In [142]:

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import datetime
import warnings
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler

from numpy import array
from numpy import split
from livelossplot.keras import PlotLossesCallback

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM, Bidirectional
from keras.layers import GRU
from keras.layers.convolutional import Conv1D
from keras.layers import Flatten
from keras.layers.convolutional import MaxPooling1D
from keras import Input, Model
from keras import backend as K
from keras.engine.topology import Layer
from keras import initializers, regularizers, constraints
from numpy import concatenate

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from math import sqrt

%matplotlib inline
plt.rcParams['image.interpolation'] = 'nearest'
plt.rcParams['image.cmap'] = 'gray'
# fix random seed for reproducibility
np.random.seed(7)
#Input Data
dataframe = pd.read_csv('Water Quality Data Monitor.csv', header
=0, index_col=0)
```

```
dataset = dataframe.values

dataset = dataset.astype('float32')

np.trim_zeros(dataset)

# normalize the dataset
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)

DO=array(dataset)
```

In [8]:

```
# split a univariate dataset into train/test sets
# split into standard weeks
train, test = DO[:int(.9*len(DO))+1], DO[int(.9*len(DO)):-11]
# restructure into windows of weekly data
train = array(split(train, int(len(train)/10)))
test = array(split(test, int(len(test)/10)))
```

In [85]:

```
# evaluate one or more weekly forecasts against expected values
def evaluate_forecasts(actual, predicted):
    scores_rmse = list()
    scores_mae = list()
    scores_r2 = list()
  # calculate an RMSE score for each day
    for i in range(actual.shape[1]):
        # calculate mse
        mse = mean_squared_error(actual[:, i], predicted[:, i])
        # calculate rmse
        rmse = sqrt(mse)
        # calculate mae
        mae = mean_absolute_error(actual[:, i], predicted[:, i])
        # calculate r square
        r2 = r2_score(actual[:, i], predicted[:, i])
        # store
        scores_rmse.append(rmse)
        scores_mae.append(mae)
        scores_r2.append(r2)
    # calculate overall RMSE
    s=0
    for row in range(actual.shape[0]):
```

```
        for col in range(actual.shape[1]):

            s += (actual[row, col] - predicted[row, col])**2
    score_rmse = sqrt(s / (actual.shape[0] * actual.shape[1]))
    # calculate overall mae
    s=0
    for row in range(actual.shape[0]):
        for col in range(actual.shape[1]):
            s += abs(actual[row, col] - predicted[row, col])
    score_mae = s / (actual.shape[0] * actual.shape[1])
    # calculate overall R square
    s1=0
    s2=0
    for row in range(actual.shape[0]):
        for col in range(actual.shape[1]):
            s1 += (actual[row, col] - predicted[row, col])**2
            s2 += (actual[row, col] - sum(actual)/len(actual))**
2
    score_r2 = 1-s1/s2
    return score_rmse, scores_rmse, score_mae, scores_mae, score
_r2, scores_r2
```

In [11]:

```
# convert history into inputs and outputs
def to_supervised(train, n_input, n_out=10):
  # flatten data
    data = train.reshape((train.shape[0]*train.shape[1], train.s
hape[2]))
    X, y = list(), list()
    in_start = 0
  # step over the entire history one time step at a time
    for _ in range(len(data)):
        # define the end of the input sequence
        in_end = in_start + n_input
        out_end = in_end + n_out
        # ensure we have enough data for this instance
        if out_end <= len(data):
            x_input = data[in_start:in_end, 0]
            x_input = x_input.reshape((len(x_input), 1))
            X.append(x_input)
            y.append(data[in_end:out_end, 0])
        # move along one time step
        in_start += 1
    return array(X), array(y)
```

In [144]:

```
def CNN_Model(train, n_input):
    # prepare data
    train_x, train_y = to_supervised(train, n_input)
    # define parameters
    n_timesteps, n_features, n_outputs = train_x.shape[1], train
_x.shape[2], train_y.shape[1] # define model
    model = Sequential()
    model.add(Conv1D(16, 3, activation='relu', input_shape=(n_ti
mesteps,n_features)))
    model.add(MaxPooling1D())
    model.add(Flatten())
    model.add(Dense(10, activation='relu'))
    model.add(Dense(n_outputs))
    model.compile(loss='mse', optimizer='adam')
    # fit network
    model.fit(train_x, train_y, epochs=20, batch_size=128, valid
ation_split=0.1, verbose=1, shuffle=True)
    return model

def evaluate_CNN(train, test, n_input):
    M_CNN = CNN_Model(train, n_input)
    # history is a list of weekly data
    history = [x for x in train]
    # walk-forward validation over each week
    predictions_CNN = list()
    for i in range(len(test)):
        yhat_sequence_CNN = forecast(M_CNN, history, n_input)
        predictions_CNN.append(yhat_sequence_CNN)
        # get real observation and add to history for predicting
the next week
        history.append(test[i, :])
    # evaluate predictions days for each week
    predictions_CNN = array(predictions_CNN)
    score_rmse_CNN, scores_rmse_CNN, score_mae_CNN, scores_mae_C
NN, score_r2_CNN, scores_r2_CNN = evaluate_forecasts(test[:, :,
0], predictions_CNN)
    return score_rmse_CNN, scores_rmse_CNN, score_mae_CNN, score
s_mae_CNN, score_r2_CNN, scores_r2_CNN
```

In [ ]:

```

```

In [91]:

```
# train the model
def LSTM_Model(train, n_input):
    # prepare data
    train_x, train_y = to_supervised(train, n_input)
    # define parameters
    n_timesteps, n_features, n_outputs = train_x.shape[1], train
_x.shape[2], train_y.shape[1] # define model
    model = Sequential()
    model.add(LSTM(50, activation='relu', input_shape=(n_timeste
ps, n_features)))
    model.add(Dense(50, activation='relu'))
    model.add(Dense(n_outputs))
    model.compile(loss='mse', optimizer='adam')
    # fit network
    model.fit(train_x, train_y, epochs=20, batch_size=128, valid
ation_split=0.1, verbose=1, shuffle=True)
    return model
```

In [100]:

```
from tcn import TCN
def TCN_Model(train, n_input):
    train_x, train_y = to_supervised(train, n_input)
    n_timesteps, n_features, n_outputs = train_x.shape[1], train
_x.shape[2], train_y.shape[1] # define model
    i = Input(shape=(n_timesteps, n_features))
    m = TCN(nb_filters=4, kernel_size=3)(i)
    m = Dense(10, activation='linear')(m)
    model = Model(inputs=[i], outputs=[m])
    model.compile(optimizer='adam', loss='mse')
    model.fit(train_x, train_y, epochs=15, verbose=1, validation
_split=0.1)
    return model
```

In [102]:

```
def GRU_Model(train, n_input):
    # prepare data
    train_x, train_y = to_supervised(train, n_input)
    n_timesteps, n_features, n_outputs = train_x.shape[1], train
_x.shape[2], train_y.shape[1] # define model
    model = Sequential()
    model.add(GRU(50, activation='relu', input_shape=(n_timestep
s, n_features)))
    model.add(Dense(50, activation='relu'))
    model.add(Dense(n_outputs))
    model.compile(loss='mse', optimizer='adam')
    # fit network
    model.fit(train_x, train_y, epochs=20, batch_size=128, valid
ation_split=0.1, verbose=1, shuffle=True)
    return model
```

In [103]:

```
from keras.layers import Bidirectional
def BiLSTM_Model(train, n_input):
    # prepare data
    train_x, train_y = to_supervised(train, n_input)
    n_timesteps, n_features, n_outputs = train_x.shape[1], train
_x.shape[2], train_y.shape[1] # define model
    model = Sequential()
    model.add(Bidirectional(LSTM(50, activation='relu', input_sh
ape=(n_timesteps, n_features))))
    model.add(Dense(50, activation='relu'))
    model.add(Dense(n_outputs))
    model.compile(loss='mse', optimizer='adam')
    # fit network
    model.fit(train_x, train_y, epochs=15, batch_size=128, valid
ation_split=0.1, verbose=1, shuffle=True)
    return model


def BiGRU_Model(train, n_input):
    # prepare data
    train_x, train_y = to_supervised(train, n_input)
    n_timesteps, n_features, n_outputs = train_x.shape[1], train
_x.shape[2], train_y.shape[1] # define model
    model = Sequential()
    model.add(Bidirectional(GRU(50, activation='relu', input_sha
pe=(n_timesteps, n_features))))
    model.add(Dense(50, activation='relu'))
    model.add(Dense(n_outputs))
    model.compile(loss='mse', optimizer='adam')
    # fit network
    model.fit(train_x, train_y, epochs=15, batch_size=128, valid
ation_split=0.1, verbose=1, shuffle=True)
    return model
```

In [16]:

```
# make a forecast
def forecast(model, history, n_input):
  # flatten data
    data = array(history)
    data = data.reshape((data.shape[0]*data.shape[1], data.shape
[2]))
    # retrieve last observations for input data
    input_x = data[-n_input:, 0]
    # reshape into [1, n_input, 1]
    input_x = input_x.reshape((1, len(input_x), 1))
    # forecast the next week
    yhat = model.predict(input_x, verbose=0)
    # we only want the vector forecast
    yhat = yhat[0]
    return yhat
```

In [77]:

```
def evaluate_LSTM (train, test, n_input):
    M_LSTM = LSTM_Model(train, n_input)
    # history is a list of weekly data
    history = [x for x in train]
    # walk-forward validation over each week
    predictions_LSTM = list()
    for i in range(len(test)):
      # predict the week
        yhat_sequence_LSTM = forecast(M_LSTM, history, n_input)
        predictions_LSTM.append(yhat_sequence_LSTM)
        # get real observation and add to history for predicting
the next week
        history.append(test[i, :])
   # evaluate predictions days for each week
    predictions_LSTM = array(predictions_LSTM)
    score_rmse_LSTM, scores_rmse_LSTM, score_mae_LSTM, scores_ma
e_LSTM, score_r2_LSTM, scores_r2_LSTM = evaluate_forecasts(test[
:, :, 0], predictions_LSTM)
    return score_rmse_LSTM, scores_rmse_LSTM, score_mae_LSTM, sc
ores_mae_LSTM, score_r2_LSTM, scores_r2_LSTM
```

In [98]:

```
def evaluate_TCN(train, test, n_input):
    M_TCN = TCN_Model(train, n_input)
    # history is a list of weekly data
    history = [x for x in train]
    # walk-forward validation over each week
    predictions_TCN = list()
    for i in range(len(test)):
        yhat_sequence_TCN = forecast(M_TCN, history, n_input)
        predictions_TCN.append(yhat_sequence_TCN)
        # get real observation and add to history for predicting
the next week
        history.append(test[i, :])
    # evaluate predictions days for each week
    predictions_TCN = array(predictions_TCN)
    score_rmse_TCN, scores_rmse_TCN, score_mae_TCN, scores_mae_T
CN, score_r2_TCN, scores_r2_TCN = evaluate_forecasts(test[:, :,
0], predictions_TCN)
    return score_rmse_TCN, scores_rmse_TCN, score_mae_TCN, score
s_mae_TCN, score_r2_TCN, scores_r2_TCN
```

In [99]:

```
def evaluate_GRU(train, test, n_input):
    M_GRU = GRU_Model(train, n_input)
    # history is a list of weekly data
    history = [x for x in train]
    # walk-forward validation over each week
    predictions_GRU = list()
    for i in range(len(test)):
        # predict the week
        yhat_sequence_GRU = forecast(M_GRU, history, n_input)
        predictions_GRU.append(yhat_sequence_GRU)
        # get real observation and add to history for predicting
the next week
        history.append(test[i, :])
    # evaluate predictions days for each week
    predictions_GRU = array(predictions_GRU)
    score_rmse_GRU, scores_rmse_GRU, score_mae_GRU, scores_mae_G
RU, score_r2_GRU, scores_r2_GRU = evaluate_forecasts(test[:, :,
0], predictions_GRU)
    return  score_rmse_GRU, scores_rmse_GRU, score_mae_GRU, scor
es_mae_GRU, score_r2_GRU, scores_r2_GRU
```

In [108]:

```
# evaluate a single model
def evaluate_BiRNN (train, test, n_input):

    M_BiLSTM = BiLSTM_Model(train, n_input)
    M_BiGRU = BiGRU_Model(train, n_input)

  # history is a list of weekly data
    history = [x for x in train]
  # walk-forward validation over each week
    predictions_BiLSTM = list()
    predictions_BiGRU = list()
    for i in range(len(test)):
      # predict the week
        yhat_sequence_BiLSTM = forecast(M_BiLSTM, history, n_inp
ut)
        predictions_BiLSTM.append(yhat_sequence_BiLSTM)

        yhat_sequence_BiGRU = forecast(M_BiGRU, history, n_input
)
        predictions_BiGRU.append(yhat_sequence_BiGRU)
        # get real observation and add to history for predicting
the next week
        history.append(test[i, :])
  # evaluate predictions days for each week

    predictions_BiLSTM = array(predictions_BiLSTM)
    score_rmse_BiLSTM, scores_rmse_BiLSTM, score_mae_BiLSTM, sco
res_mae_BiLSTM, score_r2_BiLSTM, scores_r2_BiLSTM = evaluate_for
ecasts(test[:, :, 0], predictions_BiLSTM)

    predictions_BiGRU = array(predictions_BiGRU)
    score_rmse_BiGRU, scores_rmse_BiGRU, score_mae_BiGRU, scores
_mae_BiGRU, score_r2_BiGRU, scores_r2_BiGRU = evaluate_forecasts
(test[:, :, 0], predictions_BiGRU)

    return  score_rmse_BiLSTM, scores_rmse_BiLSTM, score_mae_BiL
STM, scores_mae_BiLSTM, score_r2_BiLSTM, scores_r2_BiLSTM, score
_rmse_BiGRU, scores_rmse_BiGRU, score_mae_BiGRU, scores_mae_BiGR
U, score_r2_BiGRU, scores_r2_BiGRU
```

In [92]:

```
n_input = 10
score_rmse_LSTM, scores_rmse_LSTM, score_mae_LSTM, scores_mae_LS
TM, score_r2_LSTM, scores_r2_LSTM= evaluate_LSTM(train, test, n_
input)
```

```
Train on 52155 samples, validate on 5796 samples
Epoch 1/20
52155/52155 [==============================] - 20s 3
92us/step - loss: 0.0118 - val_loss: 1.7229e-04
Epoch 2/20
52155/52155 [==============================] - 15s 2
89us/step - loss: 6.4834e-04 - val_loss: 2.0024e-04
Epoch 3/20
52155/52155 [==============================] - 16s 3
03us/step - loss: 6.0703e-04 - val_loss: 1.5973e-04
Epoch 4/20
52155/52155 [==============================] - 16s 3
09us/step - loss: 5.9432e-04 - val_loss: 1.6389e-04
Epoch 5/20
52155/52155 [==============================] - 24s 4
55us/step - loss: 5.8896e-04 - val_loss: 2.0867e-04
Epoch 6/20
52155/52155 [==============================] - 28s 5
36us/step - loss: 5.7309e-04 - val_loss: 1.9777e-04
Epoch 7/20
52155/52155 [==============================] - 22s 4
24us/step - loss: 5.7369e-04 - val_loss: 1.4926e-04
Epoch 8/20
52155/52155 [==============================] - 20s 3
89us/step - loss: 5.6770e-04 - val_loss: 3.1019e-04
Epoch 9/20
52155/52155 [==============================] - 23s 4
44us/step - loss: 5.6659e-04 - val_loss: 1.8735e-04
Epoch 10/20
52155/52155 [==============================] - 24s 4
59us/step - loss: 5.6894e-04 - val_loss: 1.3064e-04
Epoch 11/20
52155/52155 [==============================] - 22s 4
23us/step - loss: 5.5735e-04 - val_loss: 1.3831e-04
Epoch 12/20
52155/52155 [==============================] - 24s 4
70us/step - loss: 5.5903e-04 - val_loss: 1.3009e-04
```

