We want to thank all 3 reviewers for their helpful comments that helped to improve our paper. We have now prepared a new version of our manuscript that takes full account of these comments. Specifically, it includes more tests with the neural network setup, and a test on coarse-grained reanalysis data. Additionally, we clarified numerous passages of the text based on the reviewer comments.

Below we list point-by-point responses (in red) to all issues raised by the reviewers. A draft of our new manuscript and a version with track-changes (compared to the original submission) is added at the end of this document.

Reviewer #1

This is a very interesting paper that provides an honest presentation of new results on the use of neural networks to learn the equations of motion of the atmosphere. The paper is relevant for GMD and should be published. However, a minor revision that is addressing the comments below could improve the paper.

- As we argue in Dueben and Bauer 2018, I am very surprised that you get away with 1-day timesteps. I would guess that a T21 model with a 1-day timestep would be unstable when using explicit time-stepping schemes (maybe I am wrong?) and it is hard to believe that a neural network is learning something like an implicit scheme that would allow for larger timesteps. Can you provide the timestep that is used in Plasim for comparison? If this is much smaller, can you comment why you think that the neural network model may get away with this, in particular towards the pole? (I assume that you are using a regular Gaussian grid where grid-spacing will become smaller towards the pole).

We would also expect that a T21 model with a 1-day timestep would be unstable with an explicit time-stepping scheme. However, we do not think that it is appropriate to see a neural network as an explicit time-stepping scheme. Rather, it should be seen as a general mapping function (mapping model states to model states some time later).
Regarding the timestep in the models: we used the standard configuration, which is 30 min for plasimT42, 20min for plasimT21, and 60 min for pumaT21 and pumaT42. We mention this in the new manuscript (p3 L7-8). Regarding the time-step question please see our response to you 2nd point below.
Also, note that we are not using a regular Gaussian grid, but regular lat-lon grids. We now explicitly mention this in the method section.

- I am also (positively) surprised that you are having no problems with loss of stability when using neural networks while other papers report problems when using neural networks to represent physical systems. Can you speculate why this is? Maybe due to the convolutional layers? Can you diagnose the change of global energy in climate simulations?

The reason for not losing stability might indeed be caused by the convolution layers. The reason might be that we use stacked convolution layers, whereas Dueben and Bauer use a local deep network. At a first thought these approaches are very similar, but there are important differences that might explain why the convolution network works also with longer timesteps. In the simplest setup (1 single convolution layer with stride S and convolution-depth N), the convolution approach is identical to a local network with stride S with 1 single hidden layer with size N. However, as soon as the networks get deeper the differences begin. In the case of adding mode hidden layers to the local network, the additional layers still only get information from stride S. However, when stacking convolution layers, the 2nd convolution layer has more information than from stride S. If S=3, then the second layer already has information from 5 gridpoints, and a 3rd layer from 7 and so on. So what could happen in principle is the following: if 1 single convolution layer can learn let's say a 1 hour forecast, then maybe 24 could learn a 24 hour forecast, whereas each convolution layers in the 24-layer network has exactly the same weights as the single layer network trained on 1 hours. However, we have to note that this a theoretical consideration. We added discussion on this topic in the discussion section (p15 L11-12). Regarding diagnosing the change of global energy in the climate simulations: this would unfortunately be quite difficult to do with our data. Our neural networks are trained on pressure-level data, and therefore also the network-climate simulations are on pressure levels. To our knowledge, computing global atmospheric energy is usually done on model levels, and is very tricky to do on pressure levels. We agree that analyzing global energy would be interesting, but it would require nothing short of an entirely new study with different training for the networks, and we would prefer not to address it here.

• When using convolutional networks, will the stencil of gridpoints around a grid-point in the neural network have exactly the same weights for all gridpoints? Or will the weights be changing for each gridpoint? If they are the same, how can you justify that gridpoints towards the poles (that will have a very different resolution for the stencil of surrounding points) use the same weights as for points at the equator?

In convolution networks the stencil of gridpoints around a grid-point in the neural network has exactly the same weights for all gridpoints (one way to see it is as a moving filter with fixed weights). It is true that the architecture does not take into account that gridpoints towards the poles have different resolution than close to the equator. One way to solve this is to use a spherical formulation of convolution, which has also been proposed on regular grids. We mention this in the new manuscript in the section on potential improvements of the neural network architecture (p.14 L13).

• The paper would benefit if you would provide more discussion how to improve neural network configurations in future studies at the end of the conclusion (speculations are welcome).

