https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019

Nonlinear Processes in Geophysics
Discussions
Open Access
EGU

Generalization properties of neural networks trained on Lorenz systems

Sebastian Scher[1] and Gabriele Messori[1,2]

[1]Department of Meteorology and Bolin Centre for Climate Research, Stockholm University, Stockholm, Sweden
[2]Department of Earth Sciences, Uppsala University, Uppsala, Sweden

Correspondence: Sebastian Scher sebastian.scher@misu.su.se

Abstract. Neural networks are able to approximate chaotic dynamical systems when provided with training data that covers all relevant regions of the system's phase space. However, many practical applications diverge from this idealised scenario. Here, we investigate the ability of neural networks to: 1) learn the behaviour of dynamical systems from incomplete training data, and 2) learn the influence of an external forcing on the dynamics. Our analysis is performed on the Lorenz63 and Lorenz95 models. We show that neural networks trained on data covering only part of the system's phase space struggle to make skillful short-term forecasts in the regions missed during the training. Additionally, when making long series of consecutive forecasts, the networks mostly do not reproduce trajectories exploring regions beyond those seen in the training data. We also find that it is challenging for the standard network architectures to learn the influence of a slowly changing external forcing, highlighting the limitations of a network trained on a specific forcing regime for generalising a system's behaviour. These results outline challenges for a variety of machine-learning applications. An example is climate science, which is concerned with a non-stationary chaotic system whose behaviour is known only through comparatively short data series.

1 Introduction

Neural networks are a series of interconnected – potentially nonlinear – functions, whose mutual relations are "learned" by the network by training on data. One of their many applications is predicting the time-evolution of dynamical systems. In this context, the neural networks are trained on long timeseries issued from the dynamical system of interest, and can then in principle be used to predict the system's evolution from new initial conditions. Examples of applications include classical physical systems like the double pendulum (Bakker et al., 2000), and the widely studies Lorenz toy-models of the atmosphere (e.g. Vlachas et al. (2018); Dueben and Bauer (2018)).

In recent years, neural networks have enjoyed growing attention in climate science. Applications include parameterization schemes in numerical weather prediction and climate models (Krasnopolsky and Fox-Rabinovitz, 2006; Krasnopolsky et al., 2013; Rasp et al., 2018), postprocessing of numerical weather forecasts (Rasp and Lerch, 2018), predicting weather forecast

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019

Nonlinear Processes in Geophysics
Discussions
Open Access
EGU

uncertainty (Scher and Messori, 2018) and doing actual weather forecasts and climate model emulations in simplified realities (Dueben and Bauer, 2018; Scher, 2018; Scher and Messori, 2019). These increasingly widespread practical applications warrant a more systematic evaluation of the possibilities and limitations of neural networks for the simulation of complex dynamical systems.

In in this paper, we address two open questions related to using neural networks for approximating the dynamics of chaotic systems:

1) Can neural networks infer system behaviour in regions of the phase space not included in the training dataset?

2) Can neural networks "learn" the influence of external forcing driving slow changes in the system they are trained on?

We adopt an empirical approach to answer these questions. Namely, we generate long time-series with numerical models, and then perform experiments with neural networks on this data. The first question will be investigated using the Lorenz63 system (Lorenz, 1963); the second using both the Lorenz63 and Lorenz95 systems (Lorenz, 1996) and simulating external forcing as a slow change in the parameters of the models. Both are of direct relevance to climate applications. Our knowledge of the high-frequency evolution of the climate system issues from comparatively short timeseries, which only explore a small subset of the possible states of the system. Finally, the accelerating anthropogenic forcing will likely lead to significant changes in the climate's future evolution. The two points we raise are therefore crucial in the context of using neural networks for weather forecasting and for emulating climate models. They could be reformulated in more practical terms as: do neural networks have the potential to reproduce unprecedented states of the climate system? Similarly, could they learn the influence of unprecedented greenhouse-gas concentrations on the dynamics of the climate system, given a past record of the system subjected to varying greenhouse-gas levels?

2 Emerging Challenges in Neural Networks for Dynamical Systems

Question 1) we framed above, relates to whether the network learns a "global" function mapping the state vector $\boldsymbol{x}$ from one timestep to the next:

$$f(\boldsymbol{x}) : \boldsymbol{x}_t \mapsto \boldsymbol{x}_{t+1} \tag{1}$$

or whether it learns N individual functions for N different regions of the phase space:

$$f(\boldsymbol{x}) = \begin{cases} f_1(\boldsymbol{x}) & \boldsymbol{x} \in region_1 \\ f_2(\boldsymbol{x}) & \boldsymbol{x} \in region_2 \\ \vdots & \vdots \\ f_N(\boldsymbol{x}) & \boldsymbol{x} \in region_N \end{cases} \tag{2}$$

For some applications this may be irrelevant, as long as the network forecasts work. However, it has major implications for how the network generalizes to regions of the phase space that are not covered in the training data.

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019

Nonlinear Processes in Geophysics
Discussions
Open Access
EGU

Neural networks can tend to overfit – meaning they work very well on the training data, but do not generalize and therefore do not work on new data. Therefore, they are usually tested on data not used for the training. Given a dataset, it is not trivial to decide how to split the data into a training and test set. For data without auto-correlation, a random split on a sample-by-sample basis may be suitable. For autocorrelated time-series, it is common to split the data into continuous blocks (e.g. using the first 80% of a timeseries for training, and the last 20% for evaluation). In a real-world application to the atmosphere, one could train the network on the first years of available observations, and then test on the remaining available years (e.g. Rasp and Lerch (2018); Scher and Messori (2018)). For the Lorenz63 model, the train-test splits are typically designed such that samples in the test set are not contained in the training set, but at the same time ensure that both the training and the test set cover all regions of the phase space with some reasonable density. That is, no large contiguous regions of the phase space are left out of either set of data.

Here, we consider the opposite situation, namely a scenario where the training data covers only part of the system's phase space. We know from the definition of the Lorenz63 model that the underlying equations are invariant across the phase-space. If the network can truly learn the system's dynamics, and thus successfully approximates the underlying equations, then it should be able to provide useful information concerning the system's behaviour in those regions of the phase space not included in the training data. More generally, for a long series of successive forecasts the network should thus be able to reconstruct the full attractor. However, should the network instead learn a set of functions each applicable locally, then one would expect the network to fail in regions not explored during the training.

3 Methods

3.1 The Lorenz63 and Lorenz95 models

The Lorenz63 model is a 3-variable system defined by the following ordinary differential equations:

$$\begin{aligned} \dot{x} &= \sigma\,(y - x) \\ \dot{y} &= x\,(\rho - z) - y \\ \dot{z} &= xy - \beta z \end{aligned} \tag{3}$$

We use $\sigma = 10$, $\beta = 8/3$ and $\rho = 28$, the standard parameter combination with which the system – despite its simplicity – generates chaotic behavior (the characteristic "butterfly" shape). We integrate the system with a timestep of $t = 0.01$ with the LSODA solver from ODEPACK. While the Lorenz63 model is a very rough approximation of atmosphere-like dynamics, the fact that it has only 3 variables allows to easily visualize the complete phase space and define regions that can be excluded from the training data. This makes it ideally suited to tackle the first question we pose (generalization to unseen phase-space regions).

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019

Nonlinear Processes in Geophysics
Discussions
Open Access
EGU

The Lorenz95 model is a 1-d model that approximates the atmosphere as a series of N gridpoints wrapped around a circular domain:

$$\dot{x_i} = (x_{i+1} - x_{i-2})\, x_{i-1} - x_i + F \tag{4}$$

with $i = 1...N$ and $(x_{N+1} = x_1)$. Here we choose $N = 40$. F is a forcing term. With $F = 4$ the system shows periodic behaviour; with increasing F the behaviour becomes increasingly chaotic, and with $F = 16$ it is highly turbulent. Like the Lorenz63 model, we integrate the system with the LSODA solver from ODEPACK.

