paper06_creating_models_for_train

October 31, 2021

1 Timeseries Testing and Modeling

Jorge III Altamirano-Astorga, Ita-Andehui Santiago, Luz Aurora Hernández.

Prof.: Edgar Francisco Román-Rangel.

```
                     temperature    pressure  …  wind_speed    wind_deg
datetime                                      …
2021-02-12 06:00:00    21.530000  777.410000  …    2.565310  109.799270
2021-02-12 06:05:00    21.689773  777.389432  …    2.456273  105.132299

[2 rows x 20 columns]
```

1.1 Timeseries

We use the the `timeseries_dataset_from_array` function from Keras Timeseries modeling functions. This function creates dataframes with sliding windows over time as an array.

This function work as follow:

```
tf.keras.utils.timeseries_dataset_from_array(
    X, Y,              # our dataset
    sequence_length,   #  Length of the output sequences (in number of timesteps): we need to text different s
    sequence_stride=1, # 1 is the default value for s, data[i], data[i + s], data[i + 2 * s]
    sampling_rate=1,   # 1 is the default value for timesteps data[i], data[i + r], ... data[i + sequence_leng
    shuffle=False,     # shuffle the records sorting, we set it in the default false value as the original ord
    seed=None,         # we set a fixed seed to have repeatable experiments
    #...
)
```

On our research it is relevant to have an adequate sequence length. Then we can focus a brief research on empiric good times. Our proposal are:

1.1.1 1 Minute Resampling

- 2 days before: i.e. on our resampling for every 5 min we'd have 2880 records. This is because 2 $days \times 24$ $hours \times 60$ $min \div 1$ min
- 7 days before: i.e. on our resampling for every 5 min we'd have 10,080 records. This is because 7 $days \times 24$ $hours \times 60$ $min \div 1$ min
- 15 days before: i.e. on our resampling for every 5 min we'd have 21600 records. This is because 15 $days \times 24$ $hours \times 60$ $min \div 1$ min

1.1.2 2 Minute Resampling

- 2 days before: i.e. on our resampling for every 5 min we'd have 1,440 records. This is because 2 $days \times 24$ $hours \times 60$ $min \div 2$ min
- 7 days before: i.e. on our resampling for every 5 min we'd have 5040 records. This is because 7 $days \times 24$ $hours \times 60$ $min \div 2$ min
- 15 days before: i.e. on our resampling for every 5 min we'd have 10,080 records. This is because 15 $days \times 24$ $hours \times 60$ $min \div 2$ min