We included a new subsection on this at the end of the conclusion section. In this section, we discuss potential network configurations for future studies.

• Figure 4: Why are the PUMA results not symmetric in zonal direction?

The results for PUMA are nearly symmetric when looking at 1-day forecasts (which we didn't show (see fig. R1), but for longer forecasts they are not symmetric any longer. We think this is caused by problems induced at the border of the domain since we did not use any circular wrapping in the convolution. We now mention this in the manuscript (p.9 L1-4).

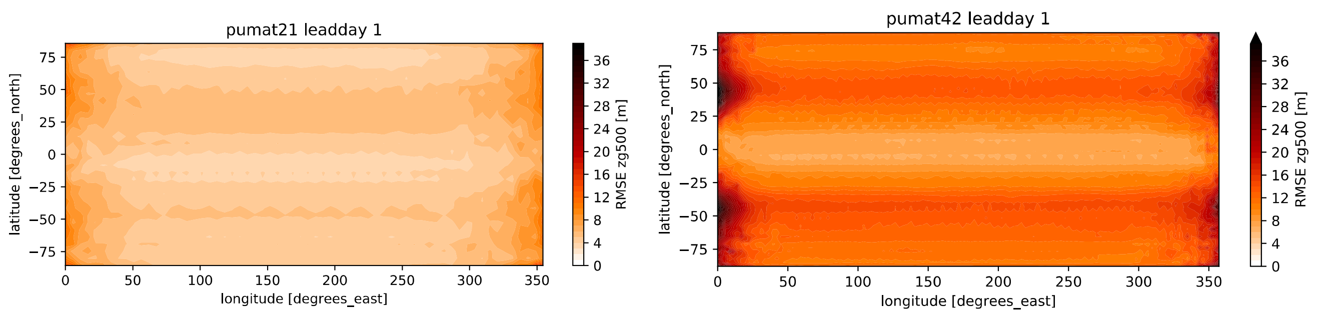

Fig. R1: RMSE of network forecasts with leadtime 1 day, for networks trained on PUMAT21(left) and PUMAT42(right).

Really minor:

Figure 2: The figure should be made bigger to improve visibility.

We increased the size of the figure.

Page 2: A "local model" as suggested in Dueben and Bauer 2018 would increase the amount of training data significantly since you would train for 40 (or 100) years of data for N grid points. Maybe worth mentioning? But only if you think this would fit.

This is in fact also happening with the convolution layers. A convolution layer is a filter of fixed size that is moved across the domain, and the weights are always the same.

Page 2: "...ambition has shifted from..." not really. This is rather a different application.

We changed the sentence to “Recently, in addition to using machine-learning to enhance numerical models, there have been ambitions to use it for weather forecasting itself.”

Page 4: "(...see Scher(2018)." add ")"

We made the suggested change

Page 6: Why zg500 (ta800) and not z500 (t800)?

We use this notation because it is the notation used in PUMA and PLASIM. In order to avoid programming errors, we make use of automatic plot-labelling directly form the data.

Page 6: "has higher error at longer lead times" At long lead times and close to zero correlation a bias in the model can cause interesting behaviours in rmse plots. It can also reduce the error.

We added “Here we have to note that at long lead-times and close to zero-correlation, RMSE can be hard to interpret since it can be strongly influenced by biases in the forecasts.” (p.7 L24-26)

Page 8: Can you comment how important it was to include the day of year as input?

We have not analyzed the error of multi-day forecasts of the networks that include day of year as input, but the 1-day forecast error is very similar to the error of the standard networks (trained without day of year as input).

Figure 5: I do not see the shading in my printout.

The shading is very narrow (since the spread is so small).

Page 11: "to to"
we removed the double “to”.

Page 12: Personally, I am very sceptical that the use for different networks for different months would produce reasonable results.

We agree that it is hard to tell a-priori whether this would work. However, considering the often un-intuitiveness of machine-learning, we prefer to keep this speculative suggestion for further research.

Reviewer #2

This paper is a substantial contribution to modelling science within the scope of this journal, containing new concept of producing statistical model for weather forecast by emulating a simplified GCM model, using neural networks. I recommend this paper for publication after minor revision. Two major points have to be clarified, otherwise it is difficult to understand the method and results presented in the paper.