```
Epoch 13/20
52155/52155 [==============================] - 27s 5
27us/step - loss: 5.5646e-04 - val_loss: 1.2875e-04
Epoch 14/20
52155/52155 [==============================] - 27s 5
25us/step - loss: 5.4796e-04 - val_loss: 1.3167e-04
Epoch 15/20
52155/52155 [==============================] - 26s 5
06us/step - loss: 5.5792e-04 - val_loss: 2.2923e-04
Epoch 16/20
52155/52155 [==============================] - 25s 4
86us/step - loss: 5.5855e-04 - val_loss: 1.3081e-04
Epoch 17/20
52155/52155 [==============================] - 25s 4
83us/step - loss: 5.4947e-04 - val_loss: 1.3456e-04
Epoch 18/20
52155/52155 [==============================] - 31s 5
85us/step - loss: 5.4861e-04 - val_loss: 4.5581e-04
Epoch 19/20
52155/52155 [==============================] - 28s 5
46us/step - loss: 5.4497e-04 - val_loss: 1.3201e-04
Epoch 20/20
52155/52155 [==============================] - 21s 4
05us/step - loss: 5.4565e-04 - val_loss: 1.7346e-04
```

In [105]:

```
score_rmse_TCN, scores_rmse_TCN, score_mae_TCN, scores_mae_TCN,
score_r2_TCN, scores_r2_TCN = evaluate_TCN(train, test, n_input)
```

```
Train on 52155 samples, validate on 5796 samples
Epoch 1/15
52155/52155 [==============================] - 38s 7
34us/step - loss: 0.0497 - val_loss: 0.0040
Epoch 2/15
52155/52155 [==============================] - 29s 5
48us/step - loss: 0.0017 - val_loss: 4.1338e-04
Epoch 3/15
52155/52155 [==============================] - 29s 5
63us/step - loss: 6.4571e-04 - val_loss: 4.5483e-04
Epoch 4/15
52155/52155 [==============================] - 32s 6
06us/step - loss: 5.9263e-04 - val_loss: 1.6541e-04
Epoch 5/15
52155/52155 [==============================] - 33s 6
```

```
38us/step - loss: 5.8016e-04 - val_loss: 1.3016e-04
Epoch 6/15
52155/52155 [==============================] - 32s 6
08us/step - loss: 5.7172e-04 - val_loss: 1.3677e-04
Epoch 7/15
52155/52155 [==============================] - 32s 6
19us/step - loss: 5.6560e-04 - val_loss: 1.2996e-04
Epoch 8/15
52155/52155 [==============================] - 33s 6
24us/step - loss: 5.5348e-04 - val_loss: 1.2644e-04
Epoch 9/15
52155/52155 [==============================] - 32s 6
22us/step - loss: 5.5520e-04 - val_loss: 1.4788e-04
Epoch 10/15
52155/52155 [==============================] - 32s 6
18us/step - loss: 5.4997e-04 - val_loss: 1.3262e-04
Epoch 11/15
52155/52155 [==============================] - 35s 6
63us/step - loss: 5.4684e-04 - val_loss: 1.6116e-04
Epoch 12/15
52155/52155 [==============================] - 37s 7
05us/step - loss: 5.4682e-04 - val_loss: 1.3019e-04
Epoch 13/15
52155/52155 [==============================] - 34s 6
44us/step - loss: 5.4454e-04 - val_loss: 1.2654e-04
Epoch 14/15
52155/52155 [==============================] - 38s 7
28us/step - loss: 5.4221e-04 - val_loss: 1.6039e-04
Epoch 15/15
52155/52155 [==============================] - 41s 7
90us/step - loss: 5.4236e-04 - val_loss: 1.8737e-04
```

In [106]:

```
score_rmse_GRU, scores_rmse_GRU, score_mae_GRU, scores_mae_GRU,
score_r2_GRU, scores_r2_GRU = evaluate_GRU(train, test, n_input)
```

```
Train on 52155 samples, validate on 5796 samples
Epoch 1/20
52155/52155 [==============================] - 22s 4
20us/step - loss: 0.0107 - val_loss: 1.4593e-04
Epoch 2/20
52155/52155 [==============================] - 18s 3
45us/step - loss: 5.7943e-04 - val_loss: 1.5307e-04
Epoch 3/20
```