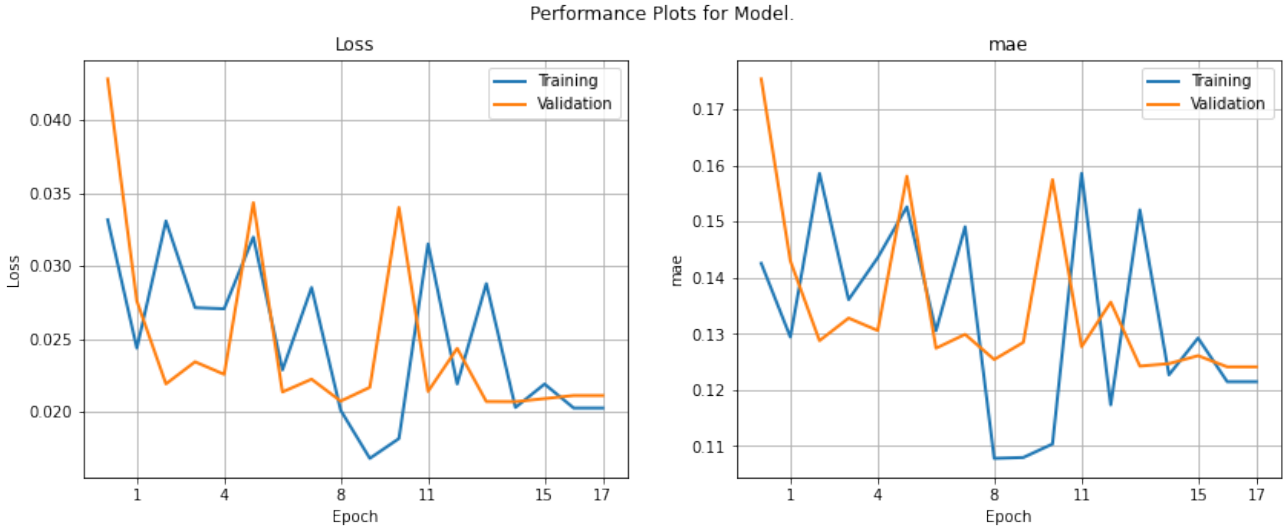
1.1.3 5 Minute Resampling

- 2 days before: i.e. on our resampling for every 5 min we'd have 576 records. This is because 2 $days \times 24\ hours \times 60\ min \div 5\ min$
- 7 days before: i.e. on our resampling for every 5 min we'd have 2016 records. This is because 7 $days \times 24\ hours \times 60\ min \div 5\ min$
- 15 days before: i.e. on our resampling for every 5 min we'd have 4320 records. This is because 15 $days \times 24\ hours \times 60\ min \div 5\ min$

We set this the number of days in a variable we set as `WINDOW_SIZE_DAYS`

1.2 5 Minute Resampling and 7 Days of History.

`Processing Time: 262.98 segundos.`

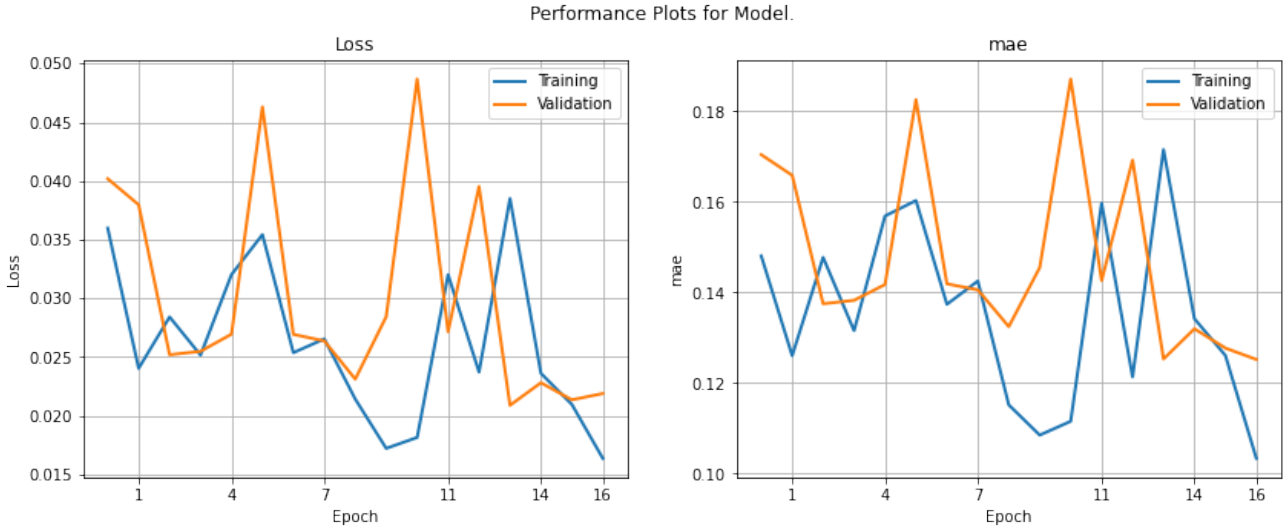


- Mean Absolute Error in Real Scale of the last Epoch: **61.5097 IAQ points**.