“1. Naming models. In the paper the same name is used for the GCM and NN emulation of this GCM (e.g., PLASIMT21). This kind of naming create confusion and makes understanding the method and results difficult.”
We agree that the naming convention we used was not ideal. In the revised text, we now always speak of “the network for PLASIMT21” or “the PLASIMT21 networks” etc.) when we refer to the neural networks and their forecasts. We hope that this makes the text clearer to the reader.

“2.What is used as "truth" in Section 2.4 and in Section 3. Are statistics shown in Section3 represent the accuracy of NN emulations of different models (e.g., NN emulatingPLASIMT21 vs. PLASIMT21) or the accuracy of NN emulations vs. reanalysis (e.g., NN emulating PLASIMT21 vs. ERA or NCEP/NCAR reanalysis”.

Throughout the whole paper, the NN emulations are evaluated against the model they were trained on (e.g., the NN trained on PLASIMT21 is evaluated on PLASIMT21). For the “weather forecasting” problem in sections 3.1 and 3.2, the networks are tested on forecasting the model a couple of timesteps ahead. This testing is done on the test-set of the model runs, so the data not used for the training. For the climate statistics in section 3.3, the statistics of the NN climate emulations are compared to the statistics of the model the network has been trained on (e.g. the statistics of the NN trained on PLASIMT21 are compared to the statistics of PLASIMT21).
We added “All network forecasts will be validated against the underlying model the network was trained on (e.g. for the forecasts of the network trained on PLASIMT21 the ``truth'' is the evolution of the PLASIMT21 run)” at the beginning of section 2.4 to make this clearer in the new manuscript (p.6 L20-21).

Anonymous Referee #3

The authors build on a previous paper in which one of the authors applied a convolutional neural net with an encoder-decoder architecture to the problem of weather forecasting and climate simulation in a simplified atmospheric GCM. Here the approach is extended to a more comprehensive atmospheric GCM and to different horizontal resolutions in the GCMs. Inclusion of a seasonal cycle proves to be an important issue when trying to reproduce the climate with the neural net. The research question is exciting and the presentation and approach is generally good. However, changes are needed to properly describe the approach that has been taken and to quantify how well the neural net is doing.

“1. As far as I can tell, the same network as was applied to the simple GCM PUMA is applied to the more comprehensive GCM PLASIM. In particular, the variables used are horizontal winds, geopotential height and temperature. This is surprising since PLASIM presumably has a hydrological cycle, and specific humidity is presumably a prognostic variable. Therefore, the state of the atmosphere in PLASIM at a given time is not described with the 4 variables used. The choice to not include humidity in the network should be justified. Presumably a network with humidity included would do better (?)”

The reviewer is correct in noting that we used the same architecture for PUMA and PLASIM, thus not including moisture, which is indeed part of PLASIM. This choice was made to keep the architecture the same across all models. However, we agree that adding moisture as variable might improve the neural network forecasts trained on PLASIM. In the revised version, we have included a test where we include moisture in one of the PLASIM trainings in section 3.5. As you will see, including the 3 hydrological state variables of PLASIM (relative humidity, cloud liquid water content and cloud cover) did not improve the forecasts, but in fact slightly made them worse. This might indicate that our architecture does not – at least not without any changes – work for hydrological variables, which might be related to their very different distributions both in time and space compared to the other variables we use.
Additionally, we now also explicitly mention in the method section that for the main simulations we do not consider the moisture variables in PLASIM in order to avoid confusion (p.6 L6-7).

“2. It is difficult to assess how well the networks are doing in their forecasts in figure 3 because they are not compared to anything else. In the preceding work, comparison was made to ’persistence’. But a more informative choice would be to plot RMSE and ACC for a ’perfect model’ forecast in which the same GCM is used to make the forecast with a small perturbation in the initial conditions (or alternatively in the tuned constants in the model physics). Comparing to a perfect model forecast would allow the reader to assess the skill of the neural net forecast - it can’t be expected to do better than a perfect model prediction (given any error and the chaotic nature of the atmosphere). This was also help the paper have more impact since the neural net will ultimately have to compete with traditional NWP, albeit in terms of both accuracy and speed and not just accuracy.”

