Artificial Neural Network and Deep Learning
Time Series Forecasting

Simone Sorrenti

1. Introduction

This report discusses the time series forecasting of 7 different features. In particular, 65528 time steps have been provided for each feature and the goal is to predict the subsequent 864 time steps.
The correlation between these characteristics varied; some had strong correlations, others slight, if not absent, correlations between them.

To make the predictions, different recurrent neural networks were designed based on the use of LSTM (Long-Short Term Memory).

In particular, the following different types of architects have been implemented:

- Residual Bidirectional LSTM
- CONV-LSTM
- CONV-Bidirectional LSTM
- Seq2Seq
- Stacked Bidirectional LSTM

2. Sub-Sequences

The dataset provided presented a single temporal sequence and since the model needs to learn a function that maps a sequence of future observations to a sequence of past observations, I proceeded to divide the single sequence into multiple subsequences composed of a smaller number of temporal observations.

As regards the past observations, 18 time windows were considered, while for the prediction of future observations 3 telescopes were considered, finally, as regards the translation between one subsequence and another, 6 different strides were used.

Logically, these parameters were not all applied for each model implemented as some models required a significant amount of memory that could not be satisfied; therefore, the parameters applied depended on the complexity of the model.

Window	9, 18, 36, 72, 144, 288, 432, 576, 864, 1152, 1440, 1728, 2016, 2304, 2592, 2880, 3168, 3456
Telescope	144, 288, 864
Stride	1, 2, 4, 8, 16, 32

3. Pre-processing

Different pre-processing of the data were carried out, in particular:

- Normalization: rescale the values into a range of [0,1] or [-1,1]
- Standardization: rescale data to have a mean of 0 and a standard deviation of 1
- Stationary: make the time series not dependent on time
- No pre-processing

4. Splitting of data

Initially, for the first models implemented, the dataset was divided into training set and validation set with the following proportions: 80% and 20% or 90% and 10%. Subsequently it was noticed that the validation set subtracted the most recent time sequences for learning and therefore providing during the model learning also these last more recent sequences the model made better predictions both on a smaller validation set (of a number limited number of examples) and on the test set.

5. RNN Model

- **CONV-Bidirectional LSTM and CONV-LSTM**

  The models that implemented this architecture were composed of a series of stacked levels that carried out the convolutional 1D on the input time series, subsequently the characteristics extracted from the filters were supplied to a series of stacked levels of LSTM and/or bi-directional LSTM and finally to a number of various dense levels.
  Batch Normalization levels have been inserted between the dense levels but also between the convolutional levels while for the latter in particular, Max Pooling and Global Averaging Pooling levels have been applied to their outputs.
  Furthermore, it has been tried to apply filters of different kernels on the same input and to concatenate their outputs for the next level.
  Finally, to mitigate overfitting, Dropout levels have been added as well as using Glorot weights initialization for all levels.

- **Residual Bidirectional LSTM**

  The models that implemented this architecture consist of a series of stacked bi-directional LSTM levels and finally a number of various dense levels.
  The peculiarity of this architecture, from which it takes its name, is the presence of shortcut paths from previous levels to subsequent levels for efficient training of deep networks with multiple LSTM layers.
  Batch normalization levels have been inserted between dense levels but also between LSTM levels.
  Finally, to mitigate overfitting, Dropout levels have been added and the use of Glorot weights initialization for all levels.

- **Seq2Seq Bidirection LSTM**

  Seq2Seq is a type of Encoder-Decoder model using RNN. In particular, stacked bidirectional LSTM levels were used for both the encoder and the decoder.
  The output of the encoder consists of the last state which is repeated N times by means of a RepeatVector level in order to provide N observations to the decoder.
  The states of the LSTM levels of the decoder are initialized with the final states of the LSTM levels of the encoder. The model terminated with a simple dense layer inserted into a TimeDistributed layer.

- **Stacked Bidirectional LSTM**
  The models that implemented this architecture consist of a series of stacked bi-directional LSTM levels, a Flatten layer and finally several dense levels.
  A TimeDistributed layer with the relu function is applied to the output of each Bi-LSTM level.
  Batch normalization levels have been inserted between dense levels but also between LSTM levels.
  Finally, to mitigate overfitting, Dropout levels have been added and the use of Glorot weights initialization for all levels.

