We thank the reviewers for their constructive comments. We have now prepared a new manuscript based on their comments. The main changes we made are:

- the network architecture for the Lorenz63 model is now based on objective tuning, and is more similar to real-world architectures than the one in the original manuscript
- the analysis of extrapolation capabilities of the neural networks is now also done on the Lorenz95 model, in addition to the Lorenz63 model
- the forcing experiments have been re-designed to be more close to real applications
- the analysis of the importance of different regions of the neural network has been extended to several different architectures

As you will see, thanks to these new experiments, the main result of the paper slightly changed. While the result that the neural networks do not extrapolate to new phase-space regions in the Lorenz63 model still holds, for the Lorenz95 this does not seem to be the case. Also, the new forcing experiments show that the networks are to some extent learning external forcing, however only when given large ranges of forcing in the training.

We provided point-by-point response to all reviewers' comments below (in red).

Reviewer #1

The authors do not reference related work on generalization properties of neural networks to unseen data, or other machine learning models designed for non-stationary time series.

We have added a new section in the introduction ("1.2 Related work on generalization properties of neural networks") that gives a better overview of the general literature on generalization properties of neural networks.

The argumentation in Section 2, about whether the network learns only one or many mappings for different regions is inconsistent. The two representations are mathematically equivalent. The impression the reader gets about what the authors are trying to express, is whether different parts of the network are responsible for different (local dynamics) parts of the training data.

Yes, this is indeed what we had meant. Even though the two expressions are mathematically the same, we think that they help explaining this idea. Therefore, we added "Even though mathematically equivalent, the latter would imply that different parts of the network are responsible for different regions of the phase-space." to make this clearer. (p3 L15)

In the Lorenz-63 system, the authors try to answer the aforementioned question (identify specific parts of the neural network that are responsible for capturing the dynamics locally), by analyzing the activation levels of the neurons of the neural network, freezing neurons that are mostly active on specific regions, to check the deterioration of performance on other regions (for a model trained on the whole training data). This argument, that parts of the neural network are responsible for local models of the training data, is very interesting. Especially for large (relative to the application) overparametrized models, this argument makes sense. How-ever, it has to be tested systematically in larger models (maybe a large model applied to the Lorenz-96), and in a more structured way to be accepted as a general attribute of neural networks, as the small network used in this study might be misleading.

In the new manuscript, we use a larger network throughout for the Lorenz63 (2 hidden layers with 128 neurons each). Additionally, we extended the analysis regarding neuron activations to a wider range of architectures. (fig. 4 in the new manuscript).

The authors do not explain the training procedure and how they cope against overfitting in the CNN applied to Lorenz-95. Especially in the low data regime, the absence of measures against overfitting can have a detrimental influence on the performance on the test dataset. Since the neural network is forecasting a deterministic system with full state information, the prediction accuracy reported in the Appendix on page 17, seems quite low. In the provided plots, the networks seem to be forecasting inaccurately, as the difference in the plots even at early timesteps is obvious.

We now added more detail on the training procedures both in the method section and in the Appendix. Additionally, we changed the plot you referred to. The wave-plot in the original manuscript showed the evolution of the real system, and the evolution of a long run of the network. However, they were not initialized from the same state, therefore giving the impression that the forecasts are very bad. The plot in the new manuscript (fig. B1 c) now shows the same, but with the network run initialized from the same initial state as the real system. From this, together with the other panels of fig. B1, it can now be seen that the network forecasts are skillful.

The generalization of the study is problematic. The study is limited to feedforward neural networks, with a one to one training scheme. By changing the loss to include some stability metric, or long-term performance, trying out different architectures, regularization techniques, etc. the generalization properties might be improved. The first problem of extrapolation is a general pitfall of data-driven approaches, however, the second problem of non-stationarity might be alleviated with more sophisticated architectures. As reported in Section 3.4, many trainedCNN networks are not stable, because they were trained for single step forecasts. This is expected, as the neural networks are not trained for long-term forecasting. RNNs can be used in these low-dimensional systems, backpropagating the gradient many timesteps in the past to ensure stability. The authors use a posteriori analysis of the networks to identify the stable ones. Moreover, the models are applied to non-stationary timeseries with external forcing, which is a really challenging application. The selected models and the training procedure used is not adequate to extract general conclusions. For example, complex RNN architectures that try to capture multiple time scales, or Reservoir Computing approaches might work better. The conclusions of the paper should be specific only to feed-forward neural networks. The arsenal of machine learning tools to counter these open problems is much wider.

We completely agree that our results are specific to feed-forward architectures only. We realize that we should have made this clearer in the original manuscript. In the new manuscript, we now clearly mention this in the abstract and the conclusion section. Also, we changed “neural networks” in the title to “feed-forward neural networks”. Additionally, we discuss in the conclusion section that other methods might alleviate the problems we found.

The second question the paper poses, is a very interesting one. Real time-series data are most of the time non-stationary. Even though many neural architectures have been successfully used in seasonal or non-stationary data, it is not clear if the networks can actually learn varying dynamics, or how efficient they are in that. One solution could be to train networks on the fly as new data come in. There is available literature on applying machine learning approaches for non-stationary data. Even though the model used in this study appears to be incapable of generalizing, this might not hold for other models. For example, the forcing was provided as an additional input to the network. However, we do know that this external forcing is not the same type of input as the rest. This information could be provided in a different way to the network.

You are right that the forcing is not of the same type as the rest of the input. We added the following sentence in the conclusion: “Regarding model architectures, for the forcing experiments it might also be possible that presenting the forcing in another way than done here (e.g. designing into the network that the forcing variable has different characteristics than the state variables) may improve the learning of the influence of the forcing.” (p20 L3) We have further significantly updated the section discussing varying forcing, for both the Lorenz63 and Lorenz95 attractors.

The statement "... the trajectories of the network forecast simply point back towards the region included in the training." Regarding the behavior of the neural network in regions of the phase space not included in the training data, seems rather arbitrary. Since the neural network is not trained in these regions the behavior can be anything.

What we meant to say is that we actually *observe* that in our trained networks: the network forecasts initialized outside the training phase space do point back to the training phase space. You are of course right that as an a-priori assumption this would be rather arbitrary. We now removed the word “simply” to make clear that this is not simply a trivial a-priori assumption, but an observation.

The first problem of generalizing to unseen data is a well-known one. As a data-driven approach, neural networks have a hard time to extrapolate to unseen regions in the dataspace. This is addressed in many

previous studies, not only related with dynamical systems. Regularization, coupling neural networks with equations, adding constraints etc. are known measures to cope with this deficiency. Most data-driven methods suffer from this problem. It is not surprising to see that a small neural network trained on the left wing of the Lorenz-63 attractor cannot generalize to the vastly different dynamics (in terms of data, not equations) of the right wing. The situation is expected to worsen as the models grows bigger (overfitting easier).

We now use a bigger network as our main architecture, and the situation is very similar to the architecture in our original manuscript. We agree that the example of training on one wing and testing on the other wing is quite “extreme” – which we indeed mention in the text – but we think it provides a good example that the network does in fact not learn the underlying equations.

Implicitly, the authors state a very interesting question, whether the neural networks learn sub-networks that are responsible for modeling the dynamics locally in parts of the training data. In the Lorenz-63 system, they manage to demonstrate this in terms of identifying the neuron that seems to be responsible for a specific part of the training data (right or left wing). Whether this argument holds for large models or other more complicated dynamics and is not specific to this study, remains open. However, it is an original and interesting finding that needs to be tested for more general settings (large networks, more applications).

We agree that in order to generalize our result, one needs to test different networks, and also different applications. To address the first point, we have now extended our analysis to a wider range of network architectures, training from shallow ones with few neurons, to deeper ones with wide layers. The finding of different parts of the network controlling different regions of phase space also – at least to some extend – holds for these architectures (see section 4.1.2). While it would indeed be very interesting to also look at other applications, we think this is beyond the scope of this study. However, in the conclusion section we now reference to a study in the field of image generation with Generative Adverserial Networks that had similar results (https://arxiv.org/pdf/1811.10597.pdf)

The second issue raised, is whether NN can learn dynamics that evolve based on external forcing. This is connected with the known open problem of neural networks learning from non-stationary data/dynamics. The architectures proposed in the study are not compared with other state-of-the art approaches, liker reservoir computers, RNNs, ARIMA models, etc. and long-term results are not presented (from iterative forecasting) so it is not straightforward to judge their efficiency.

Accept after the revision of the issues raised above, or at least referencing them in the text. Especially for the argument about the sub-networks having learned local dynamics, a bigger model needs to be tested. I doubt there is any model applied in practice with only 8 neurons. ML models applied in practice have thousands to millions of parameters. In order to support this claim, it has to be tested on large models in a systematic way, which is however, challenging to achieve.

As already mentioned above, we extended our analysis to a wider range of architectures. Also, our main architecture is now a bigger network than in the old manuscript (2 layers with 128 neurons each), and we discuss in the conclusion section that other methods might give different results.

Typos

Page 1, line 18, typo "... the widely studied ..."2.

fixed

Page 2, line 5, typo "... the widely In in this paper ..."3.

fixed

Page 4, line 24, typo "The neural networks are trained..."4.

Unfortunately, we were not able to find the typo.

Page 6, line 19, In order to avoid misconception, the following reformulation would help the reader: "Figure 2 shows the training data and the forecast error for a network …"

We changed to “Figure 1 a,b shows the training data and the attractor reconstructed by the neural network.”

5. Page 15 line 3, typo "... of how to test test the reconstruction at training time ..."C7

We deleted the sentence in the new manuscript

Reviewer #2

My overall opinion is that whilst the questions the paper addresses are interesting and important, and the results are well-presented on the whole, the experiments performed are not very close to how neural networks would be applied in reality, so it's not clear if they have real-world applications (even notwithstanding the simplicity of the systems being studied). In particular, the performance of the neural networks on reproducing the training data often looks so poor that they would not be used in an application, or the training performance is not presented in enough detail, so it's not clear that the results would apply to real-world applications that would require good validation performance. Also, the changes in forcings applied in the second part seem a lot larger than for applications that neural networks might be considered for. In addition, the authors' experiments on the Lorenz '63 system have used a particular neural network design, where the full state is predicted at every time step (it's not clear if this is the case for the Lorenz '95 system as well) – this may be expected to work worse than other designs where only the change in the state is predicted at each step, or where bias-correction of an approximate dynamical model is performed, and the results here are not clearly generalisable to those set ups.

I have given more detailed comments below. I think the work could eventually be publishable, if the comments are adequately addressed. Since my comments are quite substantial, it could be acceptable to just include the L63 experiments on training on part of the attractor and the L95 experiments on response to forcing – the L63 experiments on responding to a parameter change seem less applicable to real-world cases like predicting climate change. I do encourage the authors to continue with this line of investigation, which I think is potentially very valuable.

We thank you for your positive outlook on our study. We have decided to still include a “forcing” experiment with the L63 system. Even though we agree that it is less applicable to real-world cases, we think it is interesting to include it so as to broaden the range of examples we provide in the study. The study now addresses both questions (extrapolation to left-out phase space regions and learning of external forcing) on both systems, which we think is the most logical and complete approach

Most significant comments:

1. Training performance of neural networks:
a. For the Lorenz '63 experiments, the ability of the trained neural network to reproduce the attractor appears quite poor (fig.1), and much worse than in the results presented by Zhang (2017), whose work the authors say they are following. A model with such performance would not be used in any real-world application I can think of, and the later results may be much worse than for a well-trained system.We thank you for your positive outlook on our study. We have decided to still include a “forcing” experiment with the L63 system. Even though we agree that it is less applicable to real-world cases, we think it is interesting to include it so as to broaden the range of examples we provide in the study. The study now addresses both questions (extrapolation to left-out phase space regions and learning of external forcing) on both systems, which we think is the most logical and complete approach
It should probably be checked that all of the important steps in the prior work were followed, and if that does not resolve the problem, different architectures tried (e.g. using more neurons) until a good simulation of the attractor is produced.

In the new manuscript we have abandoned the architecture from Zhang (2017), and instead used an objective tuning procedure (similar to the one we already used for the L95, see Appendix A in the new

manuscript). The network that came out of this tuning procedure produces a much better reconstruction of the L63 attractor (fig.1 b)

b. More diagnostics for the performance of the Lorenz '95 networks on the training data and the equivalent portion of the test data should be presented to indicate the system's performance and the degree of overfitting e.g. MAE of single-timestep predictions relative to the variance of the system's tendencies.

We now included a plot that shows the distribution of MAE of 1 timesteps predictions, both on the training and on the test data. For reference, we also included the distribution of the mean absolute tendency of the model. (fig. B1c). As can be seen in the figure, the errors are much smaller than the tendencies, and there is only very slight overfitting.

c. In the experiments testing how well the networks capture the response to a changing forcing, it needs to be shown how well the networks reproduce the trend in the training data. For the results to be applicable to predicting the path of global warming, for example, there should be a discernible trend in the training data and the neural networks should reproduce it with an accuracy similar to what climate models achieve, else they would be deemed to be unsuitable for use in prediction.

Our experiments were not designed to allow the network to predict any type of trend. The idea is to test whether the networks can learn the influence of an external forcing (which might have a trend) on the short-term dynamics. Indeed, in climate change, changes in variability are arguably as important as long-term trends, even for temperature (e.g. https://journals.ametsoc.org/doi/full/10.1175/JCLI-D-18-0462.1).