```
52155/52155 [==============================] - 20s 3
91us/step - loss: 5.6838e-04 - val_loss: 1.3947e-04
Epoch 4/20
52155/52155 [==============================] - 21s 4
11us/step - loss: 5.5881e-04 - val_loss: 1.3767e-04
Epoch 5/20
52155/52155 [==============================] - 22s 4
17us/step - loss: 5.5248e-04 - val_loss: 2.3204e-04
Epoch 6/20
52155/52155 [==============================] - 21s 4
03us/step - loss: 5.5244e-04 - val_loss: 1.3045e-04
Epoch 7/20
52155/52155 [==============================] - 22s 4
27us/step - loss: 5.4595e-04 - val_loss: 1.8119e-04
Epoch 8/20
52155/52155 [==============================] - 23s 4
40us/step - loss: 5.4601e-04 - val_loss: 1.4386e-04
Epoch 9/20
52155/52155 [==============================] - 25s 4
70us/step - loss: 5.4626e-04 - val_loss: 1.2695e-04
Epoch 10/20
52155/52155 [==============================] - 20s 3
75us/step - loss: 5.4841e-04 - val_loss: 1.3517e-04
Epoch 11/20
52155/52155 [==============================] - 19s 3
61us/step - loss: 5.4471e-04 - val_loss: 1.2860e-04
Epoch 12/20
52155/52155 [==============================] - 20s 3
85us/step - loss: 5.4346e-04 - val_loss: 1.3090e-04
Epoch 13/20
52155/52155 [==============================] - 19s 3
72us/step - loss: 5.3870e-04 - val_loss: 1.5510e-04
Epoch 14/20
52155/52155 [==============================] - 16s 3
07us/step - loss: 5.4457e-04 - val_loss: 1.3504e-04
Epoch 15/20
52155/52155 [==============================] - 17s 3
20us/step - loss: 5.4196e-04 - val_loss: 1.3237e-04
Epoch 16/20
52155/52155 [==============================] - 17s 3
28us/step - loss: 5.3826e-04 - val_loss: 1.2766e-04
Epoch 17/20
52155/52155 [==============================] - 17s 3
25us/step - loss: 5.4464e-04 - val_loss: 2.9623e-04
Epoch 18/20
```

```
52155/52155 [==============================] - 17s 3
32us/step - loss: 5.4306e-04 - val_loss: 1.4274e-04
Epoch 19/20
52155/52155 [==============================] - 17s 3
20us/step - loss: 5.3988e-04 - val_loss: 1.4302e-04
Epoch 20/20
52155/52155 [==============================] - 18s 3
37us/step - loss: 5.4486e-04 - val_loss: 1.2682e-04
```

In [109]:

```
score_rmse_BiLSTM, scores_rmse_BiLSTM, score_mae_BiLSTM, scores_
mae_BiLSTM, score_r2_BiLSTM, scores_r2_BiLSTMM, score_rmse_BiGRU
, scores_rmse_BiGRU, score_mae_BiGRU, scores_mae_BiGRU, score_r2
_BiGRU, scores_r2_BiGRU = evaluate_BiRNN (train, test, n_input)
```

```
Train on 52155 samples, validate on 5796 samples
Epoch 1/15
52155/52155 [==============================] - 34s 6
50us/step - loss: 0.0090 - val_loss: 1.8568e-04
Epoch 2/15
52155/52155 [==============================] - 27s 5
19us/step - loss: 6.3241e-04 - val_loss: 1.6485e-04
Epoch 3/15
52155/52155 [==============================] - 28s 5
32us/step - loss: 5.9285e-04 - val_loss: 1.7362e-04
Epoch 4/15
52155/52155 [==============================] - 26s 4
93us/step - loss: 5.8941e-04 - val_loss: 1.3669e-04
Epoch 5/15
52155/52155 [==============================] - 32s 6
09us/step - loss: 5.8327e-04 - val_loss: 1.4439e-04
Epoch 6/15
52155/52155 [==============================] - 33s 6
26us/step - loss: 5.6869e-04 - val_loss: 1.7668e-04
Epoch 7/15
52155/52155 [==============================] - 29s 5
63us/step - loss: 5.7407e-04 - val_loss: 1.5432e-04
Epoch 8/15
52155/52155 [==============================] - 30s 5
74us/step - loss: 5.7446e-04 - val_loss: 1.7127e-04
Epoch 9/15
52155/52155 [==============================] - 31s 5
94us/step - loss: 5.6165e-04 - val_loss: 1.3414e-04
Epoch 10/15
```

```
52155/52155 [==============================] - 33s 6
42us/step - loss: 5.6493e-04 - val_loss: 1.6385e-04
Epoch 11/15
52155/52155 [==============================] - 33s 6
27us/step - loss: 5.5306e-04 - val_loss: 1.4939e-04
Epoch 12/15
52155/52155 [==============================] - 32s 6
17us/step - loss: 5.6420e-04 - val_loss: 1.2853e-04
Epoch 13/15
52155/52155 [==============================] - 32s 6
21us/step - loss: 5.4854e-04 - val_loss: 3.0260e-04
Epoch 14/15
52155/52155 [==============================] - 30s 5
82us/step - loss: 5.5238e-04 - val_loss: 1.3038e-04
Epoch 15/15
52155/52155 [==============================] - 29s 5
62us/step - loss: 5.4696e-04 - val_loss: 1.4361e-04
Train on 52155 samples, validate on 5796 samples
Epoch 1/15
52155/52155 [==============================] - 32s 6
22us/step - loss: 0.0115 - val_loss: 1.4623e-04
Epoch 2/15
52155/52155 [==============================] - 27s 5
12us/step - loss: 5.7766e-04 - val_loss: 1.3799e-04
Epoch 3/15
52155/52155 [==============================] - 28s 5
29us/step - loss: 5.5898e-04 - val_loss: 1.3631e-04
Epoch 4/15
52155/52155 [==============================] - 25s 4
85us/step - loss: 5.4894e-04 - val_loss: 1.4876e-04
Epoch 5/15
52155/52155 [==============================] - 26s 4
98us/step - loss: 5.4934e-04 - val_loss: 1.5531e-04
Epoch 6/15
52155/52155 [==============================] - 29s 5
60us/step - loss: 5.4777e-04 - val_loss: 1.3100e-04
Epoch 7/15
52155/52155 [==============================] - 27s 5
21us/step - loss: 5.4385e-04 - val_loss: 1.4326e-04
Epoch 8/15
52155/52155 [==============================] - 27s 5
21us/step - loss: 5.3989e-04 - val_loss: 1.3752e-04
Epoch 9/15
52155/52155 [==============================] - 27s 5
15us/step - loss: 5.4431e-04 - val_loss: 1.3855e-04
```