We agree that a baseline forecast would help to aid the reader. In the revised version, we have included the skill of persistence forecasts in fig. 3, and discuss it in the text both in the results and in the conclusion section (p.7, p.15 L1-2).
We also considered making perfect-model forecasts with PUMA and PLASIM as suggested. Unfortunately, this functionality is not implemented per default in PUMA/PLASIM, and to our best knowledge, related software has not been published. Doing perfect model forecasts would require both changes to the restart-routines of the model, and the implementation of an initial perturbation scheme. Especially the latter is not trivial to do. We have therefore chosen to take another approach to put the skill of our network forecasts in a broader context and make them comparable to alternative methods (in addition to including the skill of persistence forecasts). Namely, we included tests on coarse-grained reanalysis data (as in Dueben and Bauer 2018), which is presented in section 3.5 in the new manuscript. For this we used the same subset of ERA5 as in Dueben and Bauer. Dueben and Bauer have made forecasts on their dataset with the NWP model IFS with T21 resolution. This served as baseline for their neural network forecasts, and can thus also be used to compare our forecasts with via comparing our new Fig. 8 with their Fig. 3a.

3. “Another possibility to consider for why the climate prediction with a seasonal cycle does not work well is that you are forecasting over a short time frame (1 day) in which diabatic effects that vary seasonally such as changes in insolation are not very important compared to the dynamical initial condition. Perhaps training using a longer forecast lead time (e.g. 5 days) would work better for the climate simulations with a seasonal cycle”

We added a test that makes a climate run with a network trained on 5-day forecasts. Unfortunately, this did not resolve the problems with seasonality.
Specifically, we added fig S19 and Video S9, and the following text to the manuscript:

“All our networks were trained to make 1-day forecasts. The influence of the seasonal cycle on atmospheric dynamics - and especially the influence of diabatic effects - may be very small for 1-day predictions. This could make it hard for the network to learn the influence of seasonality. To test this, we repeated the training of the climate network for PLASIMT21 using 5-day forecasts. This made the seasonality of ta800 at a single gridpoint at 76 $^{\circ}$N better (fig. S19 in the supplement), but the fields of the climate run still become unrealistic (video S9 in the supplement).” (p.11 16-20).

Minor comments

1. “The neural net architecture is described as an autoencoder. My understanding of the nomenclature is that it is an encoder-decoder but not an autoencoder (since the output is not the same as the input).”

You are right, our architecture is indeed more correctly described as an encoder-decoder. Thanks for pointing this out, we have changed the naming accordingly in the manuscript.

2. “A few more lines description is needed for each GCM in section 2.1. How is the dry dynamical core of PUMA forced? (e.g. is it a Held-Suarez setup?) What makes PLASIM an ’intermediate complexity’ GCM? (e.g. how exactly does it differ from a standard GCM aside from the lack of a dynamical ocean)?”

PUMA uses Newtonian cooling for diabatic heating/cooling. The main difference between PLASIM and standard GCMs is that all included sub-systems of the Earth system except for the atmosphere (like the ocean, sea-ice and soil) are reduced to systems of low complexity.

We now mention these details in the method section (section 2.1)

“3. Figure 2 is helpful but not fully described in the caption. In particular the caption should say what the numbers are - does None, 64,128,40 refers to?, lat, lon, channels. What does ’None’ refer to here?”

None, 64,128,40 is the typical notation in Keras and refers to time (or samples), lat, lon, channels. We now explain this in the caption.

“4. Appendix A1: do you use all times t1 before and after t in the calculation”

Yes, exactly. For computing the local dimension at time t, the distances of the field at t with the fields at all other timesteps are computed.

We added “(i.e. all times before and after t)” in the manuscript to make this clearer (p. 16 L6-7).

Weather and climate forecasting with neural networks: using GCMs with different complexity as study-ground

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. Recently, there has been growing interest in the possibility of using neural networks for both weather forecasting and the generation of climate datasets. We use a bottom-up approach for assessing whether it should, in principle, be possible to do this. We use the relatively simple General Circulation Models (GCMs) PUMA and PLASIM as a simplified reality on which we train deep neural networks, which we then use for predicting the model weather at lead times of a few days. We specifically assess how the complexity of the climate model affects the neural network's forecast skill, and how dependent the skill is on the length of the provided training period. Additionally, we show that using the neural networks to reproduce the climate of general circulation models including a seasonal cycle remains challenging – in contrast to earlier promising results on a model without seasonal cycle.