2. Forcing experiments:
a.
▪ The changes in the forcing terms are rather large compared to the effects expected from anthropogenic climate forcing, for example, and I'm not aware of another real-world case where neural networks would be considered for modelling the effects of such large changes in forcing, so it's not clear to me that these results have real-world applicability. For context, anthropogenic radiative forcing of the climate system is projected to be up to a few percent of solar radiative forcing, and in the scenarios with the largest climate changes, total global warming is around 5x what has been seen in the 20 th century, and comparable in size (though not rate) to changes in between ice ages (so we arguably have some data for testing whether models can simulate such large changes well). In the Lorenz '63 experiments here, the change in the sigma parameter (meant to be analogous to radiative forcing of climate?) changes by a factor of 2. In the Lorenz '95 experiments, the forcing change is enough to change the system from being periodic to highly turbulent, which is a much larger
qualitative change than expected from climate change. I wouldn't have expected neural networks to perform well at the tasks set, namely simulating systems that are very different from what they've been trained on, so these results don't seem to provide much new information.

We agree that our original experiments were rather extreme. We have now completely redesigned the forcing experiments. Instead of relatively arbitrarily picking out a “high” and a “low” forcing, we now trained the networks on 6 different training regimes (3 relatively narrow, and 3 broader forcing regimes). For each training regime, the network is then tested on a wide range of new regimes. (see fig 7 and 8 in the new manuscript). We believe that these new experiments provide much better insights in the abilities (and inabilities) of the networks to learn external forcings.

▪ It seems reasonable to think that neural networks could perform better at simulating the effects of smaller forcing changes, that are more comparable to those in real situations. It would be interesting to test whether the neural networks can reproduce the effects of forcing at the level seen at the end of their training period (relevant for attributing observed weather events to climate change, for example e.g. National Academy of Sciences, 2016, “Attribution of Extreme Weather Events in the Context of Climate Change”) and if so, how far beyond the range of forcing they were trained on can they make good predictions for? (c.f. the Paris climate agreement global warming targets of 1.5C and 2C, which are ~1.5x and ~2x the observed warming – it would be interesting to know if neural networks could provide results that are at all useful for predicting the effects of forcing changes of that magnitude.)

As mentioned in our comment above, our forcing experiments have been completely redesigned. These experiments now address similar questions as you proposed (in the scope of the simplified systems we are using).

▪ As a further comment, it doesn't seem likely that neural networks could learn the effects of forcings outside the range of the training data without having additional information about the effects of larger forcings e.g. the radiative effects of CO2 in the climate change context. So it seems a priori likely that for the given setup the
performance will deteriorate as the forcing becomes larger. Perhaps the experiments here could demonstrate this, but I don't think it would be that surprising.

We understand your point, but in a purely data-driven approach, this is exactly what one would want to do, e.g. to extrapolate the influence of CO2 forcing to higher values outside the training regime. We don't think it is possible to a-priori answer this question. As our new forcing experiments show, the networks are to some extend able to extrapolate the influence of large forcings, albeit only when trained on wide ranges of forcing.

b. The finding that including information about the forcing as an input often worsens performance seems surprising. One reason could be that the network architecture was tuned to optimise performance without the forcing input, and a larger architecture may be needed to perform well with this information. To be a fair test, the network architecture search should be repeated for the networks using forcing as an input – this may be especially relevant for the L63 case, where the network used is quite small. The statement in the discussion that "it may be better not to include the forcing variable as network input" does not seem well-justified due to this, and also because I do not see how in principle a neural network could predict the effect of a change in forcing if it is not given information about the forcing.

We have now repeated the network architecture search for the L95 model also using the forcing as input. Also, due to our new forcing experiments, we removed the statement "it may be better not to include the forcing variable as network input".
Finally, in our new setup, we use the networks not using the forcing information as input only as baseline, as indeed is not expected that they could learn the influence of any forcing.

3. It should be made clear in the abstract and conclusions that the results apply for a particular choice of neural network design, namely feedforward networks predicting the system state at time t+1 given the state at time t (it's not clear if this is also the case for the L95 experiments, and this should be clearly stated). Also, the L63 experiments testing whether neural networks could represent the system in one wing of the attractor having been trained on the other wing only used sigmoid activation functions.

We completely agree that our results do only apply for feed-forward networks. We now mention this both in the abstract and the discussion sections. , and we also changed the manuscript title accordingly. We indeed used a full-state prediction also for the L95. This is now mentioned in the manuscript. The new architecture we chose for the L63 model uses ReLU activation functions. We repeated some of the experiments also with networks that forecast tendencies instead of full states. This leads to similar results (see Appendix C1).

• Predicting the whole state at every time step may be expected to work worse than other designs where only the change in the state is predicted at each step (e.g. Dueben and Bauer, 2018), or where bias-correction of an approximate dynamical model is performed (Watson, 2019, https://doi.org/10.1029/2018MS001597). This is because in these cases, a lot of the variance in the quantity being predicted is removed, so minimising the RMSE in training may work better to give a system that is capturing the important aspects of the variability. These different methods should also be discussed, and the abstract and conclusions should say that the results may not apply to methods like these.

We now mention this in the discussion section. However, as already mentioned, we repeated part of the experiments with networks predicting tendencies, and these were comparable to the ones predicting full states.

• The choice of sigmoid activation functions for the L63 network may be relevant for the result that the network will not make predictions outside of the range of its training data because sigmoids saturate, and may have trained to saturate at prediction values that are not far outside the boundaries of the training data region, making it difficult for the neural network to predict values outside this region. It would be good to check what happens when using an activation function that does not saturate e.g. ReLu. (Though I still

wouldn't expect it to work well when so much data is left out from training – but I'd still find the result interesting, particularly if done with a system that predicted the tendency rather than the whole state).

Our new architecture uses ReLu functions. Also, as already mentioned, we repeated part of the experiments with networks predicting tendency.

Other comments

1. p.1 L19 It would seem relevant to include citations to other recent studies using neural networks to simulate the Lorenz 95/6 system (Chattopadhyay et al., 2019, https://doi.org/10.31223/osf.io/fbxns; Watson, 2019, https://doi.org/10.1029/2018MS001597).

We now mention these 2 studies in the introduction, and we also refer to the latter in the discussion section.

2. p.2 L10 - Perhaps also mention Lorenz '95 is sometimes called Lorenz '96
we now mention this in section 3.1

3. p.2 L10-12 – Some context here may be useful. For paleoclimate variability and the oceans over multiple decades, yes, but it's less likely to be the case for the atmosphere that unforced variability would be far outside what we've observed.

we now added “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. This is particularly true for the ocean, which has much longer characteristic timescales than the atmosphere, and for applications to paleoclimatic variability.” in the introduction in order to give more context (p.2 L20).

4. p.3 L9-10 Training where no large regions of phase space is left out seems to be the most realistic case for atmospheric modelling, which is what is referred to. The experiments may be relevant for e.g. ocean modelling, where time scales are much longer. (I do think they are inherently interesting, as well.)

Thanks for pointing out that in ocean modelling the problem of not having all phase space covered in training is a realistic case. We now added “ In a climate science context, this would for example be relevant for the ocean. The latter's long characteristics timescales imply that observational datasets may cover only part of the phase-space. It is also relevant in forecasting extreme events in the atmosphere.” in the introduction. (p.4 L7).

5. p.3 L25 I'm not sure if all readers would be familiar with the Lorenz butterfly - perhaps refer to a figure.

We now refer to fig. 1a which shows a long L63 integration

6. p.3 L26 A better description of the solver is needed e.g. what software package is this from? Reference?

We used the implementation provided by scipy. We now refer both to scipy and to the original paper describing ODEPACK.

7. p.4 L1 Lorenz95 seems to be more often used to describe the 2-level model Lorenz introduced in the same paper. Perhaps use a different name to make it clear you are considering the 1-level version. (As an aside, the 2-level model could be used to test how well neural networks perform compared to a “truncated” model of the system i.e. the 1-level model – this would address whether neural networks can improve upon other reasonable
dynamical models, which is a key question.)

We now mention that the Lorenz95 model is sometimes referred to as Lorenz96, and also explicitly mention that we use the simple version of the Lorenz95 model (without additional levels).

Also, we added “Additionally, it would be interesting to extend the analysis to the 2-level version of the Lorenz95 model, which would allow to also compare the networks to ``truncated'''' in the conclusion section. (p.20 L33).

8. p.4 L4 What is the behaviour of the system like with N=40? A plot of a time series or similar may be helpful.

An example of a Lorenz95 model integration is shown in fig. B1 c (left panel). We now explicitly refer to this figure in section 3.1

9. p.4 L8-12 More detail seems necessary to make the results reproducible - e.g. Were input variables normalised? Regularisation? Minibatch size? Learning rate? Stopping rule?

All input variables were normalized to zero mean and unit variance. We used no regularization techniques except for early stopping (after the validation loss has not increased for 4 epochs). Mini-batch size is 32. This is now all mentioned in the methods section and the appendix.

10. p.4 L21-22 Why not give F as one value in the L95 system? (Analogous to using CO2 concentration as an input to a climate model.)

This is not possible in a purely convolutional architecture such as we use. It is standard practice to provide inputs that do not have a spatial pattern (such as our forcing) in the way we do, namely extending it to all points and adding it as an additional layer. For example, it is used to represent the binary game state in googles AlphaGo Zero (https://www.nature.com/articles/nature24270.pdf), and to feed the day of the year in addition to 3d fields in https://www.geosci-model-dev.net/12/2797/2019/

11. p.5 L8 – how big are the grid boxes being used?

They have a size of 0.3x0.3x0.3 on the normalized data. We now mention this in section 3.4

12. p.5 L8-9 – what does “normalized data” mean here?

We added “(normalization based on the training set, the output of the networks is always in the normalized domain).” to make this clearer. (p.6 L10)

13. p.5 L22 For testing 1), the trajectory will tend to the fixed point and only reach it after infinite time, so it's not clear that it could be identified by looking for two points to be exactly equal - perhaps set a threshold on how small a tendency is acceptable and check if the tendency magnitude falls below this and does not rise above it again?

In the new manuscript, we do not use this type of selection any longer.

14. p.5 L23-24 What is the motivation behind 2? To identify periodic behaviour? But couldn't points on a periodic orbit fall at slightly different points on that orbit and so avoid detection?

In the new manuscript, we do not use this type of selection any longer

15. p.6 L1-2 The information on training here could perhaps be merged with the earlier section about training.

We prefer to leave it in this section, as we believe it follows well the logical flow of the paragraph.

16. p.6 L5 what does "consecutive forecasts" mean as opposed to just "forecasts"?

With this we want to point out that we use one forecast to initialize the next one. We now added “(via feeding the forecast back to the input)” to make this clearer. (p.6 L28)

17. p.6 L5-6 See earlier comment about the simulated attractor not looking reasonable.

Due to our new main network architecture, the simulated attractor now closely resembles the true one (fig. 1b).

18. p.6 L9 How big are these errors compared to an average tendency magnitude? Also, could errors on the wings be larger because tendencies are typically larger? Perhaps showing a relative error would be more useful.

We now included a plot of the typical tendencies in (fig. 1d) and refer to it in the text (“To put forecast errors throughout the paper in context, panel d) shows the tendency (change between successive timesteps) of the model in different regions of phase space.”) (p7 L3).

19. p.6 L14-15 I didn't get the meaning of “the points...included point”.

If we simply remove points from the Lorenz63 model run, and then split our training set into inputs and targets (via shifting by 1 timestep), then there would be input-target pairs were the model “jumps”, because a piece of the trajectory in-between had been cut out. We added “(this is necessary because we removed parts of the models’ trajectory)” to clarify this. (p8 L1).

20. p.8 L11-12 The errors look quite a bit smaller to me, particularly for the case of fig.3f.

We discuss this in more detail in the new manuscript, since with our new network architecture it is possible to switch off more neurons at once, and the results are more complicated.

21. p.8 L12-14 I don't feel convinced by this analysis. Even if the neural network had learnt to fit the whole attractor well, there may be some neurons whose activations only vary in part of the phase space due to the values of the inputs, and fixing their values would of course be expected to degrade forecasts in areas of the phase space where they do matter.

We think this might have been a misunderstanding. In this analysis, we inspect the network that was trained on the whole attractor, and that also works well on the whole attractor. So we actually know that the network has learned the whole attractor. To make clearer that we analyzed the network trained on the whole attractor, we added *“For this, we inspect the network that was trained on the whole attractor (and forecasts well on the whole attractor).”* in section 4.1.2 (p8 L20).

22. p.11 L5-6 This experiment is not really like rising anthropogenic climate forcing because the sign of the “forcing” depends on whether y is larger than x. An additive forcing term like that used for the L95 experiments and by Palmer (1999, “A nonlinear dynamical perspective on climate prediction”, J. Clim.) would be a better analogy.

We removed the corresponding sentence. Additionally, we now discuss in the conclusion section that our experiments are only very remotely related to climate forcings.

23. p.11 L6-7 Changing the number of time steps looks to change the rate of change in the forcing as well. This is something that should be made clear. The results will conflate the effects of changing the amount of training data and the rate of change of the forcing. Perhaps a test could be done for the case with a lower rate of change of forcing with the lower number of time steps used in training to see the effect of changing each aspect of the set up.

In our new forcing experiments, we use a fixed number of training timesteps, so this is not an issue anymore.

24. p.11 L17 It's not clear to me if different training and testing runs are produced for each experimental repeat.

No, only the training is repeated (with exactly the same datasets). We now explicitly mention this.

25. p.13 L2 Given that the performance of the Lorenz systems with fixed F also vary between the situations with different training data lengths, could the different rates of forcing change in each case be affecting the system's behaviour in some important way?

