TIME SERIES FORCASTING FOR WEATHER PREDICTION

Method

Features

The dataset used is a modified version of the weather time-series dataset recorded by the Max Planck Institute for Biogeochemistry. The dataset contains in total 15 features, of these we choose the following:

- **p(mbar)**: atmospheric pressure.
- **T (degC):** air temperature.
- **rh (%):** relative humidity.
- **wv(m/s):** wind velocity.

To these we added some handmade features created from the *DATE TIME* column present in the dataset. The *DATE TIME* column contains a date in this format "01.01.2009 01:00:00" and since we cannot feed directly a network with this type of data we created four new features that encode the date. The idea is to take advantage of the circularity of the dates and the hours to create a numerical equivalent using the *sin* and *cosin* function. The new features are the following:

- day_sin: here we encode the day of the year (1-365) using the *sin* function.
- day_cos: same as above but using the *cos* function.
- hour_sin: similarly to what we did with the day we can use trigonometric functions to map the hour of the day (which is in the range of 0-23).
- hour_cos: same as above but using the *cos* function.

Preprocessing steps

Normalization

Normalization is a common preprocessing technique applied in data preparation. The goal of normalization is to transform features to be on a similar scale. This improves the performance and training stability of the model. However data normalization can cause **data leakage**, for example if we want to scale our data in the range 0-1 we first calculate the maximum and minimum of our data. If the data are splitted after the normalization the training set contains information about the distribution in the test set and this information should not be available during the training. To avoid this problem we apply **quantile** normalization in the following way:

- Perform the quantile normalization on the training dataset calculating the reference distribution using only the training data.
- Apply the same reference distribution to the test set without recalculating on the test data.

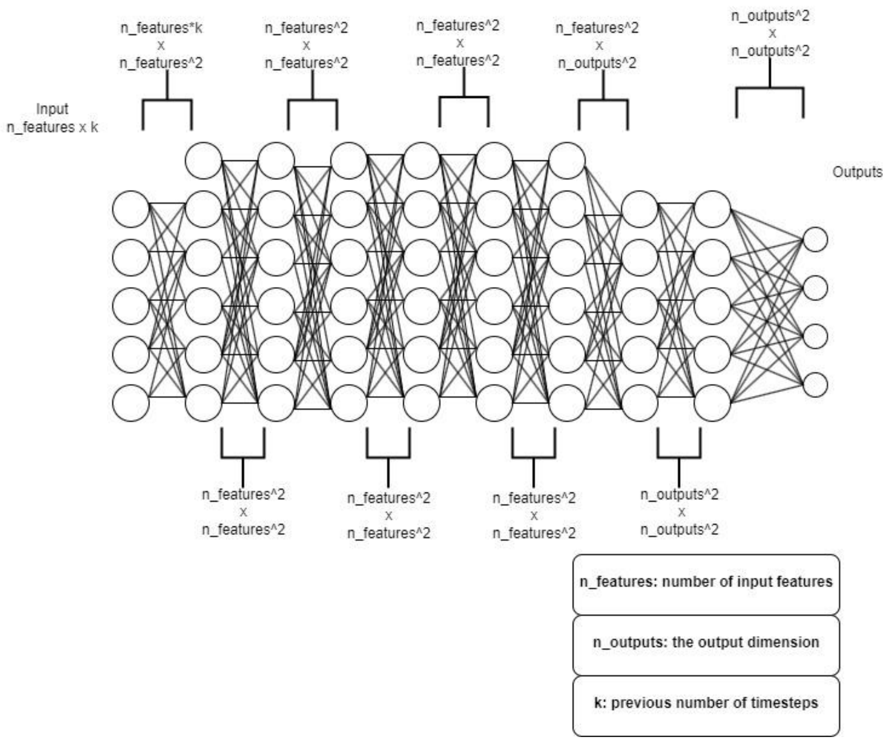
Train validation split

Since the dataset we have is missing a validation split we will manually do it starting from the training data. To perform this we must be sure, according to the chosen time step value, to preserve the sequential order of the measurement. In the current split the validation set is 10% of the training data.

