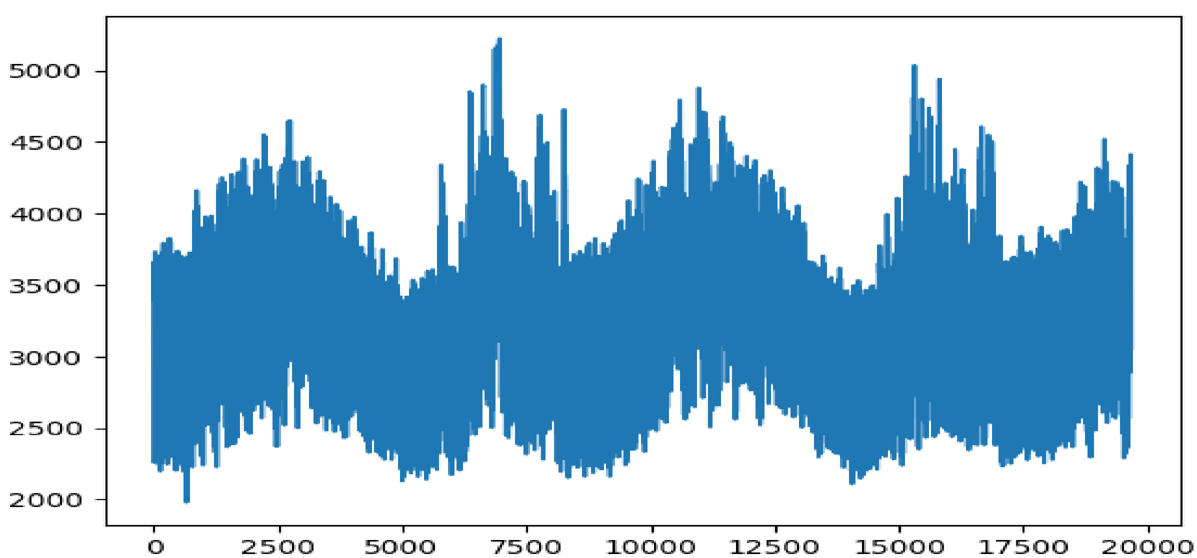
g2) True MAE obtained on the TEST SET:

```
EFFECTIVE real-scale MAE: 192.54
```

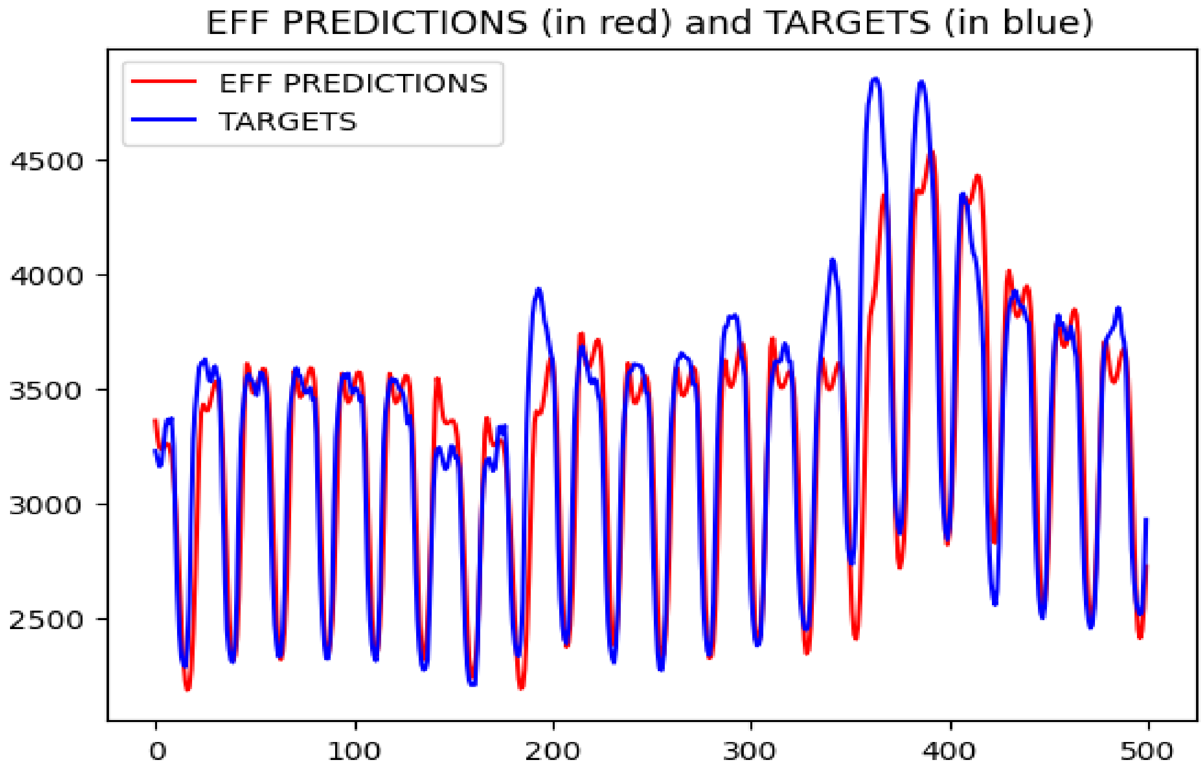
h2) Time series plot of all values predicted by the model on the test set:



i2) Time series plot of the corresponding targets:



j2) Overlay plot of the predictions (red solid line) and the targets (blue solid line) for samples 6000 to 6500 OF THE PREDICTIONS FROM THE MODEL.



k)

$$PMAE = \frac{(true)MAE\ obtained\ by\ the\ model\ on\ the\ test\ set}{Full\ range\ of\ elaod\ in\ the\ test\ set}$$

