Vision, Modeling, and Visualization (2017)
M. Hullin, R. Klein, T. Schultz, A. Yao (Eds.)

Visualization of Neural Network Predictions for Weather Forecasting

Isabelle Roesch and Tobias Günther

Computer Graphics Laboratory, ETH Zürich, Switzerland

Abstract

Recurrent neural networks are prime candidates for learning relationships and evolutions in multi-dimensional time series data. The performance of such a network is judged by the loss function, which is aggregated into a single scalar value that decreases during successful training. Observing only this number hides the variation that occurs within the typically large training and testing data sets. Understanding these variations is of highest importance to adjust hyperparameters of the network, such as the number of neurons, number of layers or even to adjust the training set to include more representative examples. In this paper, we design a comprehensive and interactive system that allows to study the output of recurrent neural networks on both the complete training data as well as the testing data. We follow a coarse-to-fine strategy, providing overviews of annual, monthly and daily patterns in the time series and directly support a comparison of different hyperparameter settings. We applied our method to a recurrent convolutional neural network that was trained and tested on 25 years of climate data to forecast meteorological attributes, such as temperature, pressure and wind speed. The presented visualization system helped us to quickly assess, adjust and improve the network design.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms

1. Introduction

Modeling relationships and trends in time series data is important to forecast the future development. In the past, recurrent neural networks (RNNs) have been successfully used to forecast time series, e.g., for market prediction [KB96], network traffic forecasting [ETFD97] and weather forecasting [HRLD15]. A common approach for RNNs is to use a sequence of previous time steps to predict the next step. Generating predictions for each time step of the testing data results in very large data sets that cannot be shown at once. Additionally, each RNN can be trained with a different set of hyperparameters like the number of neurons, the number of layers or the number of past time steps. The prediction error of the trained RNNs varies spatially and temporally across different regions of the training and test data. In this paper, we present an interactive visualization tool that allows the user to directly compare the performance of individual networks on both training and test data in the context of weather prediction. We provide multiple hierarchical views to extract the levels of detail, including annual, monthly and daily trends, down to individual data points. We support a direct comparison of forecasting models that were generated with different parameters. This allows meteorologists to analyze the output of multiple recurrent and convolutional neural networks that were trained to predict meteorological attributes, such as temperature, pressure and wind. Among others, users can spot overfitting, systematic prediction errors, outliers, trends, temporal patterns and adjust the training data and hyperparameters to improve predictions.

2. Related Work

For neural networks, most work focuses on visualizing the learned weights and the structure of the network. Karpathy et al. [KJL15] implement a method to visualize and interpret RNNs, Liu et al. [LSL*17] focus on the visual representation of convolutional filters of CNNs. The visualization method described by Zintgraf et al. [ZCAW17] helps to understand classification decisions made by artificial neural networks (ANNs). Other work that addresses visualizations of neural network classifiers include [SVZ14] and [BBM*15]. Note that forecasting weather attributes needs a regression ANN with a continuous output in contrast to classification ANNs, which were used in most previous related work.

There have been several approaches to weather forecasting using ANNs. The forecasting performance of different network structures has been analyzed by Hossain et al. [HRLD15]. Their work also describes the influence of different weather attributes on the prediction. To train the network, they use weather data from only one weather station, while we use adjacent grid data around our loca-

DOI: 10.2312/vmv.20171260

tion of interest. Grover et al. [GKH15] design a hybrid model for weather forecasting which combines an ANN with the influence of atmospheric laws on weather attributes. For the sake of simplicity, our work does not yet include physical dependencies among weather attributes and is instead completely data-driven.

3. Background

Artificial neural networks (ANNs) are an excellent tool to discover hidden patterns in nonlinear and complex structured data. Neural networks aim to learn the function $f(x)$ of the equation $f(x) = y$, where x is the input and y is the target output. Traditional feed-forward networks consist of three parts: An input layer that receives input x, one or more hidden layers and an output layer, see Fig. 1a. Each layer has a number of interconnected hidden units (neurons) whose weights are adjusted in the training to minimize the error between the network's output $f(x)$ and target value y, i.e. the *loss*.

Convolutional Neural Networks (CNNs) learn spatial features in data that is given on regular grids, e.g., images or volumes. Instead of training individual weights for the neurons, convolutional layers train small weight patches, called filters or kernels, which has two benefits: (a) Weights are shared between neurons which reduces the amount of training and redundant parameters. (b) A local connectivity assures that adjacent data in the grid is treated differently compared to grid points far away. Another necessary ingredient of CNNs are pooling layers, which reduce the number of parameters in the network by aggregating patches of data into a single value, e.g., using the maximum. A CNN is illustrated in Fig. 1b.

Recurrent Neural Networks (RNNs) are neural networks that contain loops. Instead of just propagating the input information straight through the network, a RNN layer also receives its previous output as input. RNNs have memory and are thus suitable for problems with a temporal dimension. A schematic RNN architecture can be seen in Fig 1c. However, standard RNNs do not perform well in practice when they are used to solve tasks that contain long-term dependencies, because the error gradient that is propagated back through the network is prone to either vanish or explode [BSF94].

Long Short-term Memory Networks (LSTMs) are specifically designed to avoid the vanishing gradient problem of standard RNNs and are capable to learn long-term dependencies [HS97]. LSTMs became the most commonly used type of RNN, succeeding in a wide variety of tasks including speech recognition [GMH13], text generation [SMH11] and time series prediction [SWG05].

4. Neural Network for Weather Forecasting

The ANN we designed for weather forecasting uses a time series of regular grids as input, which are centered around a location of interest. The network architecture is depicted in Fig. 2. A single time step of the input data consists of an $N \times N \times M$ grid of measured normalized meteorological attributes, such as temperature or surface pressure. The spatial $N \times N$ slices provide additional information around the location of interest (we used $N = 7$). The number of grid layers M (depth of the grid) corresponds to the number of meteorological attributes, which are used to train a particular network. The ANN consists of a convolutional part (CNN) that is followed by a recurrent part (RNN). The convolutional part applies F convolution filters (we used $F = 8$ with a support of $3 \times 3 \times M$), which results in an $N \times N \times F$ feature volume. To reduce the computational cost of the network training and to reduce overfitting, a max-pooling aggregates spatial 2×2 patches, which results in a $\lceil \frac{N}{2} \rceil \times \lceil \frac{N}{2} \rceil \times M$ grid. Convolution and pooling are done for each time step of the time series individually. After flattening the grid into a vector, the recurrent layers consider the temporal features of the time series, using multiple LSTM layers (we used 2–4), each followed by a 20% dropout to prevent overfitting. To conclude the network and to convert the output into the desired format, a dense layer is used. The network forecasts all M given meteorological attributes simultaneously for the location of interest. We used the mean squared error as objective loss function to train the network.

5. Comparative Visualization of Time Series

A common problem when designing an ANN is the optimization of its hyperparameters, such as the number of layers, number of neurons or which parts of the input data to use. Every combination of values for those parameters results in a new forecasting model. While automatic hyperparameter optimization methods exist [BB12], the performance of the network is usually only assessed by a single value (the residual or *loss*) that is plotted over the training epochs in the *Loss History*. With our visualization, we provide an in-depth analysis of the forecasting performance for individual hyperparameter settings as well as for comparisons between settings, starting from coarse overviews down to daily events.

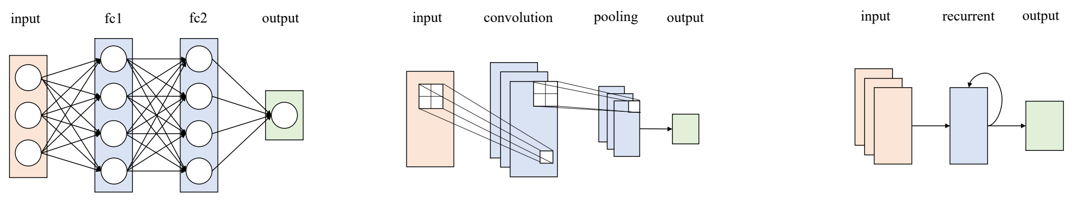
5.1. Error Metric

First, we define the error metric that we use to compare different forecasting models. Input to our method are time series of the various predictions of meteorological attributes, such as temperature, pressure and wind components for a given location on Earth. For each time series, an ensemble is available that was generated using different weather forecasting models, as described in Section 4. For a given ensemble member k, the temperature series is $\mathbf{t}^k = (t_1^k, ..., t_n^k)$, the pressure series is $\mathbf{p}^k = (p_1^k, ..., p_n^k)$ etc., where n is the number of time steps. While the weather forecasting models use one or several meteorological time series for training, our visualization will always show the performance of only one attribute at a time. Let $\mathbf{a} \in \{\mathbf{t}, \mathbf{p}, ...\}$ be the selected attribute. Then, the time series of ensemble member k is $\mathbf{a}^k = (a_1^k, ..., a_n^k)$ and the full ensemble is $\mathcal{A} = \{\mathbf{a}^1, ..., \mathbf{a}^K\}$, where K is the total number of ensemble members, i.e., neural networks trained for weather forecasting. Further, we denote the ground truth time series of the selected attribute as $\tilde{\mathbf{a}} = (\tilde{a}_1, ..., \tilde{a}_n)$, which is the measured data.

Given the ground truth $\tilde{\mathbf{a}}$, we define the error time series $\boldsymbol{\varepsilon}^k$ of a given ensemble member k as the element-wise absolute difference to the ground truth (for brevity, we drop the dependence of $\mathbf{a}$ in $\boldsymbol{\varepsilon}^k$):

$$\boldsymbol{\varepsilon}^k = \left| \mathbf{a}^k - \tilde{\mathbf{a}} \right| \tag{1}$$

To compare the errors of two ensemble members k_1 and k_2, we use their respective mean absolute errors $\overline{\boldsymbol{\varepsilon}^{k_1}}$ and $\overline{\boldsymbol{\varepsilon}^{k_2}}$ as well as their relative difference $(\overline{\boldsymbol{\varepsilon}^{k_1}} - \overline{\boldsymbol{\varepsilon}^{k_2}})$.



(a) *A standard feed-forward neural network with 2 fully connected (fc) layers.*

(b) *A convolutional neural network with a convolutional and a pooling layer.*

(c) *A recurrent neural network with one hidden recurrent layer.*

Figure 1: *Overview of common neural network architectures: feed-forward (a), convolutional (b) and recurrent neural networks (c).*

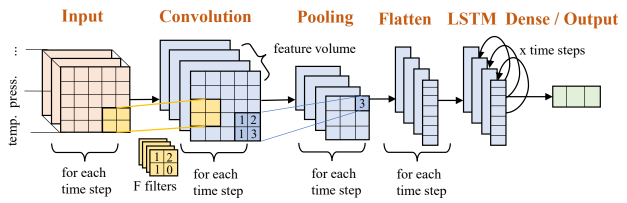


Figure 2: *Schematic illustration of our network, which combines convolutional and recurrent layers to forecast meteorological data.*

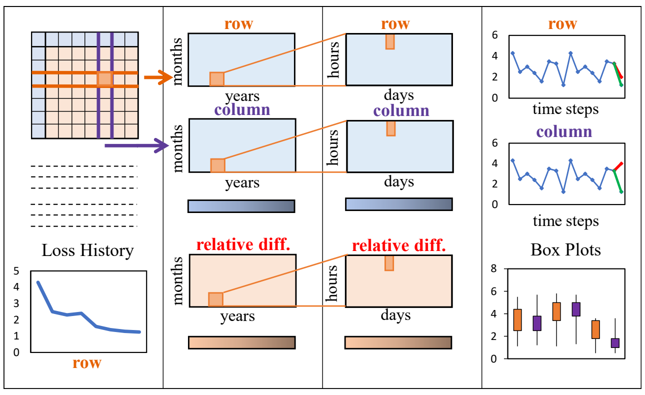
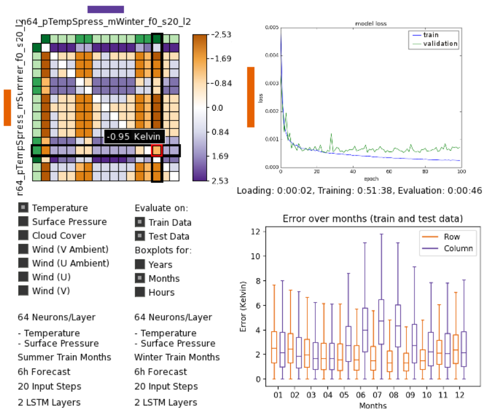


Figure 3: *Schematic overview of our interactive coarse-to-fine visualization tool, including a* Selection Matrix *(a) to select the two models to compare, a* Full Overview *(b) to display the performance and difference for the entire time sequence, a* Month Overview *(c) that displays individual data points within a month, and a* Time Step View *(d) that shows the past time steps, used to create the predictions. In addition, the* Loss History *is shown for the* row *model (a) and mean and standard deviation are aggregated in* Box Plots *(d).*


r64_pTempSpress_mWinter_f0_s20_l2
r64_pTempSpress_mSummer_f0_s20_l2
-0.95 Kelvin
-2.53
-1.69
-0.84
0.0
0.84
1.69
2.53
model loss
train
validation
loss
epoch
Error over months (train and test data)
Row
Column
Error (Kelvin)
Months

Loading: 0:00:02, Training: 0:51:38, Evaluation: 0:00:46

▣ Temperature
■ Surface Pressure
■ Cloud Cover
■ Wind (V Ambient)
■ Wind (U Ambient)
■ Wind (U)
■ Wind (V)

Evaluate on:
▣ Train Data
▣ Test Data
Boxplots for:
■ Years
▣ Months
■ Hours

64 Neurons/Layer	64 Neurons/Layer
- Temperature - Surface Pressure	- Temperature - Surface Pressure
Summer Train Months	Winter Train Months
6h Forecast	6h Forecast
20 Input Steps	20 Input Steps
2 LSTM Layers	2 LSTM Layers

Figure 4: *The entry point of the exploration is the* Selection Matrix *(top left). The variables to display, the data to evaluate and the choice of the box plot are adjusted below (bottom left). The* Loss History *(top right) reveals overfitting and underfitting for the* row *model.* Box Plots *(bottom right) show mean and standard deviation.*

5.2. Overview

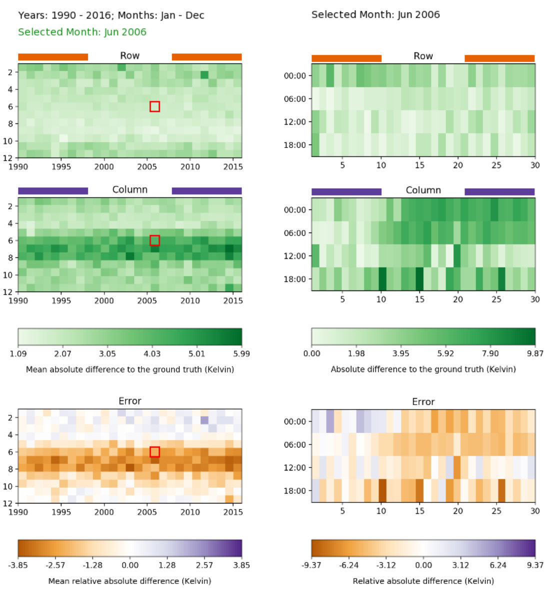
A schematic overview of our interactive tool is shown in Fig. 3. The principle of providing linked overview and detail views has been widely used in information and scientific visualization [Hau06, VWVS99, SWLL13]. In our visualization tool, the coarse-to-fine workflow follows from left to right. We precompute and store the mean absolute errors $\overline{\boldsymbol{\varepsilon}^k}$ of every model k according to Eq. (1), which are later read for the selected meteorological attribute. For comparison of the individual models, the relative differences are displayed pair-wise in a *Selection Matrix*, see Fig. 3a. The user can select two models for closer inspection by choosing a row and a column. Below the matrix, the *Loss History* is shown for the selected row. As soon as the models are selected, the mean difference $\boldsymbol{\varepsilon}^k$ to the ground truth is visualized for each model, as well as the models' relative difference, with values averaged for each month, see Fig. 3b. We denote these three plots as the *Full Overview*, as they provide an overview of the full time series. In these plots, the user can select a month, which is displayed in more detail in the *Month Overview*, which shows all data points (individual predictions) for the specified month, see Fig. 3c. After selecting a time step in the *Month Overview*, a detailed plot for this particular forecast is shown in the *Time Step View*, in which the prediction, the ground truth and the time steps are shown that the network received as input, see Fig. 3d. Below, a *Box Plot* is shown to compare the obtained mean and standard deviation of $\boldsymbol{\varepsilon}^k$ for the two selected models. In the following sections, we explain the visualizations in more detail.

5.3. Selection Matrix

As an entry point for the exploration, we devised a visualization that allows the user to quickly determine how the individual models compete with each other. Given the mean absolute errors $\overline{\boldsymbol{\varepsilon}^k}$ of each model k (averaged over entire time series), *Selection Matrix* $\mathbf{S}$ is:

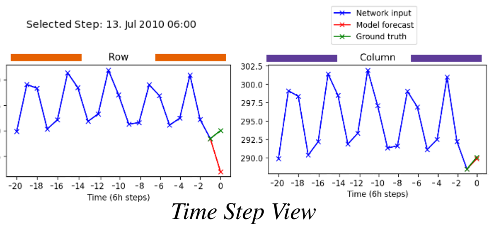
$$\mathbf{S}_{ij} = (\overline{\boldsymbol{\varepsilon}^{k_i}} - \overline{\boldsymbol{\varepsilon}^{k_j}}) \tag{2}$$

Each element of this anti-symmetric matrix directly compares two models, which are identified by the row and column index. The



(a) Full Overview **(b)** Month Overview

Figure 5: *The selected month in the* Full Overview *(left) is viewed in detail in the* Month Overview *(right).*



Time Step View

Figure 6: *Side-by-side comparison of predictions for both selected models, including all past data points received by the networks.*

models are sorted by their hyperparameters; other choices are imaginable. If an entry is positive, model k_i (row) is better and if it is negative, model k_j (column) is more accurate. We visualize this matrix in Fig. 4 (top left) using a diverging color map from [Cyn17]. The exact error value appears as text when the cursor hovers over the respective texel. Note that the colors orange and purple from now on refer to properties of the row and column model, respectively.

The green margins on the top and left of the matrix directly show the mean absolute errors $\overline{\boldsymbol{\varepsilon}^k}$ of the individual models. A brighter shade of green indicates a small error compared to the ground truth, whereas a darker shade means the opposite. While here only the mean error is shown, information on the standard deviation is available in *Box Plots*, which show the performance of the two selected models side-by-side, see Fig. 4 (bottom right). Using radio buttons, users select whether to average over years, months or hours. In addition, hyperparameters of the selected models are listed.

To the bottom left of the matrix, the weather parameter k to display can be chosen. Since not all models use the same set of weather parameters, only those with the selected parameter are displayed. Further right, the user can select on which part of the data the evaluation should happen, i.e., on the training data, the test data or both. It is useful to see how well the network performed on both training and test data to spot behavior like overfitting, e.g., in Fig. 7.

For the network of the currently selected row, the *Loss History* is shown, see Fig. 4 (top right). In addition, timing measurements for loading the data, training the network and evaluation on the data are listed. The loss is shown for 100 epochs of both the training data and the validation data, which consists of 5% of the training set. Ideally, the validation loss decreases monotonically. In case of overfitting, however, it starts to increase, since the network tends to memorize the observed training data instead of generalizing it.

5.4. Overview of Time Series

To visualize the forecast performance, we follow a coarse-to-fine approach, which includes two overview visualizations, see Fig. 5.

Full Overview. The *Full Overview* consists of three coordinated views: from top to bottom there are two individual plots that show the mean errors $\overline{\boldsymbol{\varepsilon}^{k_1}}$ and $\overline{\boldsymbol{\varepsilon}^{k_2}}$ of the currently selected models, and a difference plot that shows the relative difference $(\overline{\boldsymbol{\varepsilon}^{k_1}} - \overline{\boldsymbol{\varepsilon}^{k_2}})$. For each model, we compute an average error per month and lay out the data in a 2D matrix, where the columns show the years and the rows the respective months. Aligning months and years this way allows observing annual and seasonal patterns, see Fig. 5a for an example.

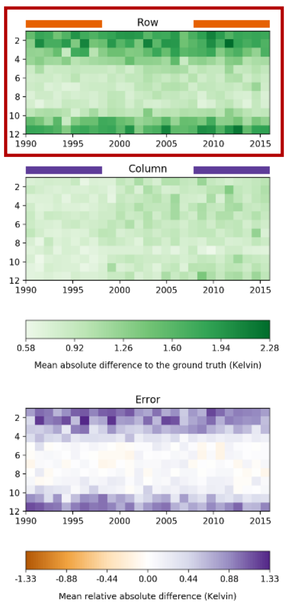
Month Overview. If the user selects a month in the previous *Full Overview*, a more detailed *Month Overview* of that month appears next to it to the right, see Fig. 5b. This view uses the same layout and colormaps as the first overview, but differs in the data it visualizes. Here, the columns show the days of the selected month and the rows the hours. The temporal resolution of our data provides a value every six hours, which results in four values per day. This overview provides insights into a model's night and day time performance.

5.5. Time Step View

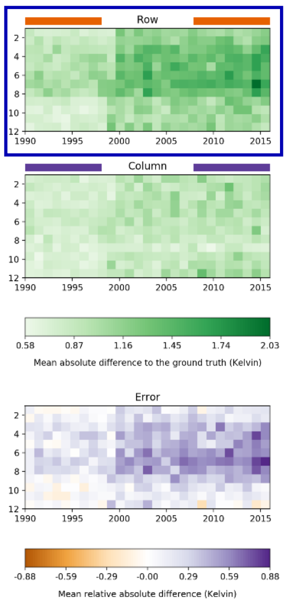
Selecting a data point in the previous *Month Overview* opens the *Time Step View*, which concludes our coarse-to-fine sequence. The respective predictions, the ground truth, as well as the previous time steps that the two selected networks received as input are plotted side-by-side in Fig. 6. While the *Month Overview* only shows the difference to the ground truth, this more detailed view sheds light onto the past time steps that led to a particular prediction.

6. Results

For the training and evaluation of the models we used an ERA-Interim reanalysis of the European Centre for Medium-Range Weather Forecast (ECMWF) [DUS*11]. We used data from 1990–1999 to train the models and data from 1990–2016 to evaluate them (separating testing and training). The data has a spatial grid spacing of 0.75/0.75 degrees (lon/lat) and a time step of six hours. Around Zürich (Switzerland), we extracted a time series of 7×7 grids. This resolution performed best in our initial tests and is another hyperparameter. The weather attributes at the grid points include temperature, surface pressure, cloud cover, and the U and V wind components. These particular attributes were used to train the networks and were selected based on advice from meteorologists. In the paper, we visualize the prediction result for temperature only. We refer to the video for visualizations of the other parameters.



(a) *Underfitting: 32 neurons (row) vs. 64 neurons (column).*



(b) *Overfitting: temperature (row) vs. temp. and pressure (column).*

Figure 7: *Left: 64 neurons (column) learn well, while 32 neurons (row) underfit, as they only learn summer months (□). Right: using only temperature as input (row) overfits the training data compared to a model with temperature and surface pressure (column), see the light green colors in the training months, i.e., 1990–1999 (□).*

6.1. Visualizations

Next, we explore different hyperparameter settings, including number of neurons in the LSTM layer, forecast range and training data.

Underfitting. The selection of the number of neurons is demanding. The models shown in Fig. 7a only differ in the number of neurons in their LSTM layers. While the column model with 64 neurons learns well on the complete domain, the row model with 32 neurons learns only a subset. We also tested a model with 128 neurons, but the results were slightly worse due to overfitting, i.e., the network memorized the training data instead of generalized from it.

Overfitting. In Fig. 7b, the opposite is observed. The row model with only temperature as input overfits on the training data (1990-1999), which appears brighter in the *Full Overview* than the testing data (1999–2016). Adding surface pressure as a second weather attribute as in the column model helps to avoid this undesirable phenomenon. Both models used 64 neurons.

Training on Specific Season. Fig. 8 gives examples of networks that trained only on certain months. If only the summer months were learned (row), the model performed slightly better in the summer than in the winter, on both training and test data. The opposite, yet far more pronounced behavior can be observed when only training on winter months (column), which performed clearly worse in the summer. In the selected region, summer months are thermodynamically much more difficult to predict (even for operational weather forecasting models). A generalization from winter models is therefore not possible. Compared to a model that could learn on all months, the specialized models do not perform better, which can be explained by the much larger variety of examples seen by a model training on the entire year than only on three months. The *Box Plots* give further insights into the performance of the models.

12h vs. 6h Forecast. The forecast distance is another parameter of interest. Most models we trained predict only one time step (6 hours) into the future. In Fig. 9, we compare a 6 hour forecast (column) with a 12 hour forecast (row). The 6h model learns to generalize better compared to the 12h model, which overfits slightly on the training data. This can be seen in the *Full Overview* as well as in the *Box Plots* summarized over years. The *Loss History* does not show an alarming decrease in validation data performance, thus the model is still capable of learning, though not as well as the 6h model.

Hourly Patterns. An interesting hourly pattern can be observed in a model (row) that uses temperature, surface pressure and cloud cover as meteorological attributes. When looking at the *Month Overview* in Fig. 10, the 12:00 step stands out. The model fails to learn this step. The *Box Plots* show this striking phenomenon even more clearly. The *Loss History* reveals a further problem of this model: Neither the training loss nor the validation loss decreased during training. They even increased over the epochs, which leads to the conclusion that this model had problems to generalize from these attributes. Further experiments involving the cloud cover parameter frequently showed a similarly poor performance. Dropping the cloud cover led to a clear improvement and led us to the conclusion that this parameter is not suitable for this network configuration.

Wind Decomposition. The 2D wind direction is a promising atmospheric attribute, since it provides directional information on the possible pathways of clouds. Following Günther et al. [GGT17], we locally decompose the air flow $\mathbf{v}$ into two components $\bar{\mathbf{v}}$ and $\tilde{\mathbf{v}}$:

$$\mathbf{v} = \bar{\mathbf{v}} + \tilde{\mathbf{v}}, \qquad \text{s.t.} \qquad \int_U \left\| \frac{\partial \bar{\mathbf{v}}}{\partial t} \right\|^2 \mathrm{d}V \to \min \tag{3}$$

to separate features such as vortices from their ambient transport. Thereby, $\bar{\mathbf{v}}$ is $\mathbf{v}$ in a near-steady reference frame (the temporal derivative is minimized at each point in a local neighborhood U) and $\tilde{\mathbf{v}}$ contains the ambient movement—ideally, the transport direction of clouds. We trained and evaluated networks with $\mathbf{v}$ (row) or $\tilde{\mathbf{v}}$ (column) as input. Fig. 11 indicates that an extraction of the ambient flow $\tilde{\mathbf{v}}$ improves predictions for small numbers of neurons, as visible in the *Box Plots*, since networks do not have to learn the feature extraction themselves, which prevents underfitting. Note that unlike $\mathbf{v}$, the decomposition of Günther et al. [GGT17] in Eq. (3) is *objective* [TN65], i.e., $\tilde{\mathbf{v}}$ adapts to any smooth rotation and translation of the reference frame.

6.2. Performance

The implementation of our network architecture in Section 4, as well as the training and testing were done with Keras [Cho15] using the Tensorflow back end with GPU support. The following timings were measured with an Intel i7-4710MQ CPU, 8GB RAM and an NVIDIA GeForce GT 730M GPU with 2GB VRAM. The training of the networks took between 51 minutes and 16 hours for 100 training

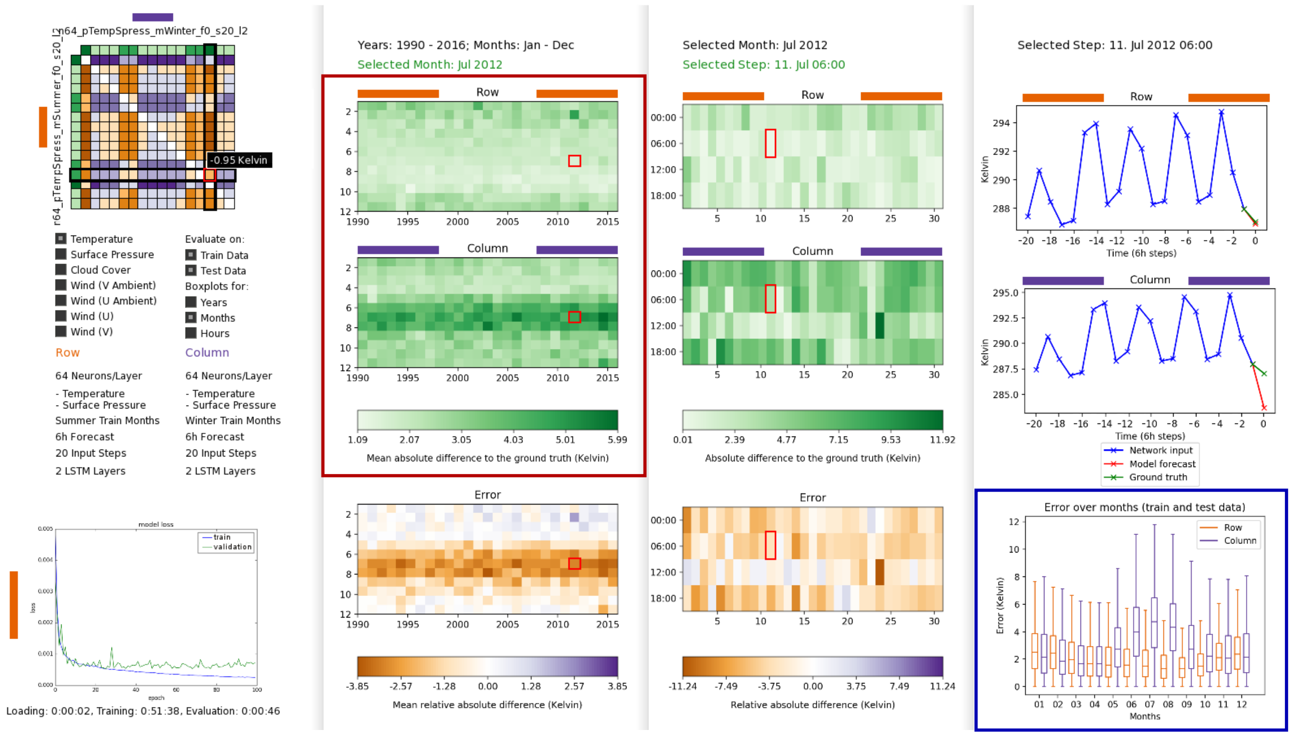


Figure 8: *Comparison of a network that has trained only on summer months (row) with one that only trained on winter months (column). The specialization is clearly visible in the* Full Overview *(□). Box Plots that summarize over months give further insight into the behavior of both networks. They show a higher mean absolute error and standard deviation in months unknown to the network (□).*

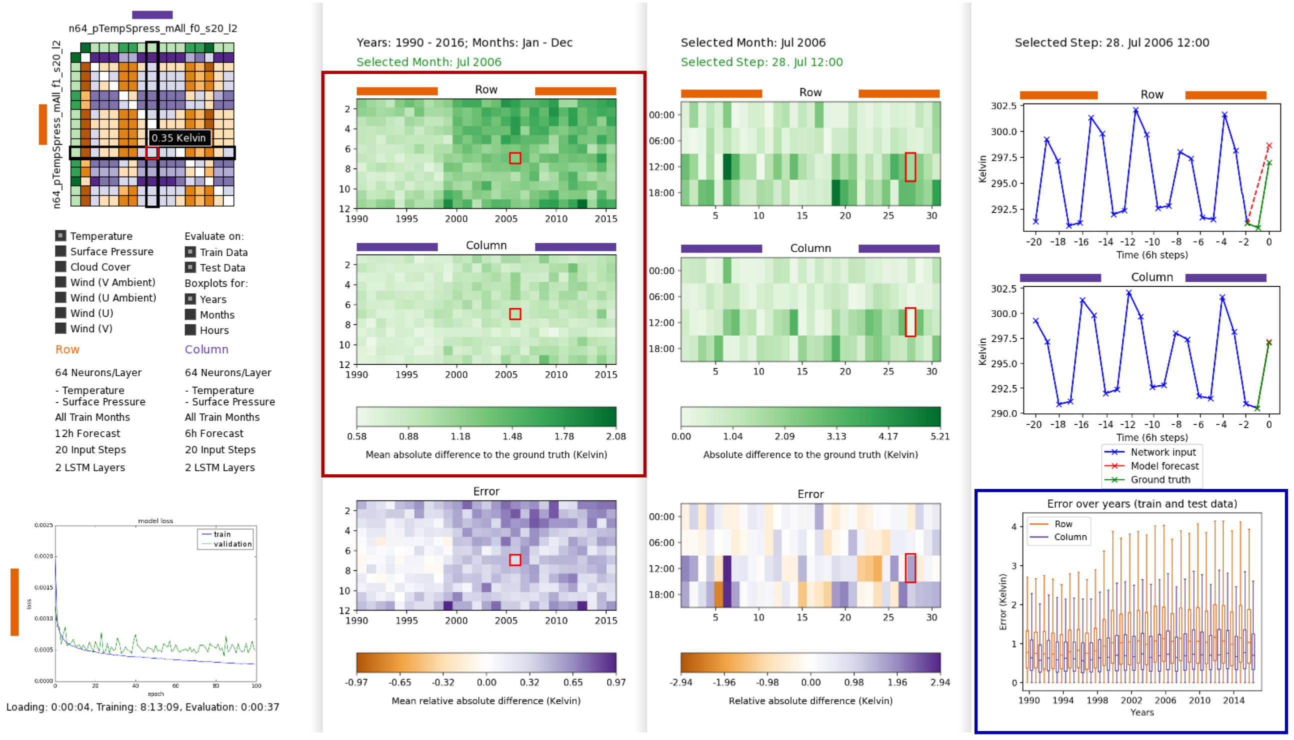


Figure 9: *Comparison of a 6 hour forecast model (column) with a 12 hour forecast (row). The 6 hour forecast performs better, since the 12 hour forecast experiences overfitting, which can be seen in the* Full Overview *(□) as well as in* Box Plots *that summarize over years (□). In the row model, we connect the 12 hour forecast with a dashed line to indicate that the skipped 6 hour data point is not existent.*

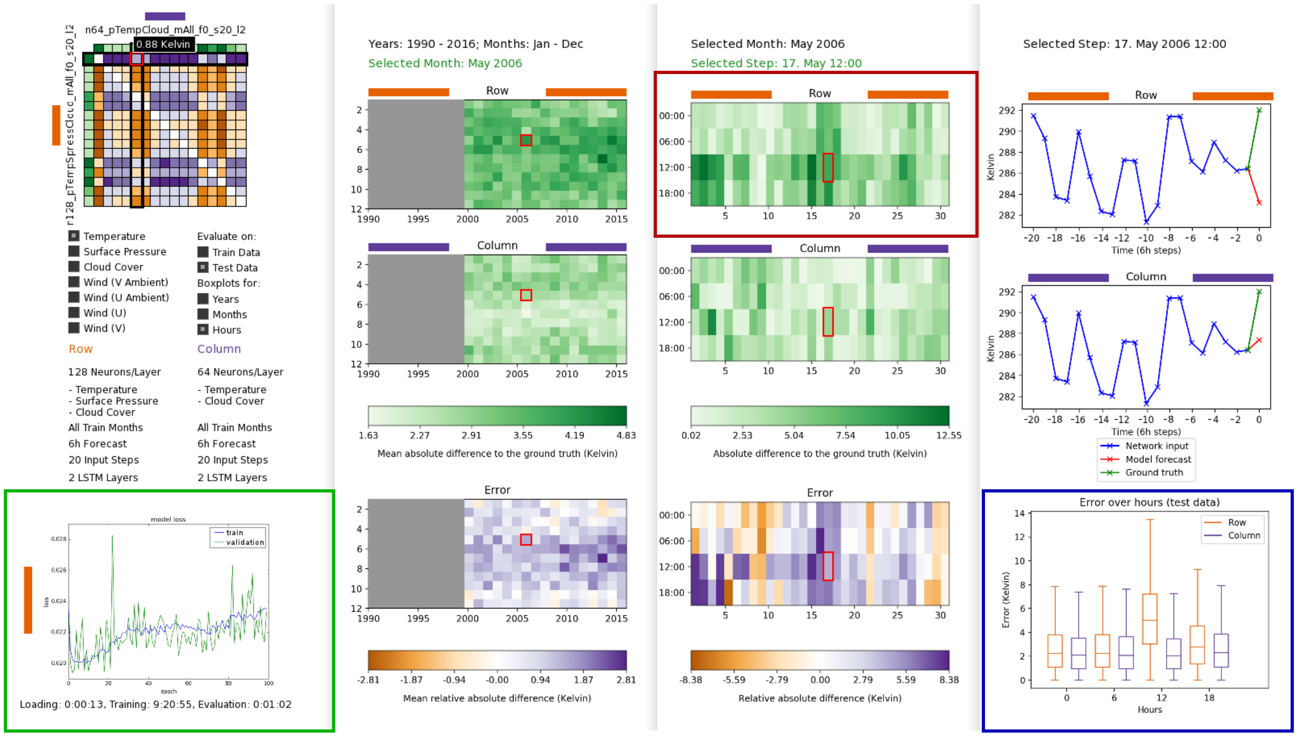


Figure 10: *The* Month Overview *(🟥) as well as the* Box Plots *(🟦) show a clear error spike around the 12:00 step for the row model. The* Loss History *(🟩) reveals that this particular model could not generalize since neither the training nor the validation loss decreased over the epochs. Here, the user chose to evaluate the performance on the test data only.*

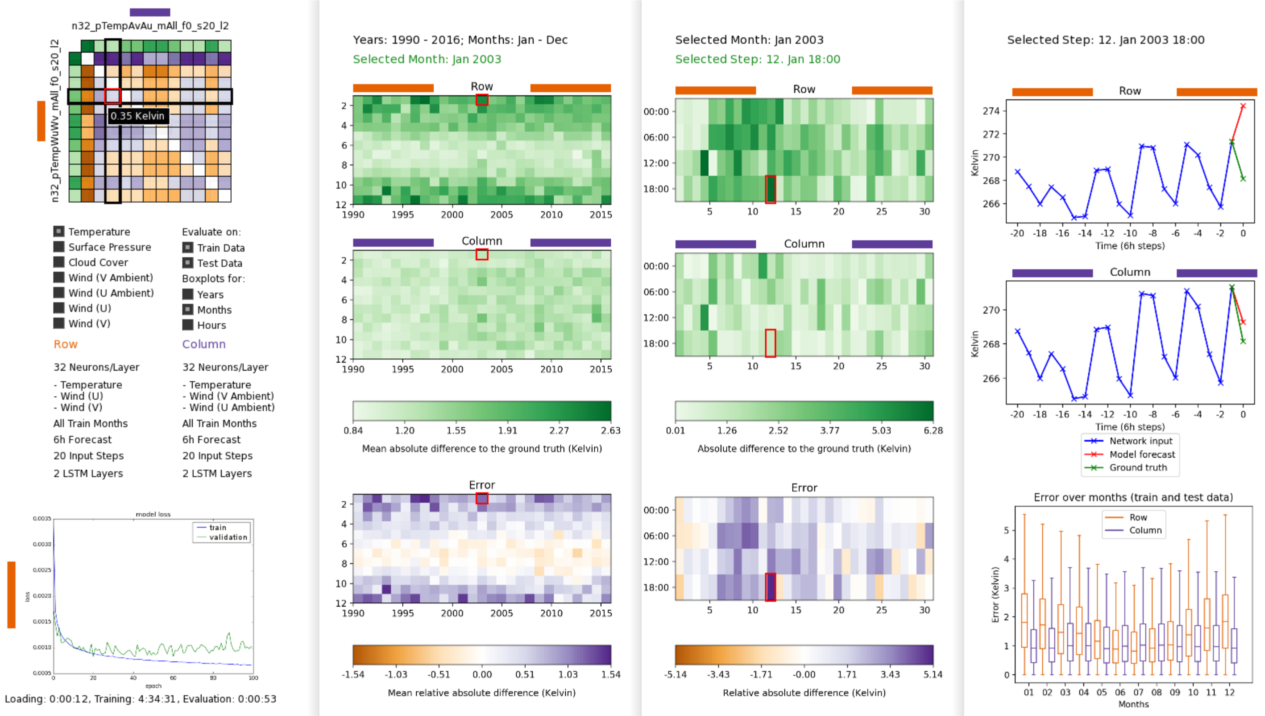


Figure 11: *Extracting the ambient transport using [GGT17] from the flow improved the predictions, as visible in the* Full Overview *and the* Selection Matrix. *The currently selected row model uses the raw wind speed components, while the column model uses the ambient transport in U and V direction. Both models also received the temperature attribute as input.*

Forecast Model	MAE	ME	SDE	Timings
COSMO-1 (Switzerland)	1.6K	-0.05K	2.1K	1 min/h
Our network (Zürich)	0.9K	0.14K	3.0K	1 ms/h

Table 1: *Forecasting statistics for the temperature 2 meters above the ground in 2016. The data of COSMO-1 (operational Swiss weather forecasting system) is courtesy of MeteoSchweiz [Wal17].*

epochs, depending primarily on the number of neurons, the number of input steps and the chosen weather attributes. After a network has been trained, a forecast of a single time step takes on average 1 millisecond. It took our models between 37 and 61 seconds to generate forecasts for the whole training and test data.

6.3. Discussion of Neural Network

Our visualizations encode the mean absolute error (MAE), which does not encode systematic overestimates or underestimates of the ground truth. This can be overcome by using the mean error (ME). MAE, ME and SDE (standard deviation of error) are listed in Table 1 for our best network (trained only for Zürich) and an operational weather forecasting system (entire Switzerland). Compared to a general meteorological model, our best network yields a lower MAE but a higher ME and SDE. For operational models a post-processing is applied, which is not taken into account here. Our neural networks can generate a prediction in 1 *ms*, whereas the meteorological model needs 1 *min* to forecast one hour. It is expected that a neural network that uses only few meteorological attributes cannot beat a physical and operational weather forecasting model. Interestingly, our network is in a similar order, yet much faster.

7. Conclusions

Our interactive visualization tool helped us to analyze the weather predictions of multiple neural networks and to directly compare their hyperparameters. We devised several linked views, including a *Selection Matrix* for an entry point of the exploration, a *Full Overview* and *Month Overview* to discover annual, monthly and daily patterns, and additionally showed *Time Step Views*, *Loss Histories* and *Box Plots* to enable deeper insight into the prediction performance. With our method, phenomena like underfitting, overfitting, and outliers are easier to spot and understand than before.

All our networks were trained for Zürich, where weather forecasts are typically difficult due to the topography. In the future, we would like to include the forecast location as an additional dimension. Instead of evaluating only one selected meteorological attribute, we would also like to analyze the correlations of multiple attributes.

References

[BB12] BERGSTRA J., BENGIO Y.: Random search for hyper-parameter optimization. *Journal of Machine Learning Research 13*, Feb (2012), 281–305. 2

[BBM*15] BACH S., BINDER A., MONTAVON G., KLAUSCHEN F., MÜLLER K.-R., SAMEK W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one 10*, 7 (2015), e0130140. 1

[BSF94] BENGIO Y., SIMARD P., FRASCONI P.: Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks 5*, 2 (Mar 1994), 157–166. 2

[Cho15] CHOLLET F.: Keras. https://github.com/fchollet/keras, 2015. access date: 12 June 2017. 5

[Cyn17] CYNTHIA A.: Brewer. http://colorbrewer2.org, 2017. access date: 12 June 2017. 4

[DUS*11] DEE D. P., UPPALA S. M., SIMMONS A. J., BERRISFORD P., POLI P., KOBAYASHI S., ANDRAE, ET AL.: The ERA-Interim reanalysis: configuration and performance of the data assimilation system. *Quarterly Journal of the Royal Meteorological Society 137*, 656 (2011), 553–597. 4

[ETFD97] EDWARDS T., TANSLEY D., FRANK R., DAVEY N.: Traffic trends analysis using neural networks. In *Procs of the Int. Workshop on Applications of Neural Networks to Telecommunications* (1997). 1

[GGT17] GÜNTHER T., GROSS M., THEISEL H.: Generic objective vortices for flow visualization. *ACM Transactions on Graphics (Proc. SIGGRAPH) 36*, 4 (2017), 141:1–141:11. 5, 7

[GKH15] GROVER A., KAPOOR A., HORVITZ E.: A deep hybrid model for weather forecasting. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015), ACM, pp. 379–386. 1

[GMH13] GRAVES A., MOHAMED A.-R., HINTON G.: Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2013), IEEE, pp. 6645–6649. 2

[Hau06] HAUSER H.: Generalizing focus+context visualization. In *Scientific visualization: The visual extraction of knowledge from data*. Springer, 2006, pp. 305–327. 3

[HRLD15] HOSSAIN M., REKABDAR B., LOUIS S. J., DASCALU S.: Forecasting the weather of Nevada: A deep learning approach. In *International Joint Conference on Neural Networks (IJCNN)* (2015), IEEE, pp. 1–6. 1

[HS97] HOCHREITER S., SCHMIDHUBER J.: Long short-term memory. *Neural Comput. 9*, 8 (Nov. 1997), 1735–1780. 2

[KB96] KAASTRA I., BOYD M.: Designing a neural network for forecasting financial and economic time series. *Neurocomputing 10*, 3 (1996), 215–236. 1

[KJL15] KARPATHY A., JOHNSON J., LI F.: Visualizing and understanding recurrent networks. *CoRR abs/1506.02078* (2015). 1

[LSL*17] LIU M., SHI J., LI Z., LI C., ZHU J., LIU S.: Towards better analysis of deep convolutional neural networks. *IEEE Trans. on Vis. and Computer Graphics (Proc. IEEE VAST 2016) 23*, 1 (Jan 2017), 91–100. 1

[SMH11] SUTSKEVER I., MARTENS J., HINTON G. E.: Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (2011), pp. 1017–1024. 2

[SVZ14] SIMONYAN K., VEDALDI A., ZISSERMAN A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop Papers* (2014). 1

[SWG05] SCHMIDHUBER J., WIERSTRA D., GOMEZ F.: Evolino: Hybrid neuroevolution/optimal linear search for sequence learning. In *Proc. International Joint Conference on Artificial Intelligence* (2005), Morgan Kaufmann Publishers Inc., pp. 853–858. 2

[SWLL13] SUN G.-D., WU Y.-C., LIANG R.-H., LIU S.-X.: A survey of visual analytics techniques and applications: State-of-the-art research and future challenges. *Journal of Computer Science and Technology 28*, 5 (2013), 852–867. 3

[TN65] TRUESDELL C., NOLL W.: *The nonlinear field theories of mechanics*. Handbuch der Physik, Band III/3, e by Flugge, S., (ed.) , Springer-Verlag, Berlin, 1965. 5

[VWVS99] VAN WIJK J. J., VAN SELOW E. R.: Cluster and calendar based visualization of time series data. In *Proc. IEEE Symposium on Information Visualization* (1999), pp. 4–9. 3

[Wal17] WALSER A.: Personal communication, 2017. MeteoSwiss. 8

[ZCAW17] ZINTGRAF L. M., COHEN T. S., ADEL T., WELLING M.: Visualizing deep neural network decisions: Prediction difference analysis. *arXiv preprint arXiv:1702.04595* (2017). 1