1 Introduction

Synoptic weather forecasting (forecasting the weather at lead times of a few days up to 2 weeks), has for decades been dominated by computer models based on physical equations – the so called Numerical Weather Predictions (NWP) models. The quality of NWP forecasts has been steadily increasing since their inception (Bauer et al., 2015), and these models remain the backbone of virtually all weather forecasts. However, the fundamental nature of the weather forecasting problem can be summarized as: "starting from today's state of the atmosphere, we want to predict the state of the atmosphere x days in the future". Thus posed, the problem is a good candidate for supervised machine learning. While long thought unfeasible, the recent success of machine learning techniques in highly complex fields such as image and speech recognition warrants a review of this possibility. Machine learning techniques have already been used to improve certain components of NWP and climate models – mainly parameterization schemes (Krasnopolsky and Fox-Rabinovitz (2006); Rasp et al. (2018); Krasnopolsky et al. (2013); O'Gorman and Dwyer (2018)), to aid real-time decision making McGovern et al. (2017), to exploit observations and targeted high-resolution simulations to enhance earth system models (Schneider et al., 2017), for El-Niño predictions (Nooteboom et al., 2018) and to predict weather forecast uncertainty (Scher and Messori, 2018).

Recently, in addition to using machine-learning to enhance numerical models, there have been ambitions to use it to tackle weather forecasting itself. The holy grail is to use machine-learning, and especially “deep learning”, to completely replace NWP models, although opinions may diverge on if and when this will happen. Additionally, it is an appealing idea to use neural networks/deep learning to emulate very expensive General Circulation Models (GCMs) for climate research. Both these ideas have been tested with some success for simplified realities (Dueben and Bauer, 2018; Scher, 2018). In Scher (2018), a neural network approach was used to skillfully forecast the “weather” of a simplified climate model, as well as emulate its climate. Dueben and Bauer (2018), based on their success in forecasting reanalysis data regridded to very low resolution, concluded that it is “fundamentally” possible to produce deep-learning based weather forecasts.

Here, we build upon the approach from Scher and apply it to a range of climate models with different complexity. We do this in order to assess: 1) how the skill of the neural network weather forecasts depends on the available amount of training data; 2) how this skill depends on the complexity of the climate models; and 3) under which conditions it may be possible to make stable climate simulations with the trained networks, and how this depends on the amount of available training data. (1) is of special interest for the idea of using historical observations in order to train a neural network for weather forecasting. As the length of historical observations is strongly constrained (~ 100 years for long renalyses assimilating only surface observations, and ~40 years for reanalyses assimilating satellite data), it is crucial to assess how many years of past data one would need to produce meaningful forecasts. The value of (2) lies in evaluating the feasibility of climate models as a “simplified reality” for studying weather forecasting with neural networks. Finally, (3) is of interest when one wants to use neural networks not only for weather forecasting, but for the distinct yet related problem of seasonal and longer forecasts, up to climate projections.

To avoid confusion, we use the following naming conventions throughout the paper: “model” always refers to physical models (i.e. the climate models used in this study), and will never refer to a “machine learning model”. The neural networks are referred to by the term “network”.

2 Methods

2.1 Climate models

To create long climate model runs of different complexity, we used the Planet Simulator (PLASIM) intermediate complexity GCM, and its dry dynamical core: the Portable University Model of the Atmosphere (PUMA) (Fraedrich et al., 2005). Each model was run for 830 years with two different horizontal resolutions (T21 and T42, corresponding to ~5.65 and ~2.8 degrees latitude, respectively) and 10 vertical levels. The first 30 years of each run were discarded as spin-up, leaving 800 years of daily data for each model. The runs will from now on be referred to as PLASIMT21, PLASIMT42, PUMAT21 and PUMAT42. All the model runs produce a stable climate without drift (fig. S1-S10 in the Supplement). Additionally, we regridded PLASIMT42 and PUMAT42 (with bi-linear interpolation) to a resolution of T21. These will be referred to as PLASIMT42_regridT21 and PUMAT42_regridT21.

The PUMA runs do not include ocean and orography and use Newtonian cooling for diabatic heating/cooling. The PLASIM runs include orography, but no ocean-model. The main difference between PLASIM and standard GCMs is that sub-systems of the Earth system other than the atmosphere (e.g. the ocean, sea-ice and soil) are reduced to systems of low complexity. We used the default integration timestep for all four model setups, namely 30 min for PLASIMT42, 20min for PLASIMT21, and 60 min for PUMAT21 and PUMAT42. We regrid all fields to regular lat-lon grids on pressure levels for analysis. An additional run was made with PUMA at resolution T21, but with the seasonal cycle switched off (eternal boreal winter). This is the same configuration as used in Scher (2018) and will be referred to as PUMAT21_noseas.

In order to contextualise our results relative to previous studies, we also run a brief trial of our networks on ERA5 (C3S, 2017) reanalysis data (Section 3.5 , similar to Dueben and Bauer (2018).