PMAE = 192.54/3245

= 0.059

= 5.9%

Part III-A: 2-Input Predictor (3 hour):

a) **sequence_length = 12**

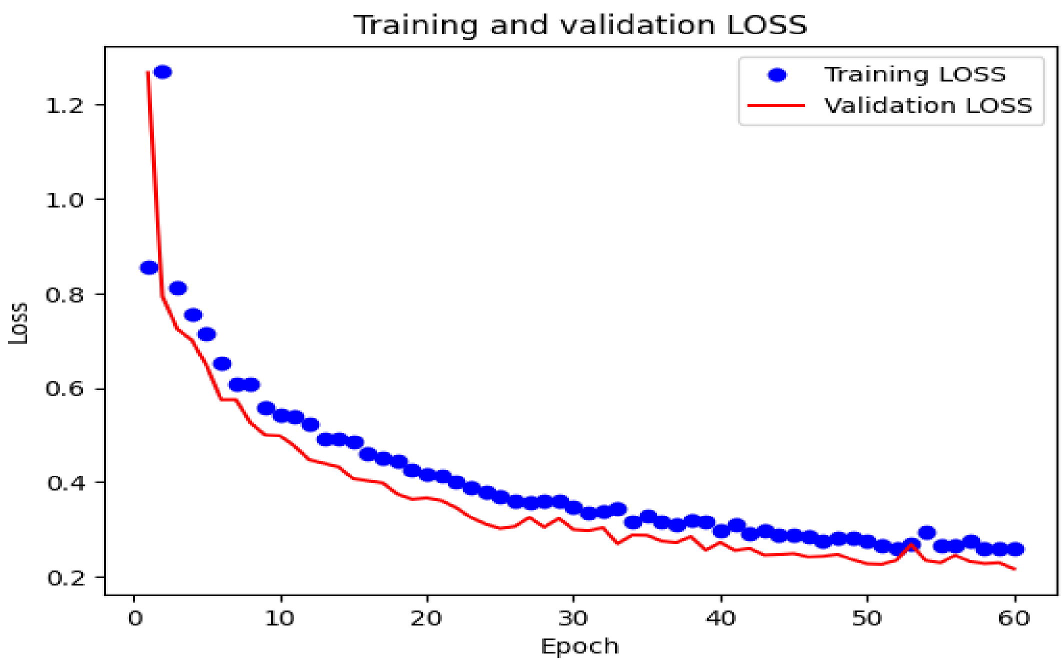
b) **COMPILE METHOD & FIT METHOD:**

```
model.compile(loss='mae' , optimizer=opt , metrics=['mae'])

# Define callbacks using the best model only
best_model_callback = ModelCheckpoint("best_model.h5", monitor='val_loss',
 save_best_only=True)
early_stop = EarlyStopping(monitor='val_loss', patience=5)

# Train model
history = model.fit(train_dataset, epochs=60, batch_size = 128, validation
_data=val_dataset, callbacks=[best_model_callback])
```

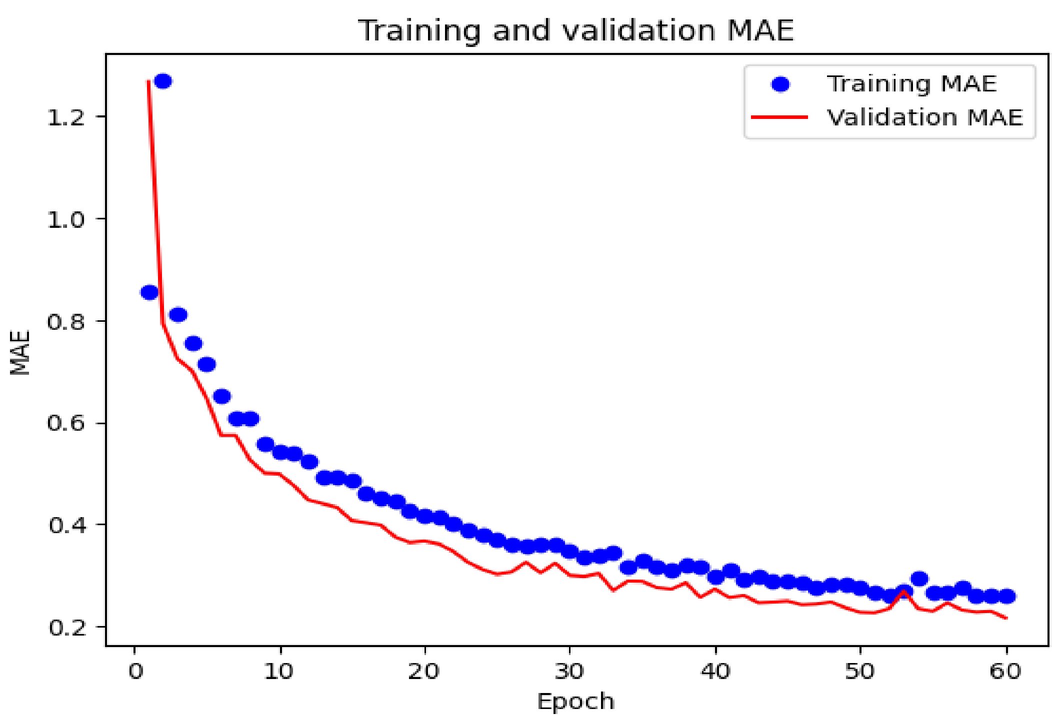
c) **`model.summary()`**

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_1 (LSTM)                (None, 64)                17152