Model architectures

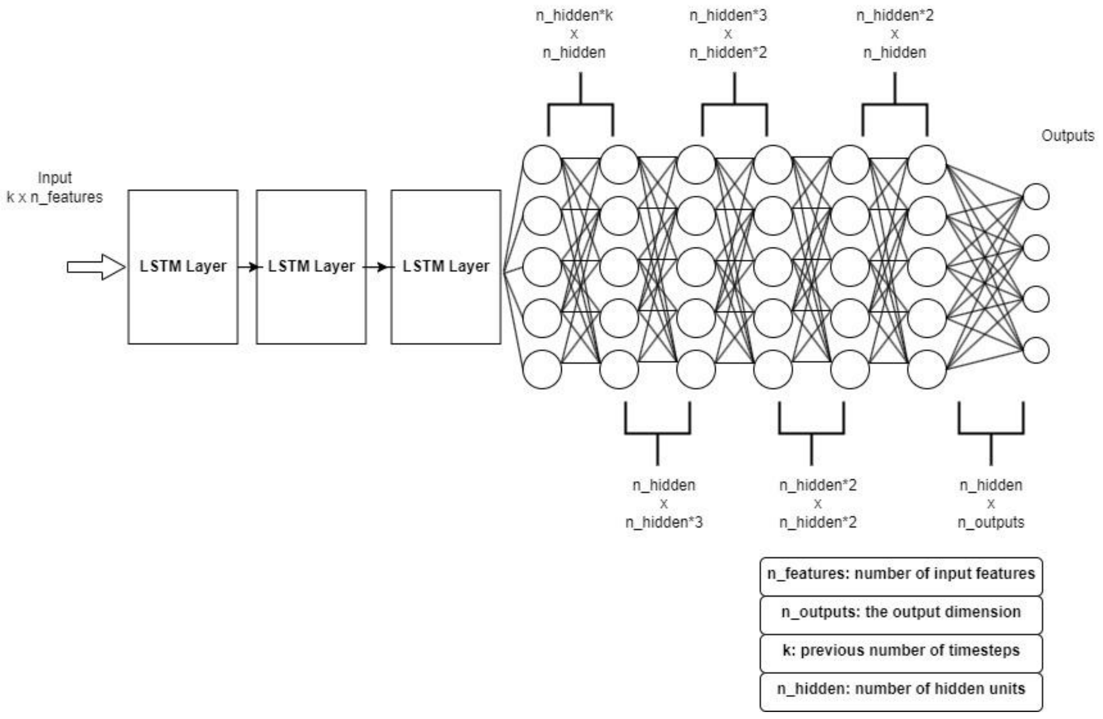
MLP baseline

The network is constructed of 9 dense layers with ReLU activation functions in between the layers. We flatten the input data to take the shape (batch size) x (number of features*number of previous timesteps). The value chosen for the number of previous timesteps is 4. The output of this model is a (number of previous timesteps) x 1 vector.


n_features*k x n_features^2
n_features^2 x n_features^2
n_features^2 x n_features^2
n_features^2 x n_outputs^2
n_outputs^2 x n_outputs^2
Input n_features x k
Outputs
n_features^2 x n_features^2
n_features^2 x n_features^2
n_features^2 x n_features^2
n_outputs^2 x n_outputs^2
n_features: number of input features
n_outputs: the output dimension
k: previous number of timesteps

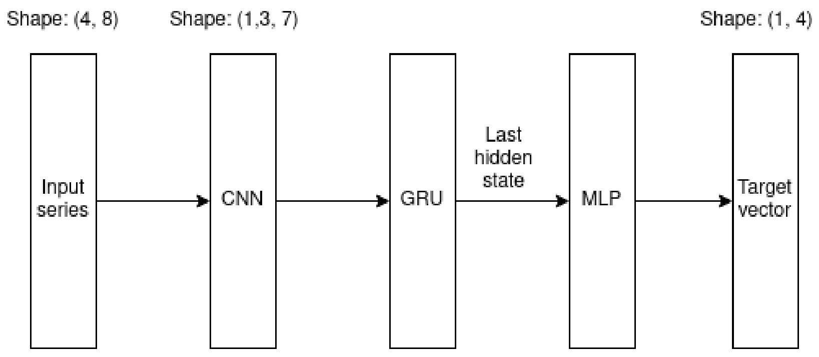
LSTM baseline

The LSTM network also comprises 9 layers in total. The first three layers are LSTM layers stacked upon each other. These layers further connect to a fully connected network with 6 layers. ReLU activation function is used between these layers. The input data takes the shape (batch size) x (number of previous timesteps) x (number of features). This is further flattened to the shape (batch size) x (number of features*number of previous timesteps) and is fed into the fully connected layers. The output of this model is (number of previous timesteps) x 1 as well.