3.2 Neural Network for Lorenz63

We use a shallow network with 1 fully-connected hidden layer, 8 neurons with a sigmoid activation function, and 3 linear output neurons. This has previously been identified as a suitable architecture for the Lorenz63 system by Zhang (2017). The network takes as input all 3 Lorenz63 variables, and outputs all 3 variables one timestep later. The training is done with the adam optimizer (Kingma and Ba, 2015). Due to the small size of the hidden layer, controlling overfitting via early stopping is not necessary. For the forcing experiments, we additionally use a second architecture, where the network has 4 input parameters (the 3 Lorenz63 variables and the parameter σ, see Eq. 3), and the same 3 output variables as the standard setup.

3.3 Neural Network for Lorenz95

For the Lorenz95 model, we use a convolutional network that works on the periodic domain. Convolutional networks have already successfully been used on gridded data from simplified general circulation models in Scher (2018) and Scher and Messori (2019). The configuration used here was tuned with an exhaustive gridsearch over different network configurations. The tuning procedure is described in Appendix A. We tuned the network for forecasting 1, 10 and 100 timesteps, where each timestep corresponds to 0.01 time units of the Lorenz95 model. The network trained for 10 timestep-forecasts (a 2-layer convolution network with a kernel-size of 5, see Appendix A) worked best for virtually all lead-times (see fig. A1), and we use this architecture in our analysis. For the forcing experiments, the parameter F at each timestep was expanded to the number of gridpoints of the Lorenz95 model and added as an additional input channel to the network.

3.4 Evaluating the reconstruction of the Lorenz63 attractor

The neural networks is trained by minimizing errors of single-step (and thus short-term) forecasts. Therefore, it might be that the network cannot reproduce a stable system when making a long series of consecutive forecasts, which is a known issue when applying neural networks to chaotic systems (e.g. Bakker et al. (2000)). In our experiments, the trained network often made very good short-term predictions, but when attempting to produce long series of iterative forecasts (which, in the context of climate science, would be analogous to producing a "climate run" from successive meteorological forecasts), the system collapsed into a fixed point. Since the training of our network is computationally inexpensive, we use a brute force method to find a network that yields both skillful short-term forecasts and a realistic long-term system evolution. We simply repeat

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019

Nonlinear Processes in Geophysics
Discussions
Open Access
EGU

the training until we find a network that has the desired characteristics. While beyond the scope of this study, it would be of interest to investigate which specific features of the training process may affect the short-term versus long-term characteristics of the network output. In order to evaluate the different networks, and provide a quantitative definition of what we mean by "realistic", we compare the density of the trajectories in phase space between the reconstructed system and the training data from the original Lorenz attractor. If the difference is below a pre-defined threshold, we accept the network as valid, and use it in our analysis (a standard approach, eg. Bakker et al. 2000). Our evaluation metric is:

$$rmse\left(\rho\right) = \sqrt{\overline{\left(\rho_{i,j,k,model} - \rho_{i,j,k,network}\right)^2}} \tag{5}$$

where $\rho_{i,j,k}$ is the density of discrete data points in the gridbox i, j, k. As threshold for $rmse\left(\rho\right)$ we chose 0.04 (on normalized data). This somewhat arbitrary value was selected based on a visual inspection of the reconstructed attractors. We will hereafter term this the "density-selection" approach.

This approach is somewhat problematic when training the network on specific regions of the phase-space. In principle, we could apply exactly the same procedure to compare the densities of the reconstructed attractor and of the training data. However, for incomplete training data – for example, only one wing of the butterfly – then a perfect reconstruction of the full attractor would fail this test, since the training data includes no information beyond the one wing. If the neural network learned a "wrong" attractor, namely one that only covers regions close to the wing included in the training, this network would pass the test and be selected, even though it clearly has undesirable characteristics. An alternative approach is to compare the reconstructed attractor with the full attractor. This solves the aforementioned problems, yet is flawed in terms of information availability at time of training. In a real world setting, we would not know what the full attractor of a complex system – for example our atmosphere – looks like. Nonetheless, in our idealised setting this approach allows to verify whether the network learns regional or global dynamics. We will hereafter term it the "density-full approach".

A third approach would be to set some a priori objective selection criteria for the desirable characteristics of the reconstructed attractor. We use the following: 1) the reconstructed attractor does not collapse into a fixed stable point. To test this, we verify that no two consecutive points in time are collocated down to machine-precision. 2) The reconstructed attractor does not have any two points in the reconstructed attractor that are equal down to machine-precision. While these two approaches leak less information from the underlying (and presumably unknown) attractor, there are still issues. In a general setting, one cannot strictly assume that there are no regions where the system collapses to a fixed point or becomes periodic. In the case of the Lorenz63 system we know that our assumptions are reasonable only because we know the full attractor. Additionally, the results of these two test might depend strongly on the length of the reconstructed attractor. We expand on these considerations in the discussion section.

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019


Nonlinear Processes in Geophysics
Discussions
Open Access
EGU

4 Experiments and Results

4.1 Reconstructing Lorenz 63 with Neural Networks

We train a network on a long Lorenz63 simulation (1e6 timesteps) meant to explore all regions of the butterfly, and make forecasts 0.01 time-units ahead. The network is then initialized with a random state out of the test dataset, and we make 1e6 consecutive forecasts. The Lorenz63 run and the network run are shown in fig. 1. The network attractor looks visually reasonable: it has the typical "butterfly" shape and, most importantly, it neither drifts into a periodic orbit nor collapses into a fixed point. The main deficiency in the network is that the inner regions of the wings are underpopulated. Figure 1 c) shows the mean absolute error (MAE) of 1-step network forecasts initialized at every point in the test set. The forecasts typically display small errors (<0.03). The highest errors occur in the edges of the wings, where recurrences are rare and the intrinsic predictability of the system is low (Faranda et al., 2017).

4.2 Training on incomplete data

Here we take a drastic approach towards training on incomplete data: namely, we select training data that explores only limited regions of the phase-space. This selection is done via "cutting out" chunks out of the phase-space. Since the training is done in pairs (timesteps t_i and t_{i+1}), the points at the locations where the trajectories are truncated are removed from the training data to avoid artificial "jumps" towards the next included point. First, we investigate whether neural networks trained on different phase space regions are able to make short-term predictions in other parts of the attractor. Then, we assess whether it may be possible to reconstruct the full attractor with these neural networks.

4.2.1 Short-term forecasting

Figure 2 shows the short-term forecast error for a network trained only on the left wing (a,d), only on the right wing (b,e) and on a butterfly with a truncated right wing tip (c, f). In the wing where training data was present, the forecast error is very similar to the error of the network trained on the full attractor (fig. 1 c). In the wing that was excluded during training, the forecast error is much higher. It is in fact so high (mean absolute error on the order of 0.7) that the forecasts have little to do with the real system. Closer examination reveals that when initialized in the "missing" wing, the forecasts point back towards the "training" wing (fig. B1, fig.4 f,i). When excluding only the tip of the right wing, the network manages to make somewhat reasonable forecasts in the "missing" region, but still the forecast error here is roughly 10 times higher than in the regions included in the training (fig. 2 c,f). These findings suggest that the network does not learn a global mapping, but a localized one which fails in previously unexplored regions. The simplicity of the system allows us to examine this behaviour further by looking at the activation of the individual neurons in the network.