In the new manuscript, we use only the long training length, so this is not an issue anymore.
In the old manuscript it might have had an influence, but since the network was trained on short-term forecasts, even for the shorter training sets the individual samples can be seen as having nearly fixed F, because the forcing varied over a long time period only.

26. p.15 L25 - “layers” should be “neurons”?

Yes indeed, thanks for pointing this out. We changed “layers” to “neurons”.

27. fig.2 - It should be pointed out clearly that the colour scales are different.

This is now pointed out in the caption

28. fig.3 - It's a bit unclear to compare the distributions across panels b and e to see which neurons change behaviour more - it might be better to put the values side by side on the same axes.

Thanks for this suggestion, we have now put the distributions for both wings on the same axis, with different colors.

29. fig.3 – panel references are wrong in the caption. It also needs to be clearer about what the distributions in b and e are.

We changed the figure to accommodate the distribution plots for the now larger network. Also, we now explain the distributions better in the caption (“*boxplots showing the distribution of neuron activations per neuron ….”*)

30. fig.5 - It needs to be made clear that single-timestep prediction errors are being shown. It would be more interesting to see metrics of longer-range forecast skill and how well theattractor is simulated, since in the end single-timestep predictions are not the target and may not indicate that well the performance on longer time scales.
31. fig.5a - I suggest using a different colour for the vertical lines.
32. fig.5 - It would be better to write "Ntrain" in exponential notation.
33. fig.5 - The orders of the Lorenz systems could be reversed so they go start to end, the same as for the neural networks.
34. fig.5 - There are no bars on the simulations with the Lorenz system, so have you used the same Truth runs for each experiment, so you could only run the Lorenz systems once, or are the bars just not visible? It would be good to make small bars visible by giving their ends a width so they stick out.

Answer to all comments above regarding fig.5: We do not use the former fig.5 anymore. We believe that the figures for the new forcing experiments are clearer, and they also include two metrics of attractor reconstruction (mean and standard deviation).

35. References – some references have repeated DOIs.

Fixed

Reviewer #3

General comments:

A first remark is that the first experiment seems quite artificial. The Lorenz 63 model is an extremely idealized model with a very peculiar bifurcation structure and distinct symmetries, making it less than ideal to represent a realistic general circulation model. A more realistic model which also displays regimes such as that of Charney and Straus (1980), would probably have been more suitable for this particular experiment. Of course, the authors mention these caveats, but in the end there are so many caveats that it seems that no conclusion can be drawn at all related to realistic climate models. Moreover, it is not because some insights related to machine learning map to more complex models, that this is a universal property. I'm not sure how the authors can address this issue without performing the analysis for a more realistic model. Nevertheless, I appreciate the value of this experiment, albeit in a more theoretical context of dynamical systems theory.

We completely agree that the Lorenz63 model is a highly idealized system. We have chosen it due to its widespread use in the study of chaotic dynamical systems, and the fact that it is very easy to visualize and inspect. However, we have now also extended our first experiment to the Lorenz95 system. While this system is also highly idealized, one may argue that it is more closely related to geophysical systems. We agree that it would be interesting to extend the analyses to more complex systems such as the Charney and Straus model, or to simple GCMs. However, considering the difficulty (not at least the required computation time), we think that this is not in the scope of the current study, which we designed for the Lorenz systems. We believe that even when focusing only on the Lorenz systems, this study provides valuable information for the use in more complex systems.

Secondly, the result of the forcing experiment of the Lorenz 95 model seems hardly surprising, as the forcing parameter is varied from the periodic regime to the turbulent regime. One cannot expect a neural network to predict qualitatively completely different behaviour.

We agree that our original experiments were rather extreme. We have now completely redesigned the forcing experiments. Instead of relatively arbitrarily picking out a "high" and a "low" forcing, we now trained the networks on 6 different training regimes (3 relatively narrow, and 3 broader forcing regimes). For each training regime, the network is then tested on a wide range of new regimes. (see fig 7 and 8 in the new manuscript). We believe that these new experiments provide much better insights in the abilities (and inabilities) of the networks to learn external forcings.

Finally, it seems to me that similar studies must have been performed in the literature on neural networks, though not necessarily in the context of geophysics. I would encourage the authors to explore the literature on this. The recent groundbreaking success of deep learning was only possible thanks to the move from few to many hidden layers, and it appears that large deep learning networks have better generalization properties
than smaller ones. It would therefore also be interesting to repeat the exercise for a deep neural network. See for example the work by Novak et al. (2018) or Wu et al. (2017) who investigate the source of these generalization properties, and references therein.

We have added a new section in the introduction ("1.2 Related work on generalization properties of neural networks") that gives a better overview of the general literature on generalization properties of neural networks. We have further tested several architectures, including ones with wider layers, in our experiments.

Specific comments and typographical errors:

p.1, L 9-10: In the abstract, the authors conclude that "These results outline challenges for a variety of machine-learning applications. [...]". The word "outline" (in the sense of summarize) goes a bit too far since the results shown are for two highly specific models and a very artificial set-up (an entire wing missing, training in periodic regime). I would just say that the results provide some examples.

We changed it to "point to potential limitations"

Figure 3: Labels in the caption don’t match with the relevant subfigures.

fixed

p. 11, L 16: lorenz -> Lorenz (2x)

fixed

p. 15, L 1: However, also the alternative methods suffer -> However, the alternative methods also suffer

We changed the paragraph to reflect the new results, and the sentence does not appear anymore.

p. 15, L 3: test test -> test

We changed the paragraph to reflect the new results, and the sentence does not appear anymore.

Check spelling consistency: throughout the manuscript, generalize / generalise are both used

we changed all to generalize

References:

Charney J G, Straus DM (1980) Form-Drag Instability, Multiple Equilibria and Propagating Planetary Waves in Baroclinic, Orographically Forced, Planetary Wave Systems. J Atmos Sci 37: 1157-1176.

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, Jascha Sohl-Dickstein (2018) Sensitivity and Generalization in Neural Networks: an Empirical Study. arXiv:1802.08760.

Lei Wu, Zhanxing Zhu, Weinan E (2017) Towards Understanding Generalization of Deep Learning: Perspective of Loss Landscapes. arXiv:1706.10239.

Interactive comment on Nonlin. Processes Geophys. Discuss., https://doi.org/10.5194/npg-2019-23, 2019.

Generalization properties of feed-forward 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~~phase-space. However, many practical applications diverge from this idealised scenario. Here, we investigate the ability of feed-forward 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. Climate science is a real world example where these questions may be relevant: it is concerned with a non-stationary chaotic system, subject to external forcing and whose behaviour is known only through comparatively short data series. Our analysis is performed on the Lorenz63 and Lorenz95 models. We show that for the Lorenz63 system, neural networks trained on data covering only part of the system's ~~phase space~~ phase-space struggle to make skillful short-term forecasts in the regions ~~missed during~~ excluded from the training. Additionally, when making long series of consecutive forecasts, the networks ~~mostly do not~~ struggle to reproduce trajectories exploring regions beyond those seen in the training data~~. We also find that it is challenging for the standard network architectures~~ , except for cases where only small parts are left out during training. We find this is due to the neural network learning a localised mapping for each region of phase-space in the training data rather than a global mapping. This manifests itself in that parts of the networks learn only particular parts of the phase-space. In contrast, for the Lorenz95 system the networks succeed in generalising to new parts of phase-space not seen in the training data. We also find that the networks are able to learn the influence of ~~a slowly changing~~ an external forcing, ~~highlighting the limitations of a network trained on a specific forcing regime for generalising~~ but only when given relatively large ranges of the forcing in the training. These results point to potential limitations of feed-forward neural networks in generalizing 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.~~ given limited initial information. Much attention must therefore be given to designing appropriate train-test splits for real-world applications.

1 Introduction

1.1 Neural networks for weather and climate applications

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~~ forecasting 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~~ forecast 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~~ studied 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~~ post-processing of numerical weather forecasts (Rasp and Lerch, 2018), empirical error correction Watson (2019), predicting weather forecast 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), as well as doing actual weather forecasts (Weyn et al., 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 focus specifically on the widely used feed-forward neural networks and address two open questions related to ~~using neural networks~~ their use for approximating the dynamics of chaotic systems:

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

2) Can neural networks "learn" the influence of an 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~~ We specifically use the Lorenz63 (**?**) and Lorenz95 ~~systems (Lorenz, 1996) and simulating external forcing as a slow change in the parameters of the models. Both~~ (**?**) models. These (and other variants of the Lorenz95 system) are widely used as toy-models for studying atmosphere-like systems, also in the context of machine learning (e.g. Vlachas et al. (2018); Watson (2019); Lu et al. (2018); Chattopadhyay et al. (2019)) and parameter optimization (e.g. Schevenhoven and Selten (2017)).

Both the questions we raise 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~~This is particularly true for the ocean, which has much longer characteristic timescales than the atmosphere, and for applications to paleoclimatic variability. Moreover, 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?

1.2 Related work on generalization properties of neural networks

The question of generalization is a central aspect in machine-learning, and is a well studied topic for neural networks (e.g. Hochreiter and Schmidhuber (1995); Hardt et al. (2015); Zhang et al. (2016)). One of the remarkable properties of deep neural networks is that, contrary to statistical learning theory, in many cases they generalise better when having more free parameters. The recent success of deep neural networks in a variety of applications and their empirically demonstrated generalisation abilities have stimulated investigations into the underlying mechanisms. For example, (Wu et al., 2017) argued that the reason for their good generalization properties are the landscape characteristics of the loss function. Novak et al. (2018) argue that generalization is favoured by high robustness to input perturbations of the trained networks in the vicinity of their training manifold, despite their large numbers of parameters. Another well-studied aspect is machine learning under covariate-shift – the situation where the probability distribution of training and test data is not the same (e.g. (Sugiyama and Kawanabe, 2012)). This amounts to a special class of non-stationarity problems, and is partly related to our question 2 (learning external forcings).

The bulk of the literature on the above topics has focussed on image recognition and related fields, and the extent to which these results may apply to dynamical systems is unclear. To the authors' knowledge, the generalization properties of neural networks applied to dynamical systems, and specifically to Lorenz systems, are yet to be studied in detail.

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~~phase-space:

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

Even though mathematically equivalent, the latter would imply that different parts of the network are responsible for different regions of the phase-space. 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~~ phase-space that are not covered in the training data.

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~~Lorenz models, 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~~ sets cover all regions of the ~~phase space~~ phase-space with some reasonable density. That is, no large contiguous regions of the ~~phase space~~ phase-space are left out of either set of data. (e.g. Pasini and Pelino (2005); Vlachas et al. (2018)).

Here, we consider the opposite situation, namely a scenario where the training data covers only part of the system's ~~phase space~~phase-space. We know from the definition of the Lorenz63 ~~model~~ and Lorenz95 models 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~~ 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. In a climate science context, this would for example be relevant for the ocean. The latter's long characteristics timescales imply that observational datasets may cover only part of the phase-space. It is also relevant in forecasting extreme events in the atmosphere.

Question 2) relates to how well a network can learn the influence of a slowly varying variable (the "forcing" in a general sense) on the evolution of the fast-varying variables (the system state). The influence of the slowly-varying forcing on the short-term dynamics is potentially very small compared to the typical variability of the systems, making the task of learning simultaneously the dynamics and the influence of the external forcing challenging, even when the forcing is provided as additional input to the network.

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, see Fig. 1a). We integrate the system with a timestep of $t = 0.01$ with the LSODA solver from ODEPACK (Hindmarsh, 1983) as provided by scipy (Jones et al., 2001). While the Lorenz63 model is a very rough approximation of atmosphere-like dynamics, the fact that it ~~has only~~ only has 3 variables allows to easily visualize the complete ~~phase space~~ 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).

The Lorenz95 model ((?), also often referred to as the Lorenz96 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. An example of a Lorenz95 model integration is shown in the left panel of Fig. B1 d. Note that there is also a second model often referred to as the Lorenz95 or Lorenz96 model, which uses a second (and sometimes a third) dimension. This model is not considered here. 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~~

For the Lorenz63 ~~system by Zhang (2017)~~model we use fully-connected networks with ReLu activation functions in the hidden layers and a linear output layer. The main configuration used in this study was determined via a tuning procedure (Appendix A). It consists of 2 hidden layers with 128 neurons each. 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~~Overfitting is controlled via an early-stopping rule. The training is stopped when the skill on a validation dataset (last 10% of the training set) has not increased for 4 training epochs, with a maximum of 100 epochs. 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. No regularization techniques are used. Part of our experiments are repeated with the same architecture,

but trained on forecasting the tendency (difference between the following and current states) rather than the following state directly.

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~~B. 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~~B) worked best for virtually all lead-times (see ~~fig. A1~~Fig. B1), 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. As for the Lorenz63 model, the network directly forecasts the next state of the system. Overfitting is controlled via an early-stopping rule. The training is stopped when the skill on a validation dataset has not increased for 4 training epochs, with a maximum of 30 epochs. No regularization techniques are used.

3.4 Evaluating the reconstruction of the Lorenz63 attractor

~~The neural networks is~~ In most of our experiments, the neural networks are trained by minimizing errors of single-step (and thus short-term) forecasts. Therefore, ~~it might be that the network cannot~~ they may not always 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~~For the Lorenz63, the trained network often made very good short-term ~~predictions~~forecasts, 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 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~~ train 10 networks, and then select the network that best reproduces the attractor when started from a random point in the training dataset. This is evaluated via comparing the reconstructed attractor to 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 isusing:

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

where $\rho_{i,j,k}$ is the density of discrete data points in the gridbox i, j, k. As threshold for $rmse(\rho)$ 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. The gridboxes have size $0.3 \times 0.3 \times 0.3$ on the normalized domain (normalization based on the training set, the output of the networks is always in the normalized domain).