Custom model

The custom model composes different neural network layers with the aim of taking advantage of the different peculiarity that each network can offer. The first layer is a convolution and also if it is usually used for images it can be also applied to time series. In our model we have a kernel size of 2x2 with stride equal to 1 and no padding. After the convolution we have a GRU layer, the last hidden state produced by this sequential model is used as input of an MLP that will output the target values. The activation function used is a ReLU and it is applied to the hidden state of the GRU and between the MLP layers.



Hyperparameters

The hyperparameters like dropout, learning rate, hidden size and number of layers are chosen in an empirical way. We tried different combinations of parameters and we kept the one that performs better on the validation dataset.

Training

The training is done using the normalized training dataset after the validation split. In all the models the optimizer chosen is Adam and the loss function is the MSE. The batch size used is equal to 50 instances and the number of epochs is set to 20. The learning rate, however, varies slightly for each model.

Results

Metrics

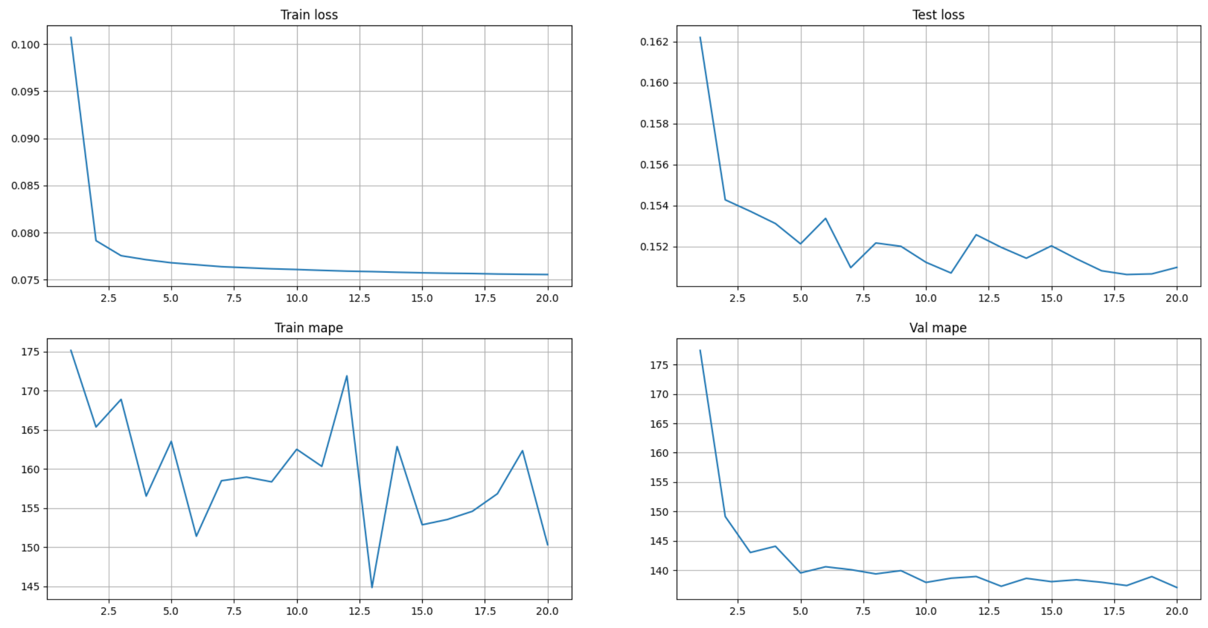
The metrics used for evaluating the models are the following:

- MSE: which is also the loss function used.
- MAPE: is a metric that defines the accuracy of a forecasting method. It represents the average of the absolute percentage errors of each entry in a dataset to calculate how accurate the forecasted quantities were in comparison with the actual quantities.