Figure 3 shows the distribution of activations (i.e. output) of the hidden neurons for the network trained on the whole attractor, when fed with input from the left wing only (a,b,c) and from the right wing only (d,e,f). The numbering of the neurons is arbitrary, but is consistent within the figure. Some neurons have very similar activations in both wings, whereas the

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019

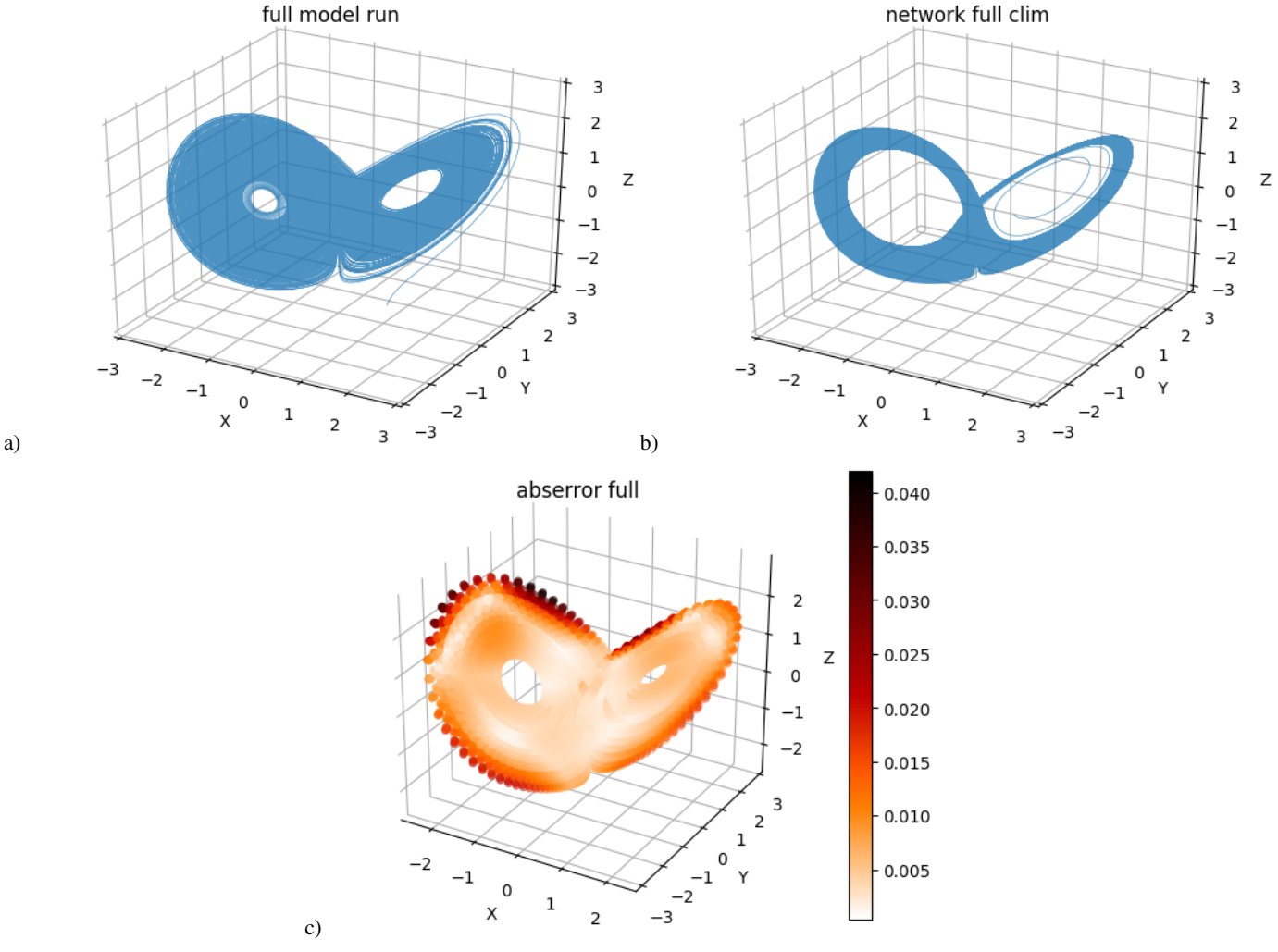


Figure 1. a) A long integration of the Lorenz63 model. b) Timeseries produced with a neural network optimized on short-term forecast error, initialized from a random initial state not used in the training. c) Short-term forecast errors of the neural network initialised at a large number of points not used for training.

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019

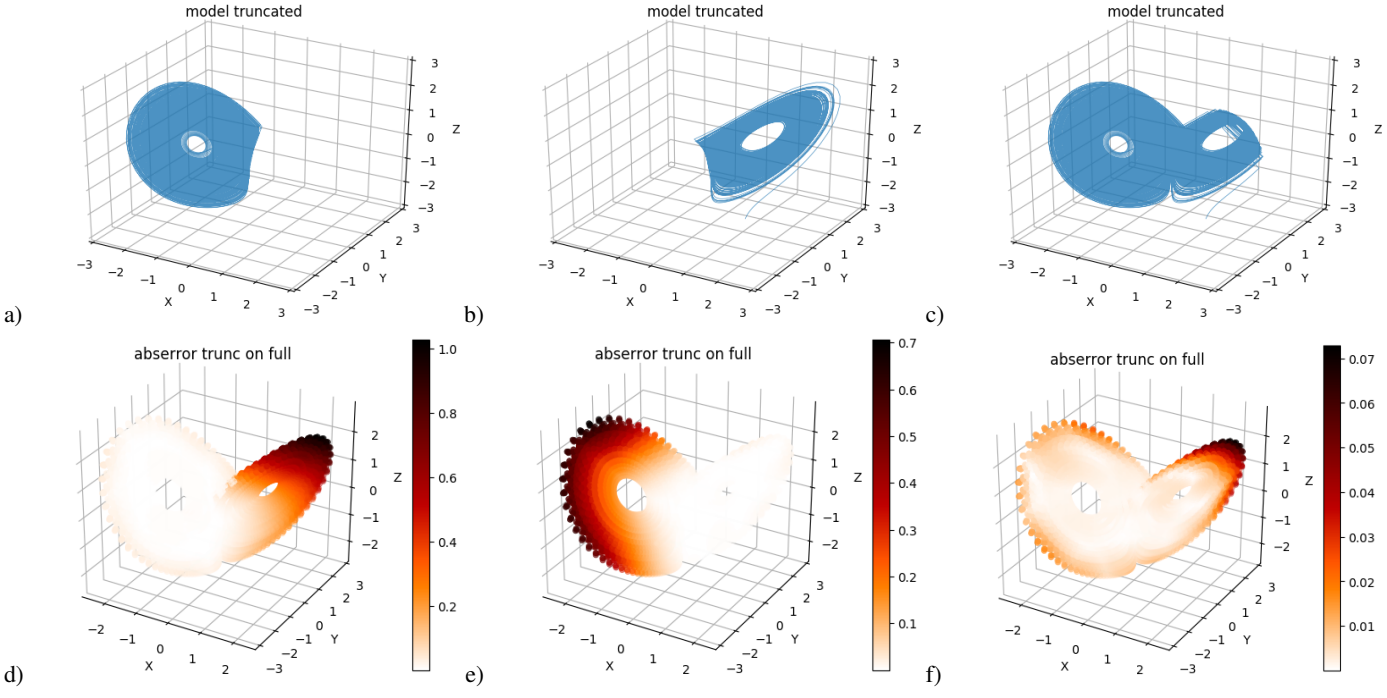


Figure 2. Truncated sets of Lorenz63 training data (a-c) and short-term forecast error (MAE) of neural networks trained on these sets (d-f). Note the different colorscales in (d-f).

distributions of other neurons change significantly. In both wings, some of the neurons have very little spread in activation, meaning that their output is relatively independent of the exact location within the wing. However, these "low-variance" neurons are not the same in the two wings. We hypothesize that they correspond to a localized mapping that the network learned for the other wing. To test this, we identify the neuron with least spread in activation (defined here as standard-deviation of the activation) for all points on each wing. These are neuron 7 in the left wing and neuron 5 in the right wing. The magnitudes of the activation values of the two neurons are not of concern to our argument here. We then create modified networks by fixing the output of each of these neuron in turn at their mean activation level for the relevant wing. With this modified network, we make predictions on the whole attractor. The result is shown in the right-hand column of fig. 3. The main effect of fixing the output of the neuron that has low spread in the left wing is that the forecast error in part of the right wing increases sharply. A similar behaviour is seen when fixing the activation of the neuron that has lowest variance in the right wing, for forecasts in the left wing. In fact, the errors this activation-fixing procedure introduces are almost comparable to those of the networks trained only on one wing (fig. 2). This suggests that these neurons correspond to the localized mapping part of the network we had speculated about earlier, and deactivating them forces the network to fall back to its global mapping, which we have seen is poor. We have tested this analysis on several different training realizations. In some cases, the lowest-variance neuron contributes also to some extent to the skill of the forecasts in the wing where it has low variance, and sometimes 2 neurons

