Deep Learning Autonomous Lab 2

Recurrent Neural Networks

Santiago Bernal

April 2018

1 Introduction

For this lab, various experiments where performed using Recurrent Neural Networks with Keras. The data used for these experiments is the Air Quality of London. This dataset has different sites that are geographically close and offers 5 variables (PM10 (Particulate Matter 10 um (ug/m3)), NO (Nitric Oxide (ug/m3)), NO2 (Nitrogen Dioxide (ug/m3)), O3 (Ozone (ug/m3)) and Wind Speed (m/s)) and other 4 which correspond to the date of the measurement (month, day, day of week and hour). The experiments consisted of the ones listed in the autonomous lab and a few others that where considered interesting to perform.

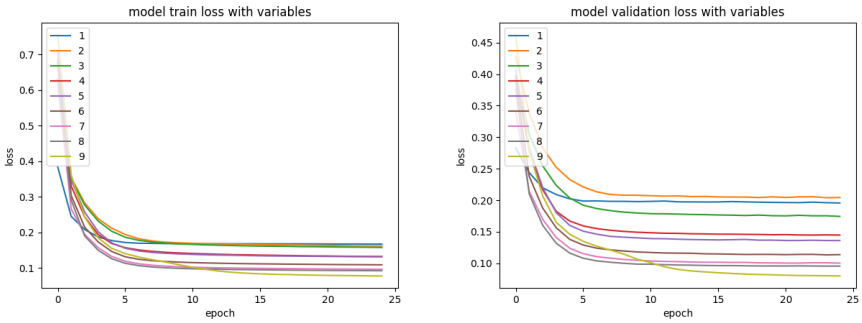
2 Experiments

2.1 Complementary Variables

This experiment consisted in changing the code in order to process the additional variables in the dataset. The original code used only the PM10 variable, the idea was to use the rest of them and observe the results. For this, the way that the dataset was generated was changed in order to include the samples with the variables needed. For this, we generate a lagged matrix of the dataset of size (samples, sequences, variables) and separate the result into the corresponding training and testing sets. Also, the model architecture was changed, considering that the original model only needed to predict one variable, it had a Dense output layer of only 1 unit, where in this case it needed to be adapted to the amount of variables to predict. The experiments made where as follows:

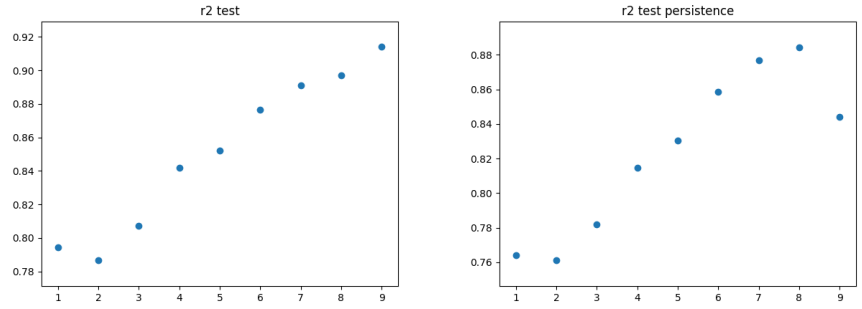
- Using various variables
  - Changes: the code was changed to train the model using various variables, calculating the MSE/R2 for each amount. The model used is a LTSM layer with the input shape as that of the training data and the output 32 units, and a final Dense layer for mapping the results using the amount of variables as outputs. The activation function used was hyperbolic tangent (tanh) and hard sigmoid for recurrent activation. The training was done using batches of size 500, over 25 epochs, using a learning rate of 0.001 and the adam optimizer.
  - Expected: With more variables, it should be easier for the model to make predictions, given that the variables are related to the output (the variables aren't random data that will confuse the model), it gives the model more context on how the output is reached and what causes it.
  - Actual results: As expected, by changing the amount of variables, the model obtains better results, with some variables like the hour having a bigger impact than the rest. With 3 variables, the training seems to maintain about the same loss, but the validation accuracy

starts increasing greatly after the 3 variables are included but there is always an increase after adding more variables, as can be seen on figure 1. The greatest change in the accuracy seems to be using 4 variables, and in the end with all the variables, the loss was able to be reduced to less than 0.10 using all the variables which is pretty high. But the increase in the last variable may not be so good as it has a huge impact in the difference between the R2 persistence and the R2 obtained, being the R2 obtained around 0.08 higher while the others only have around 0.02 difference. One of the reason for this may be that chance correlations in the sample data with this variable cause the R2 to inflate or that the data is being over-fitted, this can be later proved by changing the input window size or adding a dropout. But overall, as more variables are considered in the model, the better accuracy it was able to obtain. The R2 graphs are shown on figure 2