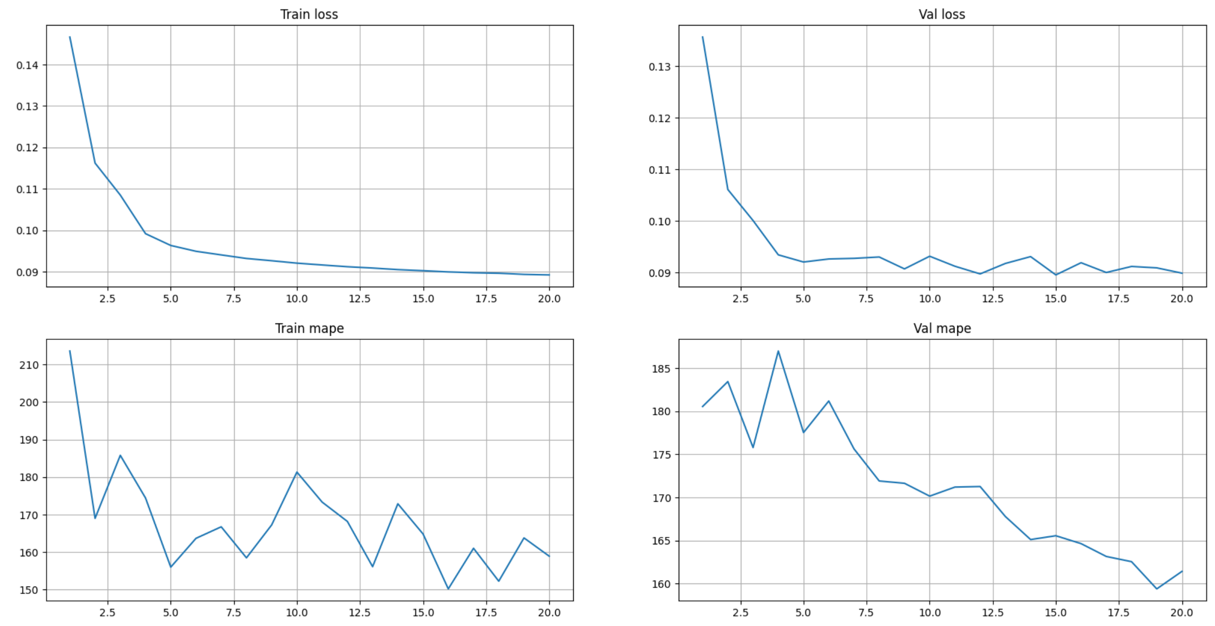
Model comparison

In this section we can see in the following figures the different metrics for the training and validation dataset for each model.