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019

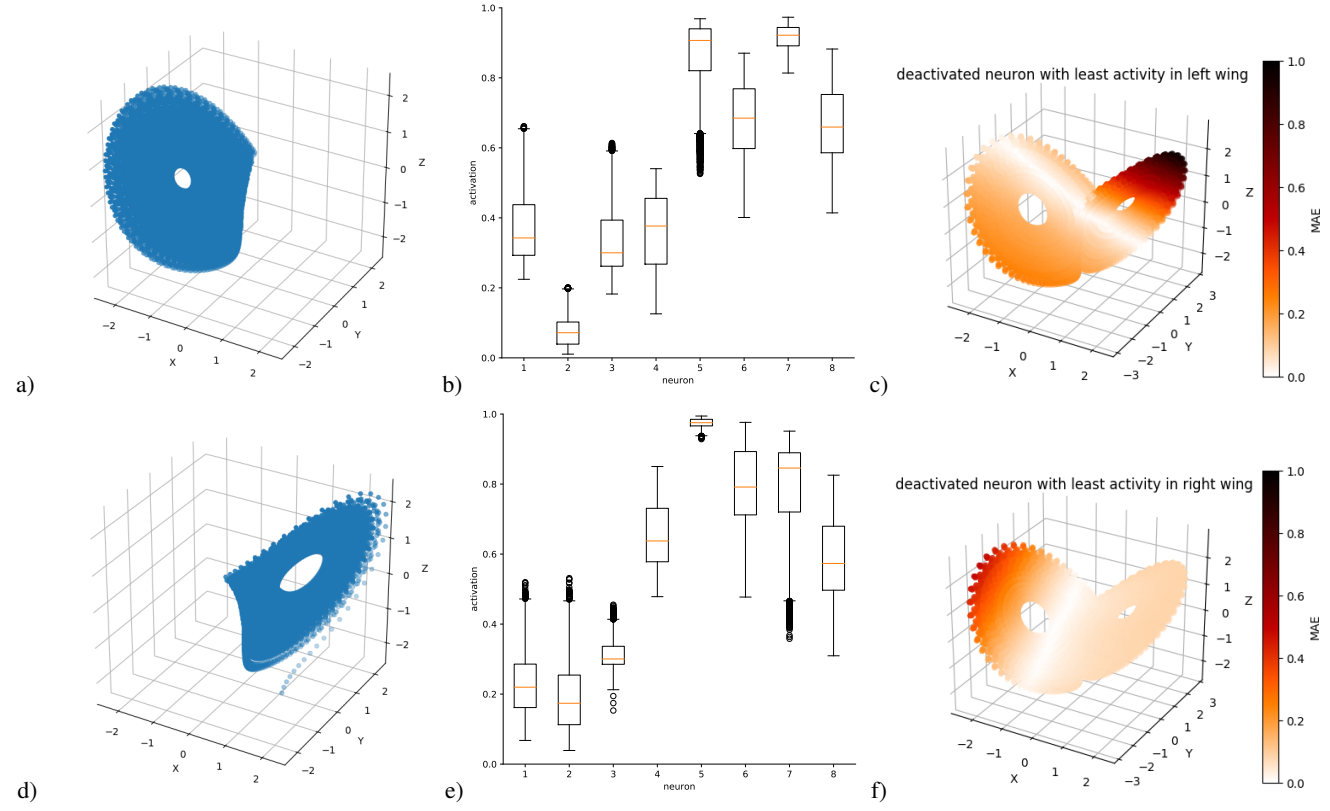


Figure 3. a,b: what we refer to as "left" and "right" wings of the Lorenz63 attractor. c,d: neuron activations for the two wings. e,f: short-term forecast errors (MAE) for the networks with the activation levels of neurons 7 and 5 ("low-variance neurons") fixed everywhere at their mean values for the left and right wings, respectively.

have very low variance in the same wing (not shown). Thus, the result that part of the network learns a specific part of the phase-space may apply to both individual or multiple neurons.

4.2.2 Reconstructing the full attractor

We next attempt to use the networks outlined in Sect. 4.2.1 to reconstruct the full attractor by making a long series of iterative forecasts with the networks. We already showed that this is possible when training on the whole attractor. When we remove only a small part of the attractor from the training data (the tip of the right wing, fig.4 a), the networks are able to reproduce a reasonable attractor regardless of whether they are selected using the density selection criterion (fig.4 b) or the density-full criterion (fig.4 c) – see Sect. 3.4. In this case, the neural networks are able to explore also the regions that are not explored by any of the trajectory segments in the training data. However, networks trained on single wings fail to reconstruct the full attractor (fig.4 e, f), independent of the selection criterion used. These networks either failed the selection tests, or produced trajectories that populate only the wing used in the training. The networks also fail to explore the other wing when they are