1.3 5 Minute Resampling and 2 Days of History with 2 Sampling Rate.

On our previous examples this quickly becomes unmanageable the we propose skipping some records and getting the previous hour by setting the `sampling_rate` parameter of 10 minute.

`Processing Time: 215.13 segundos.`

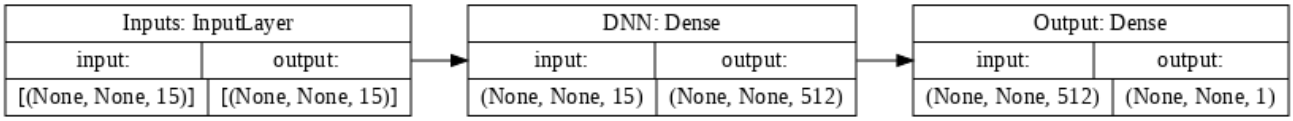


- Mean Absolute Error in Real Scale of the last Epoch: **62.0461 IAQ points**.

1.4 Model DNN01

Time series parameters:

```
timeseries_dataset_from_array(  sequence_length=576,  sampling_rate=1,  batch_size=256,  seed=175904)
```

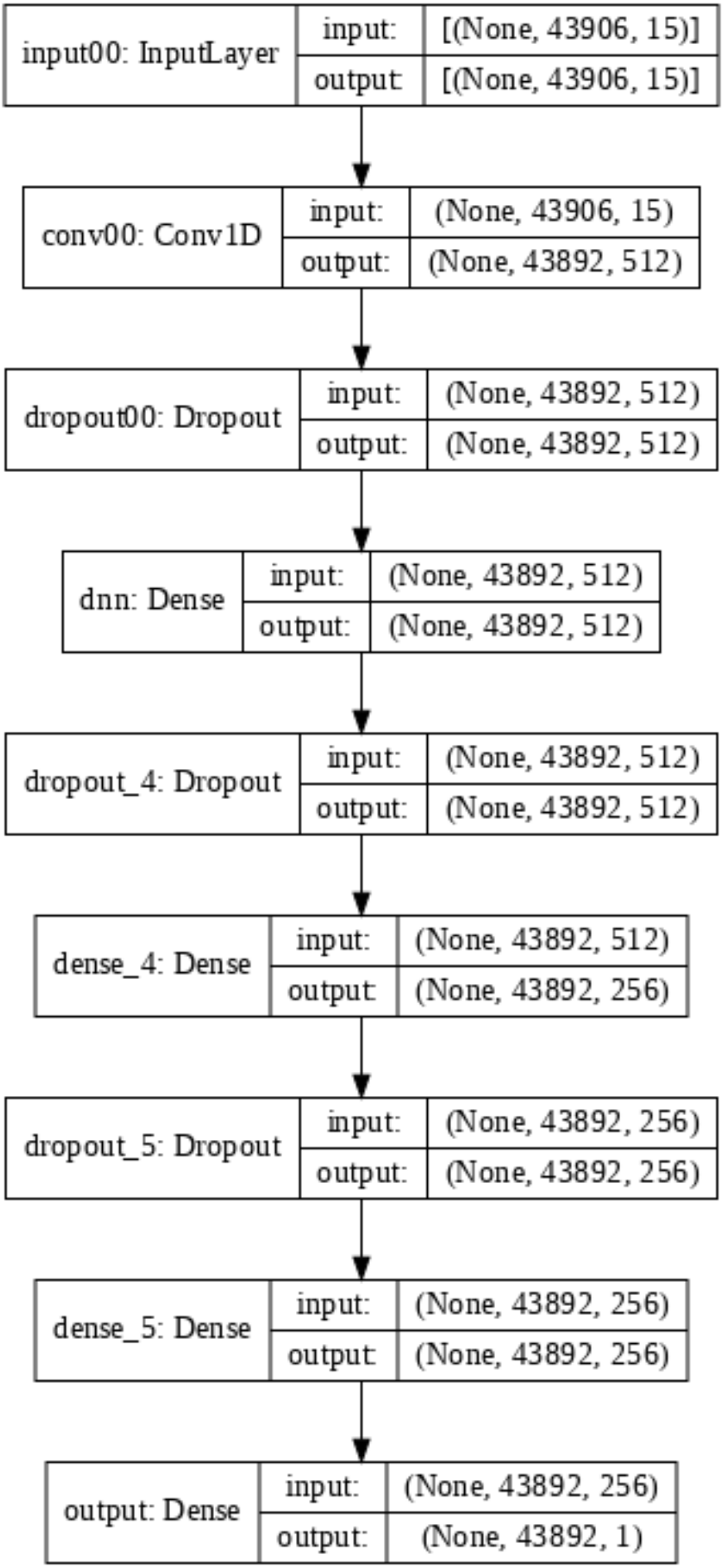



```
Saving file: drive/MyDrive/Colab Notebooks/proyecto-final/models-paper/scaler-
iaq.dill… Done!
```