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 were to learn 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 systemwe 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.

4 Experiments and Results

4 Reconstructing Lorenz systems using only part of the phase-space

4.1 Reconstructing Lorenz 63 with Neural Networks

We train a network

4.1 Lorenz63

4.1.1 Training the networks

We first verify that our networks can successfully reproduce the Lorenz63 attractor given training data from across the system's phase-space. We train 10 networks 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~~ networks are 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. ??~~(via feeding the forecast back into the input). Figure 1 a,b shows the training data and the attractor reconstructed by the neural network. The network attractor ~~looks visually reasonable: it has~~ reproduces 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~~ Its main deficiency is that the inner regions of the wings are slightly 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). To put forecast errors throughout the paper in context, panel d) shows the tendency (change over 1 timestep) of the model in different phase-space regions.

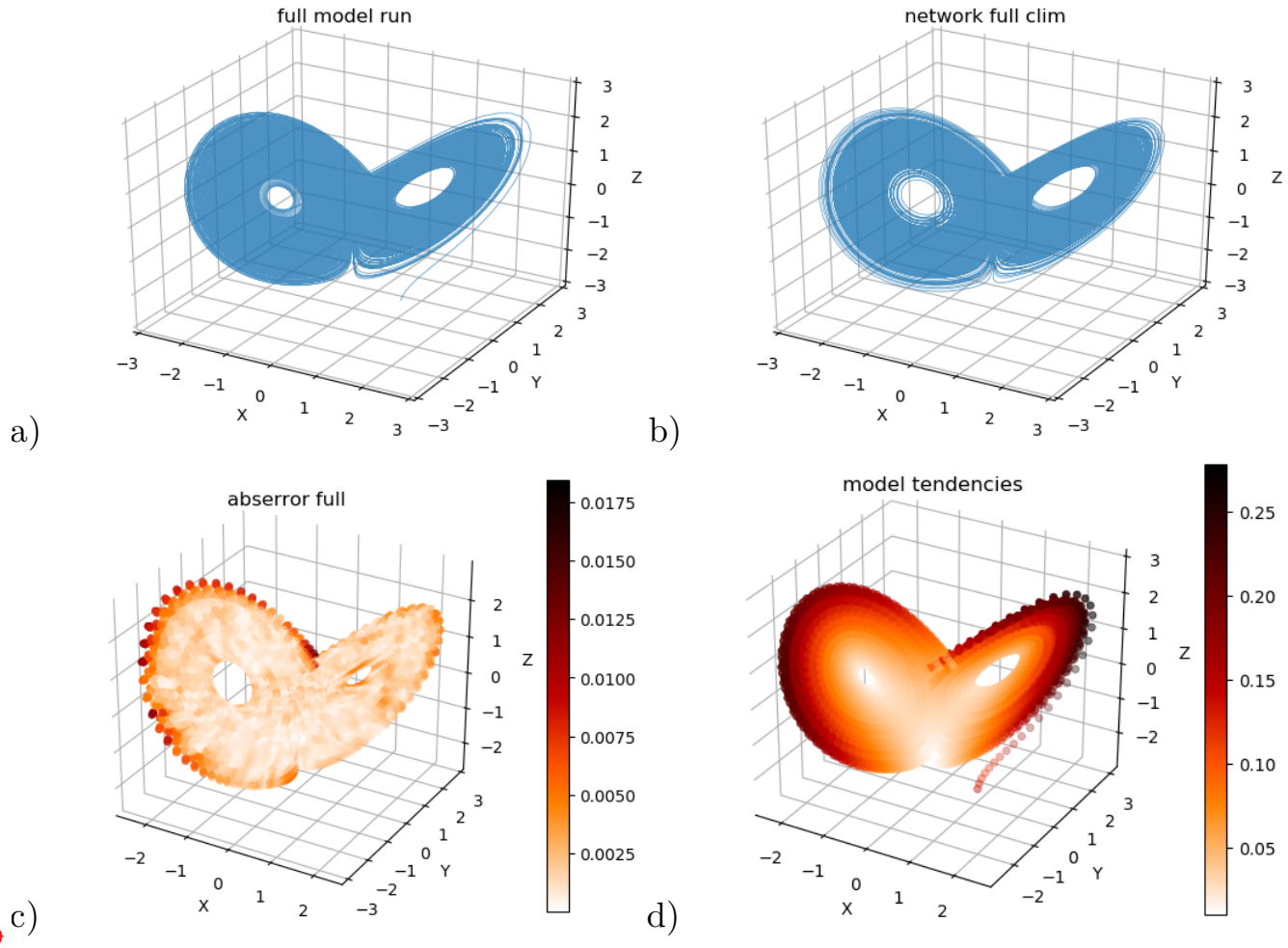
4.2 ~~Training on incomplete data~~

~~Here we take a drastic approach towards~~ We next consider the question of training on incomplete data~~: namely, we select training~~ . We take a somewhat drastic approach and we select data that explores only limited regions of the phase-space. This selection is done via "cutting out" ~~chunks out~~ contiguous regions of the phase-space. Since the training is done ~~in~~ on data 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 (this is necessary because we removed parts of the model's trajectory). First, we investigate whether neural networks trained on different ~~phase space~~ phase-space regions are able to make short-term ~~predictions~~ forecasts in other parts of the attractor. Then, we assess whether it may be possible to reconstruct the full attractor with these ~~neural~~ networks.

4.1.1 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. ??~~ 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.5 f,i~~see Sect 4.1.3). 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~~ and does

~~a) b)~~



~~c)~~

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. d) tendencies for 1 timestep ($t_{i+1} - ti$). Note different colorscales in c) and d).

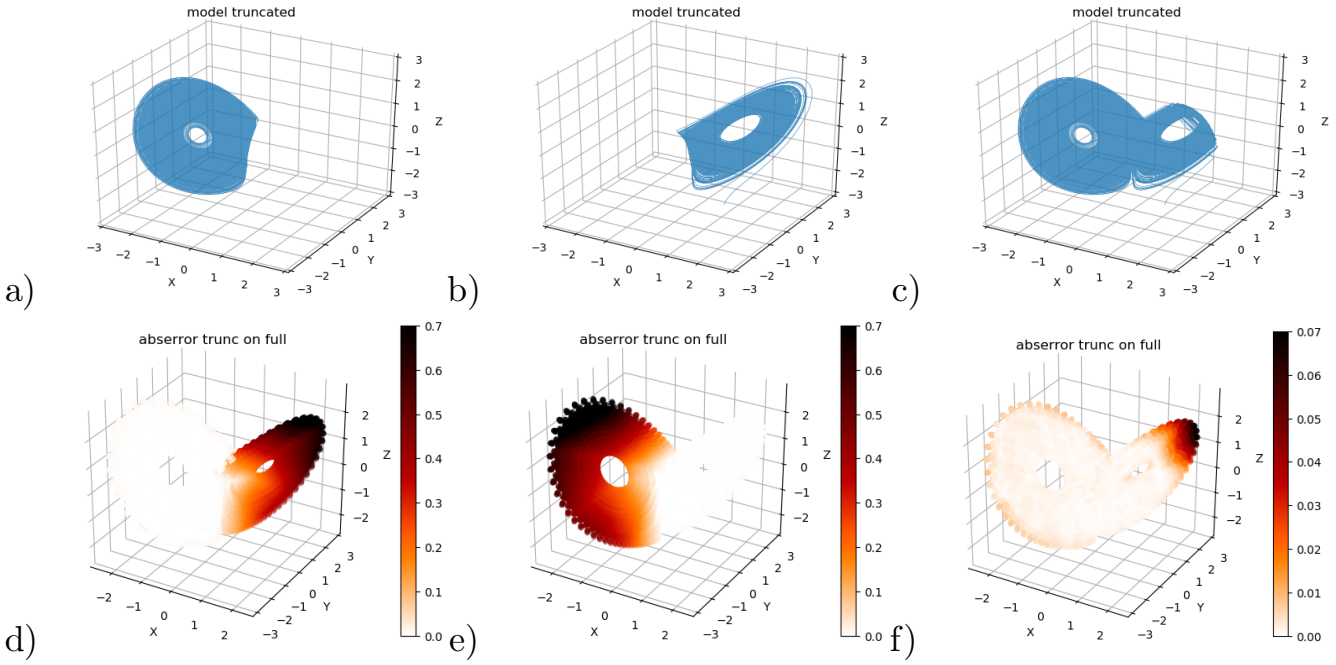


Figure 2. Truncated sets of Lorenz63 training data (a-c) and short-term forecast error (MAE) of neural networks trained on these sets (d-f). Note the different colorscales in (d-f).

not systematically point back to the region seen in the training. Nonetheless, the forecast errors in the "missing wingtip" are roughly an order of magnitude higher than in the regions included in the training (~~fig~~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 results are similar when using networks that forecast the tendency only instead of the following state (Fig. C1). The main difference is that when training on only one wing, the error in the other wing is roughly halved relative to Fig. 2, albeit still orders of magnitude higher compared to training on the whole attractor. When initializing forecasts in the left-out wing, the trajectories are unstable and drift outside of the training domain (not shown). In this respect, the architecture that forecasts tendency is doing even worse than the architecture forecasting the state. The simplicity of the system allows us to examine ~~this behaviour~~ the above results further by looking at the activation of the individual neurons in the network. For this, we inspect a network that was trained on the whole attractor (and provides good forecasts on the whole attractor).

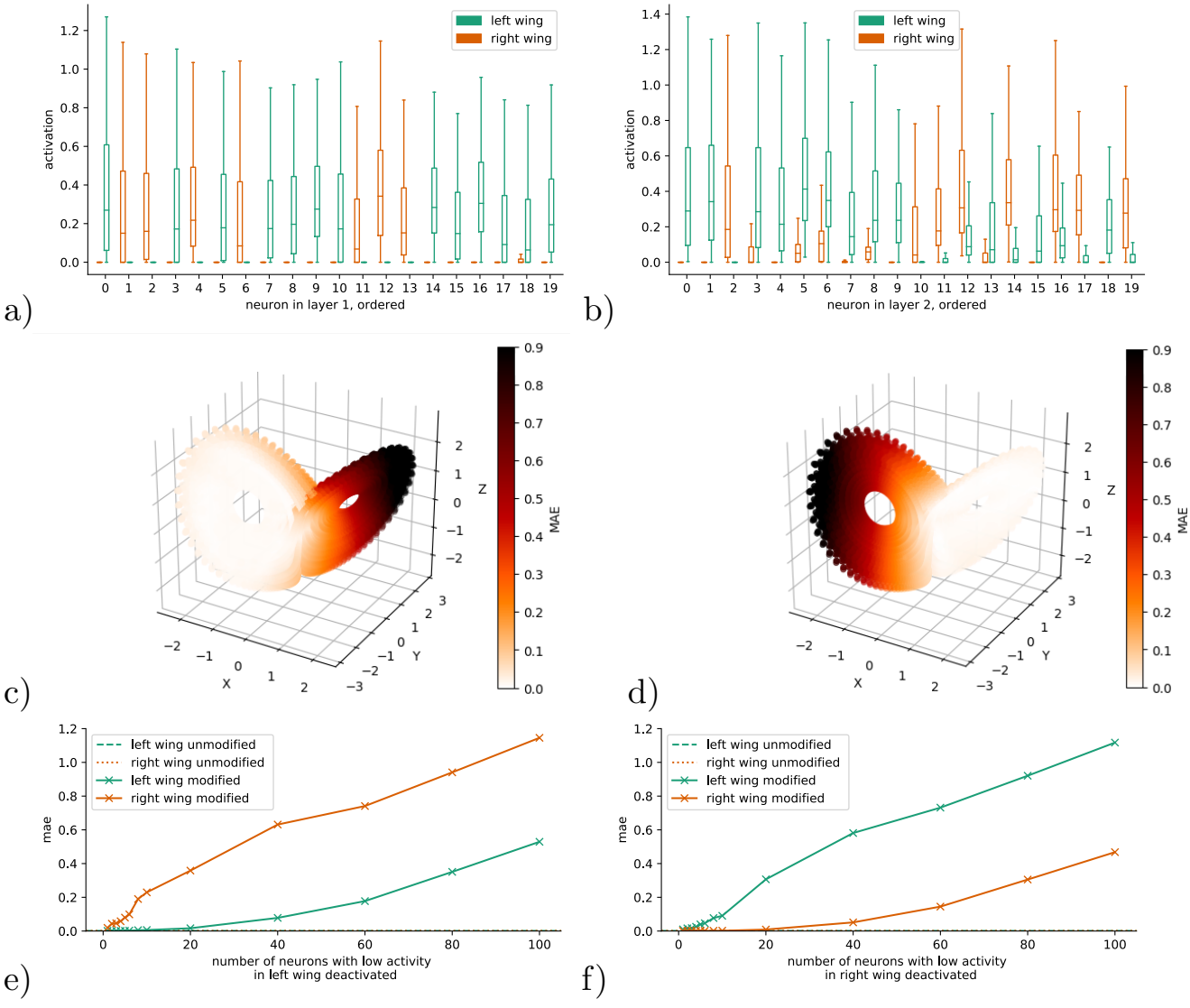
~~a)b)c)~~

~~d)e)f)~~

Figure 3a,b 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~~green) and from the right wing only (~~d,e,f). The numbering of the neurons is arbitrary, but is consistent within the figure~~orange). Shown are the 20 neurons with the largest absolute differences in the standard deviation of activations in the two wings. The distribution of activations for all neurons (without specific ordering)

is shown in Fig. C2 in the appendix. Some neurons have very similar activations in both wings, whereas the 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. This would mean that the neurons learned to correctly map the system in one wing – and are thus active and contributing to the forecasts in that wing – but they are inactive in the other wing (i.e. do not contribute to the forecasts). To test this, for each layer we identify the ~~neuron~~ n neurons with least spread in activation (defined here as ~~standard-deviation~~ 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~~ neurons in turn at their mean activation level for the relevant wing. Note that there are also some "dead" neurons, which always have zero activity in both wings. These we ignore. With this modified network, we make ~~predictions~~ forecasts on the whole attractor. The result for $n = 20$ is shown in ~~the right-hand column of fig. 3 . The main~~ Fig. 3 c,f. The effect of fixing the ~~output of the neuron that has~~ activations of the neurons that have low spread in the left wing is that the forecast error in part of the right wing increases sharply~~. A similar behaviour is seen~~ , whereas the error in the left wing is nearly unchanged. The same is seen for forecasts in the left wing when fixing the ~~activation of the neuron that has~~ activations of the neurons that have low spread in the right wing. The structure of the errors is very similar to that of the networks trained only on one wing (Fig. 2). Panels 3 e,f give a more systematic overview. They show the forecast errors of the modified networks on the left wing (green) and the right wing (orange), when fixing the 1,2,..100 neurons the have 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~~ left and right wings, respectively. When fixing the left wing "low-variance" neurons, the error in the right wing increases with even a single deactivated neuron, and rises monotonically with every additional deactivation (Fig. 3 e). In the left wing, on the other hand, the error stays very close to the error of the unmodified network, and only starts to increase beyond 20 deactivated neurons. Corresponding results are found when fixing low-activity neurons in the right wing (Fig. 3 f).