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019

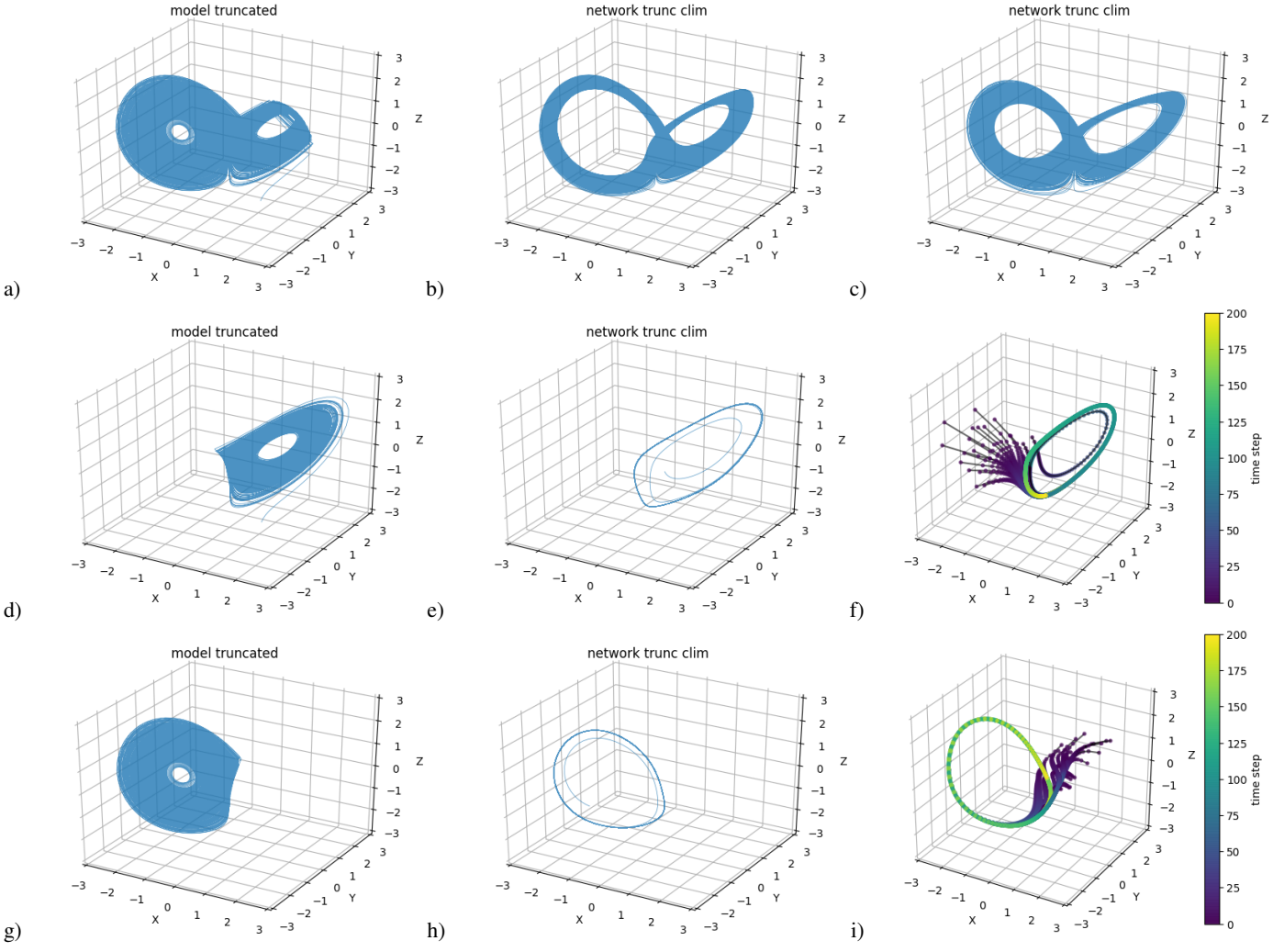


Figure 4. Reconstruction of the Lorenz63 system with neural networks trained on truncated data. a,d,g) Truncated sets of Lorenz63 training data. Reconstructed attractors with networks trained on (a) and selected using the density-selection (b) and density-full selection (c) criteria. e,h) Reconstructed attractors trained on (d,g) respectively, selected based on having no repeated points. f,i) trajectories initialized with random points from the region of the attractor that was left out in the training data in (d) and (g), respectively. The points in f,i indicate single forecast steps.

initialized from states within it. In this case, the trajectories immediately point back to the wing the network was trained on, and reach it after a couple of iterative forecasts (fig.4 f, i), implying that the network reproduces a dynamics that populates only the wing that was included in the training.

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019

Nonlinear Processes in Geophysics
Discussions
Open Access
EGU

4.3 Learning external forcings of the system

We next address the second question raised in the introduction: can our neural networks learn the influence of slowly varying external forcing on the system? We explore this on the Lorenz63 and the Lorenz95 systems.

4.3.1 Lorenz63

As external forcing scenario we consider a gradual linear increase of the σ parameter (eq. 3). This may be conceptually likened to the continuous anthropogenic forcing on the climate system. In this experiment, we run the model for 5e5 (experiment 1) and 5e6 (experiment 2) timesteps, and increase σ from 7 at the start of the run to 15 at the end. This is repeated twice with different initial conditions, creating a training and a testing run. We then train the neural networks on the first 20% (hereafter: initial period) of the training run, and test it on the last 20% (hereafter: final period) of the testing run, and vice versa (training on final period and testing on initial period). We also use the networks trained on the initial and final periods of a training run to forecast the same periods of the testing run. We do this with: 1) our standard network configuration, as described in Sect. 3.2 and 2) with the same configuration, but using σ as additional input to the network. As baseline forecast for the both the initial and final periods, we consider the Lorenz63 system with σ fixed to the mean value from the initial and final periods of the runs. This provides for each period four network forecasts (standard network trained on same period, standard network trained on other period, network including σ trained on same period, network including σ trained on other period) and two baseline measures (lorenz system with fixed σ from same period, lorenz system with fixed σ from other period). These experiments are repeated 10 times and the mean forecast skill as well as the standard deviation for the individual networks is computed.

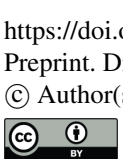
The results are shown in fig. 5. For the run with 5e5 timesteps (resulting in 1e5 training timesteps), using σ as additional input does not improve the forecasts in either the initial or the final periods. In the final period, all networks perform comparably, and adding σ as additional information if anything appears to slightly degrade the networks' performance. A similar result holds when considering forecast of the initial periods, with the difference that the loss of skill due to the variable σ in the case of networks trained on the final period is very large (Fig. 5b).

This changes somewhat when using the run with 5e6 timesteps, which has 10 times more training samples (fig. 5c). Including σ as input still provides no advantage, but at least it does not significantly degrade the forecasts for the initial period relative to the no-σ case. However, it does lead to a significant degradation of the forecasts for the final period, for the networks trained on the initial period.

4.3.2 Lorenz95

We next consider a variable forcing scenario for the Lorenz95 system, by changing the parameter F. The procedure we adopt is conceptually identical to that for the Lorenz63 scenario, with F substituting σ and F varying from 4 (periodic regime) to 16 (highly turbulent regime). The results are shown in fig. 6. The networks perform considerably worse in the forcing regimes they were not trained on. Like in the Lorenz63 system, including the forcing term in the networks does not necessarily improve the forecasts. Rather, the changes in MAE between the different networks appear to be heavily dependent on the length of the

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019



Nonlinear Processes in Geophysics
Discussions
Open Access
EGU

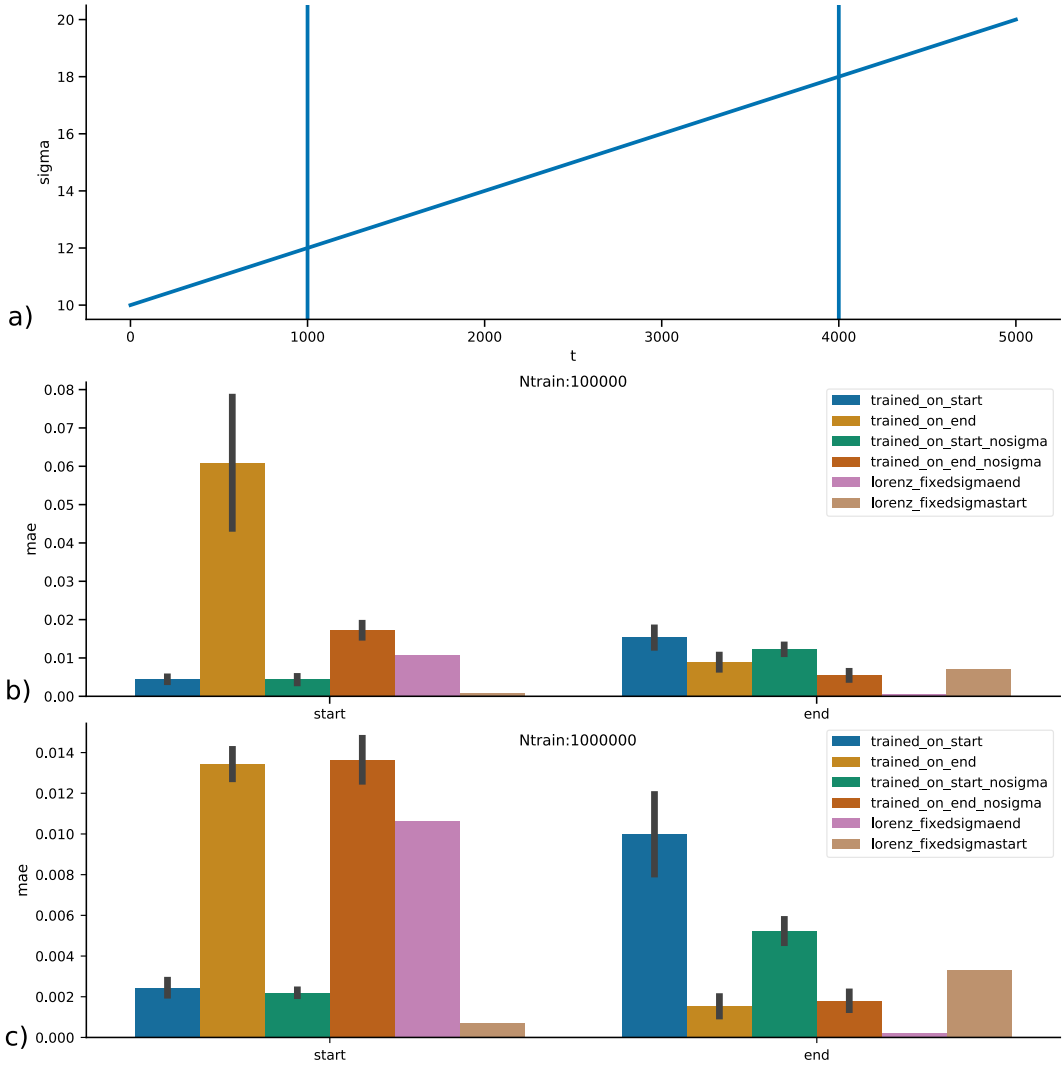


Figure 5. Experiments simulating external forcing with varying σ in the Lorenz63 system. a) Forcing vs timesteps. The vertical lines indicate where the "start" part (first 20%) ends and where the "end" part (last 20%) starts. b,c) Mean absolute error of the network forecasts and of the baseline forecasts (Lorenz63 equations with fixed σ) for runs with 1e5 and 1e6 training timesteps, respectively. The left group of bars denotes the skill on forecasting on the "start" part, the right bars on the "end" part. Some of the forecast errors are so small that the bars are barely visible. The black bars show the standard deviation over 10 iterations of each experiment.