1.5 Model Best 03a

Time series parameters:

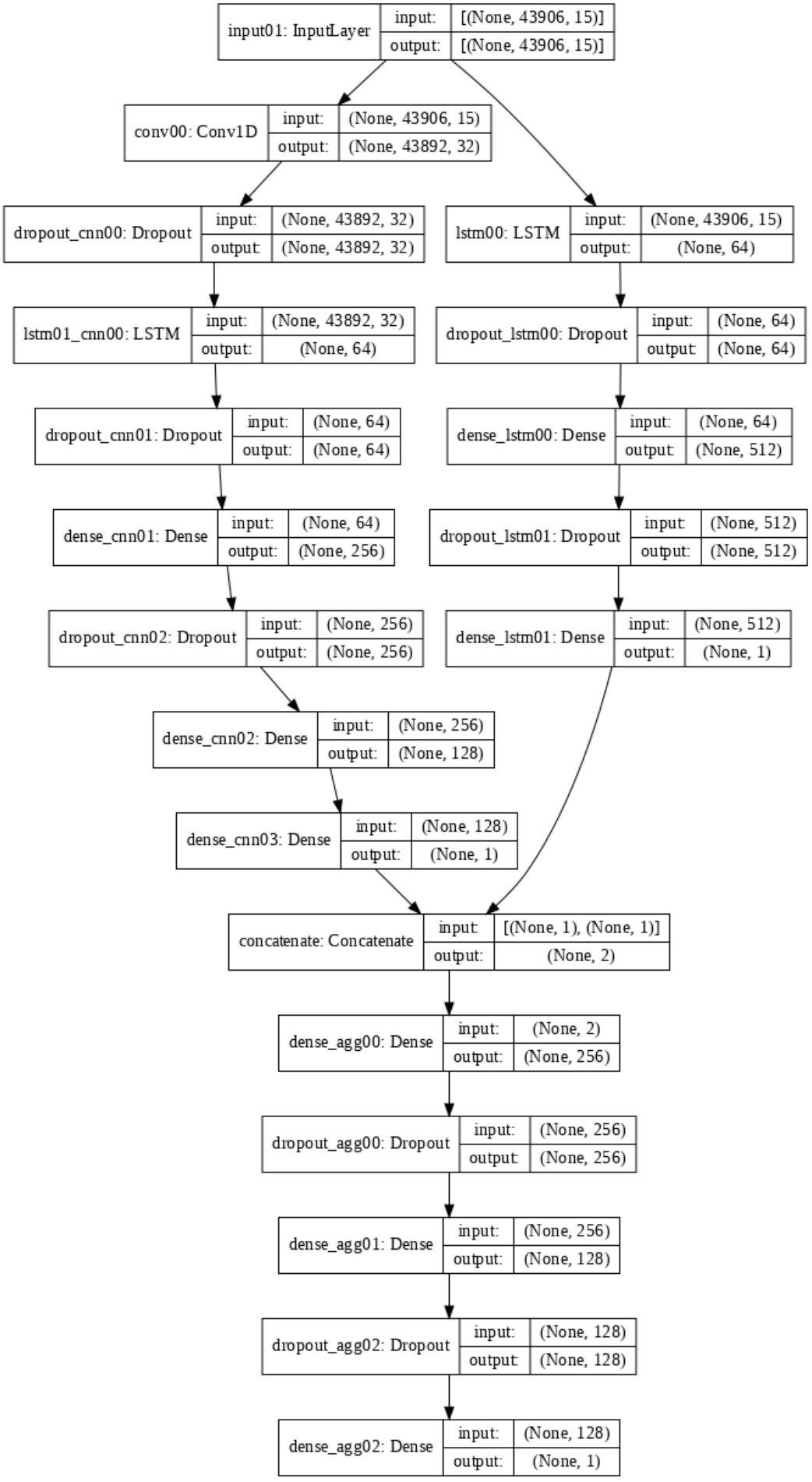
```
timeseries_dataset_from_array(  sequence_length=576,  sampling_rate=1,  batch_size=256,  seed=175904)
```