The above suggests that these roughly 20 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 have very low variance in the same wing~~ This test was repeated for different network architectures (different number of hidden layers, and different hidden layer sizes). In all we tested 20 different architectures (smallest: 1 hidden layer with 8 neurons, largest: 8 hidden layers with 128 neurons each). The result for eight of these architectures (ranging from shallow networks with narrow layers to deep networks with wide layers) are shown in Fig. 4. The behaviour is very similar to that seen in Fig. 3, except that in some cases the error does not grow monotonically with increasing number of deactivated neurons. The results for the additional architectures we tested were similar (not shown). ~~Thus, the result that part of the network learns a specific part of the phase-space may apply~~



~~a)b)c)d)c)f)~~

Figure 3. ~~a,b: what we refer to as "left" and "right" wings of~~ Boxplots showing the ~~Lorenz63 attractor. c,d:~~ distribution of neuron activations per neuron for ~~the two wings. e~~hidden layer 1 (a) and hidden layer 2 (b), ~~f: short-term~~ split by wing (color in plot). Short-term forecast errors (MAE) for the networks ~~with~~ in which in each layer the activation ~~levels~~ level of the 20 neurons ~~7 and 5 ("low-variance neurons")~~ with lowest variance are fixed ~~everywhere~~ at ~~their~~ its mean ~~values~~ value for the left (c) and right (d) wings. Short-term forecast errors of the network with 1–100 low-variance neurons per layer in the left (e) and right (f) wings deactivated, ~~respectively~~split up by wing (solid lines). The dashed lines show the forecast errors of the unmodified network.

~~to both individual or multiple neurons~~The only exception is the deepest architecture with very narrow layers (8 layers with 8 neurons each), in which deactivating a single low-activity neuron per layer degrades forecasts in both wings (not shown).

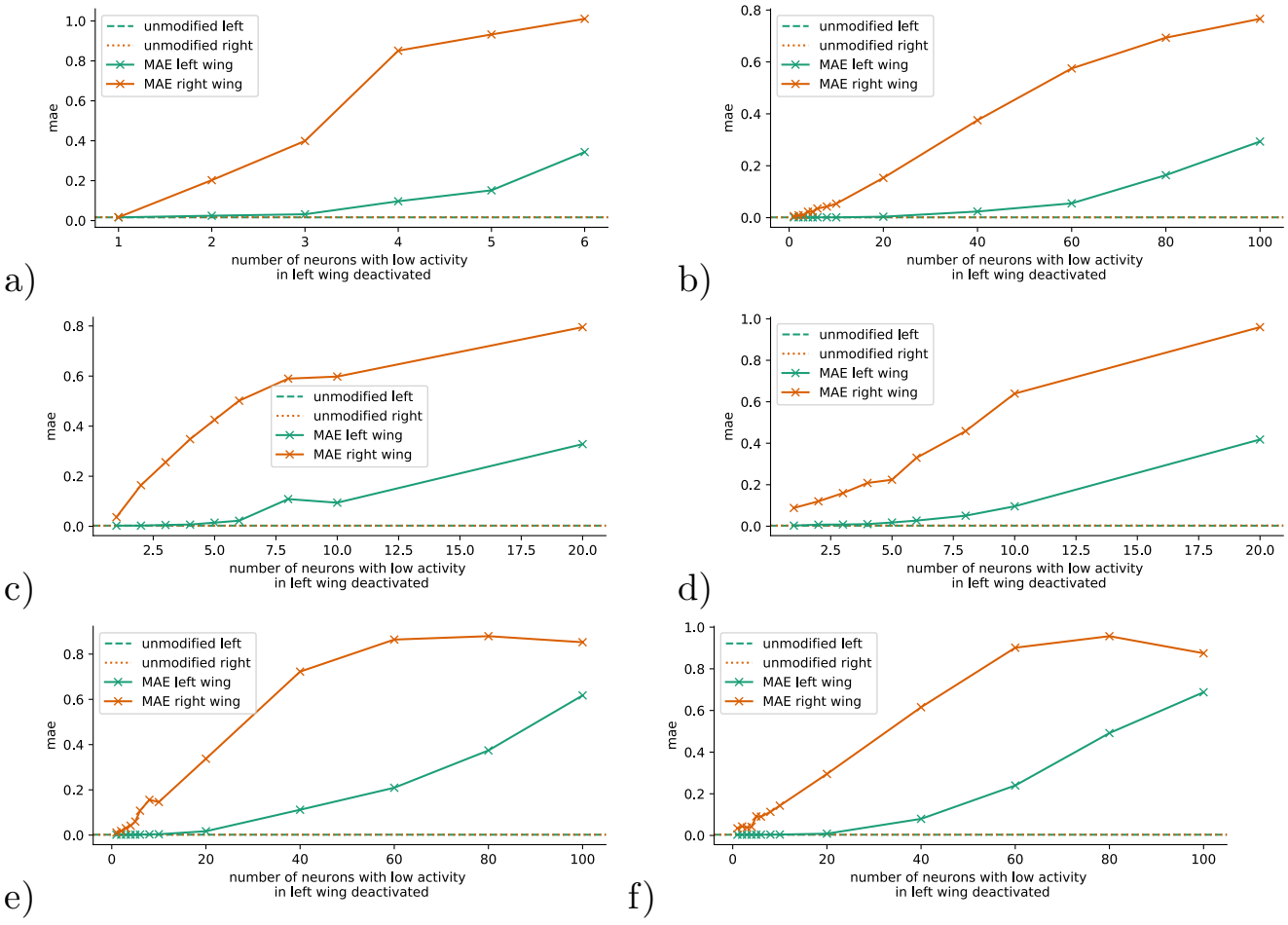


Figure 4. As Fig. 3e, but for different neural network architectures. a) shallow and narrow (1 hidden layer with 8 neurons); b) shallow and wide (1 hidden layer with 128 neurons); c) intermediate (2 hidden layers with 32 neurons each); d) deeper intermediate (4 hidden layers with 32 neurons each); e) deep and wide (4 layers with 128 neurons each); f) very deep and wide (8 layers with 128 neurons each).

4.1.2 Reconstructing the full attractor

We next attempt to use ~~the networks outlined in Sect. 4.1.1~~ neural networks trained on incomplete data to reconstruct the full attractor~~by making a long series of iterative forecasts with the networks~~. We already showed in Sect. 4.1.1 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~~Fig.5 a), the networks are able to reproduce a reasonable attractor regardless of whether they are selected using the ~~density selection criterion (fig~~density-selection (Fig.5 b) or the density-full ~~criterion (fig~~approach (Fig.5 c) – see Sect. 3.4. ~~In this case~~As could be inferred from the short-term forecasts, the neural networks are thus able to explore ~~also the~~ regions that are not ~~explored~~ visited by any of the trajectory segments in the training data. However, networks trained on single wings fail to reconstruct the full attractor (~~fig~~Fig.5 e, ~~f~~h), 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 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~~Fig.5 f, i), implying that the network reproduces a dynamics that populates only the wing that was included in the training.

4.2 ~~Learning external forcings of the system~~Lorenz95

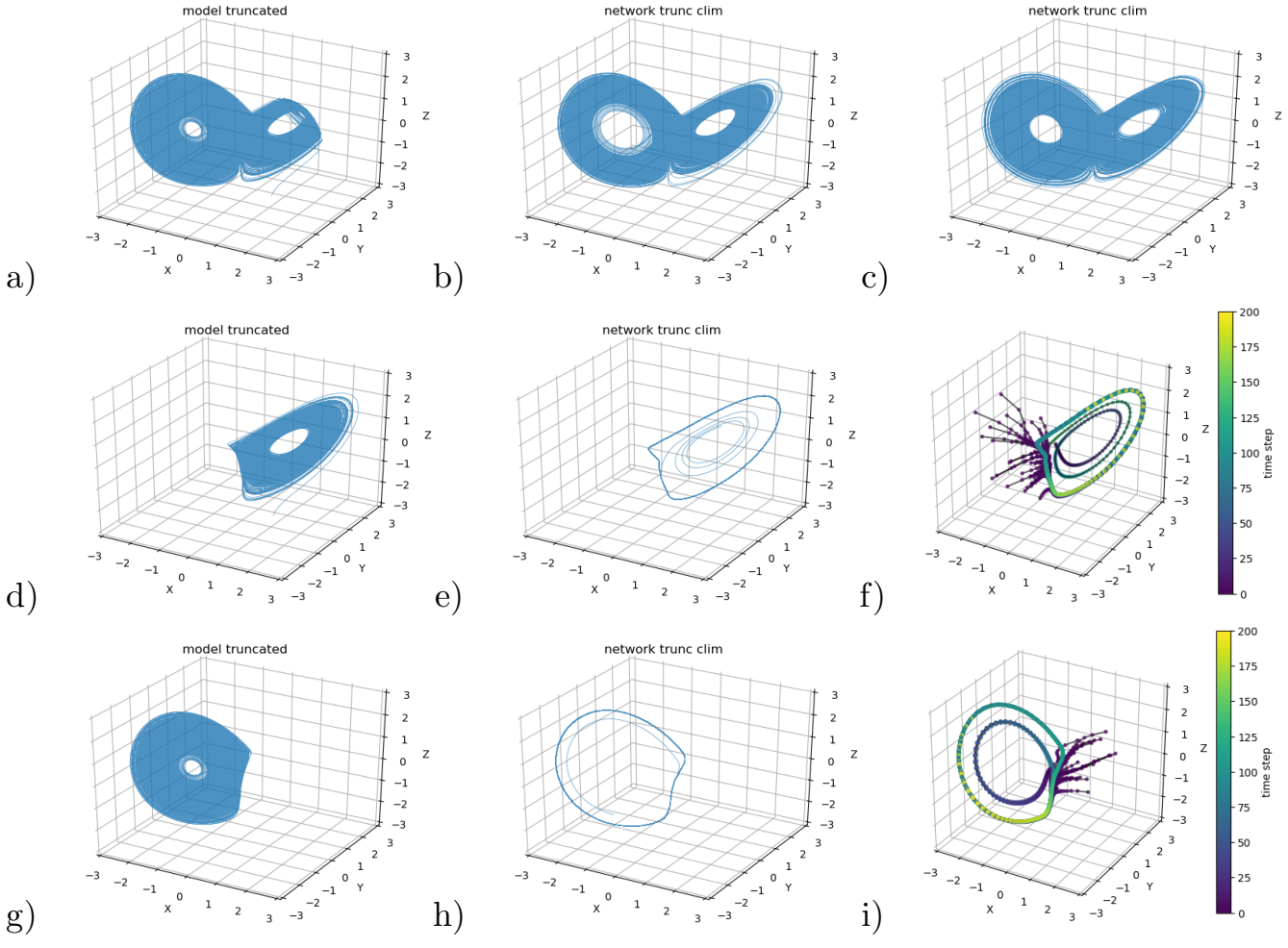
~~We next address the second question raised in the introduction: can our neural networks learn the influence of slowly varying external forcing on the system? We explore this on the Lorenz63 and the~~ In the Lorenz95 ~~systems.~~

4.2.1 ~~Lorenz63~~

~~As external forcing scenario we consider a gradual linear increase of the σ parameter (eq. 3). This may be conceptually likened to~~ system, which in our setup has a dimensionality of 40, it is harder to define reasonable regions of phase-space to be excluded from training than in the 3-dimensional Lorenz63 system. A logical step to tackle this problem would be to use a method like principal component analysis to reduce the dimensionality of the system before partitioning its phase-space. However, the leading principal component of the Lorenz95 system can only explain 8 % of the variance (not shown), meaning that it is not possible to reduce the system to a small number of principal components while still capturing most of its variance. A different approach is to look at Poincaré sections. These are 2-dimensional projections of the phase-space spanned by two variables, often used in the analysis of dynamical systems. While this approach seems intuitive, it is problematic in our context. If we define a region of the phase-space to leave out of the training (by defining a region spanned by 2 variables) we can cut out all states of the model run that fall within these regions. However, if there were identical states to these, but shifted one or more gridpoints, then these states would not be excluded. The symmetry of the system (which also translates to the symmetry in the circular convolutional network architecture used), implies that the network can forecast states excluded from 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~~training data without learning any extrapolation, as long as (near-)identical but shifted

~~a)b)c)~~

~~d)e)f)~~



~~g)h)i)~~

Figure 5. 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~~ density-full (b) and ~~density-full selection~~ density-selection (c) ~~criteria~~approaches. 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.

states are seen while training. Indeed, due to the circular convolution, original and shifted states are equivalent for the network. Based on these considerations, we use another method to define Poincaré sections of the Lorenz95 system. We first transform the system states to spectral space with a Fast Fourier Transform (FFT). 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~~ compute the amplitude of each wavenumber (absolute value of the complex wavenumber coefficients), thus removing all information about the position of the waves. We next find the pair of wavenumbers whose amplitudes have least correlation, and define a Poincaré section based on these.