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019


Nonlinear Processes in Geophysics
Discussions
Open Access
EGU

training data. For intermediate training length (fig.6 c), including F provides a clear benefit for forecasting the forcing ranges the network was not trained on. However, for shorter or longer training sets, this improvement vanishes or even reverses (fig.6 b, d).

5 Discussion and conclusion

In this study, we explored how well neural networks can 1) generalize the behaviour of a chaotic system to its full phase-space when trained only on part of said phase-space, and 2) learn the influence of a slow external forcing on a chaotic dynamical system. Both points are of direct relevance to the application of neural networks in climate science. The climate system is highly chaotic, our observational data likely includes only a small portion of the possible states of the system and we are subjecting the system to a slowly varying forcing by emitting large amounts of greenhouse gases. To address these points, we used two idealised representations of atmospheric processes, namely the Lorenz63 and the Lorenz95 models. We used neural network architectures that are shown to work well on these systems when trained on the full phase space and without external forcing.

For the first point, we showed that in general networks trained on only part of the Lorenz63 attractor are unable to reproduce trajectories outside the regions they were trained on. When making short-term forecasts initialized from points in the unknown phase space regions, the trajectories of the network forecast simply point back towards the region included in the training. This makes the forecasts so poor as to be practically useless. Similar issues arise when running a large number of iterated forecasts, so as to reproduce a long trajectory of the system using the neural networks. Again, the network trajectories do not explore regions of the phase space that were not included in the training. The only exception are cases where very small regions are excluded from the training data (and determining what is the limiting size of "very small" remains an open question). This implies that using neural networks for emulating climate models, as proposed in Scher (2018) and Scher and Messori (2019), may be more challenging than expected. The same goes for making forecasts of unprecedented events, or of events originating from unprecedented atmospheric configurations.

We interpret our results as indicating that the neural networks do not learn to approximate the equations underlying the dynamics of the system – which would be akin to a “global mapping” – but rather develop a “regionalized view” of the system, whereby specific neurons contribute to the forecasts in specific regions of the phase space. Thus, when parts of the phase space are left out, the regionalized mapping fails to produce sensible estimates of the system’s behaviour beyond the regions it has already seen. An additional challenge in this context that became obvious during the design of our experiments is the choice of criteria to judge successful attractor reconstruction after training. As discussed in the methods section, in order to reconstruct the attractor of a chaotic dynamical system with neural networks, it is not enough to minimize the error of short-term forecasts. Instead, one also needs to judge whether the trained network successfully reconstructs the attractor. When the training data contains only data from part of the phase-space, this raises issues related to information availability, as in real-world applications it would not be a valid approach to compare the reconstructed attractor with the full attractor. We also explored alternative approaches that do not rely on comparing the attractor from the trained network with the original

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019

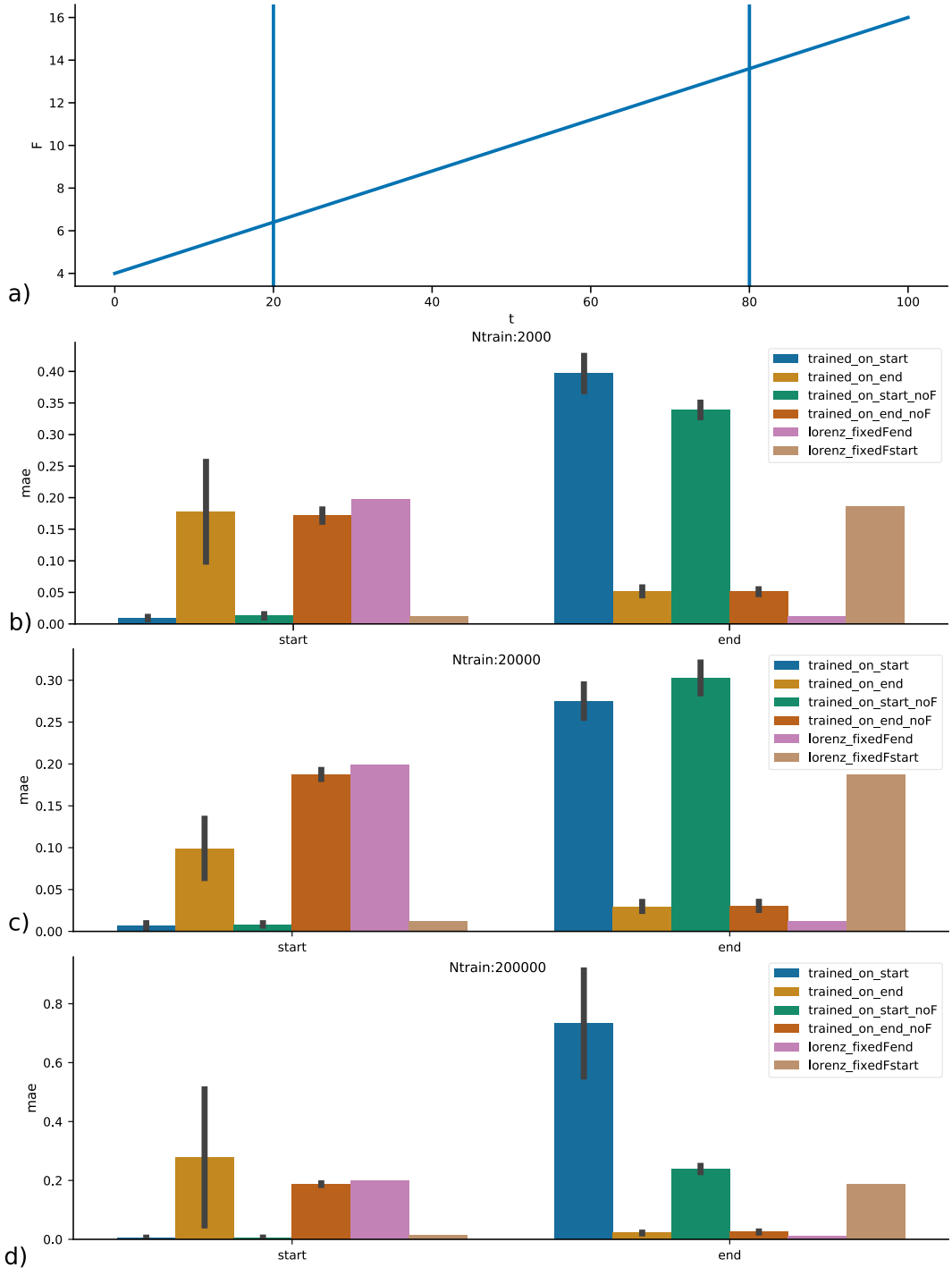


Figure 6. Same as fig. 5, but for forcing experiments with the Lorenz95 model varying the parameter F. a) Forcing vs timesteps. b,c,d) results for runs with 2e3, 2e4 and 2e5, training steps, respectively.

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019


Nonlinear Processes in Geophysics
Discussions
Open Access
EGU

attractor. However, also the alternative methods suffer from problems related to information availability at training time, and are potentially sensitive to the length of the reconstructed attractor. This implies that for applications that rely on neural networks to reconstruct the attractor of a dynamical system, very careful consideration of how to test test the reconstruction at training time is necessary.