 dropout_2 (Dropout)          (None, 64)                0

 dense_3 (Dense)              (None, 64)                4160

 dropout_3 (Dropout)          (None, 64)                0

 dense_4 (Dense)              (None, 32)                2080

 dense_5 (Dense)              (None, 1)                 33

=================================================================
Total params: 23,425
Trainable params: 23,425
Non-trainable params: 0
```

d) TRAINING AND VALIDATION LOSS



e) TRAINING AND VALIDATION MAE



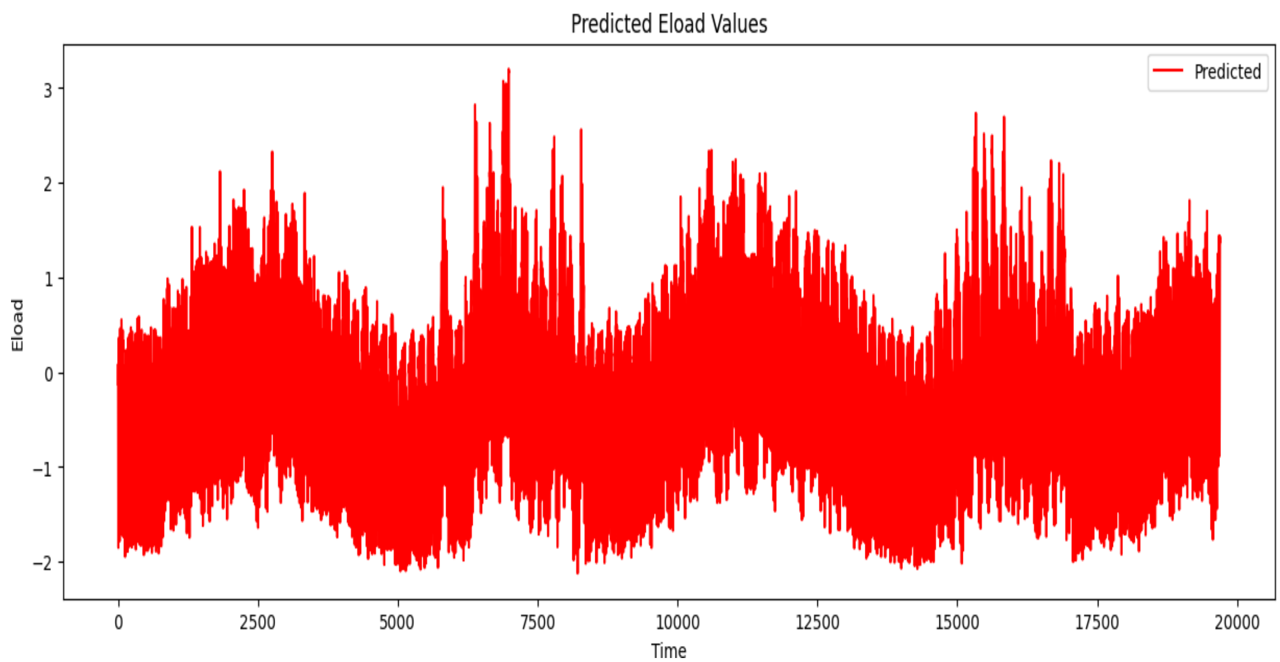
f) Final Training and Validation LOSS & MAE:

```
loss: 0.2616 - mae: 0.2616 - val_loss: 0.2165 - val_mae: 0.2165

Training Loss   : 0.2616
Training  MAE   :  0.2616
Validation LOSS : 0.2165
Validation MAE  : 0.2165
```

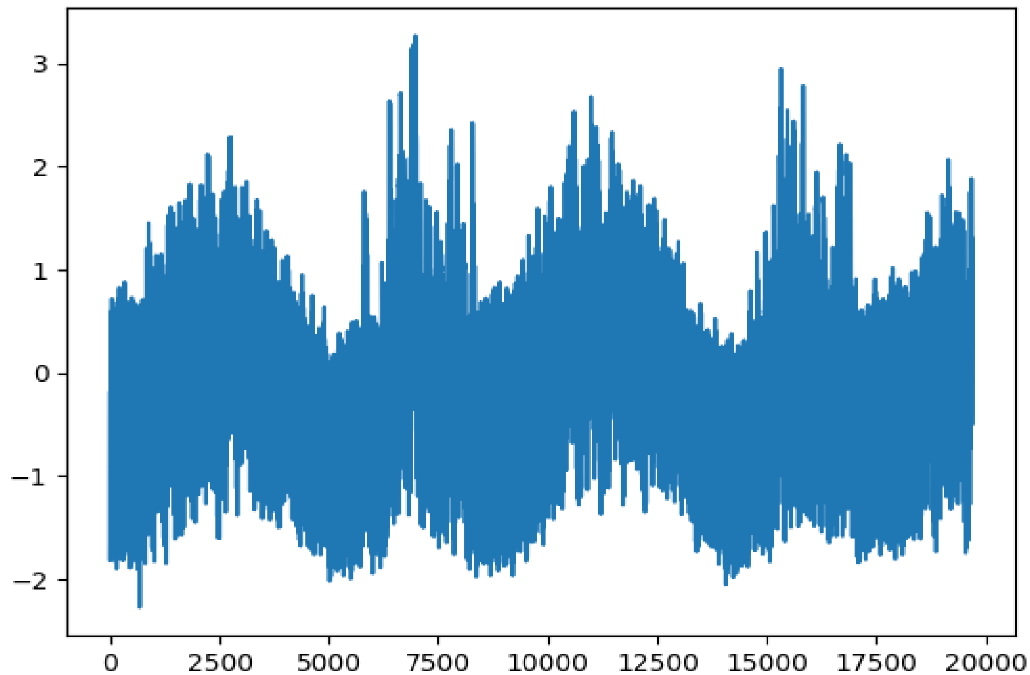
g) MAE obtained on the TEST SET:

```
Test MAE: 0.224911168217659
```

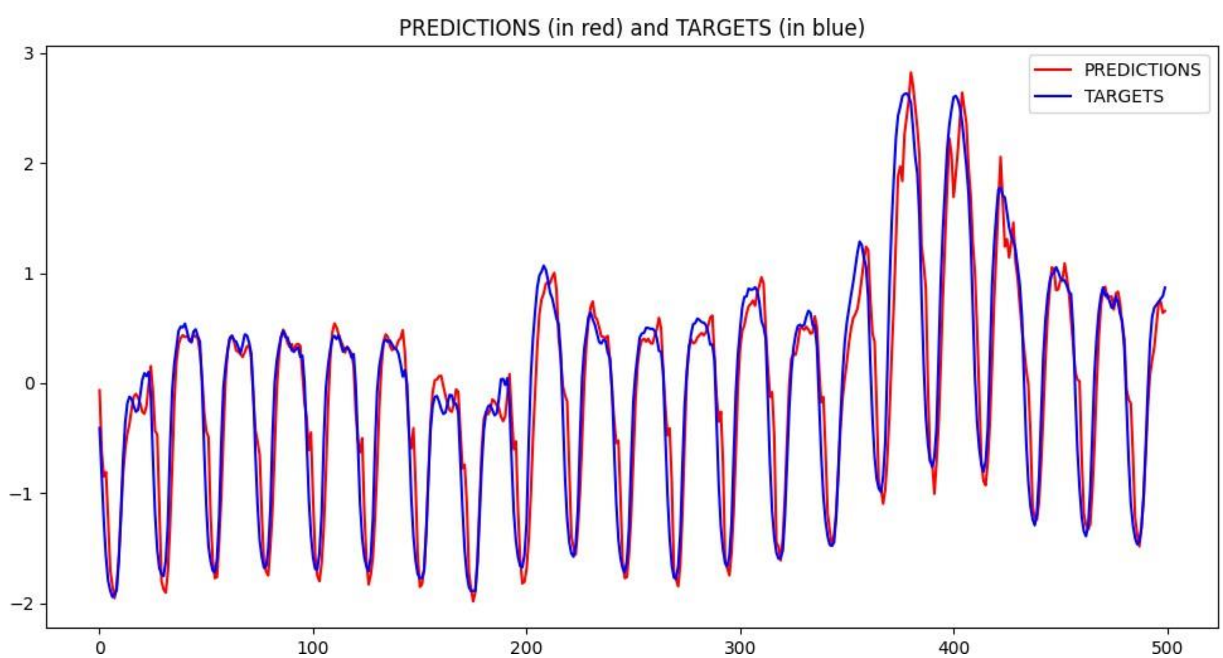
h) Time series plot of all values predicted by the model on the test set:



i) Time series plot of the corresponding targets



j) Overlay plot of the predictions (red solid line) and the targets (blue solid line) for samples 6000 to 6500 OF THE PREDICTIONS FROM THE MODEL.

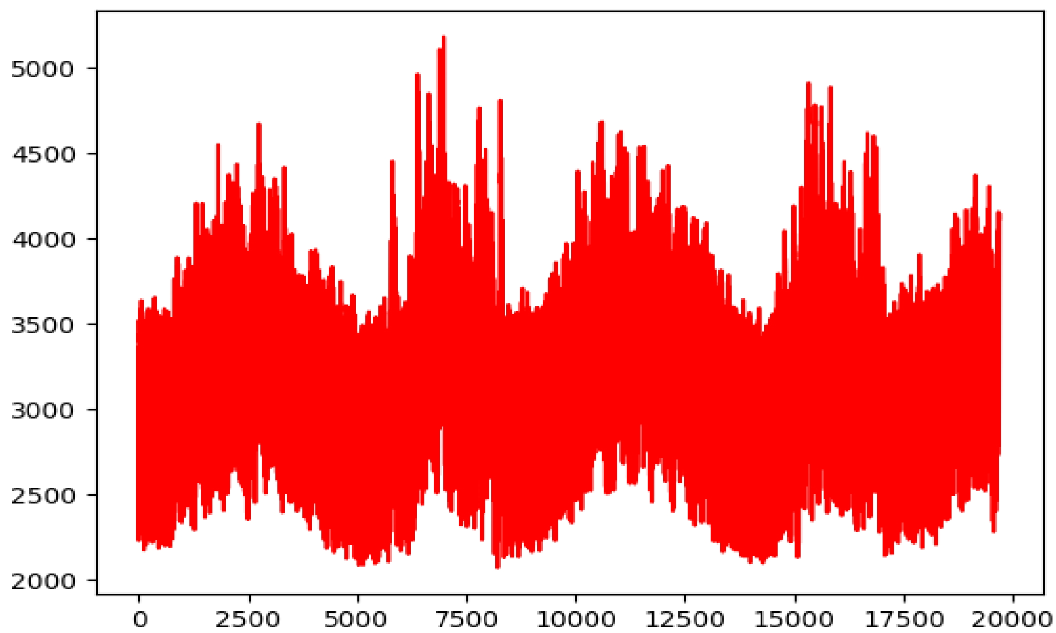


AFTER DOING DENORMALIZATION

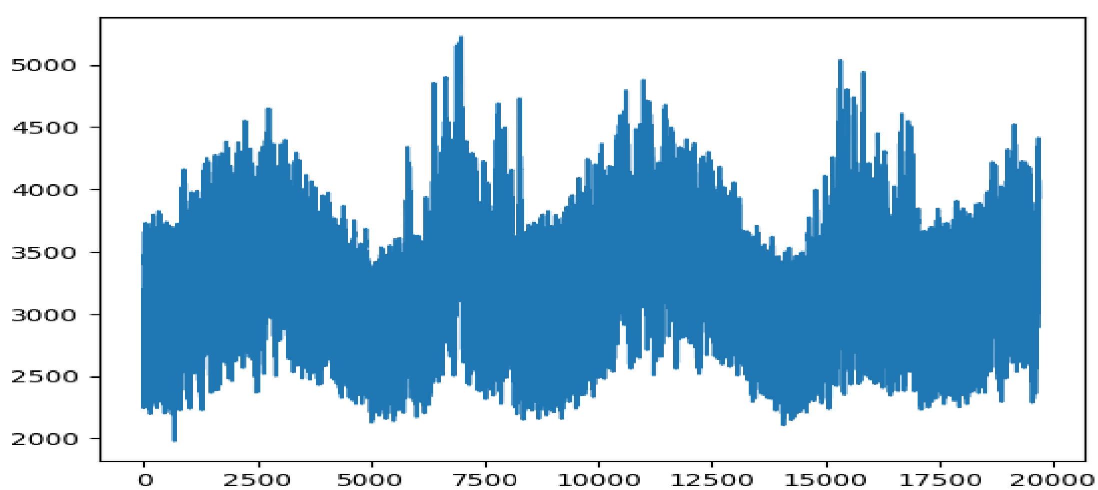
g2) True MAE obtained on the TEST SET:

```
EFFECTIVE real-scale MAE: 172.75
```

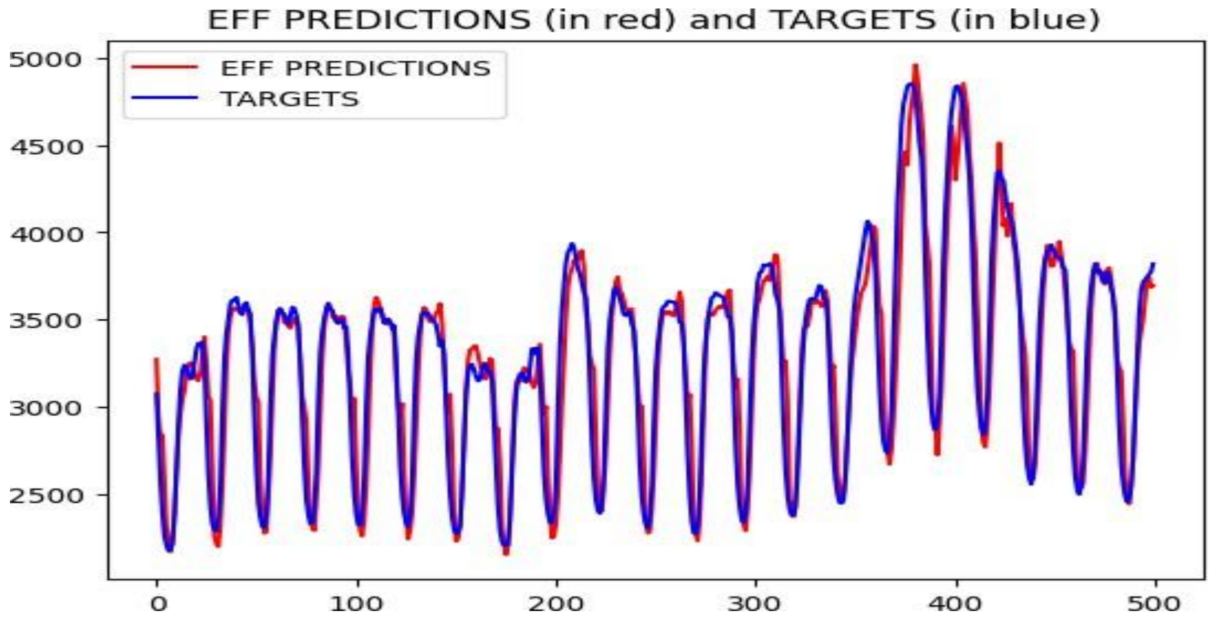
h2) Time series plot of all values predicted by the model on the test set:



i2) Time series plot of the corresponding targets:



j2) Overlay plot of the predictions (red solid line) and the targets (blue solid line) for samples 6000 to 6500 OF THE PREDICTIONS FROM THE MODEL



k)

$$PMAE = \frac{(true)MAE\ obtained\ by\ the\ model\ on\ the\ test\ set}{Full\ range\ of\ elaod\ in\ the\ test\ set}$$

PMAE = 172.75/3245

= 0.053

= 5.3%

Part III-B: 2-Input Predictor (6 hour):

a) **sequence_length = 24**

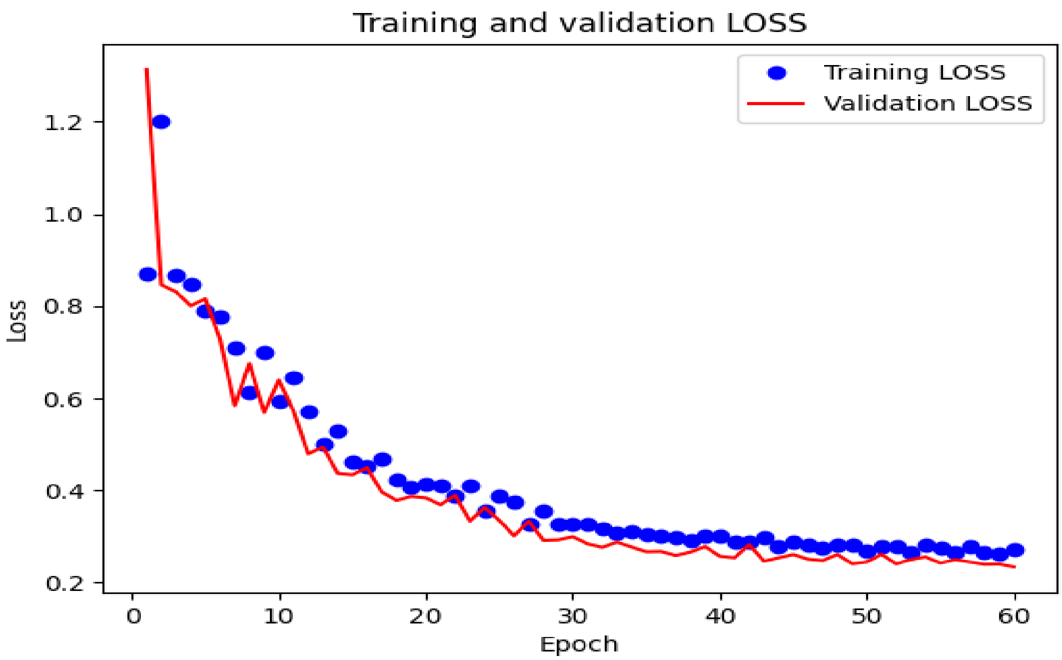
b) **COMPILE METHOD & FIT METHOD:**

```
model.compile(loss='mae' , optimizer=opt , metrics=['mae'])

# Define callbacks using the best model only
best_model_callback = ModelCheckpoint("best_model.h5", monitor='val_loss',
 save_best_only=True)
early_stop = EarlyStopping(monitor='val_loss', patience=5)

# Train model
history = model.fit(train_dataset, epochs=60, batch_size = 128, validation
_data=val_dataset, callbacks=[best_model_callback])
```

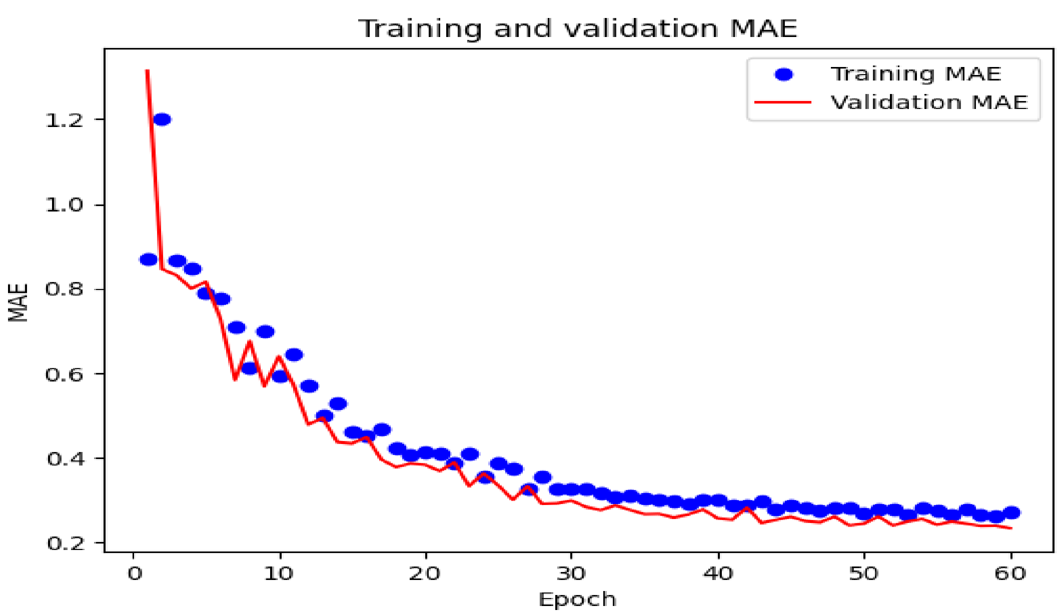
c) **`model.summary()`**

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                17152

 dropout (Dropout)           (None, 64)                0

 dense (Dense)               (None, 64)                4160

 dropout_1 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 1)                 33

=================================================================
Total params: 23,425
Trainable params: 23,425
Non-trainable params: 0
```