Since the Lorenz95 model is very cheap to run, we can also – in analogy to the Lorenz63 experiment – define a phase-space region via setting a certain range for all 40 variables. Due to the low density of data points in such a high-dimensional space, this would exclude only very few points from our standard 1e5 timesteps run, and ~~vice versa (trainingon final period and testing on initial period). We also use the networks~~ likewise, only few points in the test set would lie in this region. Therefore, for this approach we generate an additional test-set. We run the model until we have 1e3 points that lie in the region cut our from training. Due due to the symmetry considerations mentioned above, we do this in the 20-dimensional space of absolute wave-number coefficients.

To implement the first method we "cut out" squares of the spectral Poincaré section, and train a network on the rest of the data. We then use the network to forecast the whole attractor on a test set, and compare it to the skill of the same network trained on the ~~initial and final periods of a training run to forecast~~ whole attractor (which has good forecast skill, see Appendix B and Fig B1). Each training is done 10 times, and the forecast errors averaged over these 10 realizations. The results are shown in Fig. 6a,b. The short-term forecast errors in the left-out region are indistinguishable from the errors in the other regions, meaning that the network does succeed in generalizing to regions not seen in training. This is also the case for other choices of left-out regions (not shown).

For the second method, we remove all training points that lie within than range $[0, 10]$ for every wavenumber. Again, the ~~same periods of the testing run. We do this with: 1) our standard network configuration, as described in Sect.~~ experiment is repeated 10 times. The result is shown in Fig. 6 c,d. Again, the short-term forecast errors in the region left out in the training are indistinguishable from errors in other regions. Also, the difference between the errors of the network trained on all points and those of the network trained on the truncated set is smaller than the difference between different training realizations (not shown). Finally, we test whether a long run of 1e4 consecutive NN forecasts explores the regions of phase-space left out from the training data. The runs were intialized from a random state of the test set not lying in the left-out region. For all 10 trained networks, the runs did explore the left-out region (not shown).

5 Learning external forcings of Lorenz Systems

5.1 Lorenz63

As external "forcing" scenario we consider a gradual linear increase of the σ parameter (eq. 3). We train the network architecture using σ as input (see section 3.2~~and 2)with the same configuration, but using~~) on Lorenz63 runs with 1e5 timesteps, with

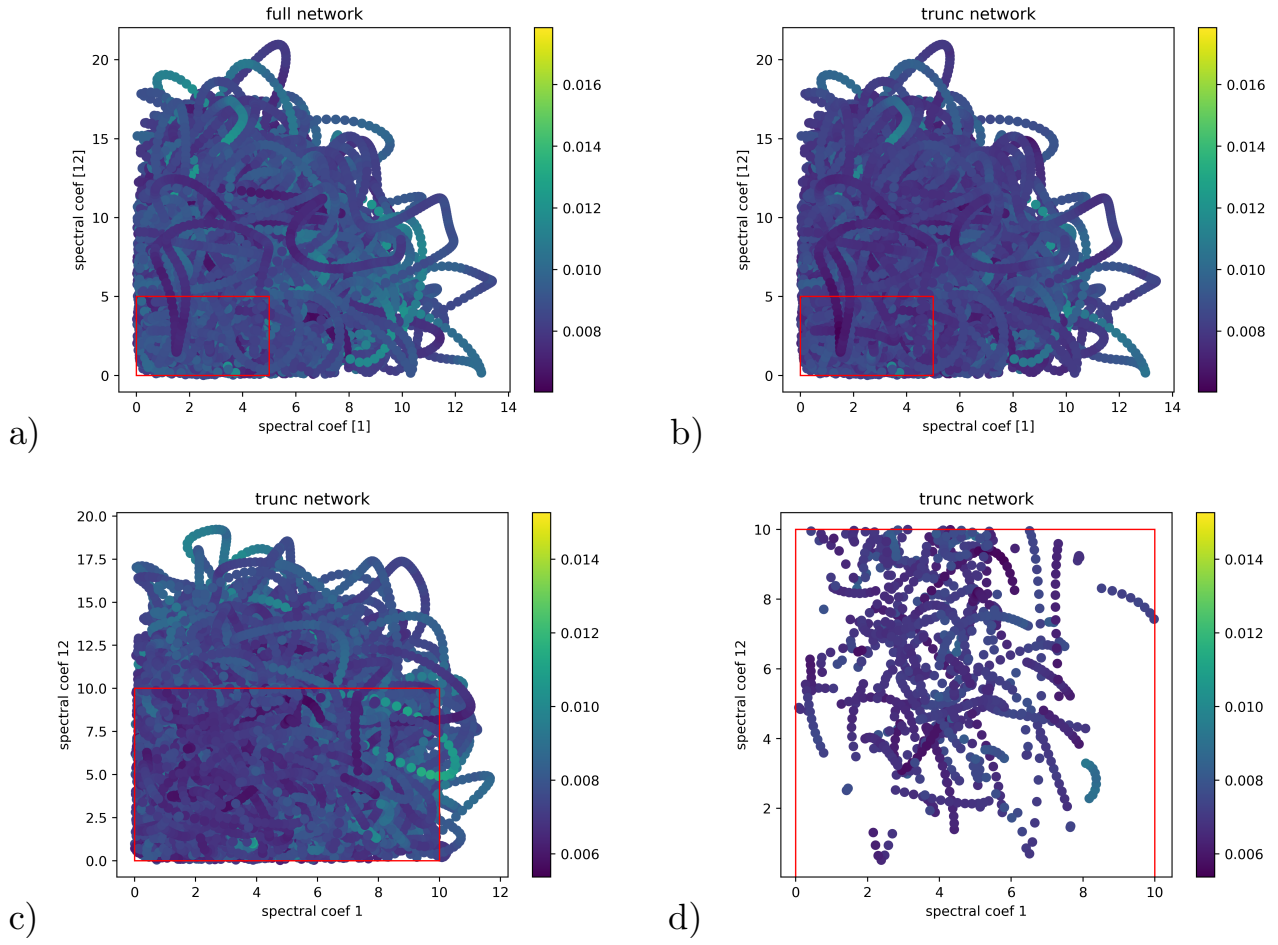


Figure 6. Networks trained only on part of the Lorenz95 phase-space. Short-term forecast errors of the network trained on full training set (a) and a truncated set selected on a Poincaré section in spectral space (b), projected onto said Poincaré section. The rectangle denotes the region of phase-space left out from training. c,d) Short term forecast errors of network trained on a truncated set selected on all 20 spectral components. c) shows all points in the test set, d) only the points that lie in the region cut out from training.

linearly increasing σ ~~as additional input to the network. As baseline forecast for the both the initial and final periods, we consider the~~ over the whole run. We perform 6 different runs, encompassing different σ-regimes regimes: two runs in a low (varying σ from 7 to 8 and 6 to 9), two in an intermediate (10 to 11 and 9 to 12), and two in a high regime (12 to 13 and 11 to 14). The networks are then evaluated on a set of 10 Lorenz63 ~~system with~~ test runs (length 1e5 timesteps) with σ ~~fixed to the mean value from the initial and final periods of the runs . This provides for each period four networkforecasts (standard~~ fixed at 4, 5, 6, 7, 8, 8.5, 9, 10, 12 and 14, respectively. In addition to the main network, two references are used. Firstly, a network trained on ~~same period, standard network trained on other period, network including~~ the Lorenz63 run with varying σ, but not using σ ~~trained on same period, network including~~ as input (termed "no input"). This network is then evaluated on the above fixed σ ~~trained on other period) and two baseline measures (lorenz system with~~ runs. Secondly, for each run with fixed σ, an identical run but with different initial conditions is made. Then, a network not using σ as input is trained on the latter run, and evaluated on the former run with the corresponding fixed σ ~~from same period, lorenz system with~~ (termed "fixed σ~~from other period). These experiments are~~"). The short-term forecast quality is assessed by initializing one-step forecasts from every state in the test runs, and computing the MAE. Each experiment is repeated 10 times~~and the mean forecast skill as well as the standard deviation for the individual networks is computed~~, using the same training and test data, to capture potential influences of random components in the training.

The results are shown in fig. **??**~~. For the run with 5e5 timesteps (resulting in 1e5 training timesteps), using~~ 7. Each panel represents a certain training range in the forcing (indicated by the grey area), and the lines show the MAE of one-step forecasts. The "fixed 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~~" networks (green lines) can be seen as a an upper baseline, as their skill is that obtained when training in the same forcing regime as used for evaluation. It is not expected that the main network (the one using σ ~~in the case of networks~~ as input and trained on the ~~final period is very large (Fig. **??**b).~~

~~This changes somewhat when using the run with 5e6 timesteps, which has 10 times more training samples (fig. **??**c) . Including~~ run with linearly increasing σ) would do better than this reference. The "no input" networks (yellow lines) can be used as a lower baseline, as this should be the skill that can be achieved without having any knowledge of the changing forcing. When trained on the narrow forcing regimes, the networks have trouble making good forecasts outside the training regime. The forecasts are indeed so poor that even the "no input" networks outperform them. In other words, the additional information provided by 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-~~actually leads to a deterioration in skill. This changes slightly when training on broader regimes. Here, the forecast errors of the networks using 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~~as input are similar to the "no input" networks, although in most cases they are still far from matching the "fixed σ" networks.

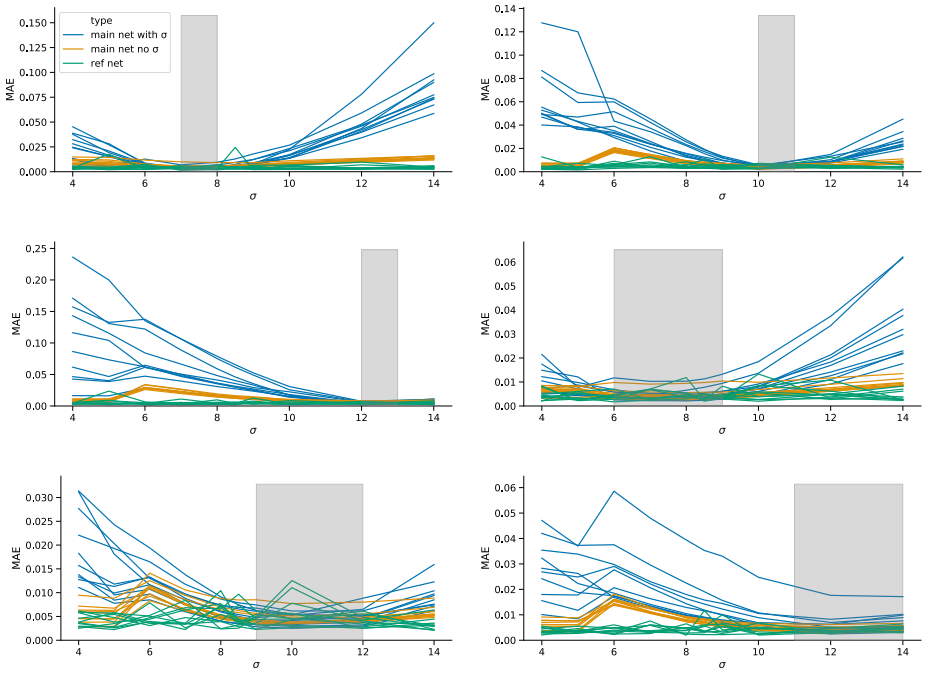


Figure 7. ~~Experiments simulating external~~ Short-term forecast errors for "forcing" experiments with ~~varying σ in the~~ Lorenz63~~system~~. Networks are trained on a ~~) Forcing vs~~ run with 1e5 timesteps ~~. The vertical lines indicate where~~ with linearly increasing parameter σ from the ~~"start" part (first 20%) ends and where~~ lower to the "upper end ~~" part (last 20%) starts. b,c) Mean absolute error~~ of the ~~network forecasts and of~~ grey shaded area in the ~~baseline forecasts (Lorenz63 equations~~ panels, and then tested on runs with fixed σ~~) for runs with 1e5 and 1e6 training timesteps, respectively~~. The ~~left group of bars denotes~~ blue lines show the ~~skill on forecasting on~~ errors for networks that include σ as input. The yellow lines show the networks without σ as input ("~~start~~no input" ~~part,~~ networks in the ~~right bars~~ text). The green lines show reference networks without σ as input that are trained on runs with σ fixed to the same value as the test runs ("~~end~~fixed σ" ~~part. Some of the forecast errors are so small that the bars are barely visible. The black bars show~~ networks in the ~~standard deviation over 10 iterations of each experiment~~text).