6. Training

The training was carried out with the aim of minimizing the ‘Mean squared error’ (MSE) function as it was dealing with a regression of several features. In addition, 2 inherent metrics were used: Mean absolute error (MAE) and ‘Root mean square error’ (RMSE), and on them was applied early stopping with a patience from 10 to 25.
Different learning rates and epochs were applied in a range from 1e-1 to 1e-4 and in a range from 50 to 150.
Finally, a various batches were applied in a range from 16 to 128.

Loss function	Mean squared error

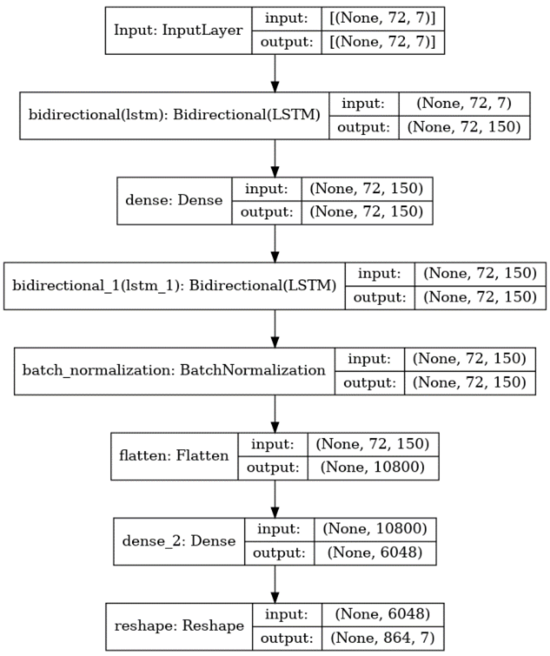
Metrics	Mean absolute error, Root mean square error
Early Stopping	Root mean square error with patience [15, 25]
Learning rates	[1e-1, 1e-4]
Epochs	[50, 150]
Cross-Validation	Hold out with 80%/20% or 90%/10% or 99%/1%
Batch	16, 32, 64, 128
Dropout	0.1, 0.2

7. Results

The best model (Model 13) scored on the test set:

- RMSE: 4.26
- MAE: 2.88
- Sponginess RMSE: 1.62
- Wonder RMSE: 2.52
- Crunchiness RMSE: 6.65
- Loudness RMSE: 2.15
- Meme RMSE: 1.23
- Soap RMSE: 4.41
- Hype RMSE: 6.95
- First Quarter RMSE: 3.63
- Second Quarter RMSE: 4.41
- Third Quarter RMSE: 4.67

Architecture of Model 13




```
Layer (type)              Output Shape          Param #
===========================================================
Input (InputLayer)        [(None, 72, 7)]       0
___________________________________________________________
bidirectional (Bidirectional (None, 72, 150)    49800
___________________________________________________________
dense (Dense)             (None, 72, 150)       22650
___________________________________________________________
bidirectional_1 (Bidirection (None, 72, 150)    13560
___________________________________________________________
batch_normalization (BatchNo (None, 72, 150)    600
___________________________________________________________
flatten (Flatten)         (None, 10800)         0
___________________________________________________________
dense_2 (Dense)           (None, 6048)          65324448
___________________________________________________________
reshape (Reshape)         (None, 864, 7)        0
===========================================================
Total params: 65,533,098
Trainable params: 65,532,798
Non-trainable params: 300
```

I got the best score using this model which implements an architecture ‘Stacked Bidirectional LSTM’.
In particular, it stacks 2 bidirectional LSTM levels and between the 2 levels there is a dense level. Finally, the output of the last LSTM is normalized and then flatten to then be supplied at a dense level with 6048 neurons. Finally, the reshape is performed.
No pre-processing was applied to the data.

Window	72
Telescope	864
Stride	16
Learning rates	1e-3
Epochs	50
Batch	64
Dropout	0.2