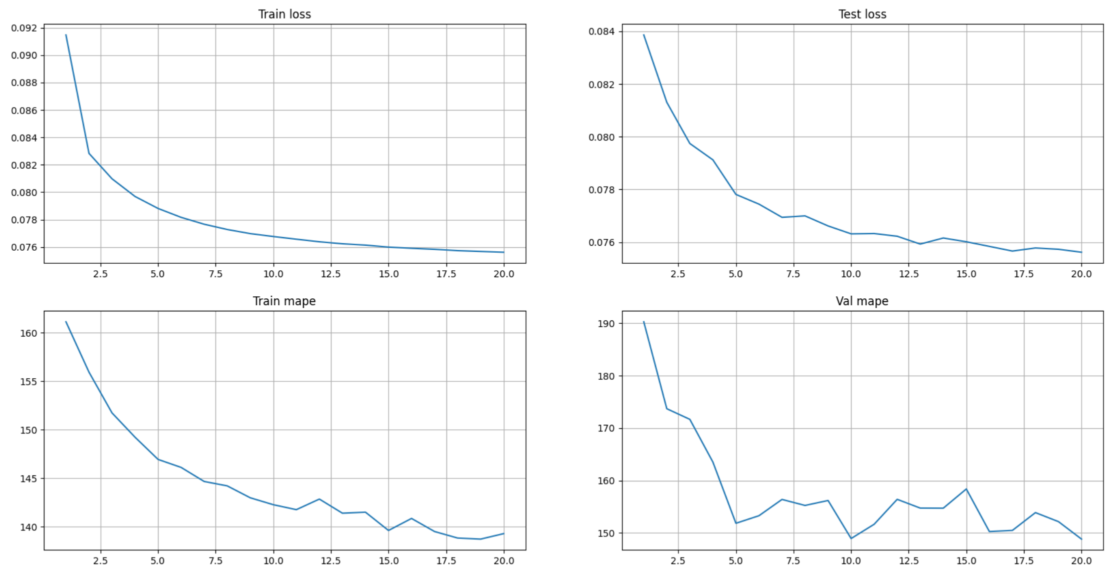
MLP baseline



LSTM baseline



Custom model



Model comparisons

Now let’s compare our model over the test set. In the first table we can see the MSE for each of the target features. In the second table we have the overall metrics for all the features combined. We can see how all the models struggle to predict the correct value of the **wind velocity.** The LSTM model and the CNN + GRU performs better than a simple MLP (as expected).

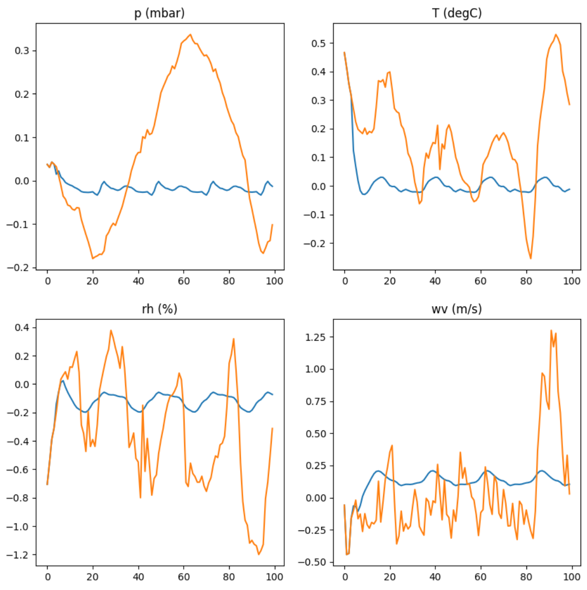
Model	**p (mbar)**	**T (degC)**	**rh (%)**	**wv(m/s)**
MLP	9.312	8.619	8.619	1564.743
LSTM	0.046	0.063	0.099	1725.503
Custom	0.036	0.044	0.072	1724.909

Model	**MSE**	**MAPE**
MLP	398.077	76.222
LSTM	431.427	81.715
Custom	431.265	77.424

Additional analysis

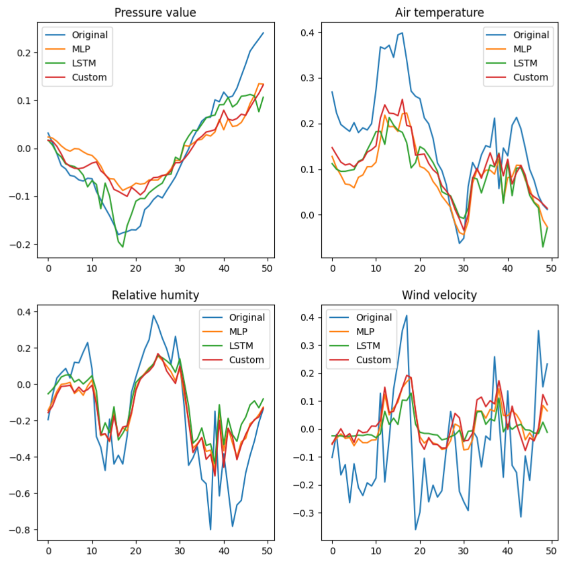
Error propagation

Error propagation is a common problem that occurs when a model uses its own predictions to make further provision. Also if we have a small error in the initial predictions it becomes bigger in the successive ones. In the plot below we can see how our custom model is affected by this problem. Instead of predicting the correct values, it starts predicting the same values periodically.



Predictions comparison

Another possible comparison between the models is the one between the predicted values and the ground truth. In this comparison, each model is feeded with the original *t-k* samples and the predicted values are plotted on the same graph to show which model gives the most accurate forecast. All the models act really similarly, as said before all of them have difficulty in predicting the wind velocity.


Pressure value
Air temperature
Relative humity
Wind velocity
Original
MLP
LSTM
Custom