5.1.1 ~~Lorenz95~~

5.2 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 setup is analogous to the Lorenz63 ~~scenario, with~~ forcing experiment, but here we change F ~~substituting~~ instead of σ ~~and~~ . With $F = 4$, the system shows periodic behaviour; as F increases, the system becomes more and more turbulent. We consider two low (varying F ~~varying~~ from 5 to 6 and from 4 ~~(periodic regime)~~to ~~16 (highly turbulent regime).~~ 7), two intermediate (8 to 9 and 7 to 11), and two high forcing regimes (12 to 13 and 11 to 14). The runs are evaluated for F fixed at 4, 5, 6, 7, 8, 8.5, 9, 10, 12 and 14. In addition to evaluating short-term forecast performance as in the Lorenz63 forcing experiment, for the Lorenz95 we also asses the ability of the trained networks to reconstruct the "climate" (or attractor) of the model by making a 1e5 timesteps climate run with the network, and then computing the mean and standard deviation of the run (averaged over all gridpoints).

The results are shown in ~~fig. **??**. The networks perform considerably worse~~ Fig. 8. Each row represents a specific training range 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 networksappear to be heavily dependent on the length of the training data. For intermediate training length (fig. **??** c), including~~ (indicated by the grey area). The left panels show the MAE of short-term forecasts, while the right panels show mean and standard deviation of the reconstructed climates, as well as mean and standard deviation of the Lorenz95 model. Each line represents on of the 10 runs made for each experiment. For the three experiments that are trained on narrow forcing regimes (5 to 6, 8 to 9 and 12 to 13), the main networks do not seem able to learn the influence of the forcing and extrapolate to new regimes. In all experiments, the main network has much higher short-term MAE than the "fixed F ~~provides a clear benefit for forecasting the forcing ranges the network was not trained on . However, for shorter or longer trainingsets, this improvement vanishes or even reverses (fig.**??** b, d).~~" networks. When trained on the low or middle regimes, the forecasts are even worse than those of the "no input" networks . As for the Lorenz63, the additional information provided by the forcing term therefore leads to a poorer performance of the network. This picture changes when training on broader forcing regimes (lower 3 rows in Fig. 8). Even though there is a large variation between the individual training realizations of the main network, both the ones trained on the high and on the intermediate forcing regimes outperform the "no input" networks. This implies that, given a wide enough forcing regime in the training, the network is able to learn – at least part of – the influence of the forcing on the dynamics, and extrapolate this influence to new forcing regimes

6 Discussion and conclusion

In this study, we explored how well feed-forward neural networks can 1) generalize the behaviour of a chaotic dynamical 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

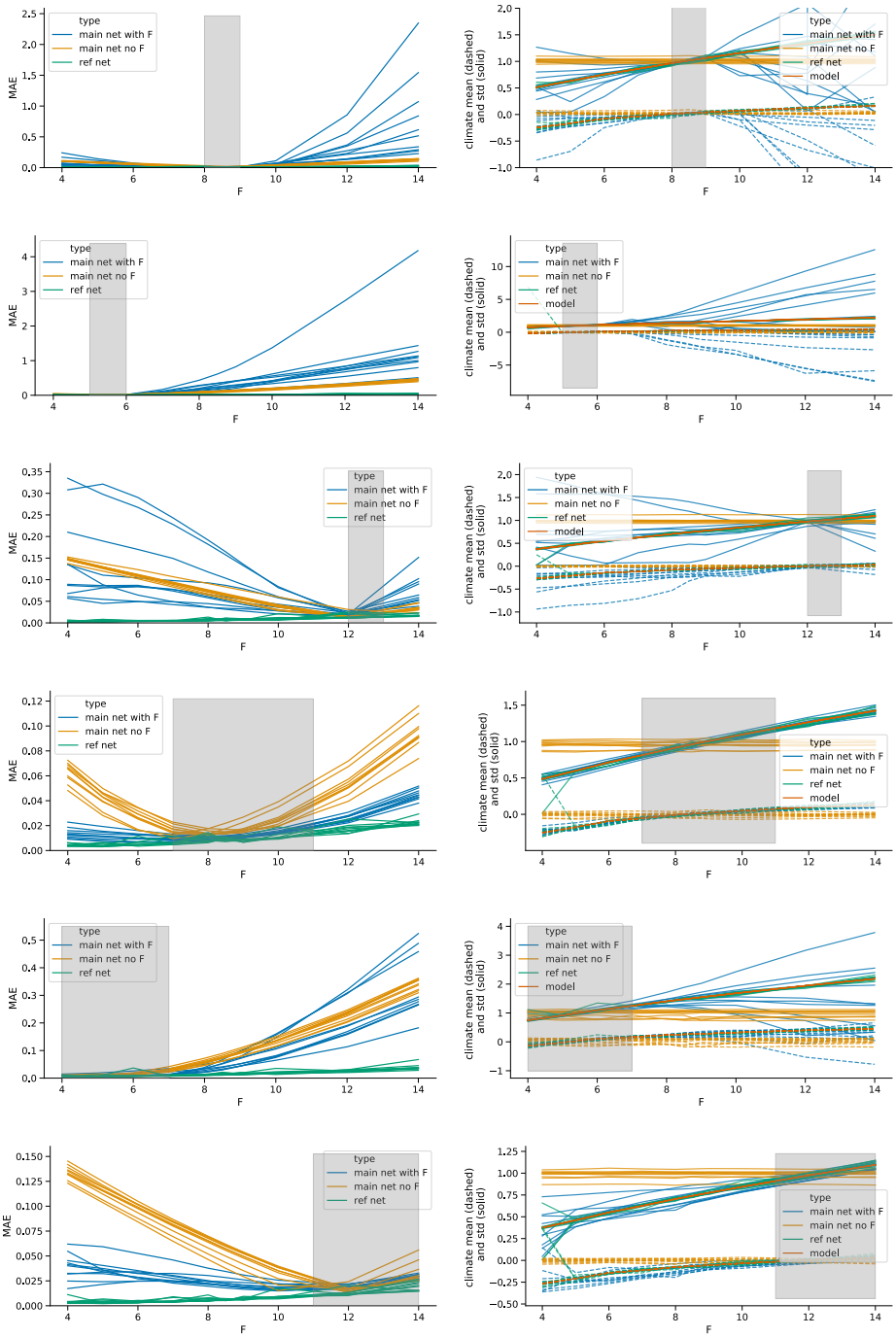


Figure 8. ~~Same as fig. ??, but~~ Short-term forecast errors and network attractor reconstructions for forcing experiments with ~~the~~ Lorenz95~~model varying the parameter *F*~~. Networks are trained on a ~~) Forcing vs timesteps. b,~~run with linearly increasing F from the lower to the upper end of the grey shaded area in the plots, ~~d~~and then tested on runs with fixed F. The blue lines show the network that include F as input. The yellow lines show the networks without F as input ("no input" networks in the text)~~results for~~ . The green lines show reference networks without F as input that are trained on runs with ~~2e3~~F fixed to the same value as the test runs ("no *F*" networks in the text). The left panels show short-term forecast error, ~~2e4~~ the right panels show mean and ~~2e5, training steps, respectively.~~standard-deviation of climate runs performed with the networks and additionally the mean and standard-deviation of Lorenz95 runs with F fixed to the reference

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 highly idealised representations of atmospheric processes, namely the Lorenz63 and the Lorenz95 models. We used feed-forward neural network architectures that are shown to work well on these systems when trained on the full ~~phase space~~ phase-space and without external forcing.

For the first point we raise, we showed that ~~in general~~ networks trained on only part of the Lorenz63 attractor are largely unable to reproduce trajectories outside the regions they were trained on. When making short-term forecasts initialized from points in ~~the unknown phase space~~ these unknown phase-space regions, the trajectories of the network ~~forecast simply~~ forecasts 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~~ phase-space that were not included in the training. The only ~~exception~~ exceptions 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 weather or climate events, or of events originating from unprecedented atmospheric ~~configurations.~~ or oceanic configurations. In contrast to the results for the Lorenz63 system, our experiments for the Lorenz95 system indicate that the networks can succesfully make forecasts in phase-space regions left out from training, and also explore these regions when making long simulations. In this respect, we have to note the difficulty of defining sensible regions of phase-space for the Lorenz95 system with its 40 dimensions. These difficulties would be even more severe for more realistic systems like numerical general circulation models. Still, this result is somewhat counter-intuitive, as one may naively consider the Lorenz95 system to be more complex than the Lorenz63 system. Therefore, our results indicate that using intuitive definitions of the complexity of a system to reason on the performance of feed-forward neural networks is problematic.

~~We~~ For the Lorenz63, 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~~phase-space. Thus, when parts of the ~~phase space~~ 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. We confirmed this by inspecting the activations of individual neurons in the trained networks, and showed that parts of the network are responsible for specific regions of the phase-space. This is similar to findings in the context of image recognition and generation, where different parts of neural networks have been shown to represent different objects/concepts (Bau et al., 2019).

As a caveat, we note that our experiments, which remove a large contiguous region of phase-space from the training data, are more penalising than what may be expected in a typical climate simulation. 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. Indeed, removing a wing is more akin to removing a season from a training set — for example asking a network to simulate a seasonal cycle without ever being trained on winter data — than having a

training set which does not include some rare extreme events — which presumably live in sparsely populated regions of the phase-space which need not be contiguous.

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~~on its performance for long series of iterated forecasts, and in particular on whether the resulting trajectories resemble those of the original dynamical system. When the training data ~~contains only data from~~ only covers part of the phase-space, this raises ~~issues related to~~ the issue of 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~~

All our main experiments were done with feed-forward neural network architectures that forecast the following state of the system. We repeated some of our experiments with networks that forecasted the systems' tendency instead. These were better in producing short-term forecasts in new regions of phase-space, but had even more trouble in producing stable trajectories outside the training space. While feed-forward architectures are widely used, there are many other architectures available, that potentially do not suffer from the issues we found (for example recurrent architectures, echo-state networks and the related reservoir computers). Chattopadhyay et al. (2019) found that echo-state networks outperform feed-forward architectures in forecasting the Lorenz95 model, and it could be that this also holds for the extrapolation issues addressed in this study. Regarding model architectures, for the forcing experiments it might also be possible that presenting the forcing in another way than done here (e.g. designing into the network that the forcing variable has different characteristics than the state variables) may improve the learning of the influence of the ~~original 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~~forcing.

To address the second ~~point~~ question 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~~ Given simulations that span a large enough range of forcing regimes, the networks that use the forcing ~~term and which may result in degraded forecasts, this poses a formidable (though not necessarily unsolvable) challenge to the~~ as input are indeed able to capture at least part of the influence of the forcing, and extrapolate it to some extent to new forcing regimes. The networks again perform better on the Lorenz95 than the Lorenz63 system. This indicates that the idea of emulating climate-change ~~projections with neural networks~~

~~. Specifically, it implies that it~~ projections with neural networks might not be entirely unrealistic. However, it would be very hard to know beforehand the range of forcing regimes one would need in the training period. Additionally, the networks trained with forcing as an input still perform worse than networks directly trained on the target forcing. Therefore, 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, without very detailed testing. Our results are similar to Rasp et al. (2018), who found that their neural network based a 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~~ Again, we should highlight that our experiments are not meant to provide a direct match to what may be seen in a climate model. For example, the forcing in the Lorenz63 system is modulated by tuning a parameter that changes the dynamics of the system, while the forcing term in the Lorenz95 system leads to transitions between periodic and turbulent regimes.

More generally, our experiments were performed on highly idealised systems and it is hard to estimate the extent to which they may ~~generalise~~ generalize 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~~ neurons (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~~ provide a starting point for ~~more~~ further discussion on the potentials and limitations of neural networks in the context of ~~complex~~ chaotic dynamical systems. Future studies could expand to more realistic systems (e.g. general circulation atmospheric models), explore neural network architectures beyond the feed-forward networks used here (e.g. recurrent architectures) and the influence of noisy training data. Additionally, it would be interesting to extend the analysis to the 2-level version of the Lorenz95 model, which would allow to also compare the networks to "truncated" versions of the model. Finally, a more mathematically rigorous approach – as opposed to the empirical approach used here – might shed interesting new light on the topic.

Code availability. The code used for this study is available in the accompanying Zenodo repository (doi:10.5281/zenodo.3461683) and on S.S.'s github repository (https://github.com/sipposip/code-for-Generalization-properties-of-neural-networks-trained-on-Lorenz-systems/tree/revision1)

Appendix A: Tuning of neural network architecture for ~~Lorenz95~~Lorenz63

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~~fully connected layers. Next, we performed an exhaustive gridsearch over network configurations and hyperparameters. The learning rate was varied from 0.00003 to 0.003, the number of hidden layers from 1 to 10, and the size of the hidden layers from 4 to 128. The activation function was fixed to the rectified linear unit ("ReLu"). A mini-batch size of 32 was used. The training data was normalized to zero mean and unit variance. The tuning was done with a Lorenz63 run with standard parameters, a timestep of 0.01 and $2e5$ timesteps. While the networks are all trained on short-term error, the final selection of network architecture was done by the ability of the network to reconstruct the attractor (see Section 3.2). The best architecture had 2 hidden layers with a hidden layer size of 128 and a learning rate of 3e-5.