```
Epoch 10/15
52155/52155 [==============================] - 27s 5
26us/step - loss: 5.3978e-04 - val_loss: 1.3355e-04
Epoch 11/15
52155/52155 [==============================] - 29s 5
53us/step - loss: 5.4013e-04 - val_loss: 1.2979e-04
Epoch 12/15
52155/52155 [==============================] - 28s 5
46us/step - loss: 5.4246e-04 - val_loss: 1.3680e-04
Epoch 13/15
52155/52155 [==============================] - 26s 5
08us/step - loss: 5.4038e-04 - val_loss: 1.3233e-04
Epoch 14/15
52155/52155 [==============================] - 27s 5
23us/step - loss: 5.3817e-04 - val_loss: 1.3095e-04
Epoch 15/15
52155/52155 [==============================] - 29s 5
60us/step - loss: 5.4508e-04 - val_loss: 1.4251e-04
```

In [145]:

```
# define the names and functions for the models we wish to evalu
ate

score_rmse_CNN, scores_rmse_CNN, score_mae_CNN, scores_mae_CNN,
score_r2_CNN, scores_r2_CNN = evaluate_CNN(train, test, n_input)
```

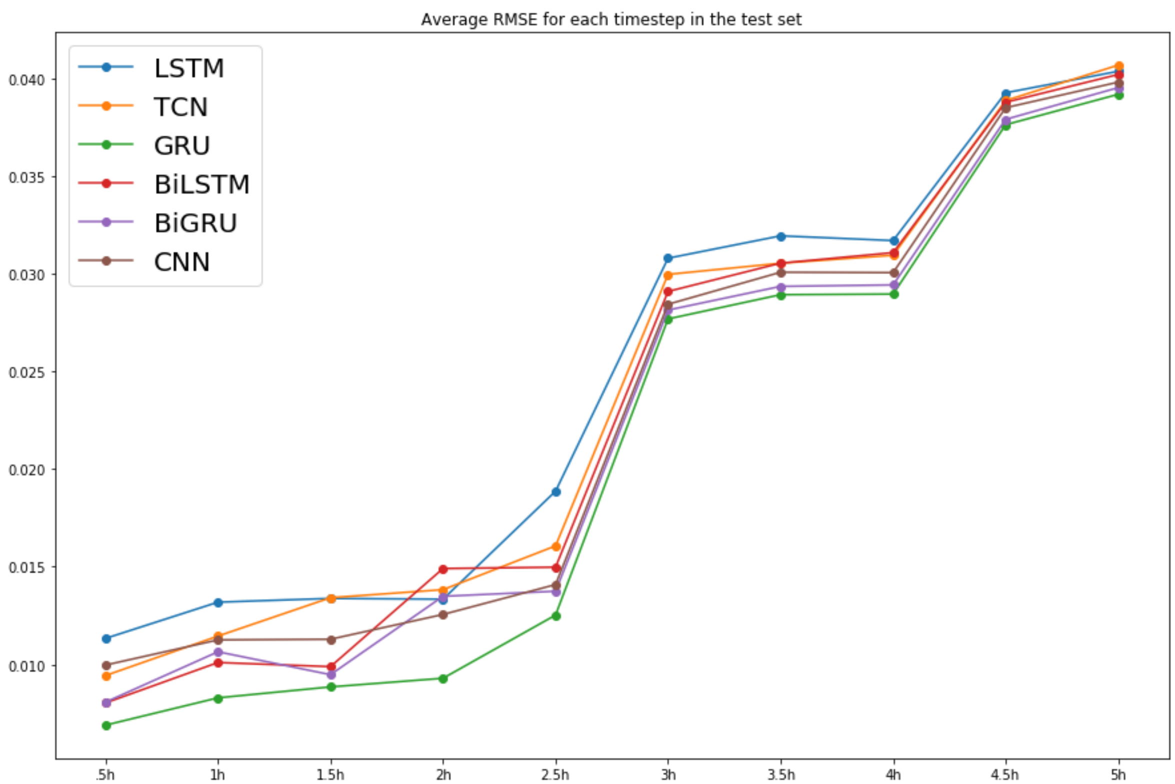
```
Train on 52155 samples, validate on 5796 samples
Epoch 1/20
52155/52155 [==============================] - 8s 15
1us/step - loss: 0.0112 - val_loss: 3.2939e-04
Epoch 2/20
52155/52155 [==============================] - 2s 30
us/step - loss: 7.3287e-04 - val_loss: 2.2902e-04
Epoch 3/20
52155/52155 [==============================] - 2s 34
us/step - loss: 6.7074e-04 - val_loss: 1.8800e-04
Epoch 4/20
52155/52155 [==============================] - 2s 32
us/step - loss: 6.3908e-04 - val_loss: 1.6425e-04
Epoch 5/20
52155/52155 [==============================] - 2s 31
us/step - loss: 6.1625e-04 - val_loss: 1.5132e-04
Epoch 6/20
```

