Recurrent Neural Networks

Juan Lao Tebar

December 11, 2017

1 Introduction

In this practice we explore the effects of the different elements of RNNs using a specific dataset[1]. In this case we extend the Wind prediction example that we used in the previous guided session.

In the original example we predicted one step of the future wind—next 15 minutes—using a window of previous measurements. Here we extend the problem in different ways:

Complementary variables In the original experiment we used only the wind speed as input data. Now we include also the air density, the temperature and the pressure.

Complementary sites In the original experiment we predicted the wind speed using the data from one site, but the dataset includes three additional sites that are geographically close (they are in the vertices of a 2 km square). Here we leverage the information of the wind speed from the other sites.

Multi step prediction Here we obtain a multi step prediction using the original model simply by adding the value predicted to the current set of measurements and discarding the oldest one.

Sequence to sequence prediction An alternative to shifting the input and adding the predictions to have a multi step prediction is a network that links a window of measurements to a window of predictions.

In this practice we change the stopping criteria of the training process, as we remove the limit of epochs and we implement a stopper based on the validation loss: when we detect a minimum value, we stop the training after 30 epochs if it does not decrease. We consider that we should not stop the training process just because we reached an arbitrary number of epochs if the model does not show signs of convergence.

In the guided laboratory we trained the model with 50,000 samples, validated with 10,000 and tested with 10,000. In this practice we use the whole dataset,

[1] `https://upc-mai-dl.github.io/rnn-lab-autonomous/`

training our model with 122,736 samples, validating with 61,368 and testing with 61,368 for no apparent reason[2].

The code related to this work is public and available at github[3].

2 Baseline

Before performing any experiment, we must define a baseline model in order to compare the results and determine how every change affects the accuracy of the system.

Our baseline model is similar to the one proposed in the guided laboratory: input window size of 6 steps, 1 output, 512 LSTM units in one single hidden layer and no dropout, trained with RMSprop—with a learning rate of 0.0001, $\rho = 0.9$, $\varepsilon = 1 \cdot 10^{-8}$ and no learning rate decay.

Table 1 shows the results obtained with this model.

Training loss	Validation loss	Test loss
0.02729	0.02924	0.03339

Table 1: Results obtained with the baseline model.

3 Complementary Variables

3.1 Plan

In this session we modify the architecture of the system adequately for taking all the variables as input—but still giving only one variable as output—and we study how the MSE changes when we change the size of hidden layer, when we apply dropout and when we change the size of input window.

- In the 1st experiment we test different layer sizes: 32, 128, 512 and 1024 units.
- In the 2nd experiment we test different dropout factors applied in the LSTM layer, including its gates: 0%, 25%, 50%, 75%.
- In the 3rd experiment we test different dropout factors only applied on inter-layer connections, between the hidden and the output layer: 0%, 25%, 50%, 75%.
- In the 4th experiment we test different input window sizes: 1, 3, 6, 12, 24, 48. In this experiment, in the case of a window size of 48, we had to clip all parameter gradients to a maximum norm of 1 in order to prevent exploding gradients.

[2] Actually, the reason behind this decision is just that we performed some dummy tests using the whole dataset and we forgot to set the default value again.

[3] `https://github.com/juanlao7/DL-lab2`

3.2 Expectations

Regarding to learning techniques:

- In theory, adding more units to the hidden layer leads to better training accuracies. However, the model may lose power of generalization, overfitting the training data and offering worse validation and test accuracies.
- If the network overfits the training data, we expect that applying a dropout factor will successfully regularize the model. However, the dropout implementation we use—"dropout" argument in Keras implementation of LSTM layers—affects also to the LSTM gates, thus it is possible that the model will end up performing even worse due to the vanishing gradient problem. For this reason we also test an inter-layer dropout, that does not affect to LSTM gates. On the other hand, if the model does not overfit, dropout will slightly reduce the gap between training and validation losses, but not in a considerable way.
- We suspect that in this dataset there are no data relations over long intervals of time, hence we will not obtain a better result by increasing the window size. However, decreasing it may lead to a worse accuracy.

Regarding to this specific problem:

- More information—variables—are always helpful and the model may find new inter-variable relations that lead to a better accuracy.

3.3 Results

- Table 2 shows the results obtained after testing different layer sizes, while figure 1 shows the evolution of the training and validation loss in each case.
- Table 3 shows the results obtained after testing different dropout factors, while figure 2 shows the evolution of the training and validation loss in each case.
- Table 4 shows the results obtained after testing different inter-layer dropout factors, while figure 3 shows the evolution of the training and validation loss in each case.
- Table 5 shows the results obtained after testing different input window sizes, while figures 4 and 5 show the evolution of the training and validation loss in each case.

	Training loss	Validation loss	Test loss
32 LSTM units	0.02770	0.02939	0.03327
128 LSTM units	0.02621	0.02835	0.03218
512 LSTM units	0.02562	0.02911	0.03266
1024 LSTM units	0.02531	0.02883	0.03247

Table 2: Results obtained after testing different layer sizes.

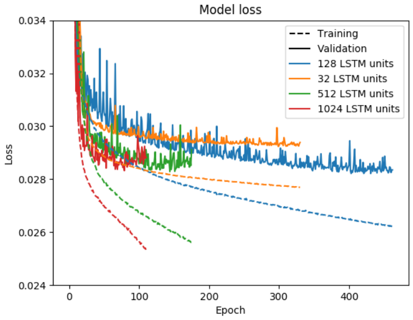

Figure 1: Validation and training loss with each layer size.

	Training loss	Validation loss	Test loss
0% dropout	0.02562	0.02911	0.03266
25% dropout	0.24172	0.08519	0.08502
50% dropout	0.47773	0.27423	0.26216
75% dropout	0.73191	0.57920	0.54833

Table 3: Results obtained after testing different dropout factors.

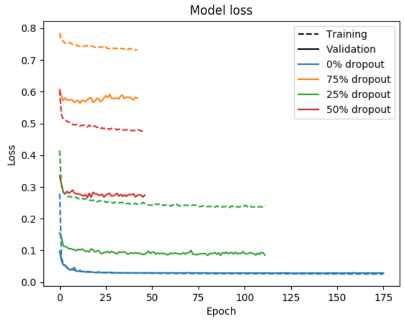

Figure 2: Validation and training loss with each dropout factor.

	Training loss	Validation loss	Test loss
0% inter-layer dropout	0.02562	0.02911	0.03266
25% inter-layer dropout	0.02830	0.02886	0.03258
50% inter-layer dropout	0.03110	0.02884	0.03262
75% inter-layer dropout	0.04012	0.02975	0.03337

Table 4: Results obtained after testing different inter-layer dropout factors.

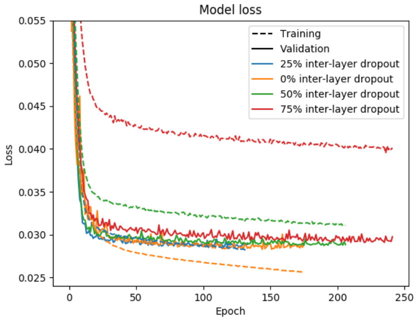

Figure 3: Validation and training loss with each inter-layer dropout factor.

	Training loss	Validation loss	Test loss
Window size of 1	0.03146	0.03305	0.03599
Window size of 3	0.02690	0.02880	0.03254
Window size of 6	0.02562	0.02911	0.03266
Window size of 12	0.02586	0.02884	0.03260
Window size of 24	0.02465	0.02941	0.03375
Window size of 48	0.02543	0.02885	0.03316

Table 5: Results obtained after testing different input window sizes.

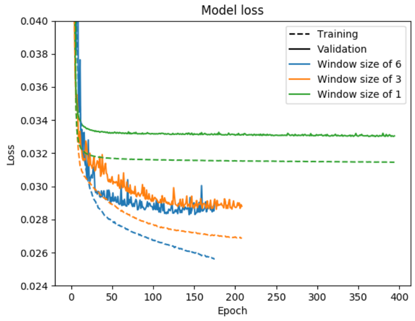

Figure 4: Validation and training loss with window sizes 1, 3 and 6.

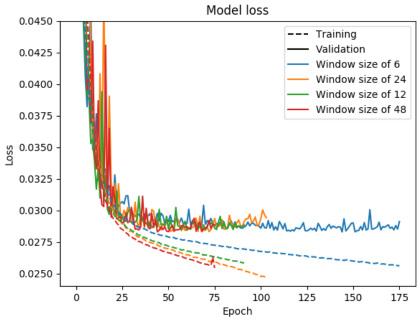

Figure 5: Validation and training loss with window sizes 6, 12, 24 and 48.

3.4 Conclusions

Regarding to learning techniques:

- *As expected*, adding more units to the hidden layer leads to better training accuracies and, for the case of 1024 units, overfits the training data and offers worse validation and test accuracies.

- *As expected*, dropout factors applied to LSTM gates leads to far worse results than the original approach. In our case, it is not a good idea to mess with the current state of the units.
- *As expected*, since our model does not overfit the training data, certain dropout factors—25% in this case—reduce the gap between the training and validation loss, but does not increase the accuracy of the system.
- *As expected*, smaller window sizes lead to worse accuracies while bigger window sizes do not offer much improvement. However, looks like the model converges in a less number of epochs.

Regarding to this specific problem:

- *As expected*, more information—variables—are helpful and the model finds new inter-variable relations that lead to a *slightly* better accuracy. The baseline model got a test accuracy of 0.03339 while our best model here got 0.03218—and several configurations got a mark under 0.03290—. We can conclude that air density, pressure and temperature help to predict wind speed at 100m in site “90-45142”.

4 Complementary Sites

4.1 Plan

In this session we leverage the information of the wind speed from the other sites and we study how the MSE changes when we change the size of hidden layer, when we apply dropout and when we change the size of input window.

- In the 1st experiment we test different layer sizes: 32, 128, 512 and 1024 units.
- In the 2nd experiment we test different dropout factors applied in the LSTM layer, including its gates: 0%, 25%, 50%, 75%.
- In the 3rd experiment we test different dropout factors only applied on inter-layer connections, between the hidden and the output layer: 0%, 25%, 50%, 75%.
- In the 4th experiment we test different input window sizes: 1, 3, 6, 12, 24, 48. In this experiment, in the case of a window size of 48, we had to clip all parameter gradients to a maximum norm of 1 in order to prevent exploding gradients.

4.2 Expectations

Exactly the same as in section Complementary Variables.

4.3 Results

- Table 6 shows the results obtained after testing different layer sizes, while figure 6 shows the evolution of the training and validation loss in each case.
- Table 7 shows the results obtained after testing different dropout factors, while figure 7 shows the evolution of the training and validation loss in each case.
- Table 8 shows the results obtained after testing different inter-layer dropout factors, while figure 8 shows the evolution of the training and validation loss in each case.
- Table 9 shows the results obtained after testing different input window sizes, while figures 9 and 10 show the evolution of the training and validation loss in each case.

	Training loss	Validation loss	Test loss
32 LSTM units	0.02666	0.02821	0.03231
128 LSTM units	0.02694	0.02855	0.03246
512 LSTM units	0.02631	0.02847	0.03225
1024 LSTM units	0.02591	0.02882	0.03242

Table 6: Results obtained after testing different layer sizes.

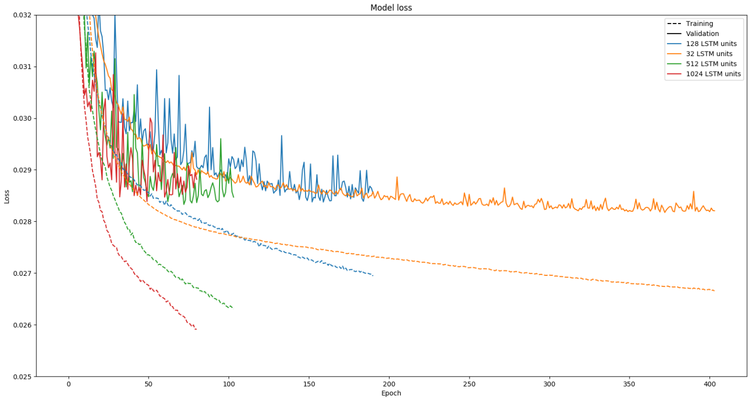

Figure 6: Validation and training loss with each layer size.

	Training loss	Validation loss	Test loss
0% dropout	0.02631	0.02847	0.03225
25% dropout	0.03554	0.08211	0.08097
50% dropout	0.09668	0.26654	0.25337
75% dropout	0.34805	0.54228	0.51081

Table 7: Results obtained after testing different dropout factors.

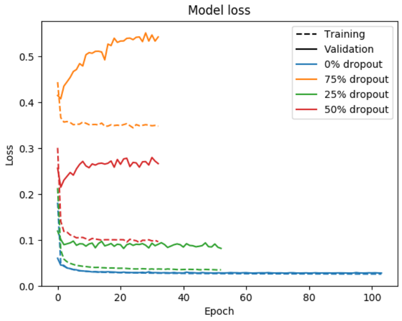

Figure 7: Validation and training loss with each dropout factor.

	Training loss	Validation loss	Test loss
0% inter-layer dropout	0.02631	0.02847	0.03225
25% inter-layer dropout	0.02836	0.02887	0.03240
50% inter-layer dropout	0.03084	0.02876	0.03277
75% inter-layer dropout	0.03898	0.02872	0.03251

Table 8: Results obtained after testing different inter-layer dropout factors.

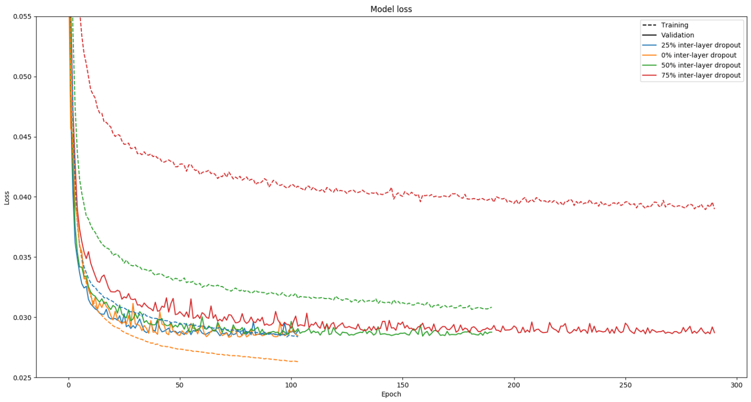

Figure 8: Validation and training loss with each inter-layer dropout factor.

	Training loss	Validation loss	Test loss
Window size of 1	0.03136	0.03277	0.03559
Window size of 3	0.02669	0.02843	0.03226
Window size of 6	0.02631	0.02847	0.03225
Window size of 12	0.02558	0.02935	0.03302
Window size of 24	0.02571	0.02879	0.03264

Table 9: Results obtained after testing different input window sizes.

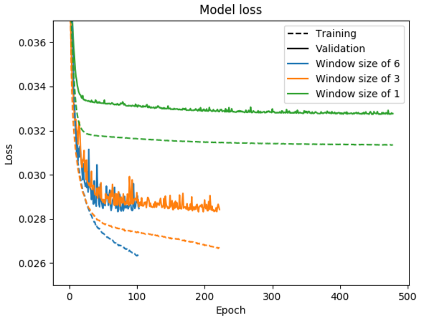

Figure 9: Validation and training loss with window sizes 1, 3 and 6.

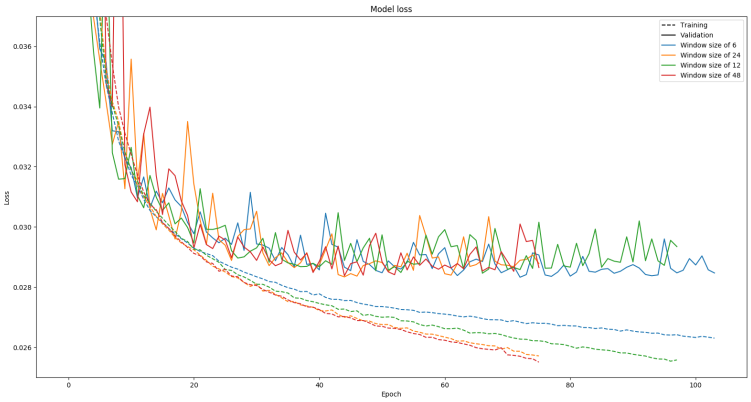

Figure 10: Validation and training loss with window sizes 6, 12, 24 and 48.

4.4 Conclusions

Regarding to learning techniques: exactly the same as in section Complementary Variables.

Regarding to this specific problem:

- *As expected*, more information—variables—are helpful and the model finds new inter-variable relations that lead to a *slightly* better accuracy. The baseline model got a test accuracy of 0.03339 while our best model here got 0.03225—and several configurations got a mark under 0.03290—. We can conclude that wind speeds at 100m of sites “90-45143’, “90-45229” and “90-45230” help to predict wind speed at 100m in site “90-45142”.

5 Multi Step Prediction

5.1 Plan

In this session we analyze the behavior of the model when it is fed with its own outputs as new inputs; we study how the MSE degrades the more steps in the future we predict and how the MSE changes if we extend the input window.

In this experiment we test different input window sizes—1, 3, 6, 12, 24—and we observe the evolution of MSE when trying to predict the next 55 steps.

5.2 Expectations

- In previous experiments we found that small window sizes lead to a higher error, and big window sizes do not show any improvement. We expect that the error of the model accumulates as more steps in the future we predict, leading to a higher error on small window sizes and more stable accuracies in big window sizes.

5.3 Results

Figure 11 shows the results obtained after testing different input window sizes.

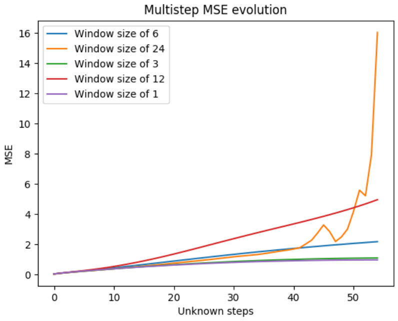

Figure 11: MSE evolution for each window size.

5.4 Conclusions

- *Surprisingly*, we found the opposite of our expectations. As we can see in figure 11, with small window sizes the error seems to converge while in big window sizes the error grows exponentially. Sincerely, we do not have an explanation for this interesting phenomena.

6 Sequence to Sequence Prediction

6.1 Plan

In this session we modify the architecture of the system adequately for linking a window of measurements to a window of predictions, and we study how the MSE of the predictions changes with the length of the predicted sequence.

In this experiment we test different prediction lengths—1, 3, 6, 12, 24—and we observe the evolution of the MSE.

6.2 Expectations

- Models that try to predict more steps forward will lead to lower accuracies.
- This sequence-to-sequence model will have a lower error than the previous multi step approach, since this model does not accumulates the error.

6.3 Results

Figure 12 shows the evolution of the training and validation loss in each case, while figure 13 shows the MSE evolution.

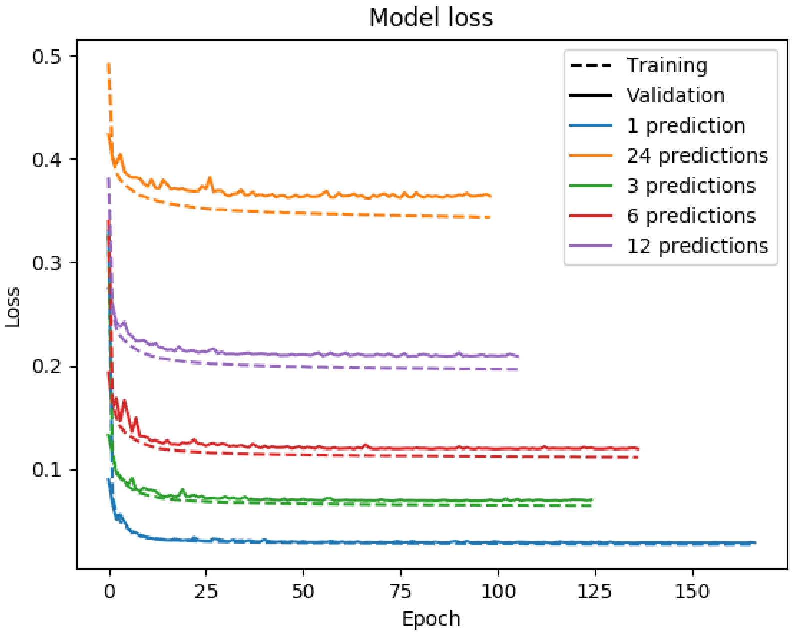

Figure 12: Validation and training loss with each number of predictions.

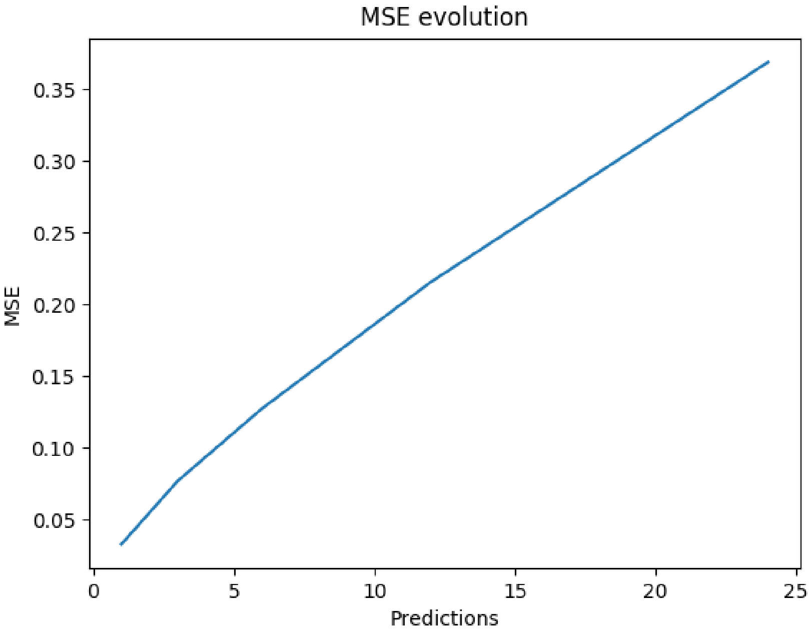

Figure 13: MSE evolution relation with the number of predictions.

6.4 Conclusions

- *As expected*, models that try to predict more steps forward lead to lower accuracies.
- *As expected*, this sequence-to-sequence model has a lower error than the previous multi step approach. The previous model, with a window size of 6 and 24 unknown inputs, has a MSE of 1.06895 while this model has a MSE of 0.368767 when predicting 24 steps.