Deep Learning Lab 4
Recurrent Neural Networks

Garben Tanghe
Olivier Van den Nest
Group 03

March 20, 2020

Data Pre-processing

There cannot be found a global trend in the pollution (not linearly increasing over time), so the year can be discarded.

The average pollution by month of the year, week of the year, and day of the year are found in Figure 1, 2, and 3 respectively. There is a seasonal trend and there is assumed that that knowledge is already encoded in the weather parameters (dew, temperature, and pressure), still the day of the year is encoded as a combination of sine and cosine values. The month is not used, as that is too coarse-grained. The day of the year is more fine-grained, which results in smoother curves (cf. Figure 4).

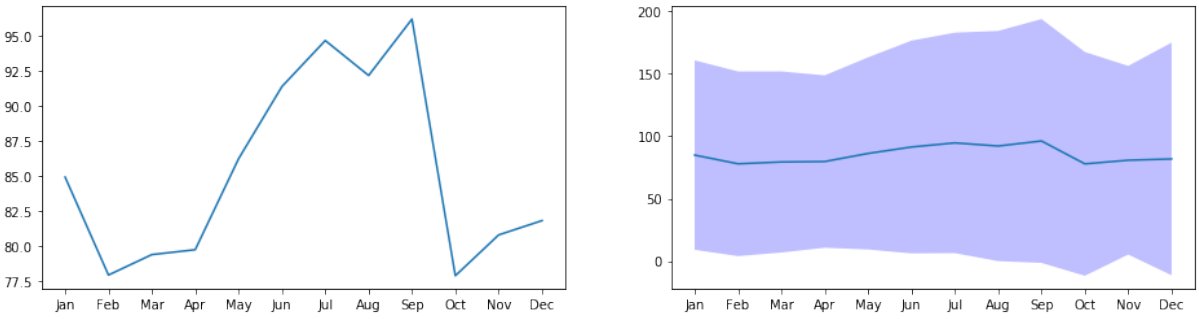


Figure 1: Average pollution by month of the year

Mistakes were found in the transformation from the day of the month to the day of the week. First of all, the dataset does not start on Friday Jan. 1, 2010, but on Saturday Jan. 2, 2010 instead. Secondly, there was not taken into account that there are 24 samples per day. The fixed transformation is found in Figure 5.

The average pollution for the day of the week and the hour of the week is found in Figure 6 and 7 respectively. From Figure 6, it is clear that the pollution is at its lowest in the beginning of the week and than increases almost linearly during weekdays. During the weekend, the pollution will than drop back quickly. In Belgium, discriminating between weekends and weekdays would be relevant, but in Beijing, there is a different

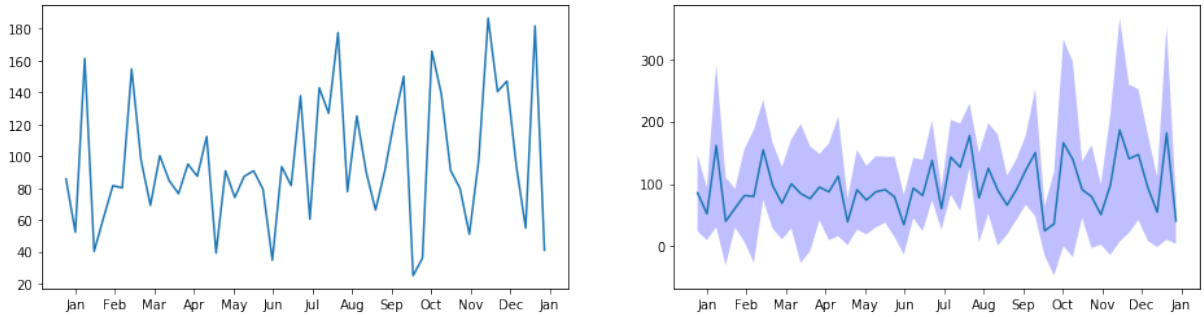


Figure 2: Average pollution by week of the year

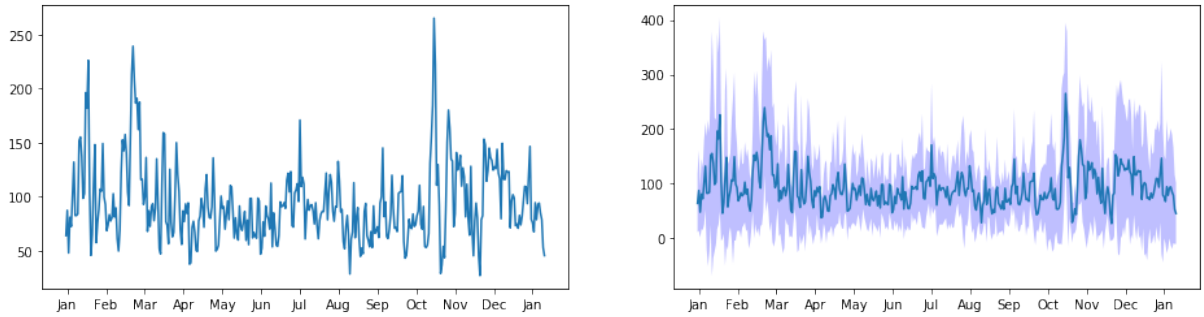


Figure 3: Average pollution by day of the year

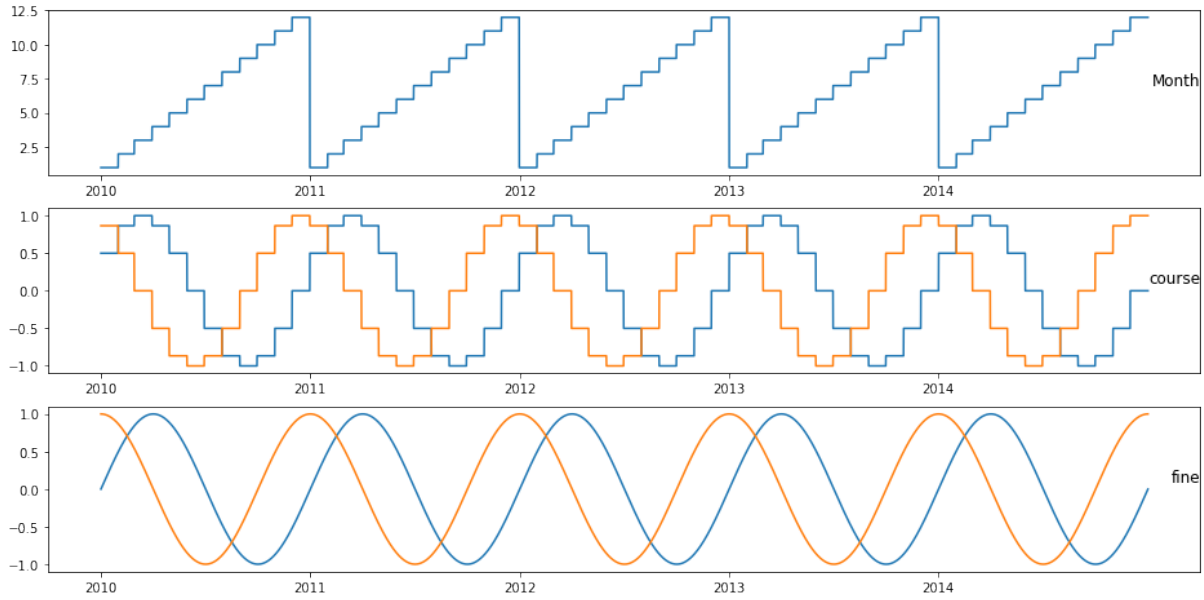


Figure 4: Encoding of the season of the year

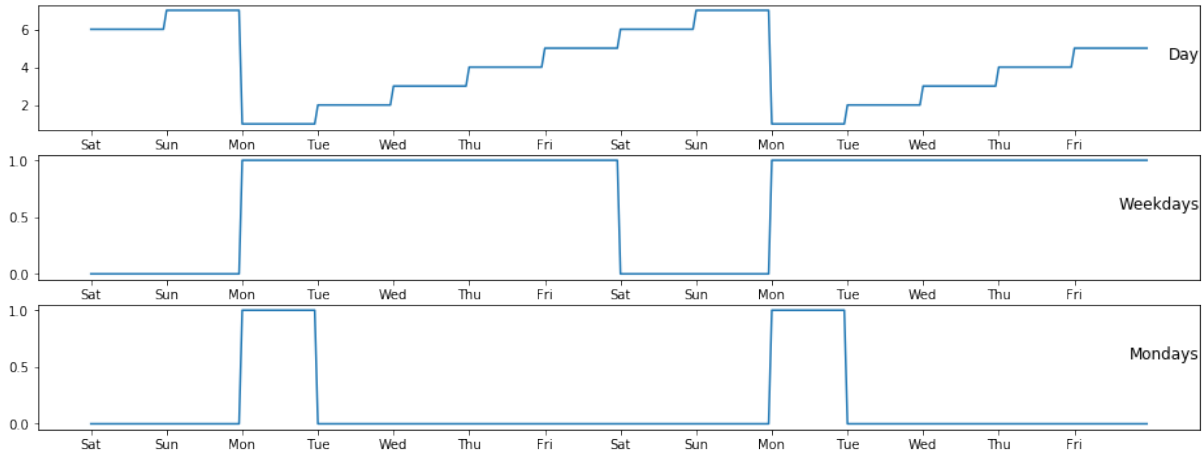


Figure 5: Transformation for day of the week

correlation. From Figure 7, the fluctuations between day and night become clear. There is decided to not one-hot encode the days of the week, but rather use the combination of a sine and cosine for the hour of the week, as those are only 2 features instead of 7 and contain more information. If the hour of the week would be one-hot encoded, there would even be 168 features! The resulting encoding can be found in Figure 8.

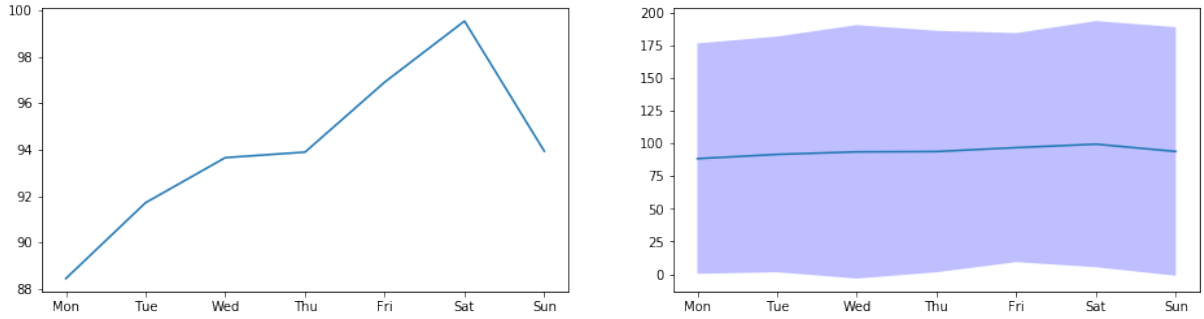


Figure 6: Average pollution by day of the week

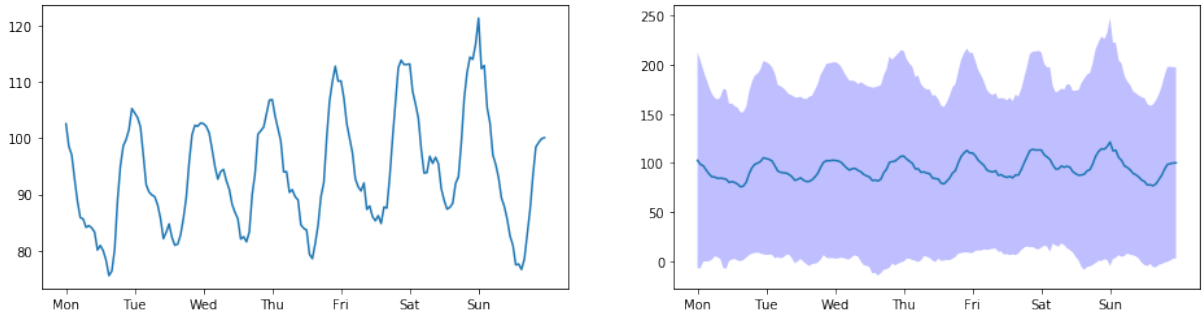


Figure 7: Average pollution by the hour of the week

The average pollution for each hour of the day can be found in Figure 9. From this, it is clear that the pollution is at its lowest around 3 o'clock the afternoon and peeks just after midnight. Next to seasons and day of the week, hour of the day is the third periodic trend. It is again encoded using the sine and cosine combination. The resulting encoding can be found in Figure 10.

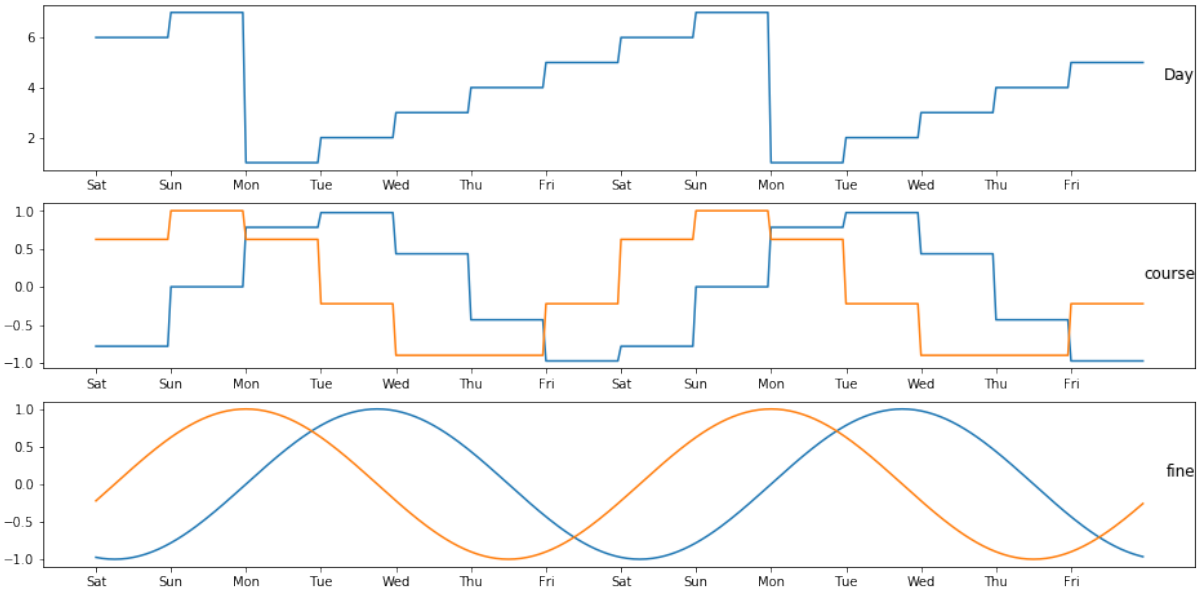


Figure 8: Encoding of the day of the week

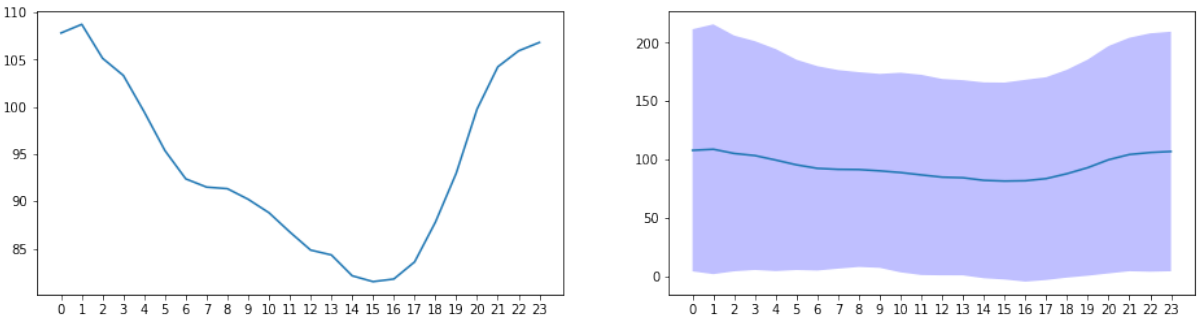


Figure 9: Average pollution by hour of the day

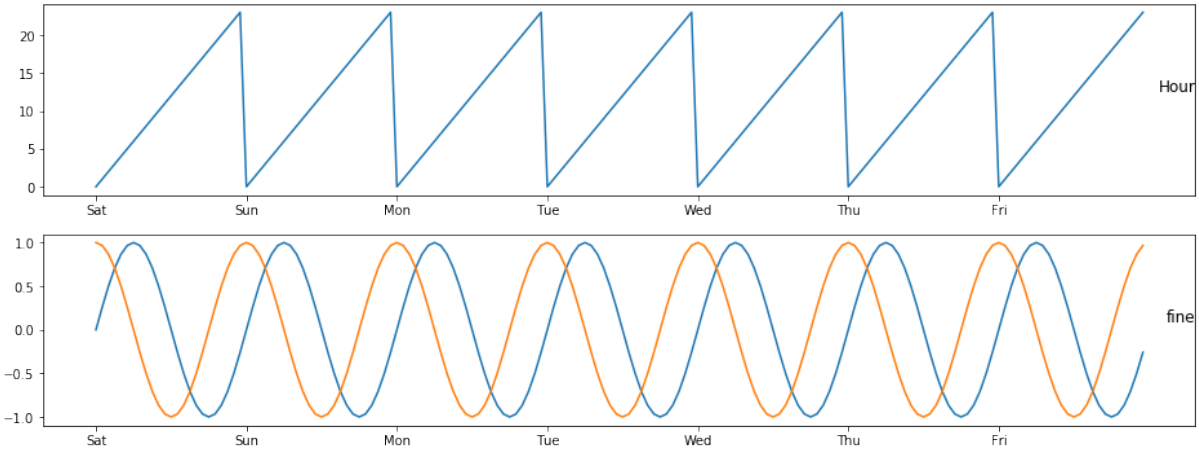


Figure 10: Encoding of the hour of the day

So far, 3 periodic trends are found in the data: seasonal, weekly, and daily. The day of the year, hour of the week, and hour of the day are each time encoded as a sine and cosine pair, which results in 6 very informative features.

The provided encoding of the wind strengths are being used. This comes down to 4 features (one for 'NE', 'NW', 'SE', and 'cv') that contain the strength of the wind coming from that direction.

The pollution, weather, and precipitation features are left as they are, which results in total to 16 features.

2010-2012 are used for training, 2013 as the year for validation, and 2014 for testing. On the edges between data sets, there are gaps to reduce the amount of data leakage. A plot of the data for the first 3 years can be found in Figure 11.

PowerTransformer is used to scale the features independently, since it maps all features to have zero-mean and unit variance. This is often wanted for neural networks, since the features with large values would otherwise have a much bigger impact than others. The weights to compensate and make all features equally important at the beginning, would first have to be learned. If those weights are too big, regularization techniques as L1 and L2 will result in a very large cost. As a result, the network will not learn anything. Now that all features have values in the same order of magnitude, the network can immediately start learning what features are more important than others. A plot of the data for the first 3 years can be found in Figure 12.

Step 1: Dense Network

The first baseline model looks as follows: 5 Dense layers with 256, 1024, 1024, 128, and 1 node(s) per layer in that order. All layers, except the last, have a 'He Uniform' initialization scheme, Batch Normalization, and a ReLU activation function. The last layer makes use of 'Glorot Uniform' or 'Xavier' weight initialization, no Batch Normalization, and a linear activation function. Between all dense layers, Dropout is used to regularize. The dropout rates are 0.5, 0.5, 0.5, and 0.1 in that order. The best results are achieved from using the Adam optimizer (with a starting learning rate of 0.1 and decay it when the validation mean absolute error plateaus) for 70 epochs and with a batch size of 5 weeks (5*168). The validation curves for training on the first 3 years and validation on 2013, are found in Figure 13a. For retraining on the complete training set, the validation curve is depicted in Figure 13b. The results of the baseline model are plotted in Figure 14. The total mean absolute error on the validation and test set is 37.89 and **34.73** respectively. The predicted pollution is almost a shifted version of the ground truth. This is called 'lagging behind'.

To find the most important features, the Local Interpretable Model-Agnostic Explanations (LIME) framework is tried out. This framework performs IO perturbations to try finding the most impactful features on the predictions. By applying LIME, there is found that the model's decisions are mostly explained by the *Pollution* feature, which is expected. Other important features are *temp*, *dew*, *NW_strength*, and *hour_of_day_sin*, also features that make a lot of sense for this type of model. Explanations from LIME for a good and a bad prediction is shown in Figure 15.

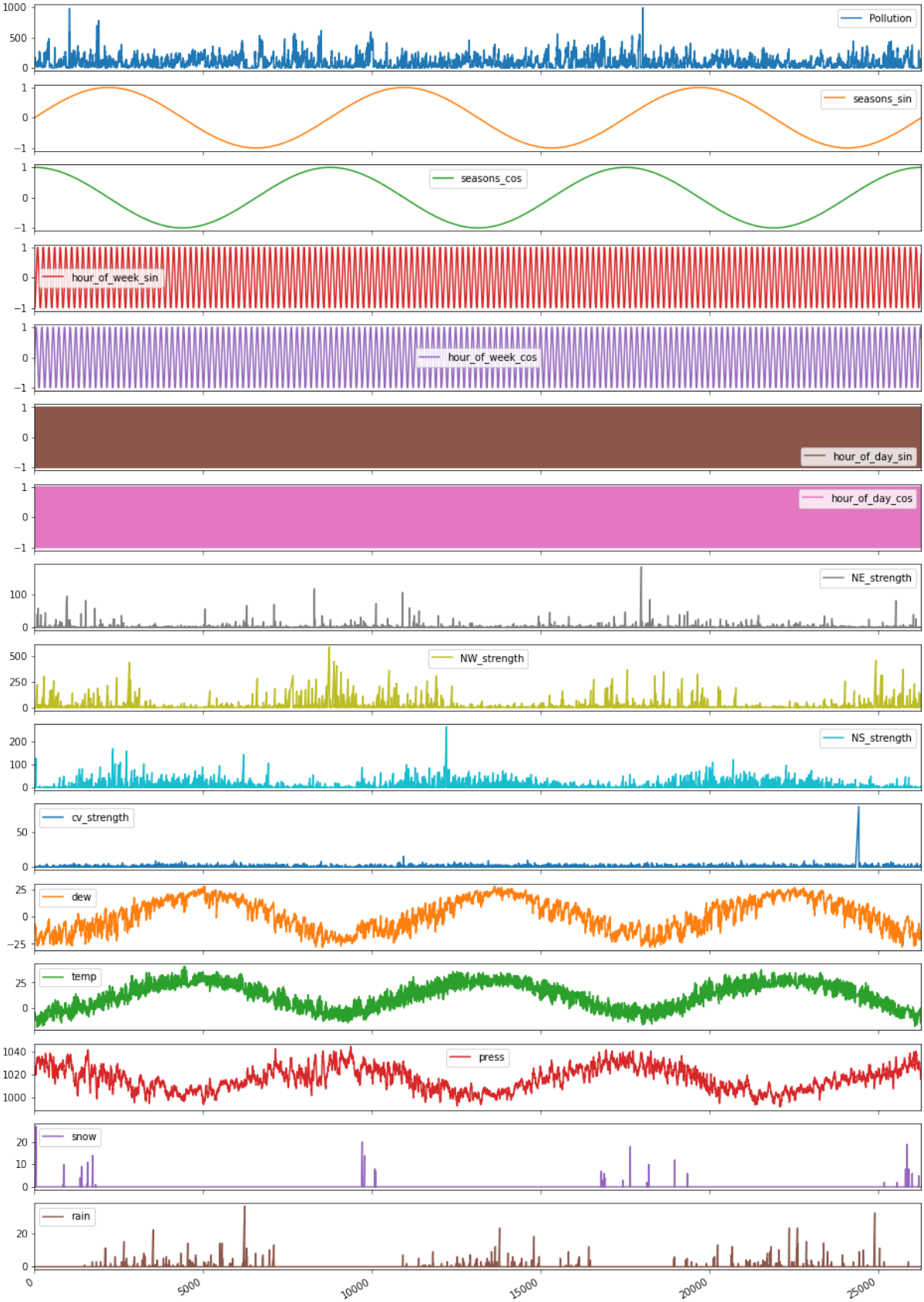


Figure 11: Raw data (2010-2012)

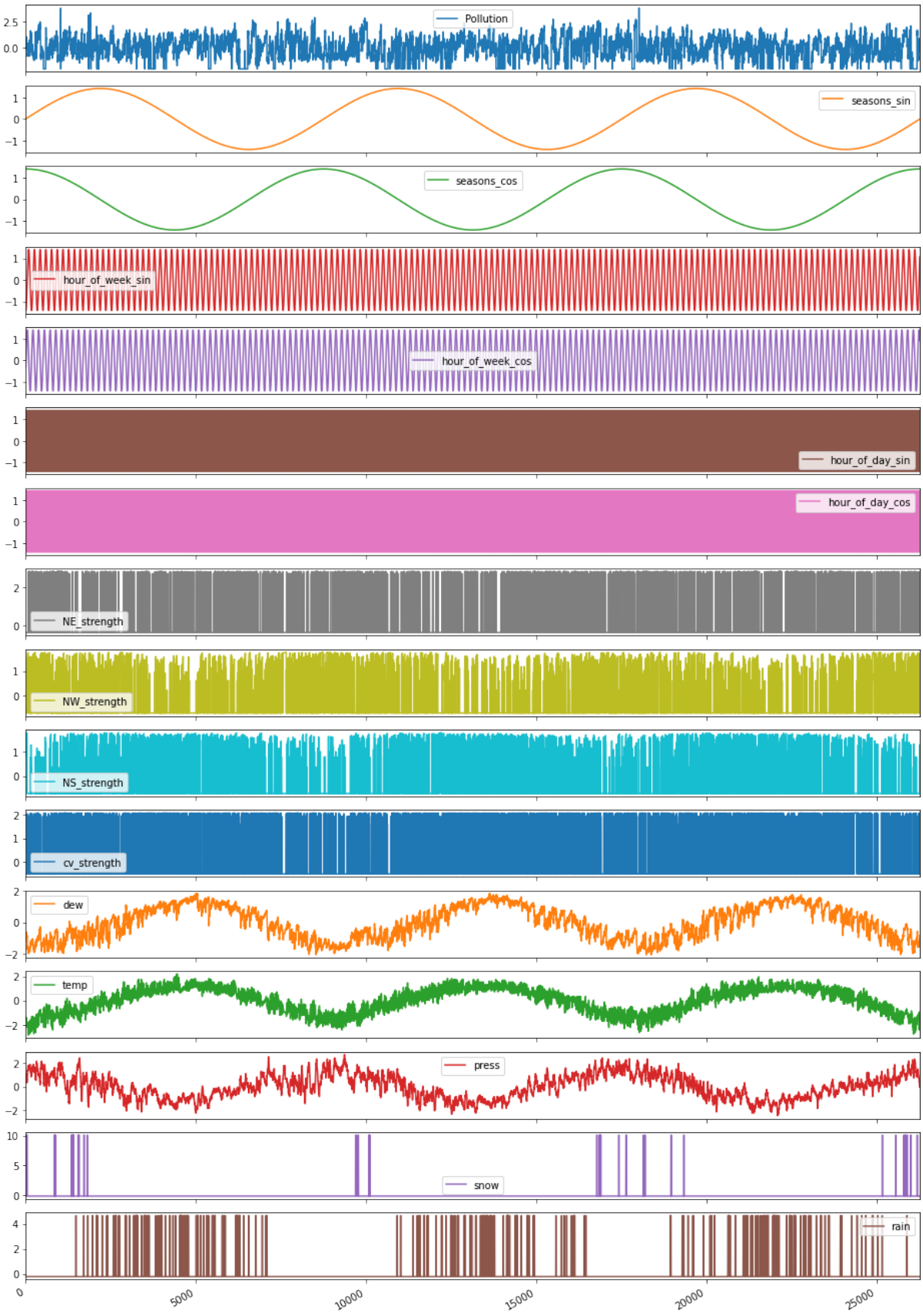
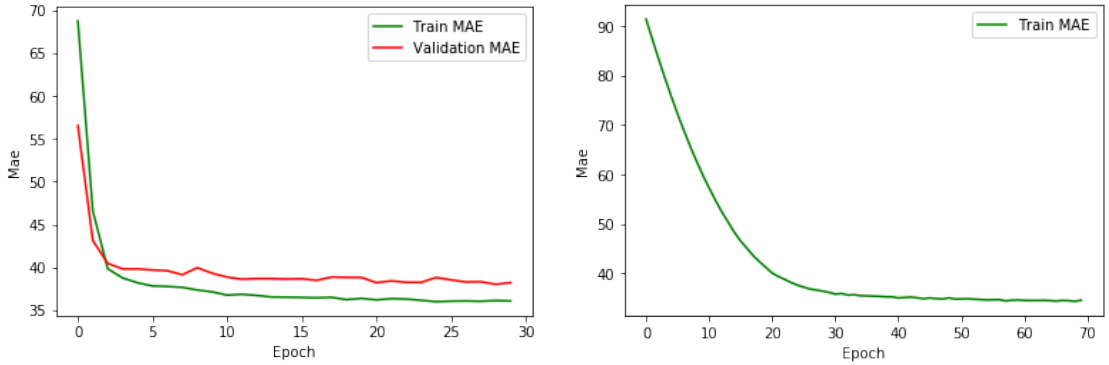


Figure 12: Normalized data (2010-2012)



(a) Small training set (b) All training data

Figure 13: Validation curves for the Dense network

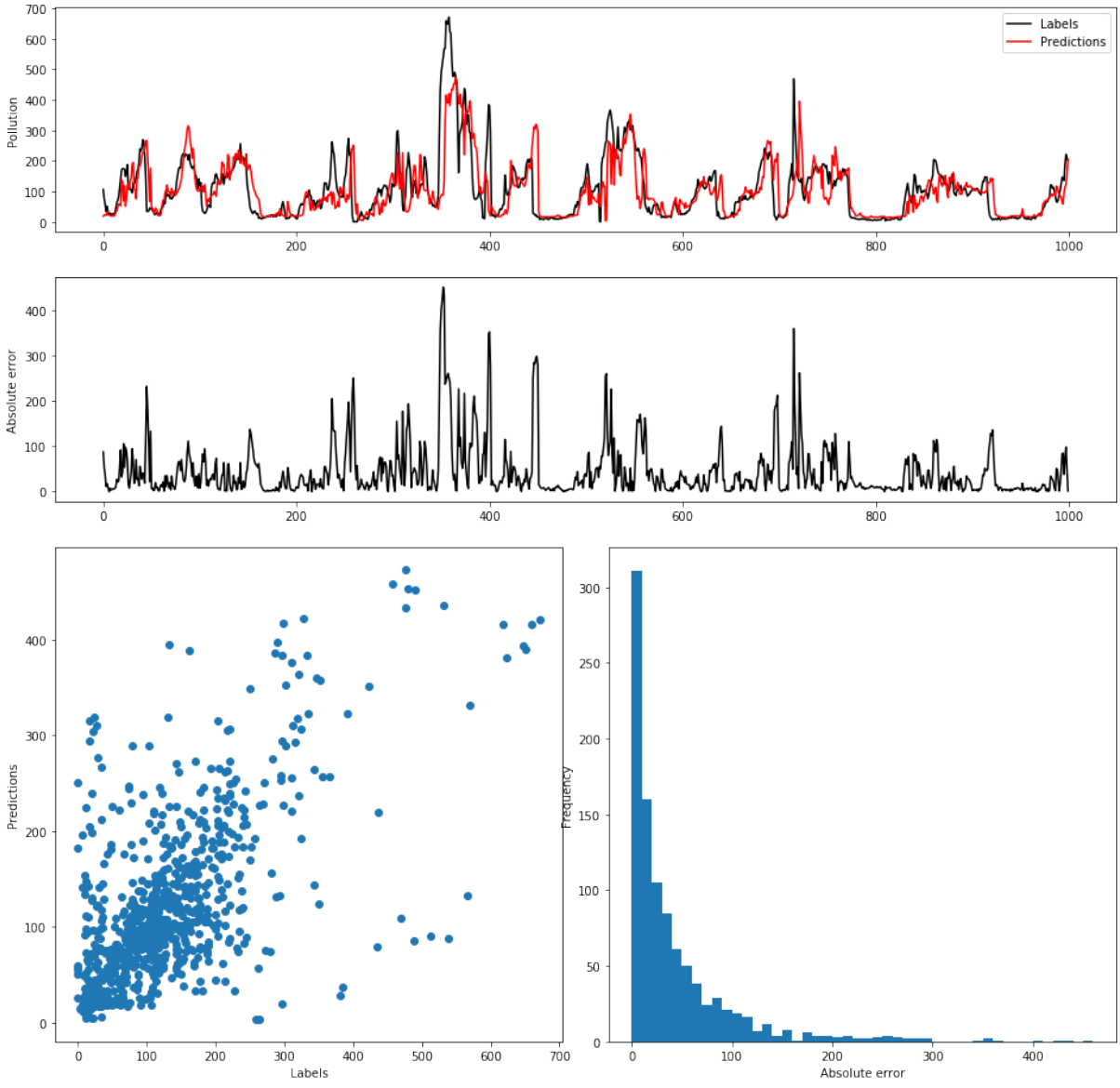
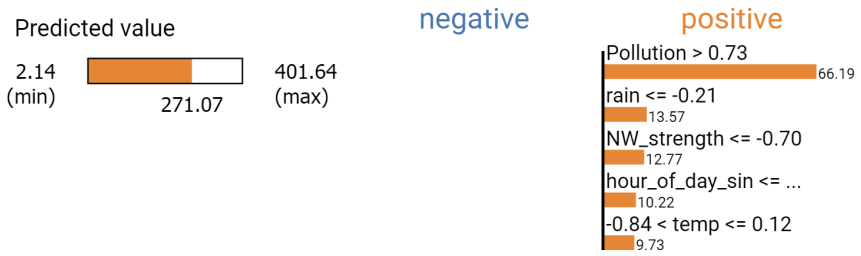
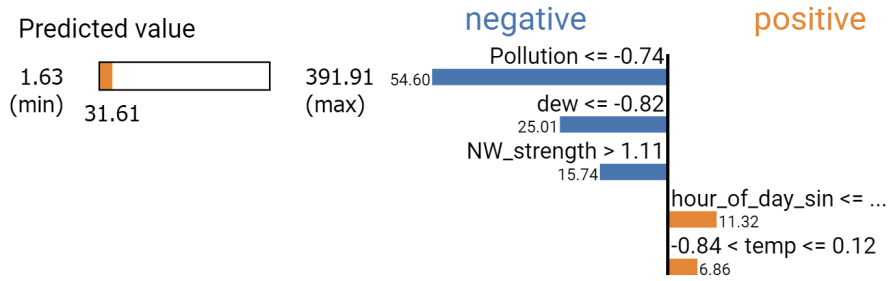


Figure 14: Results for the Dense network



(a) Bad prediction



(b) Good prediction

Figure 15: Explanation

Step 2: CNN and RNN

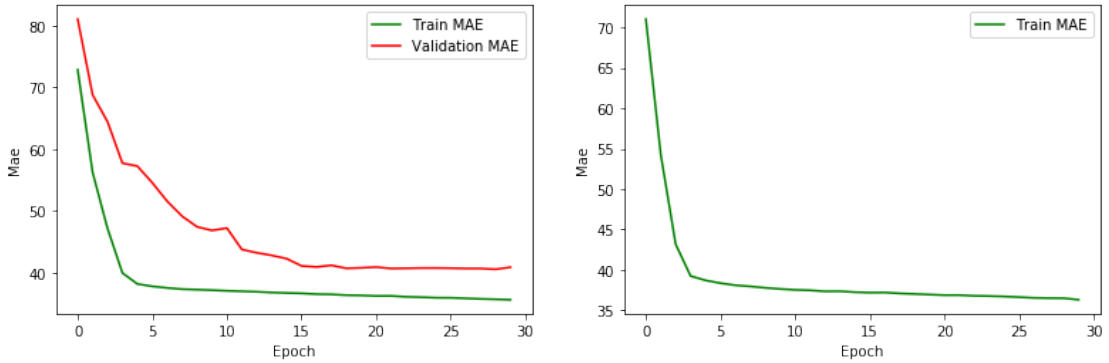
CNN

The window-based convolutional neural network (CNN) looks as follows: 3 Conv1D layers with 64, 128 and 256 nodes per layer in that order, as well as a last Dense layer with 1 output. All Conv1D layers have a 'He Uniform' initialization scheme, Batch Normalization and ReLu activation function. MaxPooling with a pool size of 2 is also applied after each Conv1D layer. The Dense layer makes use of 'Glorot Uniform' or 'Xavier' weight initialization and a linear activation function. No dropout is used, as that regularized too much. The Adam optimizer with a fixed learning rate of 0.001 is used. The batch size is set to 8 weeks (8*168) and there is trained for 30 epochs. The window size for making predictions is set to 1 week (168 hours).

The validation curves for training on the first 3 years and validation on 2013, are found in Figure 16a. For retraining on the complete training set, the validation curve is depicted in Figure 16b. The results of the window-based CNN model are plotted in Figure 17. Again, the model seems to be lagging behind. The total mean absolute error on the validation and test set is 40.49 and **37.04** respectively. This is a lot worse than the baseline model. Therefore there can be concluded that the other important features now function as a latent variable that cannot be learned.

RNN

There is decided to implement a Gated Recurrent Unit (GRU) instead of a Long short-Term Memory (LSTM), because for this task the model memory may not be too useful. The current state is intuitively more relevant to predict future pollution difference. Next to this advantage, a GRU is also trained faster. The GRU is easily implemented in Tensorflow Keras with a GRU layer. The model is constructed with 1 or multiple GRU layers,



(a) Small training set (b) All training data

Figure 16: Validation curves for the CNN

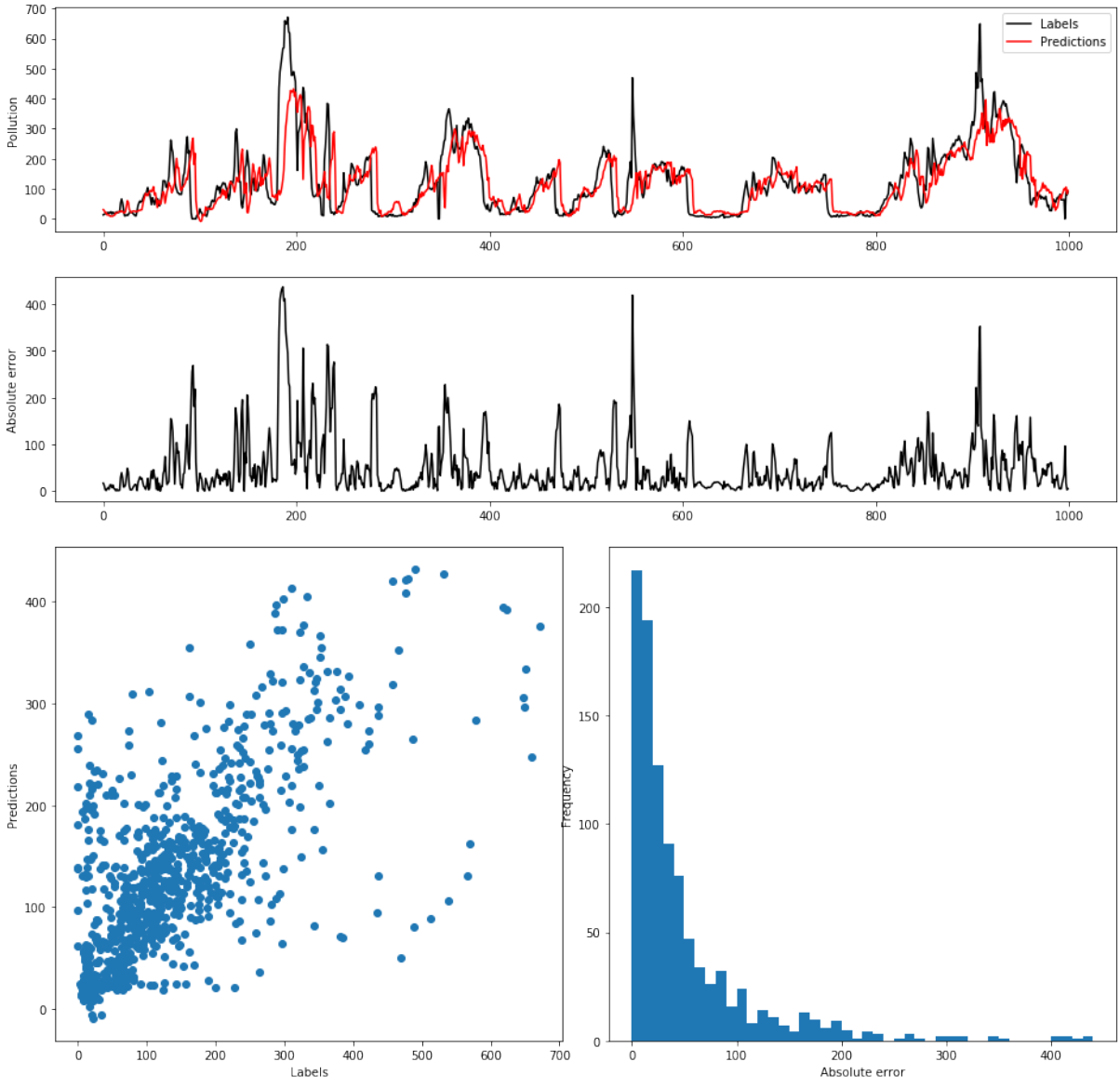
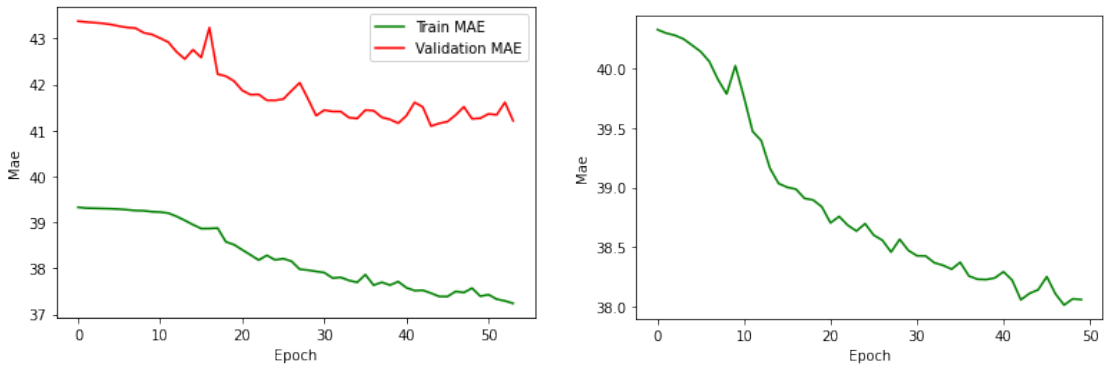


Figure 17: Results of the CNN

followed by a Dense layer with 1 output node and a linear activation function. The model is finetuned by varying the GRU window size, the number of GRU layers and the amount of units every GRU layer has. The SGD optimizer is chosen, with a learning rate of 0.01, momentum of 0.9, clipnorm of 0.9 and a batch size of 2 weeks (2*168) and 200 epochs combined with early stopping patience 10. Training the models on the small training set and validating them on the validation set, the optimal values for these parameters are: a window size of 20, 1 GRU layer and 512 units. The optimal model achieved a training MAE of 37.37 on the small training set and a validation MAE of 41.32 on the validation set, achieving this after 49 epochs. This is illustrated in Figure 18a which shows the validation curves for training on the small training set. The results of the GRU model are shown in Figure 19. Retraining this model on the large training set and evaluating on the test set, resulted in a training MAE of 38.11 and a testing MAE of **39.19**. The validation curve for training on the whole training set is shown in Figure 18b.



(a) Small training set (b) All training data

Figure 18: Validation curves for the RNN

Step 3: Hybrid

The network structure used in this step is very similar to the window-based CNN. The only difference is that there are now 4 Conv1D layers with 32, 64, 128, and 256 nodes per layer and that batch normalization is removed. The window size is set to 2 weeks (2*168) and the batch size is lowered to 24. The Adam optimizer uses a fixed learning rate of 0.0001. The validation curves for training on the first 3 years and validating on the 4^{th} for 25 epochs, are found in Figure 20a. The best validation mean absolute error is 37.63. For retraining on the whole data set for 20 epochs, it can be found in Figure 20b. The resulting test mean absolute error is 36.0. According to Tao et al. (2019) [1], the features that are least correlated with the pollution, are snow, rain, pressure, and temperature (in that order). When snow and rain are left out, the same model scores **35.36**. Also leaving pressure out, results in a score of 35.79. Lastly, when leaving temperature out as well, the model scores 35.49. Leaving snow and rain out of the equation, clearly improves the model. Leaving even more variables out, does not necessarily make it any better. The predictions can be seen in Figure 21. The predictions are still lagging behind a bit, but less than previously.

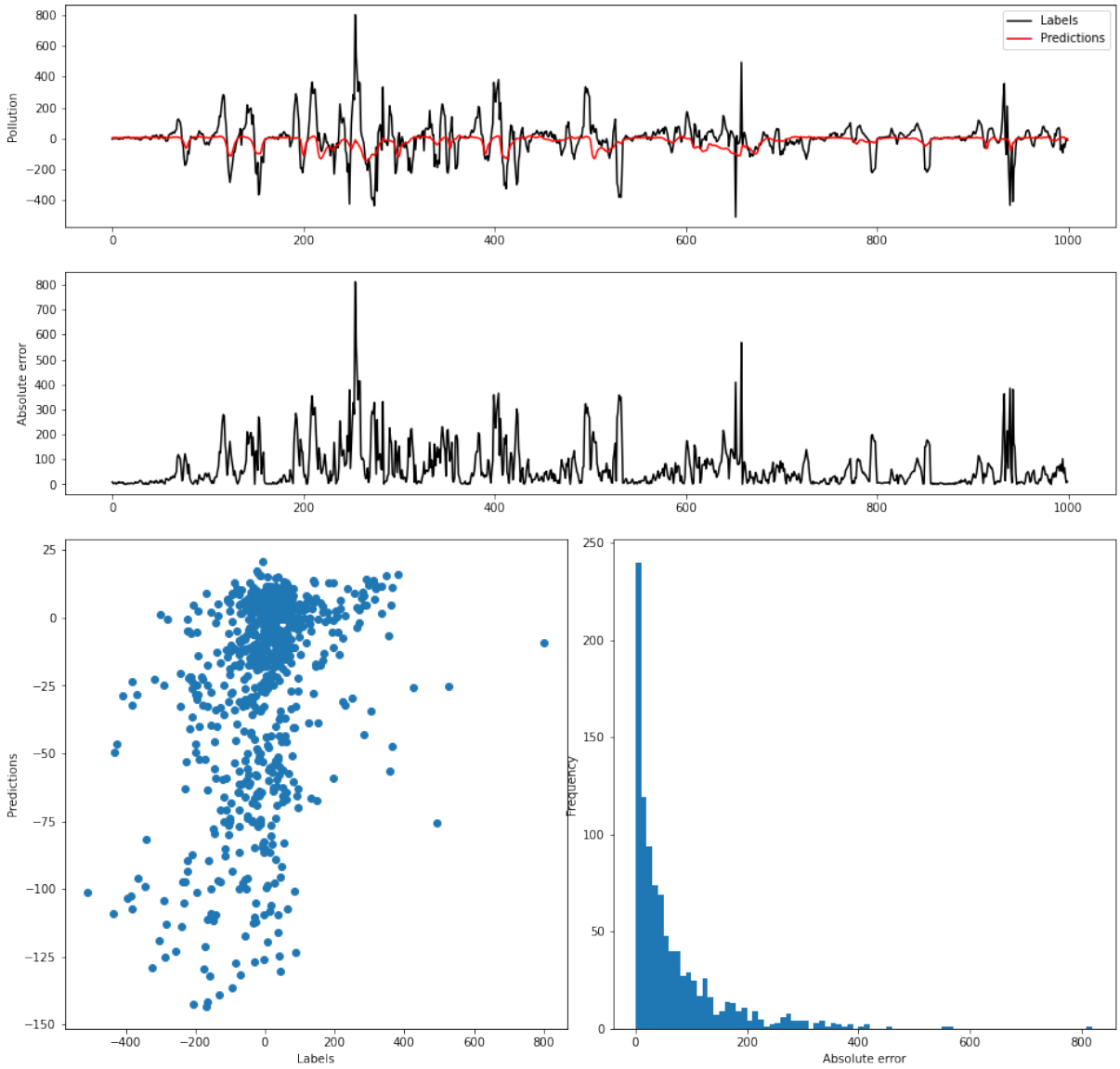
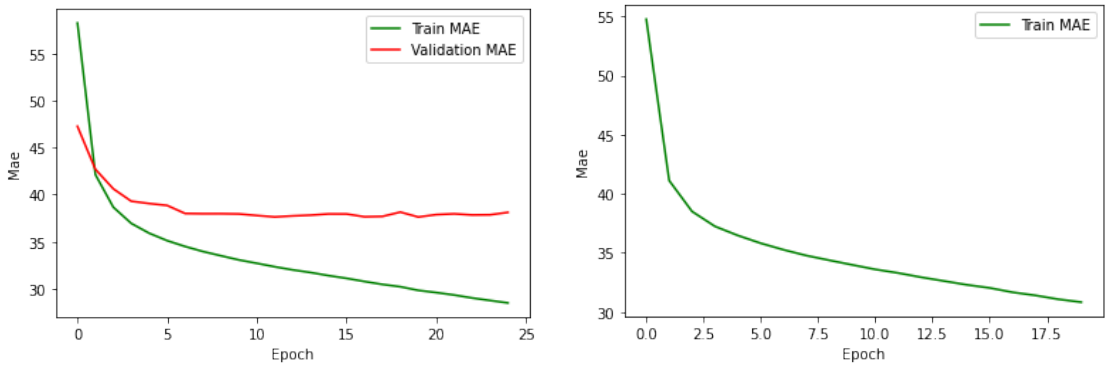


Figure 19: Results of the RNN



(a) Small training set (b) All training data

Figure 20: Validation curves for the hybrid model

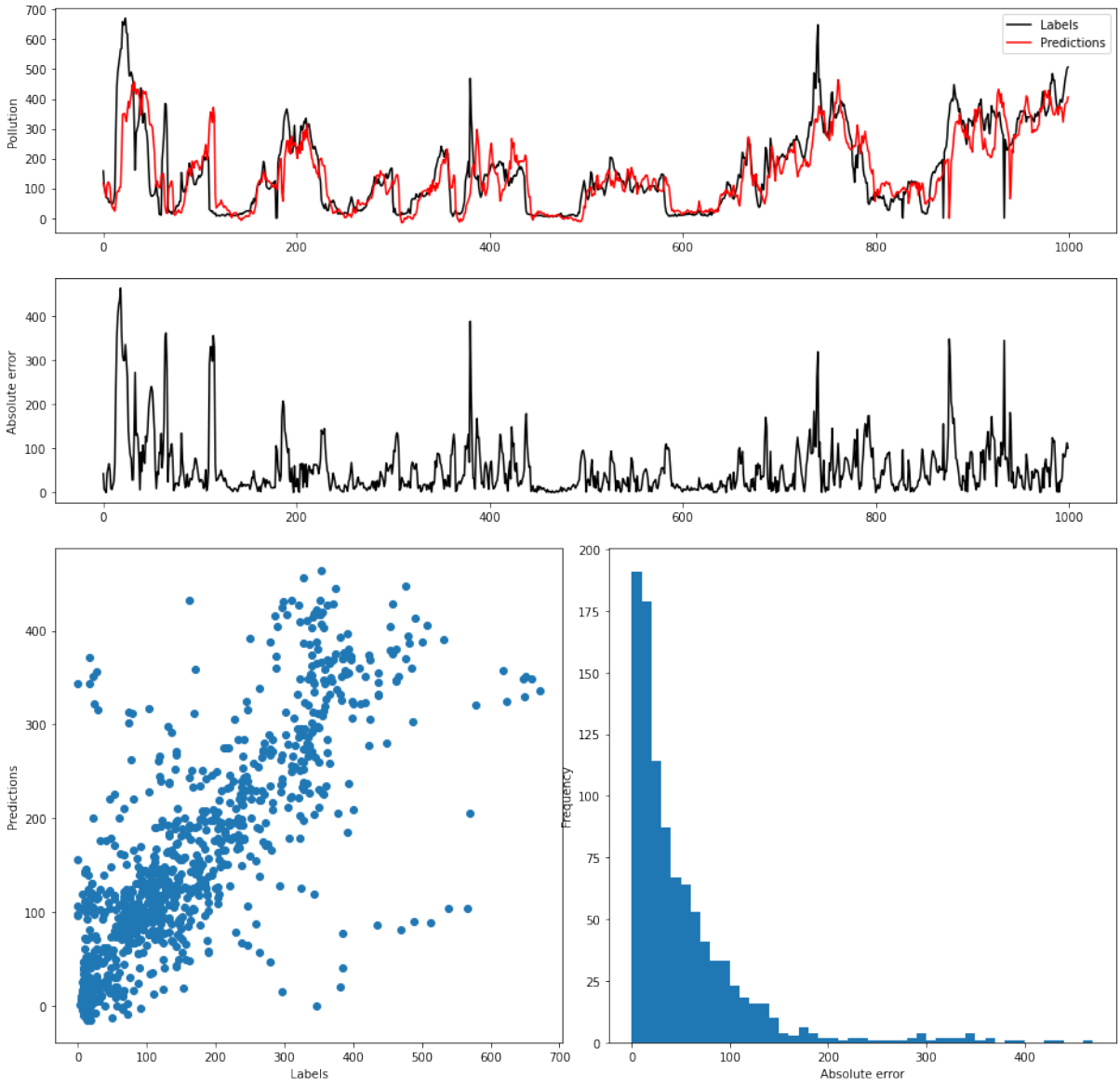


Figure 21: Results for the hybrid model

Step 4: Ensemble

The final model is an ensemble of 5 models, each trained on a different feature set, but the same data. By training on different features, the models should learn to make different predictions on the pollution. The model architecture is each time taken from Step 3. The first model uses all features, the second leaves snow and rain out, the third also leaves pressure out, the fourth leaves temperature out as well, and the fifth only uses the pollution.

Average

The first ensemble simply takes the average of the model predictions. The validation mean absolute error equals 36.95, while the test mean absolute error improved to **34.07**. The resulting predictions can be found in Figure 22

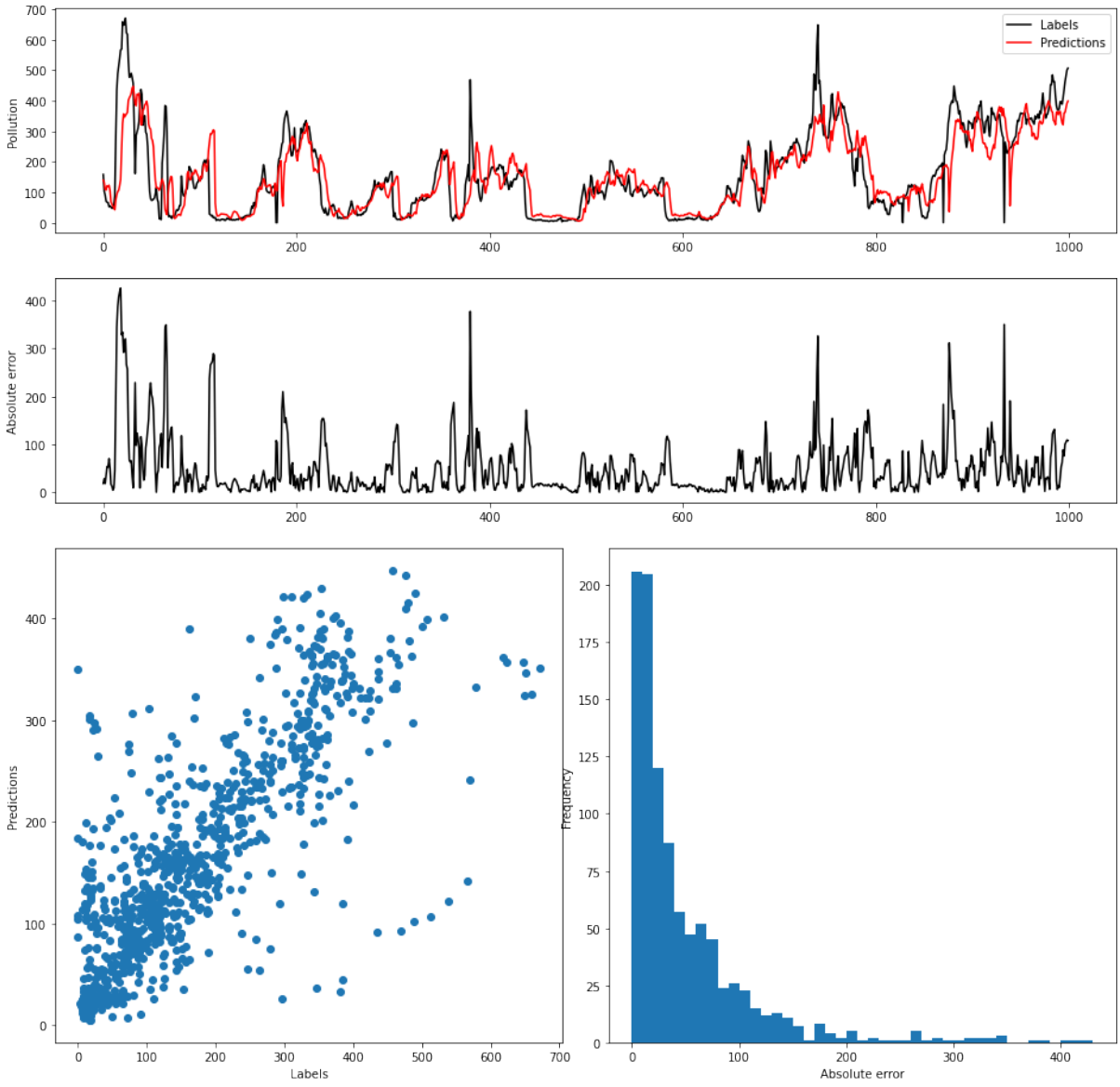
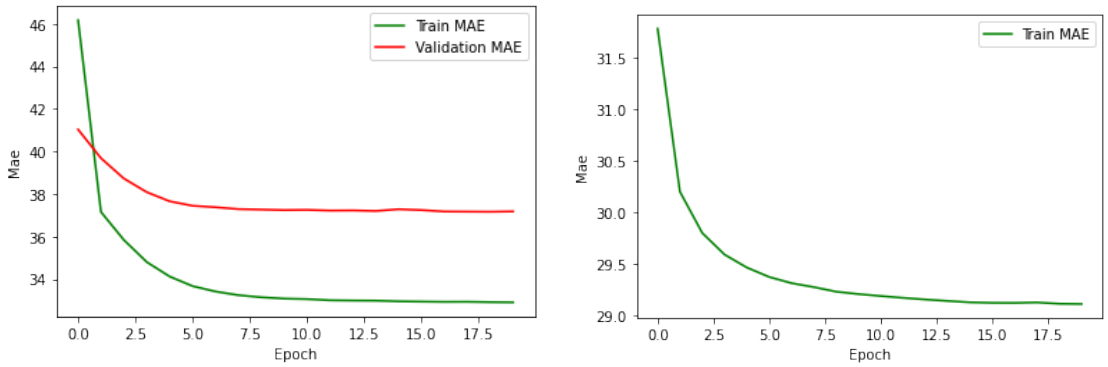


Figure 22: Results for the ensemble, using average

Weighted average

A weighted average might improve the scores further. However, picking the right weights by hand is not easy. Luckily, the weights can be learned by adding another model on top of the rest. The top model consists of one Dense layer with 1 output, no bias addition, and a linear activation function. Its kernel is initialized using a uniform distribution and regularized by a MaxNorm constraint of 2. The batch size is set to 32 and the learning rate of the Adam optimizer is fixed to 0.001. The validation mean absolute error gets very close to the one of the average above: 37.16. The validation curves for 20 epochs can be seen in Figure 23a. After retraining on the whole dataset, the test mean absolute error is **34.46**, which is worse than for just taking the average. The validation curve for retraining on all training data is found in Figure 23b, the resulting predictions are found in Figure 24. The learned weights are as follows: [0.394, 0.262, 0.301, 0.134, -0.087]. This means that the first model (with all features) is most important, then the third (without rain, snow, and pressure), and that the last model (with only the pollution as feature) is least important. This ordering is approximately equal to the order of the best scoring models.



(a) Small training set

(b) All training data

Figure 23: Validation curves for the ensemble, using weighted average

Conclusions

The results of all different models are summarized in Table 1. The ensemble, using average, is clearly the best choice, even though none of the models in the ensemble are better than the Dense network. To see the variance of a single architecture, trained on different data, cross-validation on the first 3 or 4 years could be used. This also improves the comparison between multiple models. Further, each model from cross-validation could be used in the ensemble. Then, there would not only be models trained on different features, but also models trained on a different part of the dataset. The biggest improvement would probably come from data augmentation. This could be done by adding some small Gaussian noise to the data.

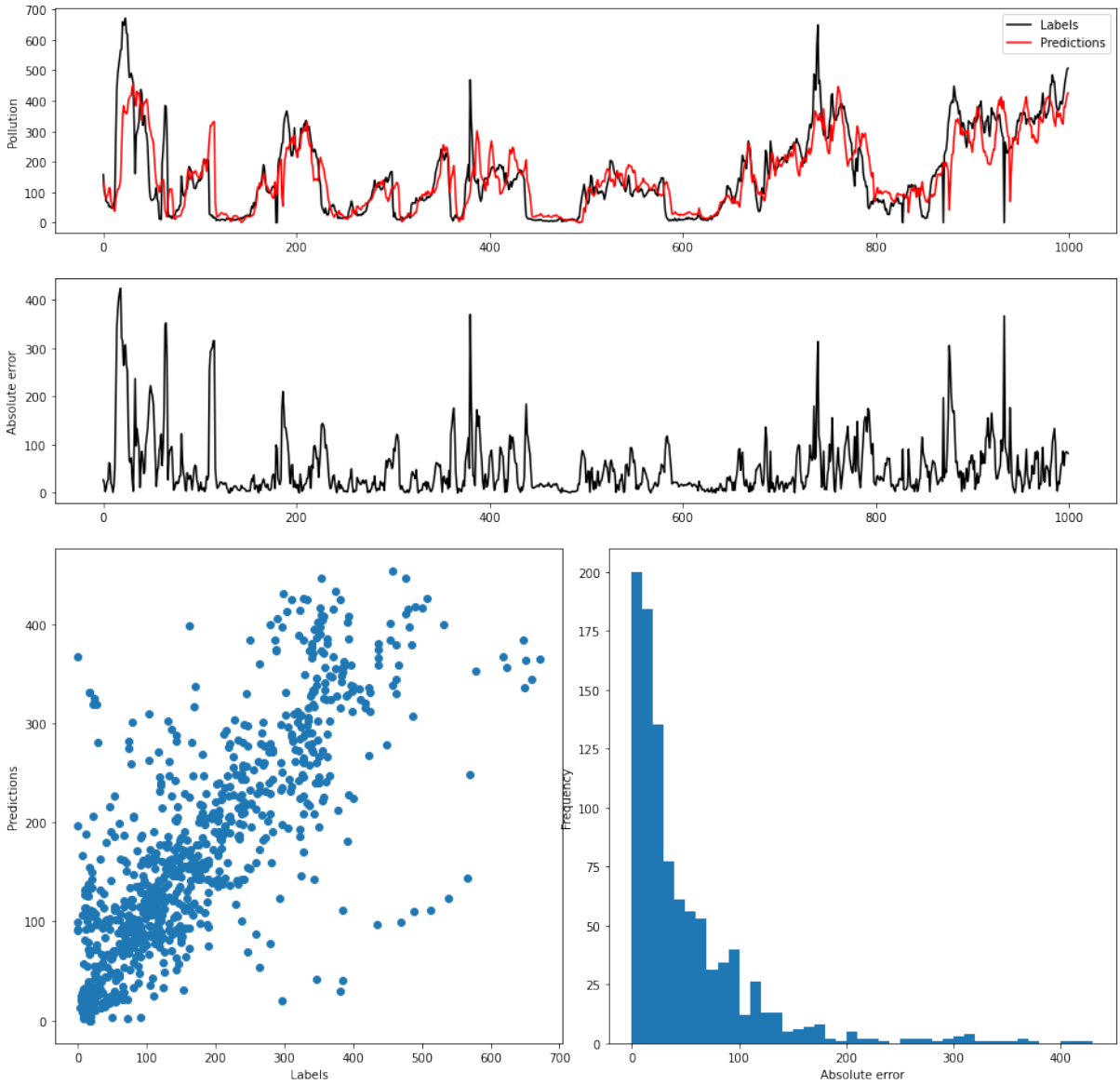


Figure 24: Results for the ensemble, using weighted average

	Dense	CNN	RNN	Hybrid	Average	Weighted
Validation MAE	37.89	40.49	41.32	37.63	**36.95**	37.16
Test MAE	34.73	37.04	39.19	36.0	**34.07**	34.46

Table 1: Results of the different models

References

[1] Q. Tao, F. Liu, Y. Li, and D. Sidorov, "Air pollution forecasting using a deep learning model based on 1d convnets and bidirectional gru", *IEEE Access*, vol. 7, pp. 76 690–76 698, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2921578.

[2] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: A review", *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.