```
52155/52155 [==============================] - 2s 31
us/step - loss: 5.9984e-04 - val_loss: 1.4367e-04
Epoch 7/20
52155/52155 [==============================] - 2s 31
us/step - loss: 5.8618e-04 - val_loss: 1.8215e-04
Epoch 8/20
52155/52155 [==============================] - 2s 31
us/step - loss: 5.7658e-04 - val_loss: 1.3892e-04
Epoch 9/20
52155/52155 [==============================] - 2s 32
us/step - loss: 5.7410e-04 - val_loss: 1.5984e-04
Epoch 10/20
52155/52155 [==============================] - 2s 31
us/step - loss: 5.6960e-04 - val_loss: 2.6646e-04
Epoch 11/20
52155/52155 [==============================] - 2s 30
us/step - loss: 5.6745e-04 - val_loss: 1.5619e-04
Epoch 12/20
52155/52155 [==============================] - 2s 31
us/step - loss: 5.6940e-04 - val_loss: 1.3686e-04
Epoch 13/20
52155/52155 [==============================] - 2s 31
us/step - loss: 5.6097e-04 - val_loss: 1.3285e-04
Epoch 14/20
52155/52155 [==============================] - 2s 30
us/step - loss: 5.6393e-04 - val_loss: 1.3405e-04
Epoch 15/20
52155/52155 [==============================] - 2s 31
us/step - loss: 5.5976e-04 - val_loss: 1.3858e-04
Epoch 16/20
52155/52155 [==============================] - 2s 32
us/step - loss: 5.5846e-04 - val_loss: 1.3554e-04
Epoch 17/20
52155/52155 [==============================] - 2s 31
us/step - loss: 5.5931e-04 - val_loss: 1.3265e-04
Epoch 18/20
52155/52155 [==============================] - 2s 31
us/step - loss: 5.5846e-04 - val_loss: 1.4083e-04
Epoch 19/20
52155/52155 [==============================] - 2s 31
us/step - loss: 5.6220e-04 - val_loss: 1.9075e-04
Epoch 20/20
52155/52155 [==============================] - 2s 31
us/step - loss: 5.5359e-04 - val_loss: 1.3674e-04
```

In [ ]:

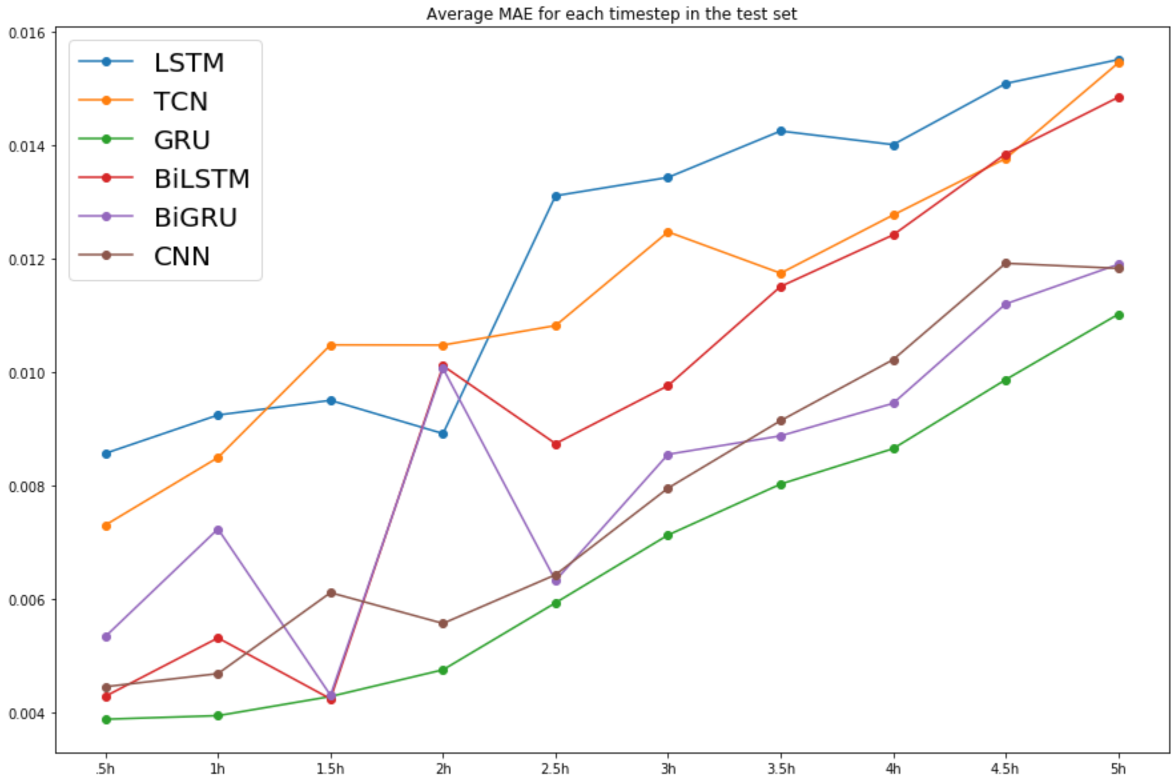
In [147]:

```
# plot scores
fig=plt.figure()
plt.title('Average RMSE for each timestep in the test set')
plt.rcParams['figure.figsize'] = (15, 10)
timesteps = ['.5h', '1h', '1.5h', '2h', '2.5h', '3h', '3.5h','4h
', '4.5h','5h']
plt.plot(timesteps, scores_rmse_LSTM, marker='o', label='LSTM')
plt.plot(timesteps, scores_rmse_TCN, marker='o', label='TCN')
plt.plot(timesteps, scores_rmse_GRU, marker='o', label='GRU')
plt.plot(timesteps, scores_rmse_BiLSTM, marker='o', label='BiLST
M')
plt.plot(timesteps, scores_rmse_BiGRU, marker='o', label='BiGRU'
)
plt.plot(timesteps, scores_rmse_CNN, marker='o', label='CNN')
plt.legend(fontsize=20)
plt.show()
fig.savefig('RMSE.png')
```



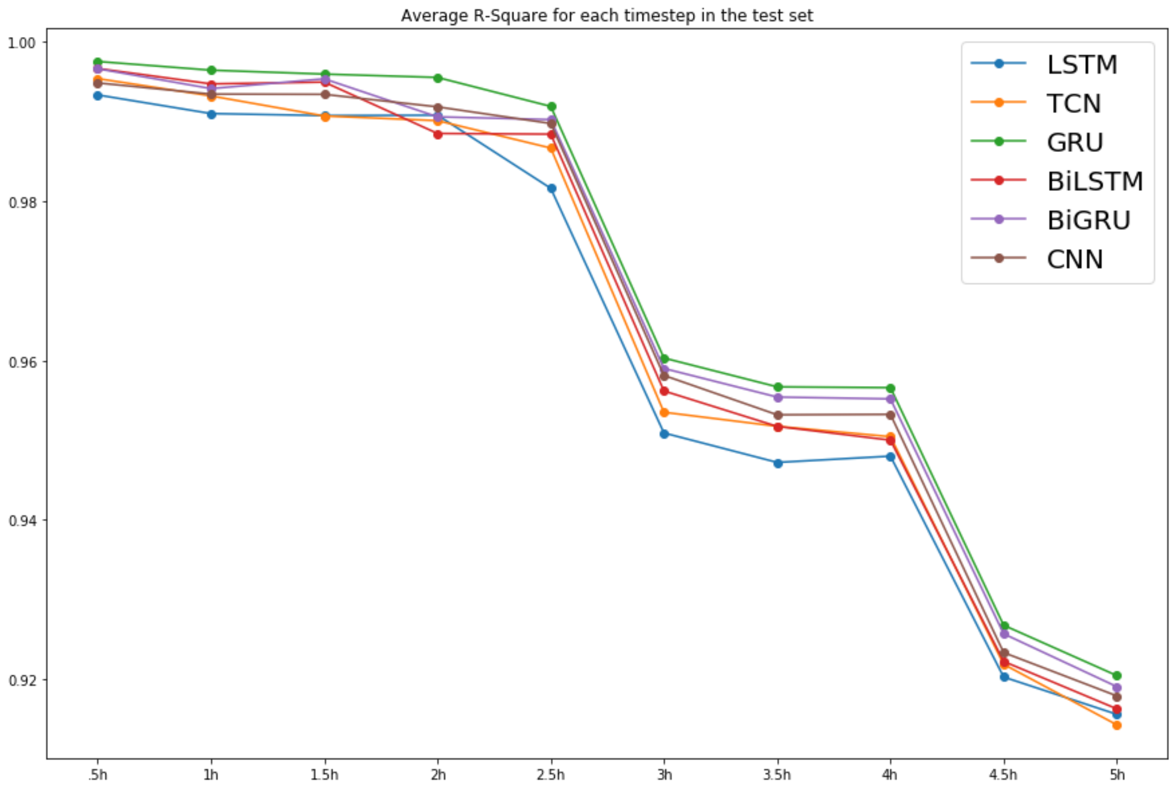
In [148]:

```
# plot scores
fig=plt.figure()
plt.title('Average MAE for each timestep in the test set')
plt.rcParams['figure.figsize'] = (15, 10)
timesteps = ['.5h', '1h', '1.5h', '2h', '2.5h', '3h', '3.5h','4h
', '4.5h','5h']
plt.plot(timesteps, scores_mae_LSTM, marker='o', label='LSTM')
plt.plot(timesteps, scores_mae_TCN, marker='o', label='TCN')
plt.plot(timesteps, scores_mae_GRU, marker='o', label='GRU')
plt.plot(timesteps, scores_mae_BiLSTM, marker='o', label='BiLSTM
')
plt.plot(timesteps, scores_mae_BiGRU, marker='o', label='BiGRU')
plt.plot(timesteps, scores_mae_CNN, marker='o', label='CNN')
plt.legend(fontsize=20)
plt.show()
fig.savefig('MAE.png')
```



In [149]:

```
fig=plt.figure()
plt.title('Average R-Square for each timestep in the test set')
plt.rcParams['figure.figsize'] = (15, 10)
timesteps = ['.5h', '1h', '1.5h', '2h', '2.5h', '3h', '3.5h','4h
', '4.5h','5h']
plt.plot(timesteps, scores_r2_LSTM, marker='o', label='LSTM')
plt.plot(timesteps, scores_r2_TCN, marker='o', label='TCN')
plt.plot(timesteps, scores_r2_GRU, marker='o', label='GRU')
plt.plot(timesteps, scores_r2_BiLSTMM, marker='o', label='BiLSTM
')
plt.plot(timesteps, scores_r2_BiGRU, marker='o', label='BiGRU')
plt.plot(timesteps, scores_r2_CNN, marker='o', label='CNN')
plt.legend(fontsize=20)
plt.show()
fig.savefig('R2.png')
```



In [ ]: