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 evolutions in multi-dimensional time series data. The performance of such a network is judged by the loss function, which is aggregated into a scalar value that decreases during 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 network hyperparameters, such as the number of neurons, number of layers or to adjust the training set to include more representative examples. In this paper, we design a comprehensive and interactive system that allows users to study the output of recurrent neural networks on both the complete training data and 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 velocity. We further visualize the quality of the forecasting models, when applied to various locations on Earth and we examine the combination of several forecasting models.

This is the authors preprint. The definitive version is available at http://diglib.eg.org/ and http://onlinelibrary.wiley.com/.

1. Introduction

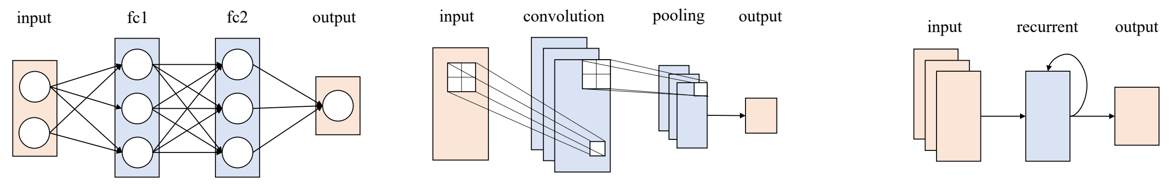
Modeling relationships and trends in time series data is important to forecast the future development. In the past, recurrent neural networks (RNNs) have been successfully used to forecast time series, e.g., for market prediction [KB96], network traffic forecasting [ETFD97] and weather forecasting [HRLD15]. 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. We further extend the method by combining network models trained for selected locations on Earth to create predictions for each grid point on Earth. The performance of the combined network is then evaluated and visualized.

2. Related Work

In recent years, neural networks enabled significant improvements in many scientific disciplines. Their strength comes from the capability to learn non-linear relationships, which can often out-perform hand-crafted features. Neural networks, however, have certain limitations. It is difficult to adjust their hyperparameters and to infer how and why a network works, which makes it difficult to make theoretical promises. Since networks are not transparent, it is not straightforward to extract new scientific insights about the learned problem, which might be from a scientific point of view not satisfactory. This is where visualization is needed: to build trust by analyzing internal data flows and to enable the extraction of scientific insights.

Visualization of Neural Networks. For neural networks, most work focused on visualizing the learned weights and the structure of the network. Karpathy et al. [KJL15] visualized and in-

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.



(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).*

terpreted RNNs, Liu et al. [LSL*17] focused on the visual representation of convolutional filters of CNNs. The method of Zintgraf et al. [ZCAW17] helps to understand classification decisions made by artificial neural networks (ANNs). Other work that addressed 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.

Neural Networks for Weather Forecasting. The forecasting performance of different network structures and weather attributes has been analyzed by Hossain et al. [HRLD15]. They used weather data from only one weather station, while we use adjacent grid data around our location of interest. Grover et al. [GKH15] designed a hybrid model for weather forecasting which combines an ANN with the influence of atmospheric laws on weather attributes. For simplicity, our work does not yet include physical dependencies among weather attributes and is instead completely data-driven.

Visualization of Time Series. Sun et al. [SWLL13] compiled a detailed summary of visual analytics approaches, including time series and geospatial data. Dang et al. [DAW13] explored high-dimensional and multivariate economic and financial time series and focused on cross-correlation between the different variables. Shi et al. [SCL*12] visualized ranking changes of search queries in large time series. Krstajic et al. [KBK11] detected events in multiple time series. McLachlan et al. [MMKN08] facilitated the analysis of large collections of system management time series data. We refer to Aigner et al. [AMM*08] for further related work on time series.

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 ANNs we designed for weather forecasting use 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 $N \times N \times K$ grid of 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 K (depth of the grid) corresponds to the number of meteorological attributes, 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 $3 \times 3 \times K$), resulting in an $N \times N \times F$ feature volume. To reduce the training time and to reduce overfitting, a max-pooling aggregates spatial 2×2 patches, which results in a $\lceil \frac{N}{2} \rceil \times \lceil \frac{N}{2} \rceil \times K$ grid. Convolution and pooling are done for each time step of the time series individually. After

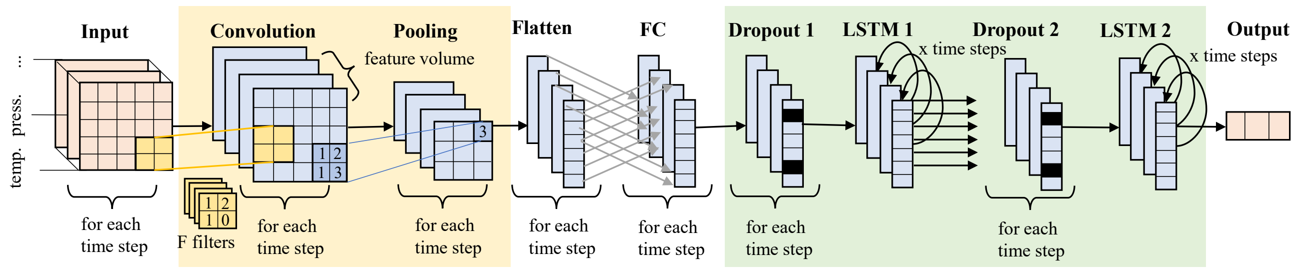


Figure 2: *Schematic illustration: the convolutional part which consists of the Convolution and Pooling layer is highlighted in yellow, while the Dropout and LSTM layers which form the recurrent part are highlighted in green. A flattening and a fully connected (FC) layer are in-between.*

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 K 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 metrics that we use to compare different forecasting models. Input to our method are time series of various predictions of meteorological attributes for a given location on Earth. For each time series, an ensemble was generated using different weather forecasting models, see Section 4. For a given ensemble member m, the temperature series is $\mathbf{t}^m = (t_1^m, ..., t_n^m)$, the pressure series is $\mathbf{p}^m = (p_1^m, ..., p_n^m)$ etc., where n is the number of time steps.

While our networks use one or several weather attributes 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 m is $\mathbf{a}^m = (a_1^m, ..., a_n^m)$ and the full ensemble is $\mathcal{A} = \{\mathbf{a}^1, ..., \mathbf{a}^M\}$, where M is the total number of ensemble members, i.e., the number of 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 provided meteorological weather data.

Given the ground truth $\tilde{\mathbf{a}}$, we define different error time series for a given ensemble member m. As error time series, we consider error (E), absolute error (AE), normalized error (NE) and the normalized absolute error (NAE). For brevity, we drop the dependence of $\mathbf{a}$ in the error time series notation.

$$\boldsymbol{\varepsilon}_{\mathrm{E}}^m = \mathbf{a}^m - \tilde{\mathbf{a}} \tag{1}$$

$$\boldsymbol{\varepsilon}_{\mathrm{AE}}^m = |\mathbf{a}^m - \tilde{\mathbf{a}}| \tag{2}$$

$$\boldsymbol{\varepsilon}_{\mathrm{NE}}^m = \frac{\mathbf{a}^m - \tilde{\mathbf{a}}}{\sigma(\tilde{\mathbf{a}})} \tag{3}$$

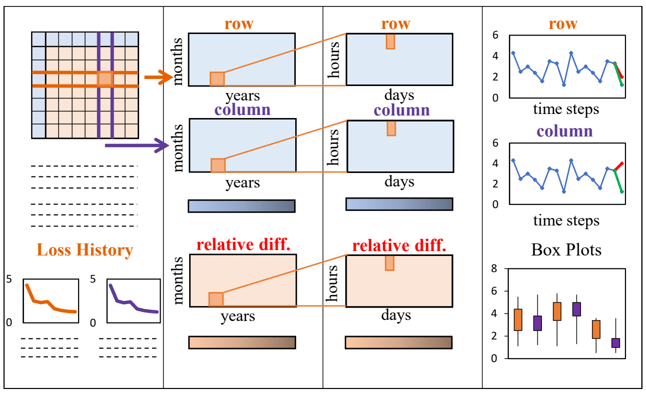
$$\boldsymbol{\varepsilon}_{\mathrm{NAE}}^m = \frac{|\mathbf{a}^m - \tilde{\mathbf{a}}|}{\sigma(\tilde{\mathbf{a}})} \tag{4}$$

Eq. (1) computes the signed difference, whereas Eq. (2) computes the absolute error. Eqs. (3) and (4) contain a normalization to account for the difficulty of predicting values with a higher standard deviation. To compare the mean errors of two ensemble members m_1 and m_2, we use their respective mean errors $\overline{\boldsymbol{\varepsilon}_*^{m_1}}$ and $\overline{\boldsymbol{\varepsilon}_*^{m_2}}$ as well as their relative difference $(\overline{\boldsymbol{\varepsilon}_*^{m_1}} - \overline{\boldsymbol{\varepsilon}_*^{m_2}})$, where $* \in \{\mathrm{E}, \mathrm{AE}, \mathrm{NE}, \mathrm{NAE}\}$. While $\overline{\boldsymbol{\varepsilon}_{\mathrm{AE}}^{m_1}}$ and $\overline{\boldsymbol{\varepsilon}_{\mathrm{NAE}}^{m_1}}$ are used to directly compare the performance of the networks, the $\overline{\boldsymbol{\varepsilon}_{\mathrm{E}}^{m_1}}$ and $\overline{\boldsymbol{\varepsilon}_{\mathrm{NE}}^{m_1}}$ show whether models are biased to overestimate or underestimate. The normalized versions of the errors are better suited to compare results from different training locations, since areas with low standard deviation are easier to predict than those with high 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 visualization 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 errors $\boldsymbol{\varepsilon}_*^m$ of every model m according to Eqs. (1)–(4), which are later read for the selected meteorological attribute. The following visualization tasks are supported by our tool:

- a high-level comparison of the overall network performance, e.g., for the comparison of different hyperparameters,
- a view onto statistics including the loss history and box plots,
- a coarse-to-fine analysis from years to months, days and hours, with comparisons to the ground truth and among models,
- a comparison of the results obtained with the training and test data to locate overfitting and underfitting,



a) *Selection Matrix* b) *Full Overview* c) *Month Overview* d) *Time Step View*

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. The* Loss History *is shown for both the row and the column model (a) and mean and standard deviation are aggregated in* Box Plots *(d).*

- an inspection of different error metrics and attributes.

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 both selected models. As soon as the models are selected, the difference $\boldsymbol{\varepsilon}_*^m$ 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 $\boldsymbol{\varepsilon}_*^m$ for the two selected models.

In the following, 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 errors $\overline{\boldsymbol{\varepsilon}_*^m}$ (according to the respective error metric $*$) of each model m (averaged over the entire time series), we define the *Selection Matrix* $\mathbf{S}$ as:

$$\mathbf{S}_{ij} = (\overline{\boldsymbol{\varepsilon}_*^{m_i}} - \overline{\boldsymbol{\varepsilon}_*^{m_j}}) \tag{5}$$

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

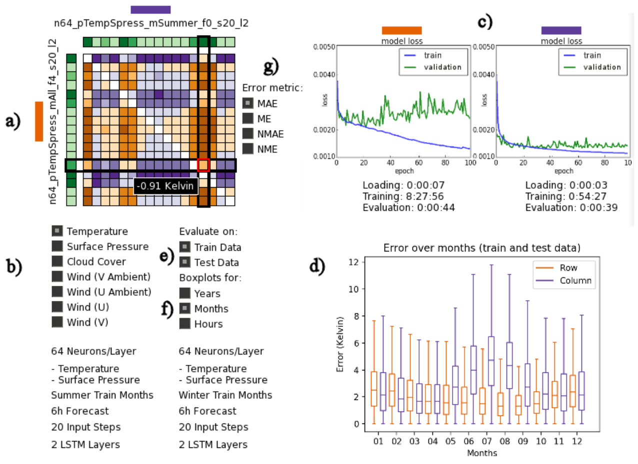
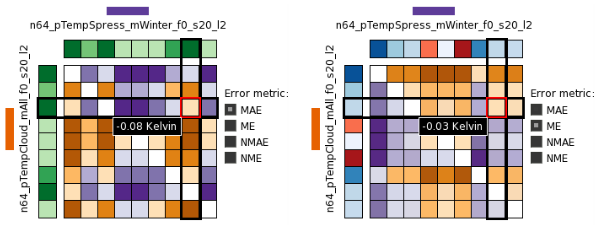


Figure 4: *The entry point of the exploration is the* Selection Matrix *(a). The variables to display, the share of the data to evaluate and the choice of the box plot are adjusted below (b). The* Loss History *(c) reveals overfitting and underfitting and the* Box Plots *(d) display the median and the deviation of the error of both selected models.*


n64_pTempSpress_mWinter_f0_s20_l2
n64_pTempCloud_mAll_f0_s20_l2
-0.08 Kelvin
Error metric:
MAE
ME
NMAE
NME
n64_pTempSpress_mWinter_f0_s20_l2
n64_pTempCloud_mAll_f0_s20_l2
-0.03 Kelvin
Error metric:
MAE
ME
NMAE
NME

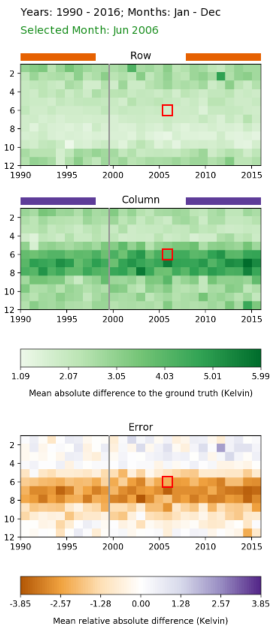
(a) *Mean Absolute Error (MAE)* **(b)** *Mean Error (ME)*

Figure 5: *Different colormaps are used for the* Selection Matrix *depending on the chosen error metric.*

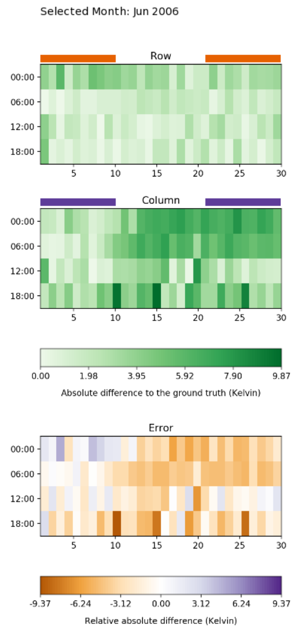
models are initially sorted alphanumerically by their hyperparameters; other choices are imaginable. We visualize this matrix in Fig. 4a. To provide all available error metrics, the radio buttons to the right of the interactive visualization (cf. Fig. 4g) can be used to switch to the desired error metric. While the entries inside the matrix visualize the relative difference between two model scores $(\overline{\boldsymbol{\varepsilon}_*^{m_i}} - \overline{\boldsymbol{\varepsilon}_*^{m_j}})$, the left and top margins of the matrix represent the column and row scores $\overline{\boldsymbol{\varepsilon}_*^{m_i}}$ and $\overline{\boldsymbol{\varepsilon}_*^{m_j}}$, respectively. The specific colormaps used in the visualizations depend on the selected error metric (cf. Fig 5) and are all based on colormaps by ColorBrewer2 [Cyn17].

If an entry is negative, model k_i (row) is better and if it is positive, model k_j (column) is more accurate. We visualize this matrix in Fig. 4 (top left) using a diverging color map. 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.

AE and NAE use a diverging orange-purple colormap for the relative difference and a sequential green one for the column and row scores. A brighter shade of green indicates a small error com-



(a) *Full Overview*



(b) *Month Overview*

Figure 6: *The selected month in the* Full Overview *(left) is viewed in detail in the* Month Overview *(right).*

pared to the ground truth, whereas a darker shade means the opposite.

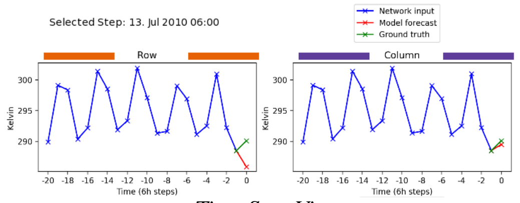
E and NE use a diverging orange-purple for the relative difference and a diverging red-blue one for the column and row scores. A red entry means that the mean error is positive, a blue color shows a negative mean error. The brighter the colors in the margin texels, the closer are the mean errors to zero.

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 using 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.

For the currently selected row and column model, 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. 6.



Time Step View

Figure 7: *Side-by-side comparison of predictions for both selected models, including all past data points received by the networks.*

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 $(\overline{\boldsymbol{\varepsilon}^{k_1} - \boldsymbol{\varepsilon}^{k_2}})$. 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, cf. Fig. 6a.

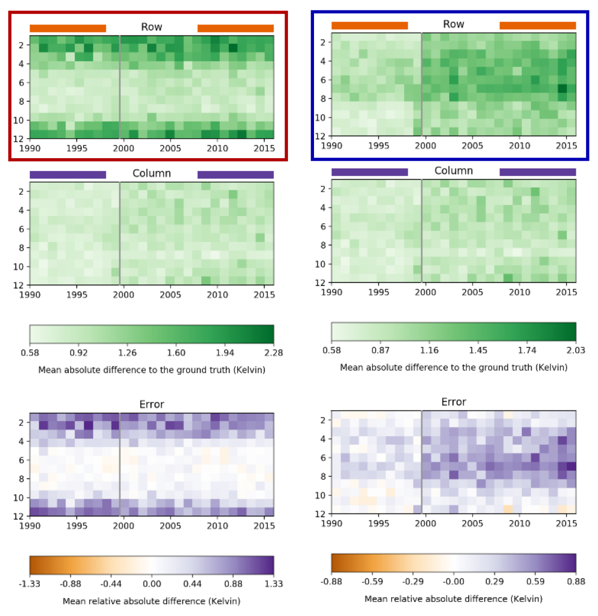
Month Overview. If the user selects a month in the previous *Full Overview*, a more detailed *Month Overview* of that month appears next to it to the right, see Fig. 6b. This view uses the same layout and colormaps as the first overview, but differs in the data it visualizes. Here, the columns show the days of the selected month and the rows the hours. The temporal resolution of our data provides a value every six hours, resulting in four values per day. This view 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. 7. 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, 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 Zurich (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 a meteorologist. In the paper, we visualize prediction results for temperature only. We refer to the video for visualizations of other parameters and to the additional material for detailed listings of forecasting errors for all tested networks.



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

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

Figure 8: *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. 8a 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. 8b, the opposite case 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. 9 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. 10, 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. 11, 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 an overfitting problem: 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. When it comes to feature extraction from unsteady flows, the choice of the right reference frame is important [GT18]. To test, whether a neural network would learn to extract the reference frame internally on its own, we precompute the optimal reference frame and compare the network output with the output obtained from the original vector field. Following Günther et al. [GGT17], we locally decompose the air flow $\mathbf{v}$ into two components $\bar{\mathbf{v}}$ and $\tilde{\mathbf{v}}$:

$$\mathbf{v} = \bar{\mathbf{v}} + \tilde{\mathbf{v}}, \quad \text{s.t.} \quad \int_U \left\| \frac{\partial \bar{\mathbf{v}}}{\partial t} \right\|^2 \mathrm{d}V \rightarrow \min \tag{6}$$

to separate vortices from the ambient transport. Thereby, $\bar{\mathbf{v}}$ is the same vector field as $\mathbf{v}$ at a transient moment in a rotating and/or translating reference frame in which the flow is near-steady (the temporal derivative is minimized at each point in a local neighborhood U) and $\tilde{\mathbf{v}}$ accordingly 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. 12 indicates that using the ambient flow $\tilde{\mathbf{v}}$ improves predictions for small numbers of neurons, as visible in the *Box Plots*, since networks must not learn the feature extraction themselves. If not enough neurons are available for learning, such a preprocessing of the input prevents

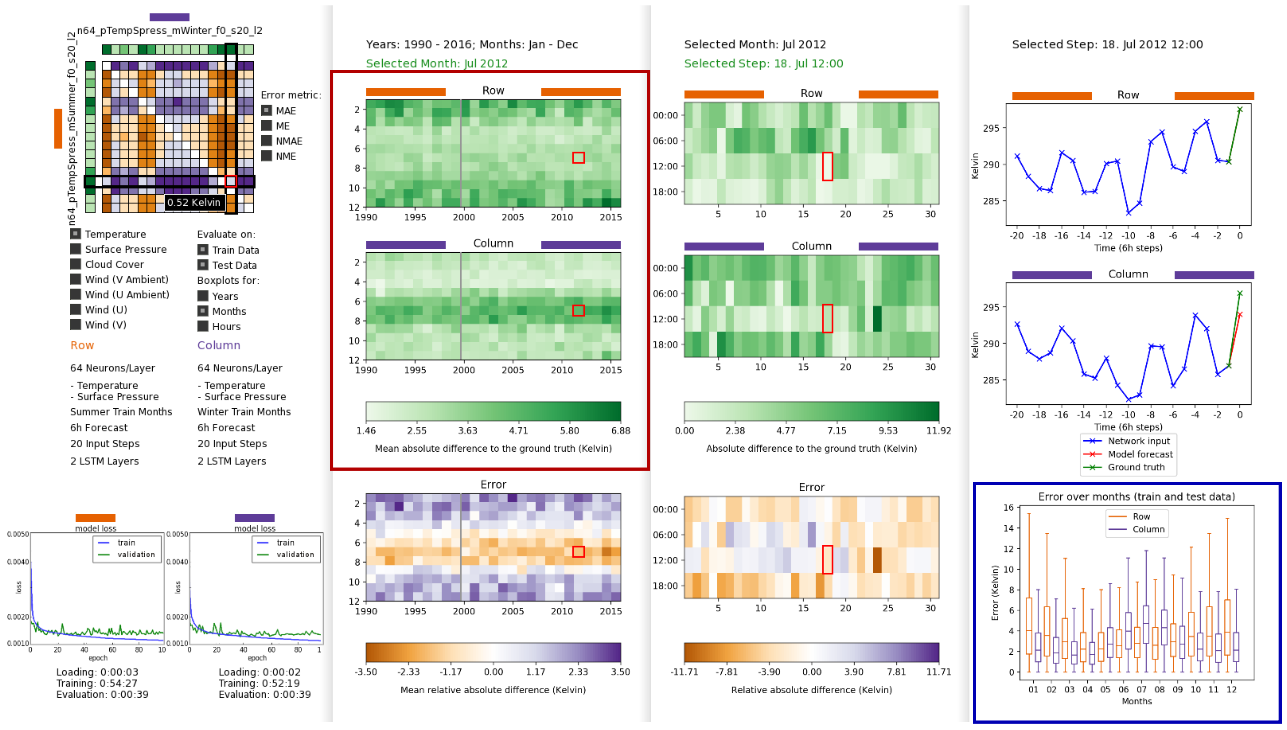


Figure 9: *Comparison of a network that has trained only on summer months (row) with one that only trained on winter months (column). The specialization is clearly visible in the* Full Overview *(□).* Box Plots *that summarize over months give further insight into the behavior of both networks. They show a higher mean absolute error and standard deviation in months unknown to the network (□).*

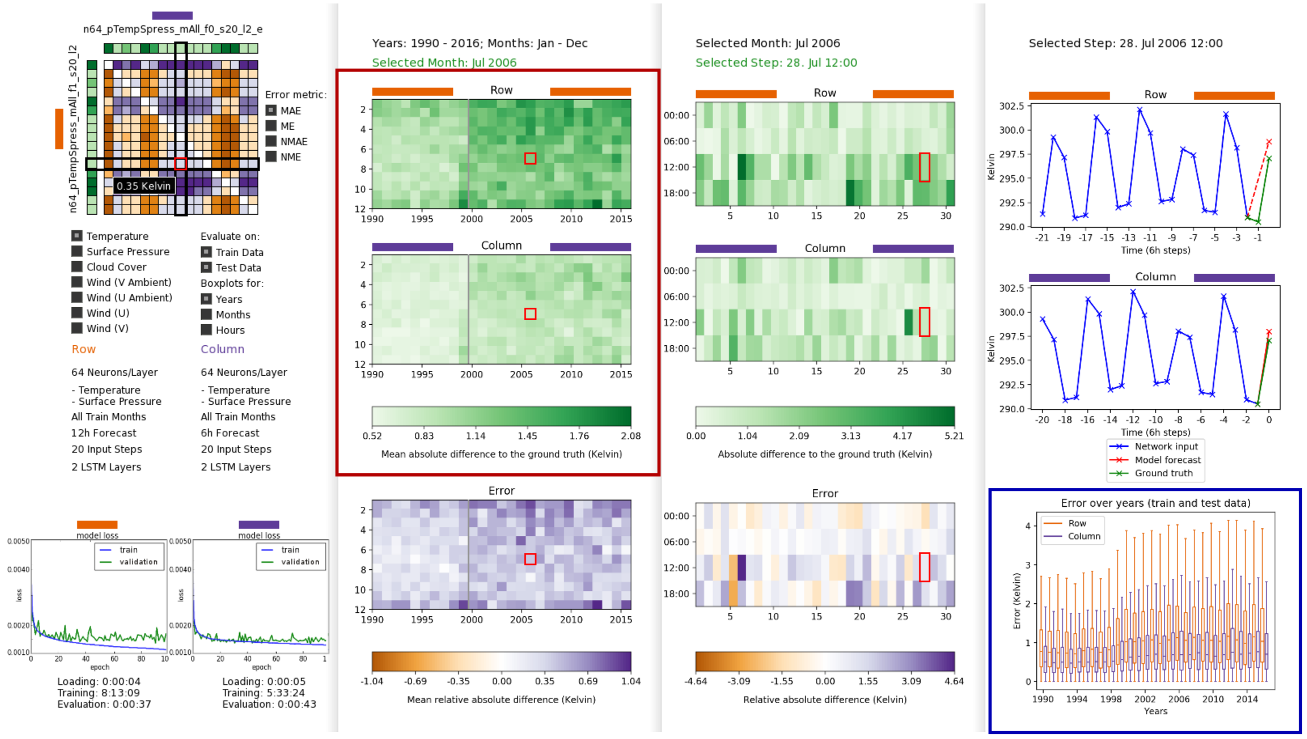

n64_pTempSpress_mAll_f0_s20_l2_e
n64_pTempSpress_mAll_f1_s20_l2
0.35 Kelvin
Error metric: MAE ME NMAE NME
Row
Column
64 Neurons/Layer
All Train Months
12h Forecast
6h Forecast
20 Input Steps
2 LSTM Layers
Years: 1990 - 2016; Months: Jan - Dec
Selected Month: Jul 2006
Selected Month: Jul 2006
Selected Step: 28. Jul 12:00
Selected Step: 28. Jul 2006 12:00
Mean absolute difference to the ground truth (Kelvin)
Absolute difference to the ground truth (Kelvin)
Time (6h steps)
Network input
Model forecast
Ground truth
Loading: 0:00:04
Training: 8:13:09
Evaluation: 0:00:37
Loading: 0:00:05
Training: 5:33:24
Evaluation: 0:00:43
Mean relative absolute difference (Kelvin)
Relative absolute difference (Kelvin)
Error over years (train and test data)
Error (Kelvin)
Years

Figure 10: *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.*

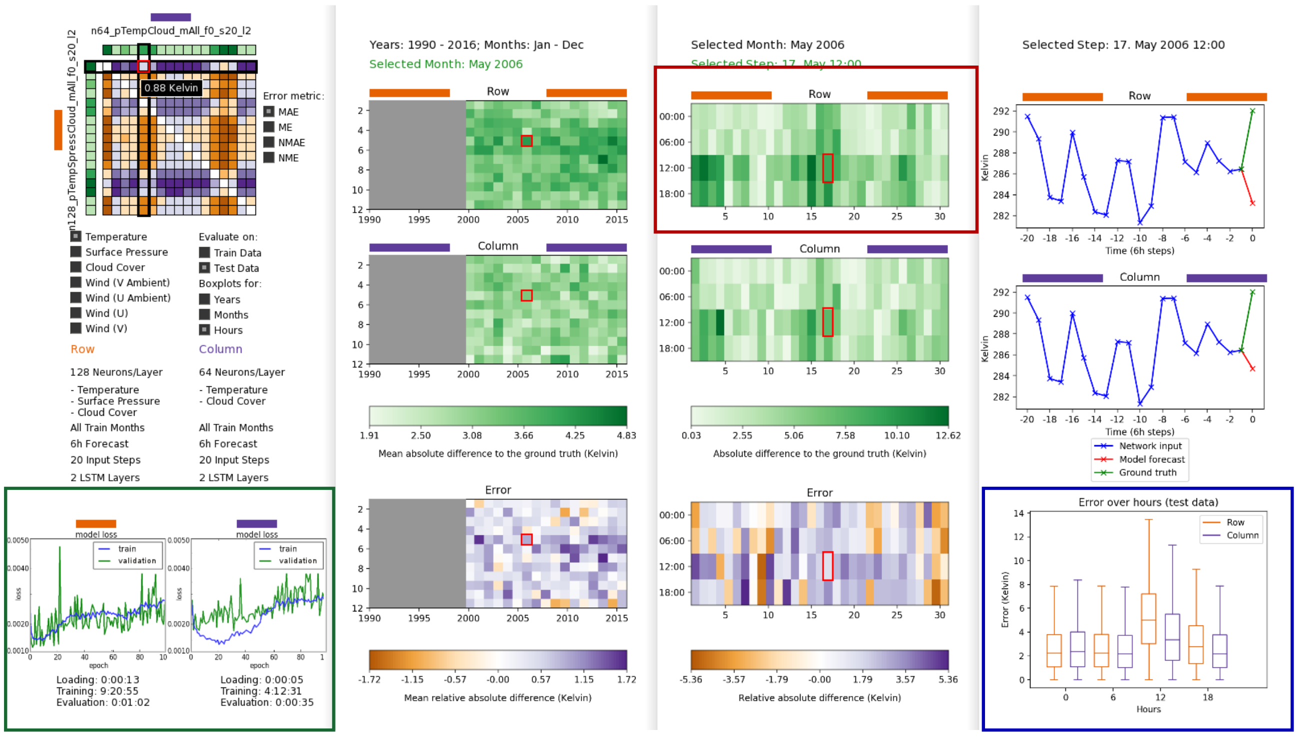


Figure 11: *The* Month Overview *(□) as well as the* Box Plots *(□) show a clear error spike around the 12:00 step for the row model. The* Loss History *(□) reveals that this particular model could not generalize since neither the training nor the validation loss decreased over the epochs. This is an indicator of overfitting. Here, the user chose to evaluate the performance on the test data only.*

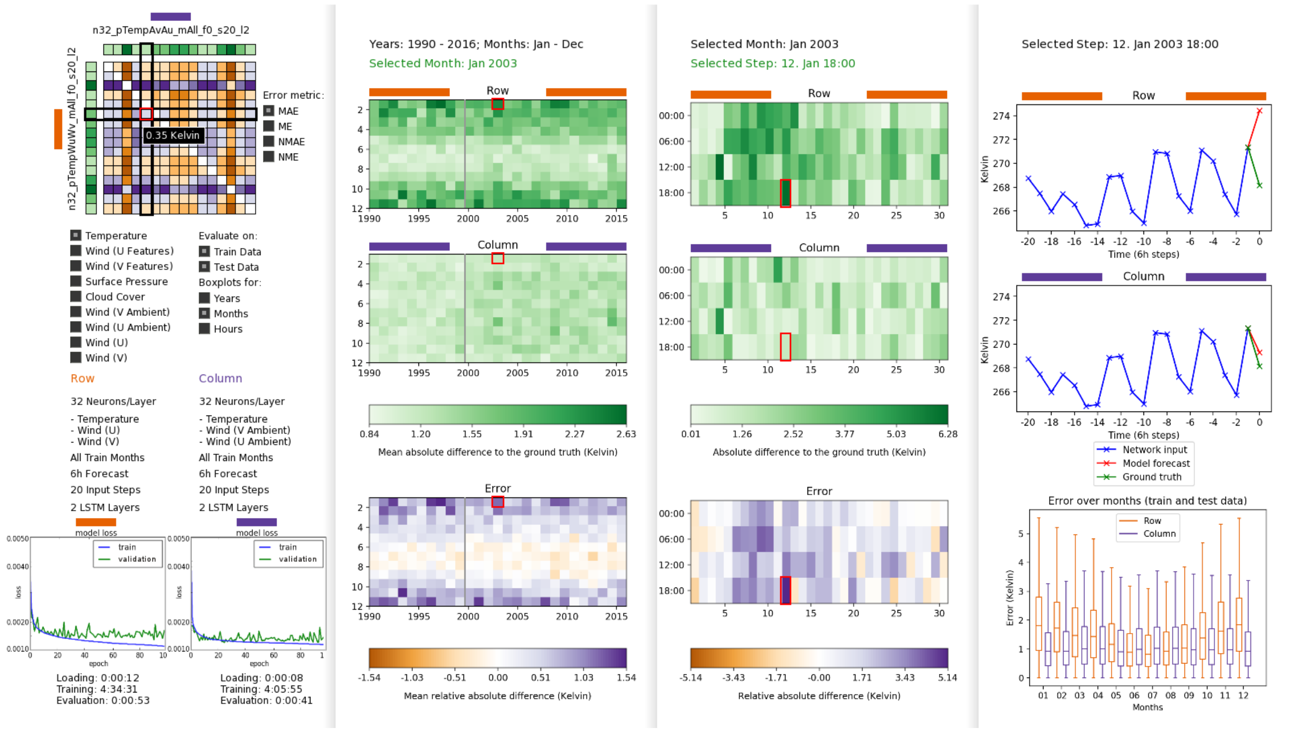


Figure 12: *Extracting the ambient transport using [GGT17] from the flow improved the predictions, as visible in the* Full Overview *and the* Selection Matrix. *The currently selected row model uses the raw wind speed components, while the column model uses the ambient transport in U and V direction. Both models also received the temperature attribute as input.*

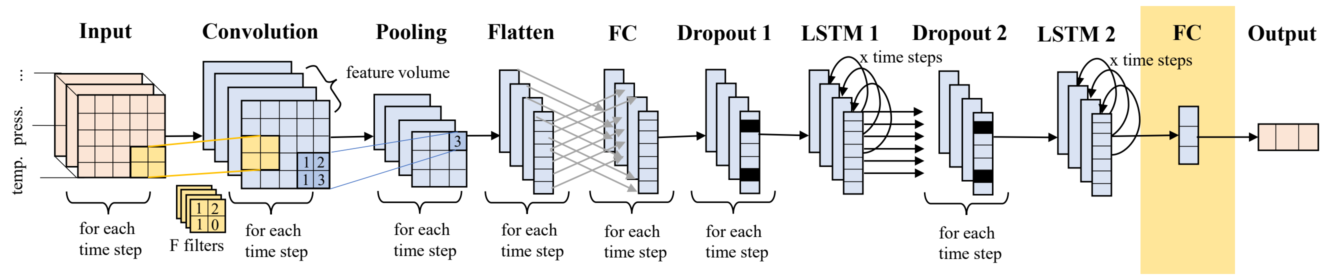


Figure 13: *Adding a fully connected layer (highlighted in yellow) at the end of the network improves the predictions as the network uses the last layer for balancing over- and underestimations.*

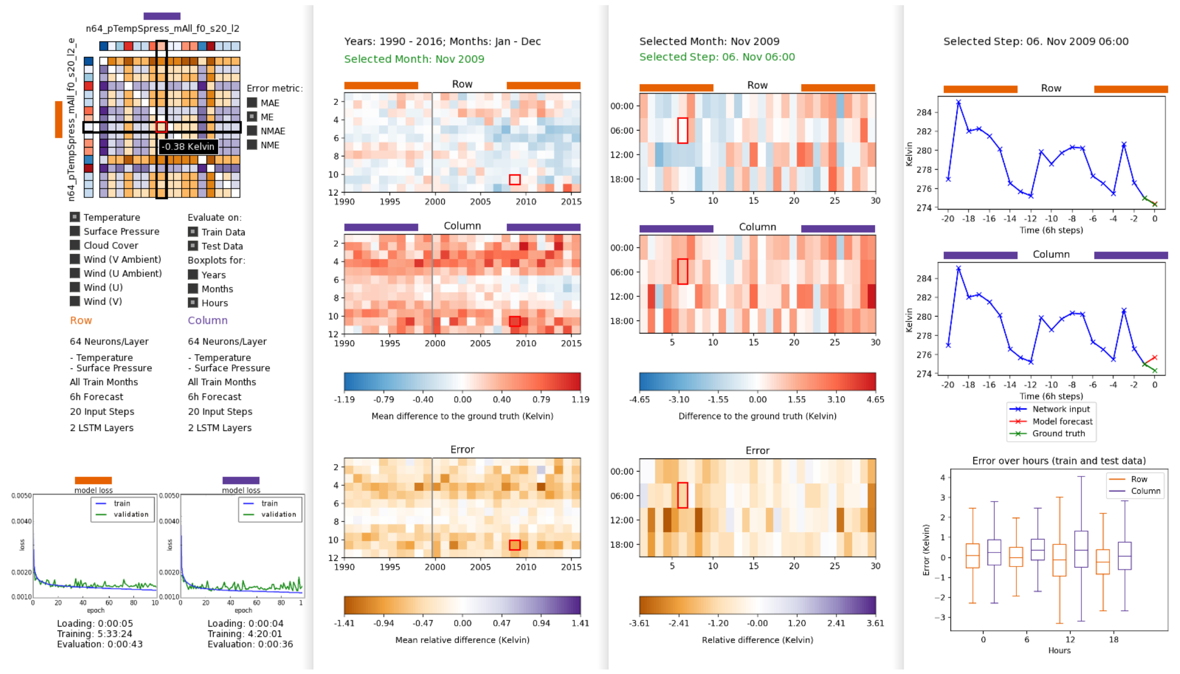


Figure 14: *The* column *model was the best prediction network for Zurich for the network architecture described in this work. When looking at the* Full Overview*, it is apparent that this model generally overpredicts temperature. Adding a last layer as shown in the* row *model helps to balance the network and generates reduced MEs as well as MAEs.*

underfitting. Note that unlike $\mathbf{v}$, the decomposition of Günther et al. [GGT17] in Eq. (6) is *objective* [TN65], i.e., $\tilde{\mathbf{v}}$ is independent of smooth rotations and translations of the input reference frame.

Post-Processing While showing the MAE helps to visualize the general quality of the predictions, the ME setting also proved to be useful. The best network model for Zurich using the network architecture described in Section 4 has the lowest MAE on the testing data but still yields a ME of 0.31. This means that the model tends to predict too high temperature values in general.

Adding a last dense layer makes the model balance itself. The network structure including the last fully connected layer is depicted in Fig. 13. Comparing the best Zurich model with the current network architecture with one that uses an additional last dense layer reduces the ME from $0.31K$ to $-0.07K$ (by 77.4%) and the MAE from $0.95K$ to 0.88 (by 7.4%), which can be seen in Table 1. The visualizations in Fig. 14 show the comparison of the best model so far and the model using an additional layer. The colors in the *Full Overview* are evidently more balanced now.

Forecast Model	MAE	ME	SDAE	SDE
With last layer	$0.88K$	$-0.07K$	$0.76K$	$1.17K$
Without last layer	$0.95K$	$0.31K$	$0.83K$	$1.22K$

Table 1: *Network performance with and without the last dense layer: mean absolute error (MAE), mean error (ME), standard deviation of absolute error (SDAE) and standard deviation of error (SDE).*

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 [ABC*16] back end with GPU support. The

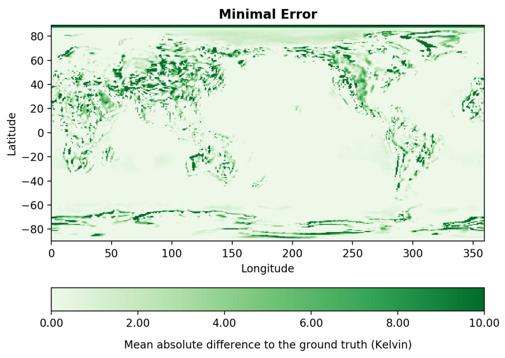


Figure 15: *A network model that has been trained on a single location (Zurich) is evaluated on all available grid locations on Earth. The evaluation error is averaged over temperature predictions for the year 2016 and truncated at a maximum error of 10 Kelvin.*

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 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.

7. Combination of Forecasting Models

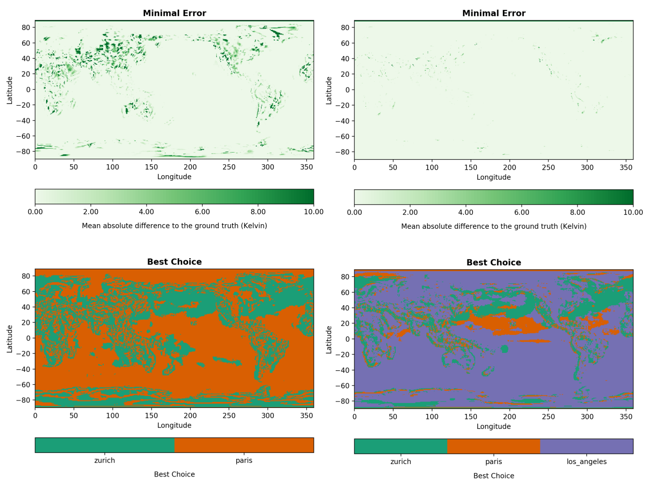
The models generated with the network described in Section 4 are optimized for a particular location on Earth where they have been trained to forecast weather attributes. However, a model that was trained for one location can be evaluated at different locations, as well. As the evaluation of a model is much less time-consuming than its training, training on a few selected locations and evaluating on the whole grid will save time. From a visualization point of view, the step from a single point to the whole grid introduces spatial dimensions, for which we add additional views, as described next.

7.1. Full Evaluation of a Single Model

Evaluating a network model trained on one specific location on all available grid points on Earth results in a *Prediction Quality Map* as seen in Fig. 15 for a network trained for Zurich. There exists a set of grid points where the model performs well and which are therefore similar to Zurich from the network's perspective.

7.2. Full Evaluation of Combined Models

Further developing this idea leads to a combination of multiple networks that were trained for different locations. If we look at the *Prediction Quality Map* generated by the Zurich model in Fig. 15, a second network trained on a grid point with a high MAE can be used to predict weather attributes of locations where the Zurich model did not succeed.



(a) Zurich and Paris **(b)** Zurich, Paris and Los Angeles

Figure 16: *Combination of Zurich, Paris and Los Angeles models. The minimal error decreases every time a new model is added and reaches a good score when using three different network models.*

An example of a suitable location for a second network model is the grid point near Paris. Choosing the better prediction of the Zurich and Paris networks for each grid point decreases the overall MAE of the predictions when evaluating on every grid point on Earth, which can be seen in Fig. 16a.

The minimal error plot depicted in Fig. 16a does still have spots where both models fail to predict the temperature attribute correctly. We chose the nearest grid point to Los Angeles to train the third model. Combining the Zurich, Paris and Los Angeles models further decreases the overall MAE as shown in Fig. 16b. The Los Angeles model seems to be a good choice for ocean regions. Note that for this approach two testing rounds with different data sets have to be used: One to determine the best model for each location and one to evaluate the choice on separate data. Both data sets are distinct parts of the testing data which have not been used for training before. In our example, determining the best model for each grid point can be done by evaluating the models on data from 2000 and then use the resulting combined model to predict weather data from 2001–2016.

This approach was compared with a single model that was trained on the data of all three locations. The result of this comparison favors the combination of individual models, which can be inspected in the additional materials.

8. Conclusions

In this paper, we described a comprehensive and interactive system to visually analyze and compare time series predictions. The input to our system were time series generated by convolutional and recurrent neural networks that we trained for meteorological weather forecasting. 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 provide a deeper insight into the prediction performance. Our method enables to spot phenomena like underfitting, overfitting, and outliers easily. Using not only the MAE but also the ME as an error metric helped to identify unbalanced models and to improve the network design. Finally, we added the spatial dimension and visualized the prediction errors globally for the entire Earth.

In the future, we would like to use neural networks to predict the residual error of operational weather forecasting systems. Further, analyzing which parts of the input data led to a prediction would also be interesting for a comparison even if networks were trained to forecast other attributes. While we concentrated on meteorological data, an application of the visualization model to other ensembles of time series is imaginable as well. An example would be stock market data, which can also be generated using ANNs [KB96].

References

[ABC*16] Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., Devin M., Ghemawat S., Irving G., Isard M., Kudlur M., Levenberg J., Monga R., Moore S., Murray D. G., Steiner B., Tucker P., Vasudevan V., Warden P., Wicke M., Yu Y., Zheng X.: Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2016), OSDI'16, USENIX Association, pp. 265–283. 9

[AMM*08] Aigner W., Miksch S., Müller W., Schumann H., Tominski C.: Visual methods for analyzing time-oriented data. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (Jan 2008), 47–60. 2

[BB12] Bergstra J., Bengio Y.: Random search for hyper-parameter optimization. *Journal of Machine Learning Research 13*, Feb (2012), 281–305. 3

[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. 2

[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. 9

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

[DAW13] Dang T. N., Anand A., Wilkinson L.: Timeseer: Scagnostics for high-dimensional time series. *IEEE Transactions on Visualization and Computer Graphics 19*, 3 (March 2013), 470–483. 2

[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. 5

[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. 6, 8, 9

[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. 2

[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

[GT18] Günther T., Theisel H.: The state of the art in vortex extraction. *Computer Graphics Forum* (2018), to appear. 6

[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, 2

[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, 11

[KBK11] Krstajic M., Bertini E., Keim D.: Cloudlines: Compact display of event episodes in multiple time-series. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec 2011), 2432–2439. 2

[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. 2

[MMKN08] McLachlan P., Munzner T., Koutsofios E., North S.: Liverac: Interactive visual exploration of system management time-series data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2008), CHI '08, ACM, pp. 1483–1492. 2

[SCL*12] Shi C., Cui W., Liu S., Xu P., Chen W., Qu H.: Rankexplorer: Visualization of ranking changes in large time series data. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (Dec 2012), 2669–2678. 2

[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). 2

[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. 2, 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. 9

[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

[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). 2