To address the second point we raised, we simulated an external forcing on the Lorenz63 and Lorenz95 systems via slowly changing model parameters. We then trained neural networks both with and without the changing model parameters as additional input. The results vary considerably depending on length of the training set and iteration of the experiment, but in many cases including forcing information degrades the network forecasts. That is, it may be better not to include the forcing variable as network input, even when the system undergoes a forcing which is exactly known. In general, the neural networks thus struggle to leverage the additional information concerning the external forcing. Since it is impossible to know a priori which specific combinations of parameters may result in improved forecasts when including the forcing term and which may result in degraded forecasts, this poses a formidable (though not necessarily unsolvable) challenge to the idea of emulating climate-change projections with neural networks. Specifically, it implies that it may be unwise to apply an architecture that in principle works reasonably on past atmospheric data (like the one proposed by Dueben and Bauer (2018)) to future climates. Our results are similar to Rasp et al. (2018), who found that their neural network based subgrid-model is not able to extrapolate very far into new climate states, even though it is able to interpolate between different extreme climate states.

As a caveat, we underline that our experiments were performed on highly idealised systems and it is hard to estimate the extent to which they may generalise to more complex systems such as atmospheric general circulation models or even global climate models. Nonetheless, Scher and Messori (2019) have shown that some insights drawn from simple models in the context of machine learning do map to more complex systems. A second caveat is that our approach to truncating the training data was somewhat extreme: it is likely that the regions of the phase-space explored by the climate system during the satellite era are more representative of the hypothetical climate attractor than a single wing of the butterfly is for the Lorenz63 system. Finally, it is virtually impossible to robustly demonstrate that neural networks cannot fulfill a specific task. In fact, the Universal Approximation Theorem loosely states that a feed-forward neural network can approximate any continuous function with any desired accuracy, as long as it has a large enough number of hidden layers (Hornik, 1991). However, this does not mean that there is a practically feasible way to *find* the optimal network (network meaning here both architecture and weights) and train it with sufficient data.

We hope that this study can pose as a starting point for more discussion on the potentials and limitations of neural networks in the context of complex dynamical systems. Future studies could expand to more realistic systems (e.g. general circulation models), explore neural network architectures beyond the feed-forward networks used here (e.g. recurrent architectures) and the influence of noisy training data. Additionally, a more mathematically rigorous approach – as opposed to the empirical approach used here – might shed interesting new light on the topic.

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019


Nonlinear Processes in Geophysics
Discussions
Open Access
EGU

Code availability. The code used for this study is available in the accompanying Zenodo repository (10.5281/zenodo.2649879) and on S.S.s github repository (https://github.com/seb1000/code-for-Generalization-properties-of-neural-networks-trained-on-Lorenz-systems)

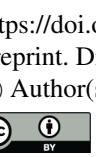
Appendix A: Tuning of neural network architecture for Lorenz95

The use of neural networks requires a large number of somewhat arbitrary choices to be made before the training of the network even begins. The first step is to select a specific network architecture, and choose the so-called hyperparameters. As basic architecture here we chose stacked convolution layers, which wrap around the circular domain. Next, we performed an exhaustive gridsearch over network configurations and hyperparameters. The learning rate was varied from 0.00001 to 0.003; the kernel size of the convolution layers (the "stencil" the convolution operations uses) from 3 to 9; the number of convolution layers from 1 to 9, and the depth of each convolution layer from 32 to 128. Furthermore, both sigmoid and rectified linear units ("ReLu") activation functions were tested.

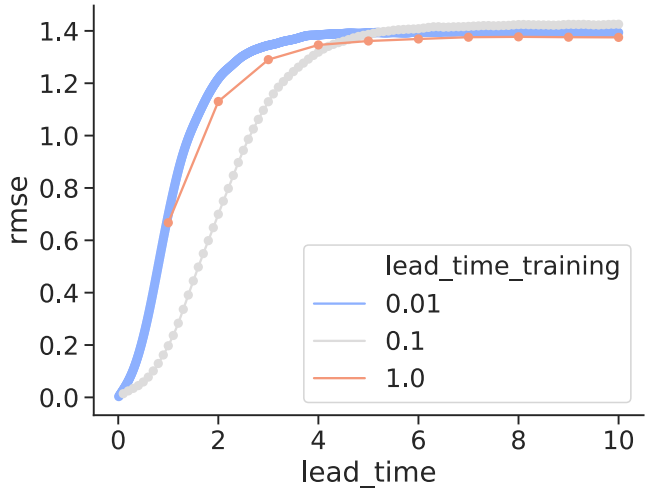
The tuning was done with a Lorenz95 run with $F = 8$, a timestep of 0.01 and $1e4$ timesteps. It was performed independently for forecast lead-times of 0.01, 0.1 and 1. For each lead-time, a different network architecture worked best. When training on lead-times of 0.01, a single convolution layer with kernel size 5 worked best. For lead-time 0.1, 2 convolution layers with kernel size 5 worked best, and for lead-time 1, 9 convolution layers with kernel size 3 were the optimal choice. When considering how stacked convolution layers work, this result is not surprising. The information available for forecasting the target value for a specific gridpoint is kernel-sized for a single layer, and increases with each additional convolution layer. From a physical point of view, the information affecting the dynamics of a specific gridpoint comes only from the immediate neighborhood for very short forecasts (given the local nature of the Lorenz95 equations). With increasing lead-time the information from an increasingly large part of the domain becomes important. Therefore, it is intuitive that for making a longer forecast in a single step, the network should have more convolution layers.

The network architecture trained on a timestep of 0.1 made the best forecasts over lead-times up to ~4 timeunits, both in terms of RMSE and anomaly correlation (when making longer forecasts through iteratively making forecasts with the network, see fig. A1). We therefore chose this network architecture (conv_depth = 128, kernel_size = 5, learning_rate= 0.003, and 2 convolution layers with ReLu activation) for the analyses presented in the study. This result also suggests that there could be an "optimal" lead-time that neural networks should be trained on for chaotic dynamical systems. The network was trained until validation loss did not increase for 4 epochs with a maximum of 30 epochs.

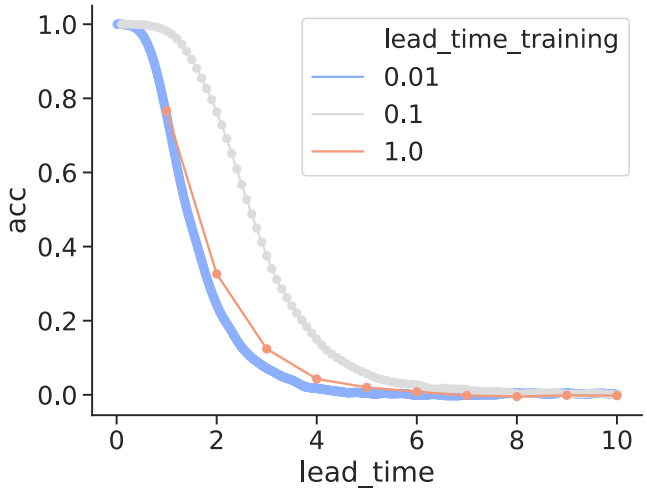
https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019



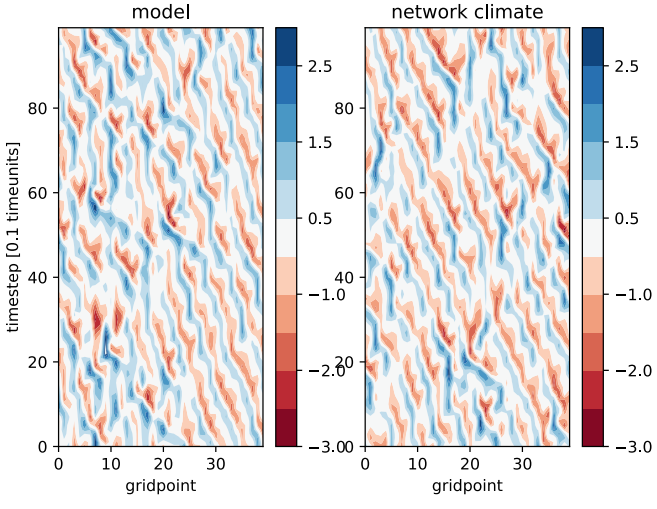
Nonlinear Processes in Geophysics
Discussions
Open Access
EGU



a)



b)



c)

Figure A1. Evaluation of network architecture for the Lorenz95 system. a,b) Forecast error (on test data) for the best network configurations when training on lead-times of 0.01, 0.1 and 1 (different colors). c) Examples of the Lorenz95 model (left) and the network model obtained through iterated forecasts trained on a lead-time of 0.1 (right)

Appendix B: Supplementary figures

Author contributions. Both authors developed the ideas underlying this study. S.S. designed the study, implemented the software, performed the analysis and drafted the manuscript. Both authors helped in interpreting the results and improving the manuscript.

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019

Nonlinear Processes in Geophysics
Discussions
Open Access
EGU

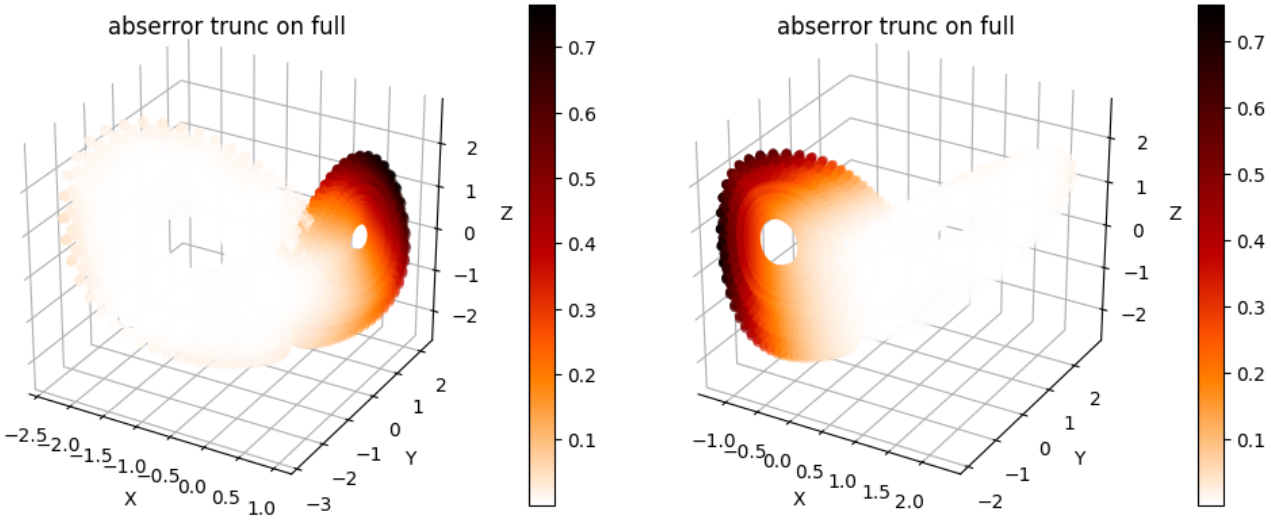


Figure B1. As fig. 2, but the points are located at the predicted state instead of at the initial state of each forecast.

Competing interests. The authors declare that they have no competing interests.

Acknowledgements. S.S. was funded by the Dept. of Meteorology of Stockholm University. G.M. was supported by the Swedish Research Council Vetenskapsrådet (grant no.: 2016-03724).

https://doi.org/10.5194/npg-2019-23
Preprint. Discussion started: 14 June 2019

Nonlinear Processes in Geophysics
Discussions
Open Access
EGU

References

Bakker, R., Schouten, J. C., Giles, C. L., Takens, F., and Bleek, C. M. v. d.: Learning Chaotic Attractors by Neural Networks, Neural Computation, 12, 2355–2383, https://doi.org/10.1162/089976600300014971, https://doi.org/10.1162/089976600300014971, 2000.

Dueben, P. D. and Bauer, P.: Challenges and design choices for global weather and climate models based on machine learning, Geoscientific Model Development, 11, 3999–4009, https://doi.org/https://doi.org/10.5194/gmd-11-3999-2018, https://www.geosci-model-dev.net/11/3999/2018/gmd-11-3999-2018.html, 2018.

Faranda, D., Messori, G., and Yiou, P.: Dynamical proxies of North Atlantic predictability and extremes, Scientific Reports, 7, 41 278, https://doi.org/10.1038/srep41278, https://www.nature.com/articles/srep41278, 2017.

Hornik, K.: Approximation capabilities of multilayer feedforward networks, Neural Networks, 4, 251–257, https://doi.org/10.1016/0893-6080(91)90009-T, http://www.sciencedirect.com/science/article/pii/089360809190009T, 1991.

Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization, CoRR, abs/1412.6980, 2015.

Krasnopolsky, V. M. and Fox-Rabinovitz, M. S.: Complex hybrid models combining deterministic and machine learning components for numerical climate modeling and weather prediction, Neural Networks, 19, 122–134, https://doi.org/10.1016/j.neunet.2006.01.002, http://www.sciencedirect.com/science/article/pii/S0893608006000050, 2006.

Krasnopolsky, V. M., Fox-Rabinovitz, M. S., and Belochitski, A. A.: Using ensemble of neural networks to learn stochastic convection parameterizations for climate and numerical weather prediction models from data simulated by a cloud resolving model, Advances in Artificial Neural Systems, 2013, 5, 2013.

Lorenz, E. N.: Deterministic nonperiodic flow, Journal of the atmospheric sciences, 20, 130–141, 1963.

Lorenz, E. N.: Predictability: A problem partly solved, in: Proc. Seminar on predictability, vol. 1, 1996.

Rasp, S. and Lerch, S.: Neural Networks for Postprocessing Ensemble Weather Forecasts, Monthly Weather Review, 146, 3885–3900, https://doi.org/10.1175/MWR-D-18-0187.1, https://doi.org/10.1175/MWR-D-18-0187.1, 2018.

Rasp, S., Pritchard, M. S., and Gentine, P.: Deep learning to represent subgrid processes in climate models, Proceedings of the National Academy of Sciences, p. 201810286, https://doi.org/10.1073/pnas.1810286115, http://www.pnas.org/content/early/2018/09/05/1810286115, 2018.

Scher, S.: Toward Data-Driven Weather and Climate Forecasting: Approximating a Simple General Circulation Model With Deep Learning, Geophysical Research Letters, 0, https://doi.org/10.1029/2018GL080704, https://agupubs.onlinelibrary.wiley.com/doi/full/10.1029/2018GL080704, 2018.

Scher, S. and Messori, G.: Predicting weather forecast uncertainty with machine learning, Quarterly Journal of the Royal Meteorological Society, 144, 2830–2841, https://doi.org/10.1002/qj.3410, https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.3410, 2018.

Scher, S. and Messori, G.: Weather and climate forecasting with neural networks: using GCMs with different complexity as study-ground, Geoscientific Model Development Discussions, pp. 1–15, https://doi.org/https://doi.org/10.5194/gmd-2019-53, https://www.geosci-model-dev-discuss.net/gmd-2019-53/, 2019.

Vlachas, P. R., Byeon, W., Wan, Z. Y., Sapsis, T. P., and Koumoutsakos, P.: Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks, Proc. R. Soc. A, 474, 20170 844, https://doi.org/10.1098/rspa.2017.0844, http://rspa.royalsocietypublishing.org/content/474/2213/20170844, 2018.

Zhang, L.: Artificial neural networks model design of Lorenz chaotic system for EEG pattern recognition and prediction, in: 2017 IEEE Life Sciences Conference (LSC), pp. 39–42, https://doi.org/10.1109/LSC.2017.8268138, 2017.