d) TRAINING AND VALIDATION LOSS



e) TRAINING AND VALIDATION MAE



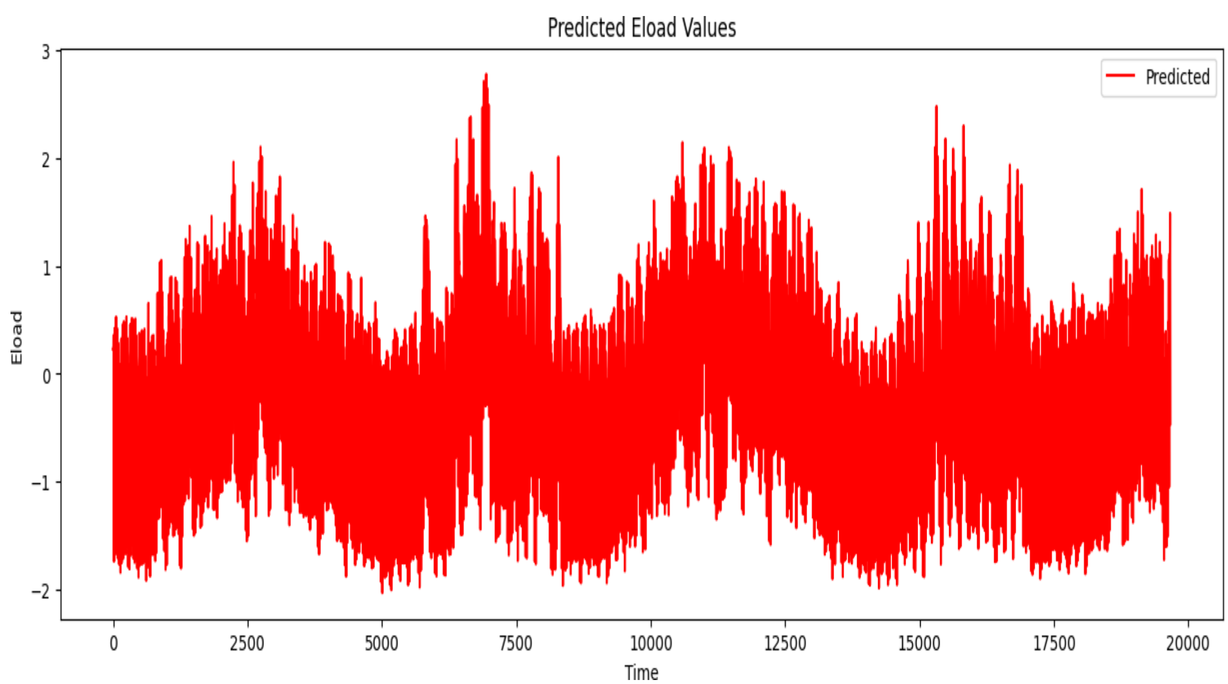
f) Final Training and Validation LOSS & MAE:

```
loss: 0.2723 - mae: 0.2723 - val_loss: 0.2335 - val_mae: 0.2335

Training Loss   : 0.2723
Training  MAE   :  0.2723
Validation LOSS : 0.2335
Validation MAE  : 0.2335
```

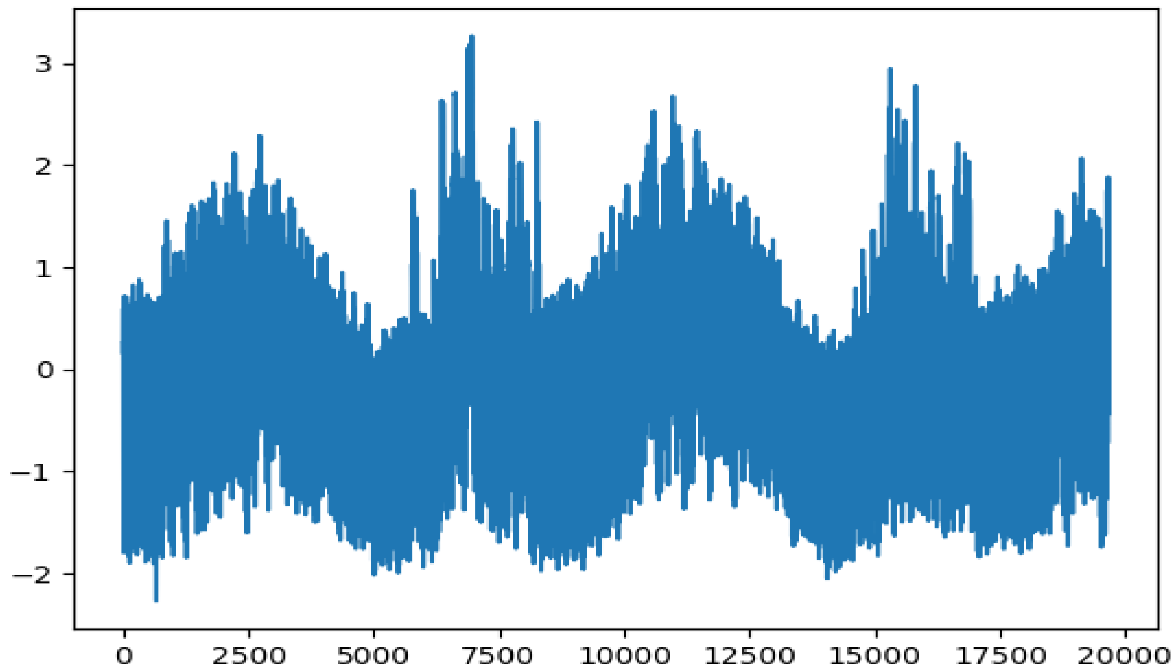
g) MAE obtained on the TEST SET:

```
Test MAE: 0.23702189326286316
```