input00: InputLayer
input: [(None, 43906, 15)]
output: [(None, 43906, 15)]
conv00: Conv1D
input: (None, 43906, 15)
output: (None, 43892, 512)
dropout00: Dropout
input: (None, 43892, 512)
output: (None, 43892, 512)
dnn: Dense
input: (None, 43892, 512)
output: (None, 43892, 512)
dropout_4: Dropout
input: (None, 43892, 512)
output: (None, 43892, 512)
dense_4: Dense
input: (None, 43892, 512)
output: (None, 43892, 256)
dropout_5: Dropout
input: (None, 43892, 256)
output: (None, 43892, 256)
dense_5: Dense
input: (None, 43892, 256)
output: (None, 43892, 256)
output: Dense
input: (None, 43892, 256)
output: (None, 43892, 1)

1.6 Model Best 03b

Time series parameters:

```
timeseries_dataset_from_array(  sequence_length=576,  sampling_rate=1,  batch_size=256,  seed=175904)
```


input01: InputLayer
input: [(None, 43906, 15)]
output: [(None, 43906, 15)]
conv00: Conv1D
input: (None, 43906, 15)
output: (None, 43892, 32)
dropout_cnn00: Dropout
input: (None, 43892, 32)
output: (None, 43892, 32)
lstm00: LSTM
input: (None, 43906, 15)
output: (None, 64)
lstm01_cnn00: LSTM
input: (None, 43892, 32)
output: (None, 64)
dropout_lstm00: Dropout
input: (None, 64)
output: (None, 64)
dropout_cnn01: Dropout
input: (None, 64)
output: (None, 64)
dense_lstm00: Dense
input: (None, 64)
output: (None, 512)
dense_cnn01: Dense
input: (None, 64)
output: (None, 256)
dropout_lstm01: Dropout
input: (None, 512)
output: (None, 512)
dropout_cnn02: Dropout
input: (None, 256)
output: (None, 256)
dense_lstm01: Dense
input: (None, 512)
output: (None, 1)
dense_cnn02: Dense
input: (None, 256)
output: (None, 128)
dense_cnn03: Dense
input: (None, 128)
output: (None, 1)
concatenate: Concatenate
input: [(None, 1), (None, 1)]
output: (None, 2)
dense_agg00: Dense
input: (None, 2)
output: (None, 256)
dropout_agg00: Dropout
input: (None, 256)
output: (None, 256)
dense_agg01: Dense
input: (None, 256)
output: (None, 128)
dropout_agg02: Dropout
input: (None, 128)
output: (None, 128)
dense_agg02: Dense
input: (None, 128)
output: (None, 1)