Appendix B: Tuning of neural network architecture for Lorenz95

For the Lorenz95 model we chose as basic architecture stacked convolution layers, which wrap around the circular domain. The gridsearch was done over the following parameters: 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. A mini-batch size of 32 was used. The training data was normalized to zero mean and unit variance.

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~~ a lead-times of 0.1, 2 convolution layers with kernel size 5 worked best, and for a lead-time of 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~~~4 time units, both in terms of RMSE and anomaly correlation (when making longer forecasts through iteratively making forecasts with the network, see ~~fig. A1~~Fig. B1). 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 and is contrary

to what Scher and Messori (2019) found on coarse-grained reanalysis data. Indeed, the latter study concluded that the longer the training lead-time, the lower the forecast error. Our architecture only slightly overfits (Fig. B1 c); that is, error on the test data is slightly higher than on the training data. The network was trained until validation loss did not increase for 4 epochs, with a maximum of 30 epochs. The network architecture for the experiments including the forcing F as input was tuned separately. For this, a Lorenz95 run of 1e4 with linearly increasing F from 6 to 7 was used. The last 10 % of the run was used as validation set.

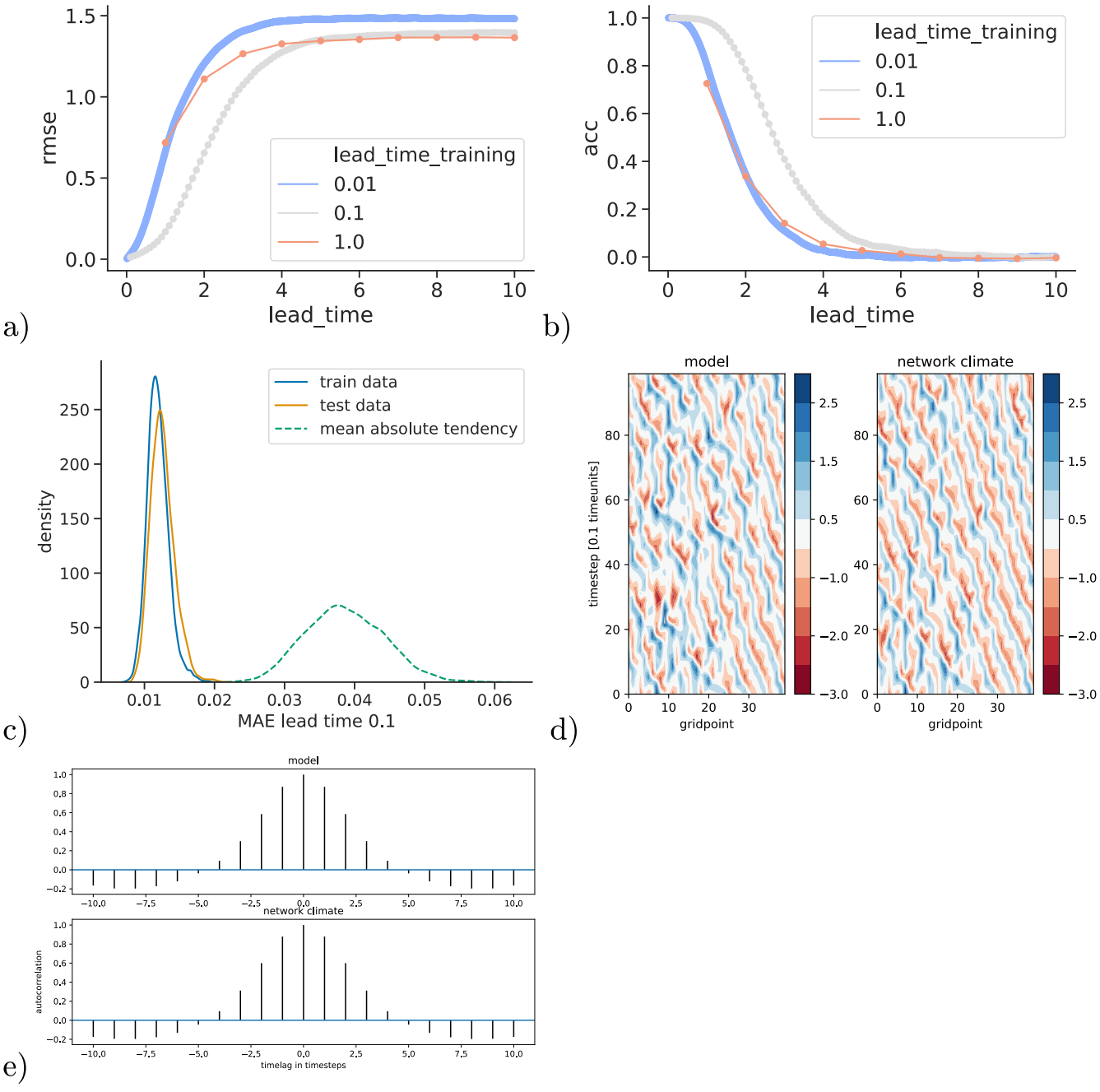


Figure B1. Evaluation of network architecture for the Lorenz95 system without F as input. a,b) Forecast error (on test data) for the best network configurations when training on lead-times of 0.01, 0.1 and 1 (different colors). c) Kernel density estimate of mean absolute forecast error on training and test data for 1-step forecasts of the network trained with a lead time of 0.1, and kernel density estimate of the mean absolute 1-step tendencies of the model (dashed line). d) Examples of the Lorenz95 model (left) and the network model obtained through iterated forecasts trained on a lead-time of 0.1 (right), both initialized from the same initial state. e) Autocorrelation for different timelags of the model and the network "climate".

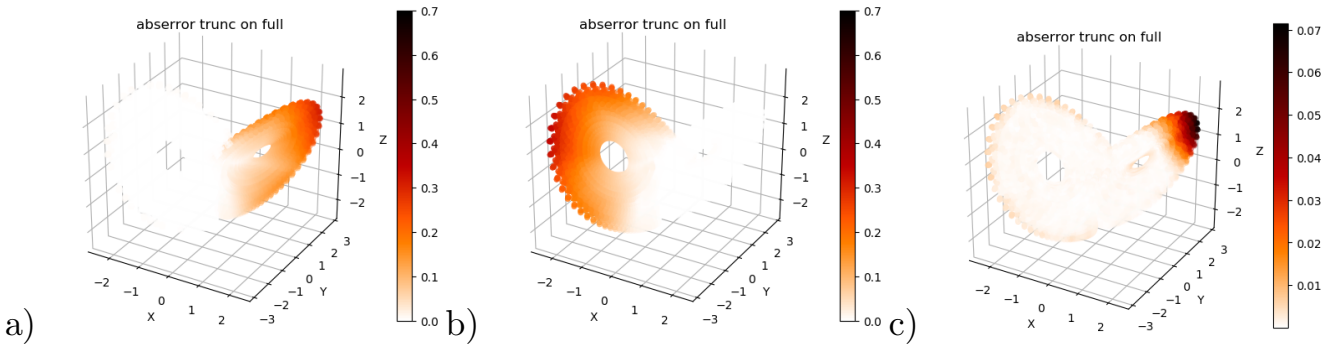


Figure C1. ~~As fig~~Same as Fig. ~~2~~2 d-f, but for networks forecasting the ~~points are located at the predicted state~~ tendency instead of ~~at~~ the ~~initial~~ following state~~of each forecast~~. Note the different colorscale in c)

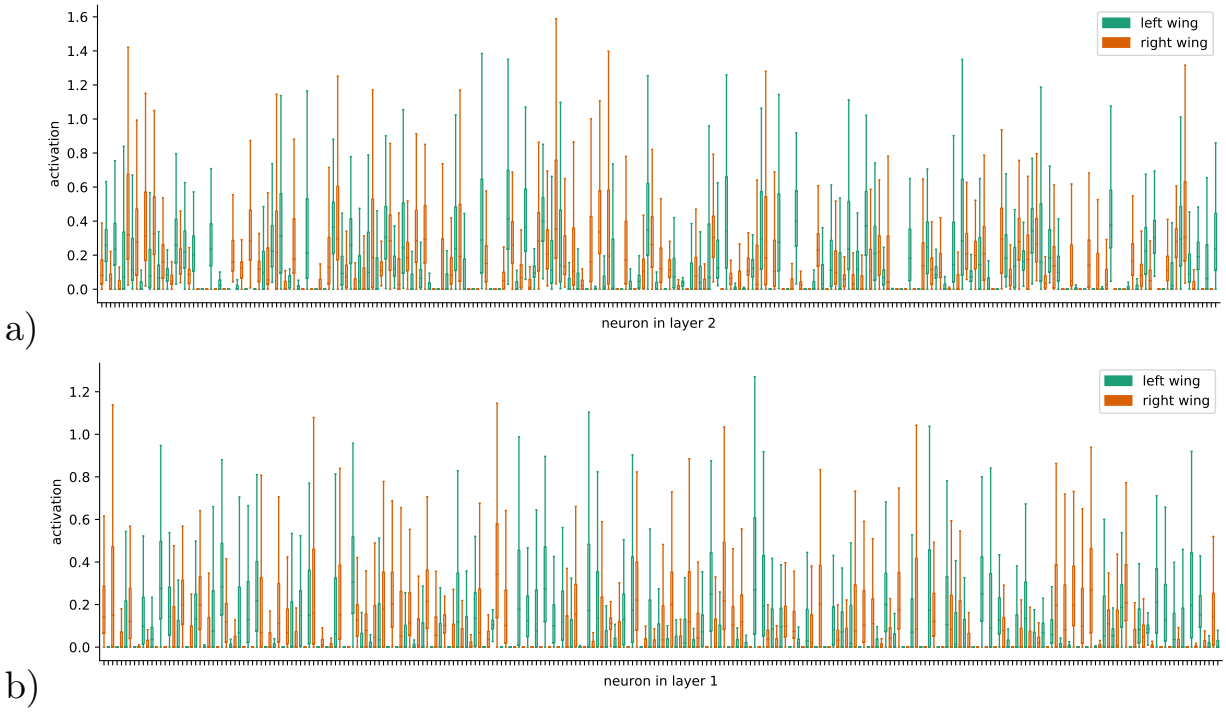


Figure C2. Same as Fig. 3b,e, but showing all neurons without specific ordering

Appendix C: ~~Supplementary figures~~

Appendix C: 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.

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 partly supported by the Swedish Research Council Vetenskapsrådet (grant no.: 2016-03724). The computations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at the High Performance Computing Center North (HPC2N) and National Supercomputer Centre (NSC).

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, 2000.

Bau, D., Zhu, J.-Y., Strobelt, H., Bolei, Z., Tenenbaum, J. B., Freeman, W. T., and Torralba, A.: GAN Dissection: Visualizing and Understanding Generative Adversarial Networks, in: Proceedings of the International Conference on Learning Representations (ICLR), 2019.

Chattopadhyay, A., Hassanzadeh, P., Palem, K., and Subramanian, D.: Data-driven prediction of a multi-scale Lorenz 96 chaotic system using a hierarchy of deep learning methods: Reservoir computing, ANN, and RNN-LSTM, arXiv preprint arXiv:1906.08829, 2019.

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.

Hardt, M., Recht, B., and Singer, Y.: Train faster, generalize better: Stability of stochastic gradient descent, arXiv preprint arXiv:1509.01240, 2015.

Hindmarsh, A. C.: ODEPACK, a systematized collection of ODE solvers, Scientific computing, pp. 55–64, 1983.

Hochreiter, S. and Schmidhuber, J.: Simplifying neural nets by discovering flat minima, in: Advances in neural information processing systems, pp. 529–536, 1995.

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.

Jones, E., Oliphant, T., Peterson, P., et al.: SciPy: Open source scientific tools for Python, http://www.scipy.org/, online; accessed 12th September 2019, 2001.

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.

Lu, Z., Hunt, B. R., and Ott, E.: Attractor reconstruction by machine learning, Chaos: An Interdisciplinary Journal of Nonlinear Science, 28, 061 104, 2018.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J.: Sensitivity and generalization in neural networks: an empirical study, arXiv preprint arXiv:1802.08760, 2018.

Pasini, A. and Pelino, V.: Can we estimate atmospheric predictability by performance of neural network forecasting? the toy case studies of unforced and forced lorenz models, in: CIMSA. 2005 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, 2005., pp. 69–74, https://doi.org/10.1109/CIMSA.2005.1522829, 2005.

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, 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.

Schevenhoven, F. J. and Selten, F. M.: An efficient training scheme for supermodels, Earth System Dynamics, 8, 429–438, https://doi.org/10.5194/esd-8-429-2017, https://www.earth-syst-dynam.net/8/429/2017/, 2017.

Sugiyama, M. and Kawanabe, M.: Machine learning in non-stationary environments: Introduction to covariate shift adaptation, MIT press, 2012.

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.

Watson, P. A. G.: Applying Machine Learning to Improve Simulations of a Chaotic Dynamical System Using Empirical Error Correction, Journal of Advances in Modeling Earth Systems, 11, 1402–1417, https://doi.org/10.1029/2018MS001597, https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018MS001597, 2019.

Weyn, J. A., Durran, D. R., and Caruana, R.: Can Machines Learn to Predict Weather? Using Deep Learning to Predict Gridded 500-hPa Geopotential Height From Historical Weather Data, Journal of Advances in Modeling Earth Systems, 2019.

Wu, L., Zhu, Z., et al.: Towards understanding generalization of deep learning: Perspective of loss landscapes, arXiv preprint arXiv:1706.10239, 2017.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O.: Understanding deep learning requires rethinking generalization, arXiv preprint arXiv:1611.03530, 2016.

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.