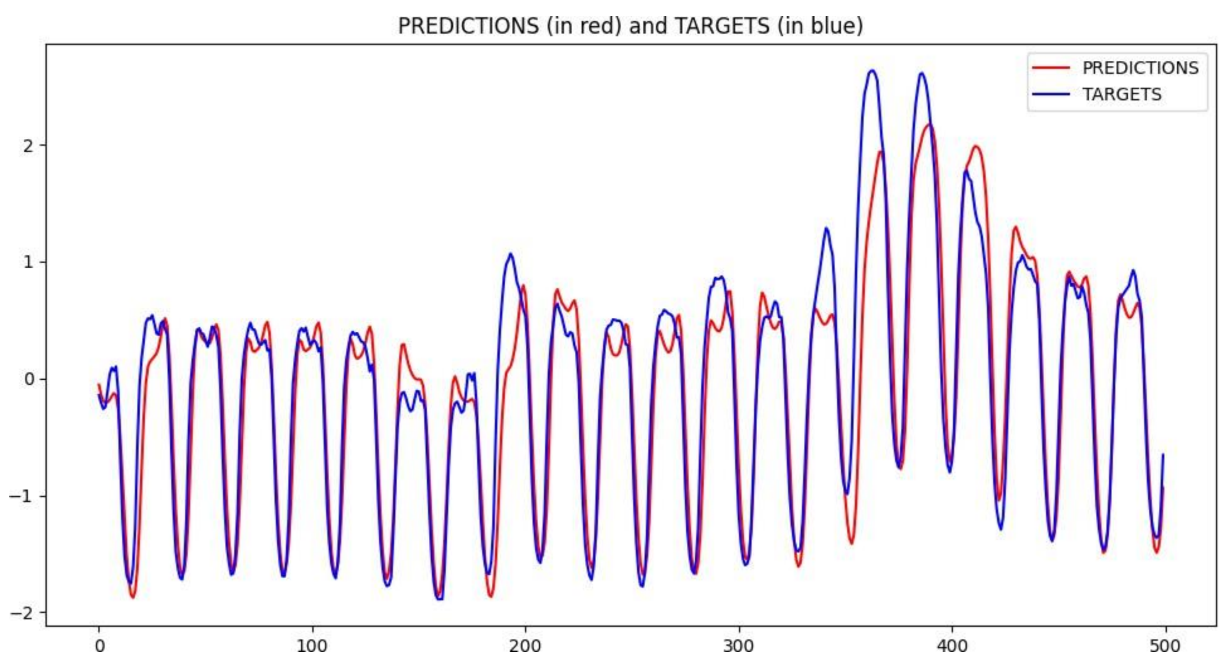
h) Time series plot of all values predicted by the model on the test set:



i) **Time series plot of the corresponding targets**



j) **Overlay plot of the predictions (red solid line) and the targets (blue solid line) for samples 6000 to 6500 OF THE PREDICTIONS FROM THE MODEL**