(a) Train loss (b) Validation loss

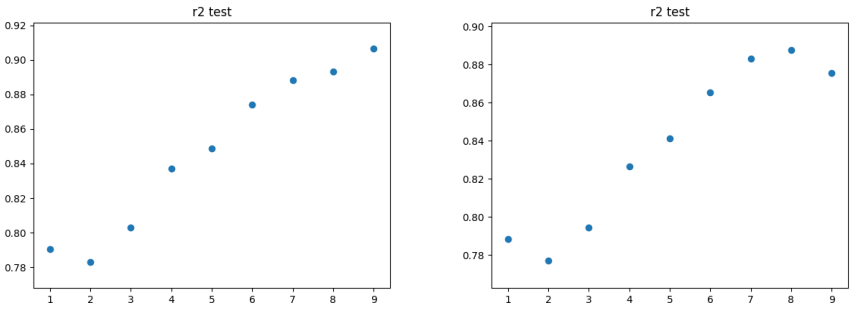
Figure 1: Train and validation loss for first experiment



(a) R2 test (b) R2 Persistence

Figure 2: R2 test and persistence for first experiment

- Changing the dropout used in the model
  - Changes: The configuration of the model was changed to use the same architecture as before but adding a dropout of 0.5 and 0.9 on the data.
  - Expected: As mentioned before, some changes may occur to the resulting scores and R2 measures if the data was over-fitted, there is a suspicion that it is, this based on the difference in R2 and R2 persistence obtained after evaluating the model with all the 9 variables
  - Actual results: The suspicion seems to have been right, considering the difference obtained with these test to the original ones, especially looking at the 0.9 dropout. We will focus mostly on the case of using 9 variables and the 0.9 dropout for the comparison since it has the most difference. The first thing to notice is that loss in the 0.9 dropout experiments with 9 variables is higher than the original case with actually the results with 8 variables being reaching the best loss, being the lowest obtained. Also, the R2 results and the R2 persistence for the last variable are closer than the original model. These changes seem to only greatly affect the last variable, the rest of the variables have changes as well but they're not as steep as with 9 variables. With this we can assume that the model seems to tend to over-fit when including the last variable. The graphs of the R2 for both are on figure 3



(a) R2 test 0.5 dropout

(b) R2 Test 0.9 dropout

Figure 3: R2 test for 0.5 and 0.9 dropout

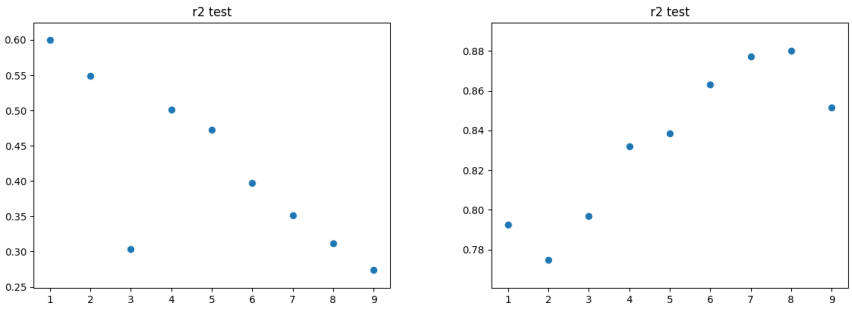
- Changing the amount of units in the model
  - Changes: The output units of the LSTMs are changed to: 1, 9, 64 and 128. The final model would then be: a LSTM layer same as before but with the changed output for each case, and the final dense layer for activation of the output variables. The model is re-trained in each case and the prediction re-calculated

– Expected: With 1 unit, the model should be really bad, and not make good predictions, especially when having more than one variable, since there wouldn't be much separating the activation of the different variables, making the prediction task harder for the dense layer. For the rest of the output units, answers shouldn't be too far apart from the previous experiments, since a high accuracy was already obtained. There might be a slight increase but it shouldn't be too significant.

– Actual results:

  * 1 unit: as expected the results are worse than with the original architecture, obtaining the best results with only one variable and then spiraling down as more variables where used.
  * 9 units: surprisingly, with 9 units we get pretty much the same as the original output, being the original a little better, but without the over-fitting that was detected before when using 9 variables.
  * 64 and 128 units: for more than 32 units, the results seem to change very slightly, but not enough to be able to establish any actual difference between using 32 or more.

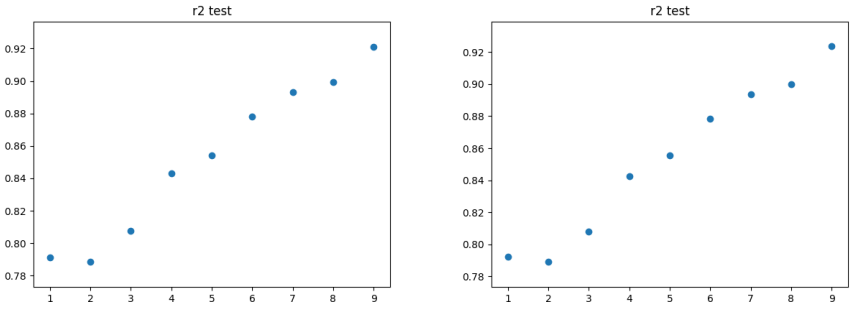
  The r2 results for 1 and 9 units can be found on figure 4 and the results for 64 and 128 on figure 5



(a) R2 test 1 neuron

(b) R2 Test 9 neurons

Figure 4: R2 test for 1 and 9 units

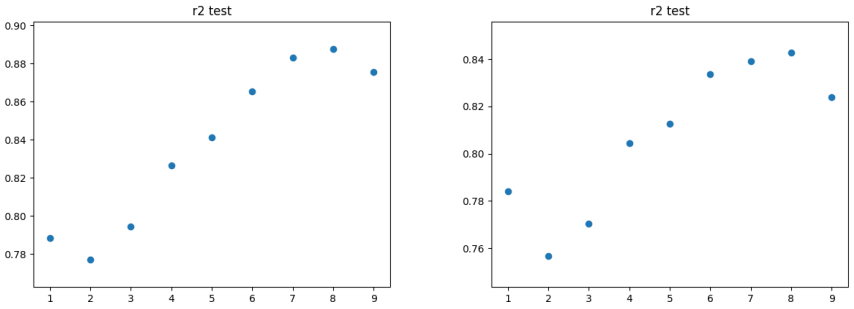


(a) R2 test 64 neurons

(b) R2 Test 128 neurons

Figure 5: R2 test for 64 and 128 neurons

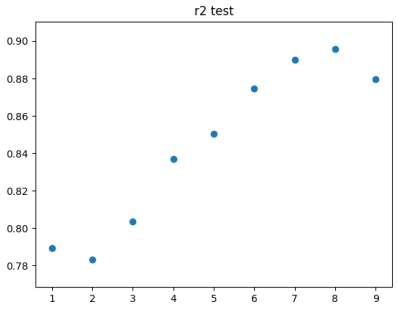
- Changing the amount of units and the dropout
  - Changes: Combines the same that was done before for the changes of units and the dropout, also considering the results obtained before, only configurations that cause notable changes where used. The combinations where: 1 unit 0.9 dropout, 9 units 0.9 dropout, since 32 units with dropout was already done and in the previous experiment it was shown that more units doesn't add much changes, they weren't considered.
  - Expected: The dropout shouldn't change much for the results obtained for the changes of the units, considering that there wasn't any signs of over-fitting that could be softened using dropout.
  - Actual results: Surprisingly, for 1 neuron and 0.9 dropout, the results where greatly increased when predicting only one and three variables. As for, 9 neurons and 0.9 dropout, the results where actually worse, looking at the R2, we can see a general decrease with all the options of variables comparing with the previous tests. These results can be seen in figure 6



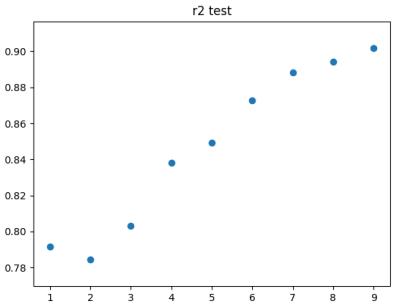
(a) R2 test 1 neuron with 0.9 dropout (b) R2 Test 9 neurons with 0.9 dropout

Figure 6: R2 test for 1 and 9 neurons with 0.9 dropout

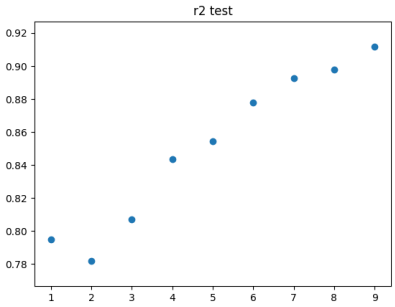
- Changing the input window
  - Changes: Using the original architecture (with 32 neurons and 0.0 dropout) we change configurations for the lag value, changing the input window size. Configurations chosen: 2, 12 and 32
  - Expected: changing the input window size may have various effects, one may be that by increasing it too much, the model will be able to make more accurate predictions since it will have more context but it may also over-fit on the data. By decreasing the window size, the model may not be precise since it will be harder for it to make predictions if it doesn't have enough context beforehand.
  - Actual results:
    - 2 lag: it seems to obtain pretty much the same results as with the original experiments, it even obtains an R2 result closer to the persistence model without the difference that was found originally when using all 9 variables. This could mean that the data may be easily predictable, so it doesn't need much context to learn it and be able to make predictions as seen on figure 7a
    - 12 and 32 lag: For more lag, the results are very similar to the original measurements made in the first experiment, having basically the same characteristics, as seen on figure 8



(a) R2 test 2 lag



(a) R2 test 12 lag

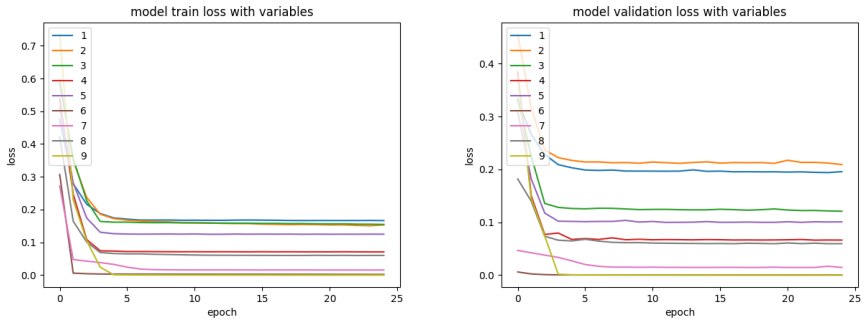


(b) R2 Test 32 lag

Figure 8: R2 test for 12 and 32 lag

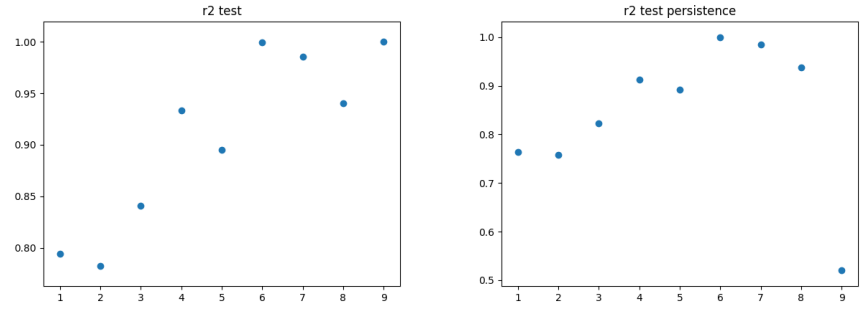
- Using the variables separately Instead of processing using an additional variable on each iteration, a few tests where done separating each variable and training the model with only that information.
  - Changes: Using the same architecture from the previous experiments, we train the model and test it using only one variable, changing the one used over each iteration.
  - Expected: The processing of each individual variable should yield results similar to the ones found before, although it's hard to say if the increase in accuracy would still be present, considering that the model would be getting less data for it to base it's prediction on. Although, this could as well help it by not getting biased by unrelated.
  - Actual results: unlike the previous results, there is not a curve like increase in the results, rather, it has more spikes and sudden changes. This is understandable considering that only one variable is processed

in this test so its only that result, instead of the previous test in which an accumulation of the variables made it soften the spikes thanks to previous recollected data. Interestingly, and similarly to the previous tests, it seems that the last variable has some weird effects on the model, where even though the R2 score is pretty high the R2 persistence is at a very low value, this can be seen on figure 10. The train and validation loss graphs also show how the model reduced the loss to 0 within a few iterations as shown on figure 9.



(a) Train loss for separate variables (b) Validation loss

Figure 9: Train and validation loss for separate variables



(a) R2 test (b) R2 Persistence

Figure 10: R2 test and persistence for separate variables

- Using the variables separately with dropout Considering the results obtained before, training the model with a dropout may have interesting effects, especially considering what was obtained in the final variable.
  - Changes: Using the previous experiment on each variable separately, a dropout of 0.9 was added to the model.

- Expected results: the results should be about the same as previously, but variables like the last one should decrease in accuracy, since it had a tendency of being over-fit.
- Actual results: As expected, and similarly like in previous experiments, the results pretty much maintained the same, expect for the last variable which had a significant drop, as seen in 11

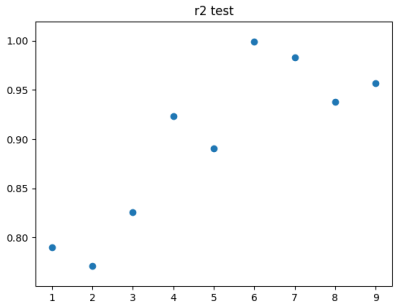
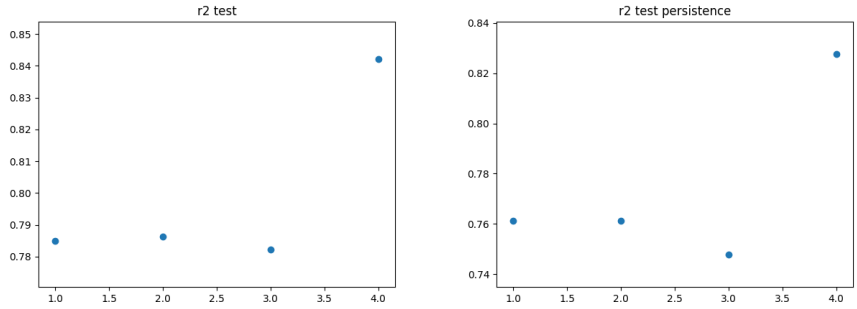


Figure 11: R2 test for separate vars with dropout

2.2 Complementary Sites

The original experiments where made using data measurements on one location. The dataset also offers multiple locations that are geographically close with the same measured variables. For these experiments, the predictions are made using data from other sites as well as the original one.

- One variable and multiple sites
  - Changes: The model was trained using the data of different sites instead of just one. All the options are then compared to each other with the results obtained when using 1, 2, 3 or 4 sites combined.
  - Expected: With information of different sites, given that the sites are geographically close, the model may be able to establish a better context of how the predictions can be made to obtain better results. For the rest of the configurations, the same from the first experiment was used.
  - Actual results: For the first sites it doesn't seem to have much effect, in fact, for two sites the accuracy increases slightly but then decreases again when it uses three sites. But, there is a significant increase when having four sites, which can be seen on figure 12

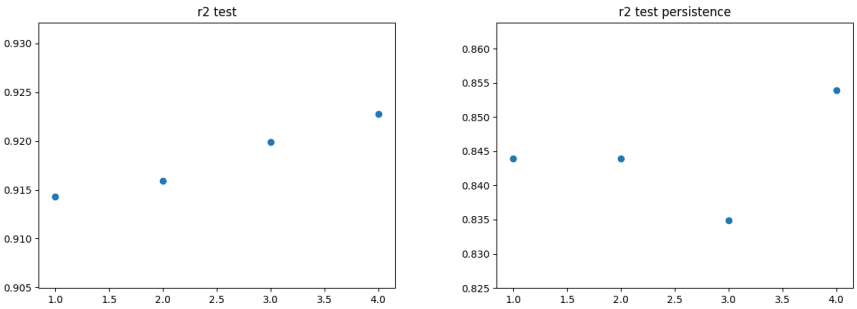


(a) R2 test with multiple sites

(b) R2 persistence with multiple sites

Figure 12: R2 test and persistence for multiple sites

- All variables and all sites
  - Changes: Using the same code as before but with all 9 of the variables, the model was trained to also use the information with varying amounts of sites.
  - Expected: Considering the results that have been obtained so far, when using 9 variables the model was able to make predictions with a high R2 score. Combining that with what was found in the previous experiment, the model should obtain high accuracy and improve with more sites, with a possible spike when using all 4 sites.
  - Actual results: Basically what was expected became true, but surprisingly the spike when using all 4 sites is no longer present. In fact, there is about the same increase in accuracy between 2 and 3 sites, as there is in between 3 and 4 sites. The results for this experiment are on figure 13



(a) R2 test with multiple sites and all variables

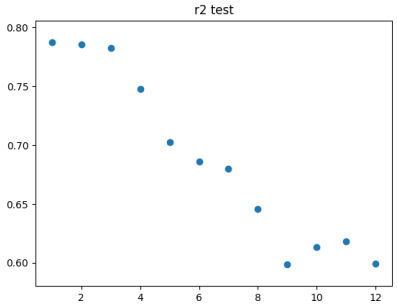
(b) R2 persistence with multiple sites and all variables

Figure 13: R2 test and persistence for multiple sites

2.3 Multi step Prediction

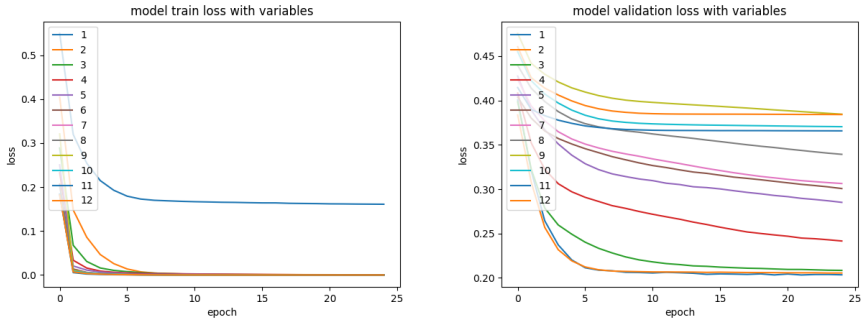
Multi-steps can be predicted using the model by adding the predicted value to the measurements and discarding the oldest one and moving on and so forth until we advance to t+n measurements in "the future". For these experiments, we use the air quality dataset to evaluate the models prediction capabilities in this multi-step.

- Multi-step prediction
  - Changes: Using the original architecture from the first experiment, we use the values predicted by the model as a new measurement and discard the oldest one in the input window. This way, we attempt to predict more into the future, several steps ahead. An additional of 24 steps where added to see how the model changes over the course of the predictions
  - Expected: Considering the fact that the predictions are going to be made using as reference other predictions, it's highly likely that the accuracy of the new predictions aren't going to be that good. It's like the text prediction in a virtual keyboard, it may be good at guessing the first few words that you wanted to input, but after a while the next words added don't make much sense.
  - Actual results: As expected, as we progress adding more and more predictions, the quality of them decrease. The training and validation loss can be viewed in 15 and the results on figure 14



(a) R2 test with 12 steps

Figure 14: R2 test for 12 steps

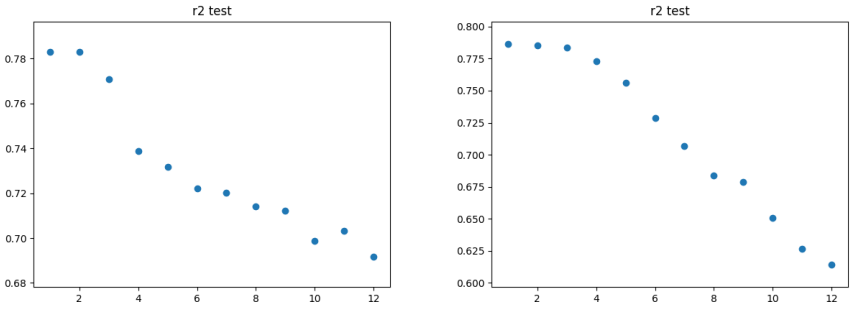


(a) Train loss with 12 steps

(b) Validation loss with 12 steps

Figure 15: Validation and training loss for multiple steps

- Multi-step prediction with different input window
  - Changes: changing the input sequence size to a smaller and larger sizes and seeing how that affects the multi-step predictions. The sizes chosen where an input window of: 2 and 10, using 12 steps prediction.
  - Expected: For the small input window, the accuracy of the results should decrease faster since more predictions will be used. For the large input window this should be the other way around, with the predictions not being able to affect the outcome too much, though there should be some decrease in accuracy.
  - Actual results: For a small input window, the accuracy is less at the beginning and decreases slowly, not as fast as the original test with an input window of 6. After 4 predictions, the decreasing curve flattens a bit and becomes slower. For a larger input window, the first few iterations get a pretty high accuracy but then it decreases rapidly getting even less accuracy than with the small input window. The results can be seen on figure 16



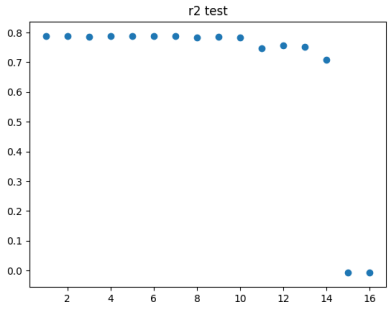
(a) 2 lag r2 test

(b) 10 lag R2 test

Figure 16: 2 and 10 lag with 12 steps

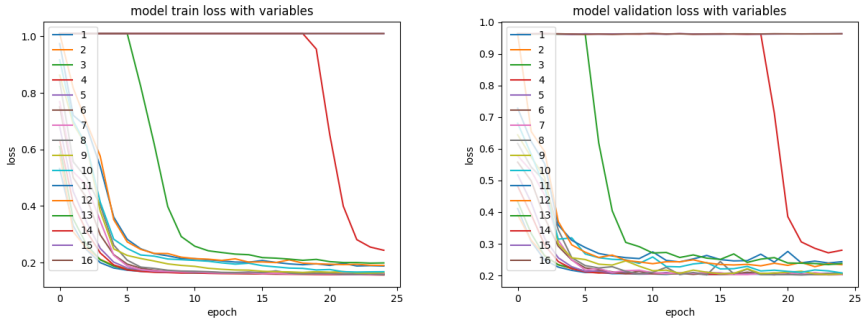
2.4 Sequence to Sequence

- Sequence to sequence prediction
  - Changes: The original network from the first experiment was adapted to use sequence to sequence predictions. For this we chose different lengths for the sequence, using up to 16 sequences.
  - Expected: Since the model's objective is to predict different variable relating to air quality, using sequence to sequence doesn't actually make much sense, since it shouldn't be needed. Considering this, the results shouldn't really vary from the original measurements.
  - Actual results: the results really don't vary that much at the beginning it seems, with some times getting a bit better and others a bit worse. After around 12 sequences, it seems that the accuracy drops a bit more and then at 15 it completely plummets and the model becomes really inaccurate. The results can be viewed on figure 18



(a) R2 test with 12 steps

Figure 17: R2 test for 16 sequence



(a) Train loss for 16 sequence (b) Validation loss for 16 sequence

Figure 18: Validation and training loss for multiple steps

3 Appendix