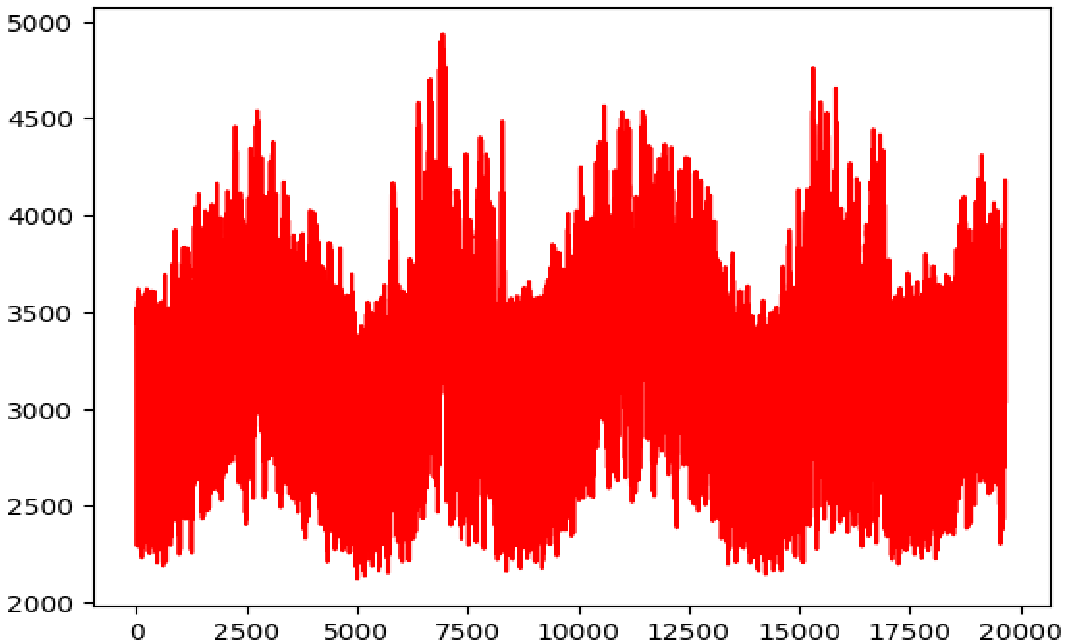


<u>AFTER DOING DENORMALIZATION</u>

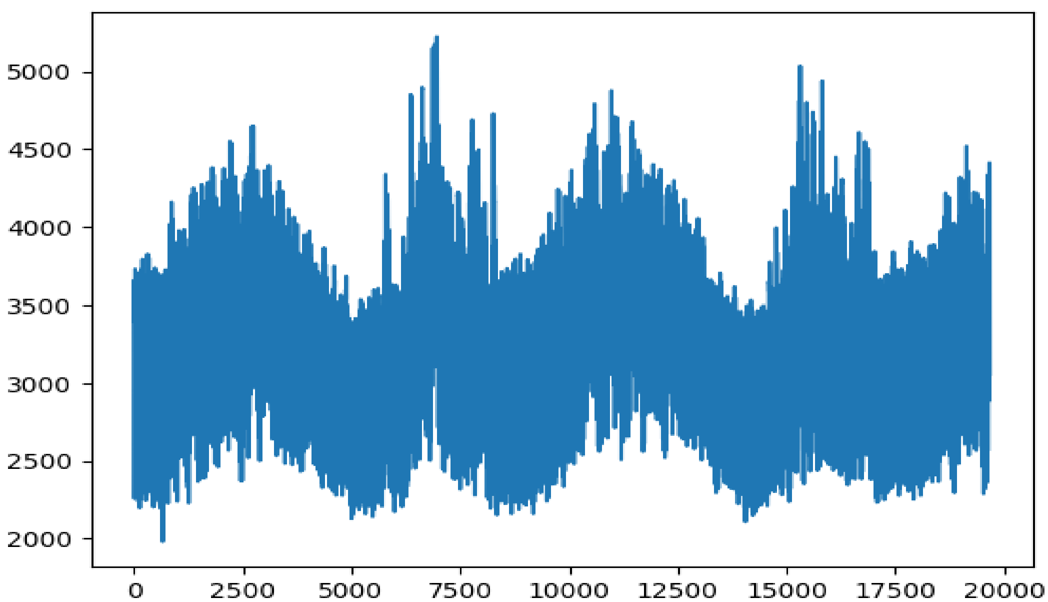
g2) True MAE obtained on the TEST SET:

```
EFFECTIVE real-scale MAE: 158.03
```

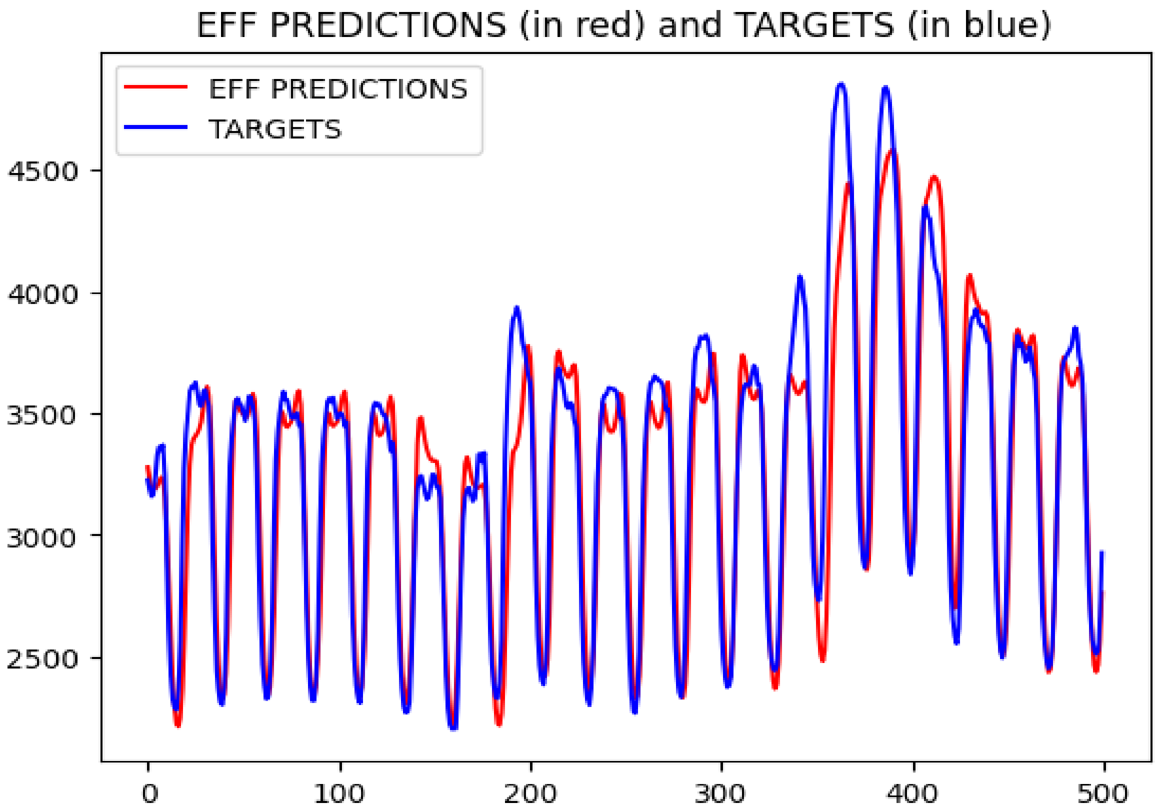
h2) Time series plot of all values predicted by the model on the test set:



i2) Time series plot of the corresponding targets:



j2) Overlay plot of the predictions (red solid line) and the targets (blue solid line) for samples 6000 to 6500 OF THE PREDICTIONS FROM THE MODEL



k)

$$PMAE = \frac{(true)MAE\ obtained\ by\ the\ model\ on\ the\ test\ set}{Full\ range\ of\ elaod\ in\ the\ test\ set}$$

PMAE = 158.03/3245

= 0.048

= 4.8%

CONCLUSIONS

What I learned & difficulties faced:

Initially I was not able to get the better training and validation loss. After increasing the number of epochs and passing a learning rate argument to the optimizer, I was able to get the better results and obtained the best models after tuning a lot of hypermeters. I was able to achieve finally Real-Scale of MAE of 158.03.

The difficulties I faced initially was without normalizing the targets I am not able to train and do predicitions. The methodolody which was provided helps me in doing normalization to get the better results. I think that the data with these large values we need to preprocess the data necessarily inorder to scale down the values or else there will be lot of misinterpretations and will lead to wrong results and predictions. So additional preprocessig of the data was a lot beneficial.

The PMAE of 1 vs 2-input predictors with a 3-hour horizon was 6.1 & 5.3, I observed some reduce in PMAE if increase the sequence length might get the better PMAE. I am still trying to reduce the PMAE of both the predictors. And the PMAE of of 1 vs 2-input predictors with a 6-hour horizon was 5.9 & 4.8, I was able to get the best results for this predictors , So I can conclude that from my results I trained , the model was able to perform better when providing predicitons of eload 6 hours into the future.