2.2 Complexity

Ranking climate models according to their "complexity" is a non-trivial task, as it is very hard to define what complexity actually means in this context. We note that here we use the term loosely, and do not refer to any of the very precise definitions of "complexity" that exists in various scientific fields (e.g. Johnson (2009)). Intuitively, one might simply rank the models according to their horizontal and vertical resolutions and the number of physical processes they include. However, it is not clear which effects would be more important (e.g. is a model with higher resolution but less components/processes more or less complex than a low-resolution model with a larger number of processes?). Additionally, more physical processes do not necessarily imply a more complex *output*. For example, very simple models like the Lorenz63 model (Lorenz, 1963) display chaotic behaviour, yet it is certainly possible to design a model with more parameters and a deterministic behaviour.

To circumvent this conundrum, we adopt a very pragmatic approach based solely on the output of the models, and grounded in dynamical systems theory. We quantify model complexity in terms of the the local dimension d: a measure of the number of degrees of freedom of a system active locally around a given instantaneous state. In our case, this means that we can compute a value of d for every timestep in a given model simulation. While not a measure of complexity in the strict mathematical or computational senses of the term, d provides an objective indication of a system's dynamics around a given state and, when averaged over a long timeseries, of the system's average attractor dimension. An example of how d may be computed for climate fields is provided in Faranda et al. (2017), while for a more formal discussion and derivation of d we point the reader to Appendix A in Faranda et al. (2019). The approach is very flexible, and may be applied to individual variables of a system (which represent projections of the full phase-space dynamics onto specific sub-spaces, called Poincaré sections), multiple variables or, with adequate computational resources, to the whole available dataset. The exact algorithm used here is outlined in Appendix A1.

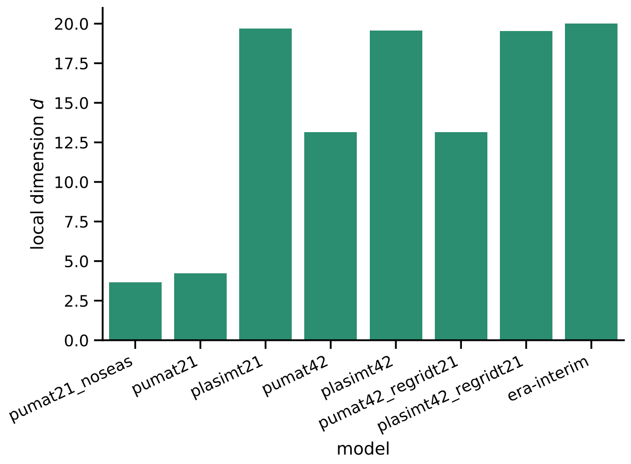

Figure 1. Averages of the local dimension d (here used as a data-based measure of "complexity") for the 500hPa geopotential height of the models used in this study and of the ERA-Interim reanalysis.

The local dimension was computed for 38 years of each model run, as well as for the ERA-Interim reanalysis on a 1x1 degree grid over 1979-2016 (Dee et al., 2011). The choice of 38 years was made because this is the amount of available years in ERA-Interim, and the length of the dataseries can affect the estimate of d (Buschow and Friederichs, 2018). Figure 1 shows the results for 500hPa geopotential height. The complexity of PUMA increases with increasing resolution, whereas both the low and the high resolution PLASIM model have a complexity approaching that of ERA-Interim. Thus – at least by this measure – they are comparable to the real atmosphere. The high resolution runs regridded to T21 have nearly the same complexity as the original high resolution runs. The ranking is the same for nearly all variables and levels (fig. S11 in Supplement). For the rest of the paper, the term "complexity" or "complex" always refers to the local dimension d.

2.3 Neural networks

Neural networks are in principle a series of non-linear functions with weights determined through training on data. Before the training, one has to decide the *architecture* of the network. Here, we use the architecture proposed by Scher (2018), which is a convolutional decoder-encoder, taking as input 3d model fields and outputting 3d model fields of exactly the same dimension. It was designed and tuned in order to work well on PUMAT21 without seasonality (for details see Scher (2018)). In order to ease comparison with previous results, in the main part of this study we use the same network layout and hyperparameters as in (Scher, 2018), except for the number of epochs (iterations over the training set) the network is trained over. In the original configuration only 10 epochs were used. It turned out that, especially for the more complex models, the training was not saturated after 10 epochs. Therefore, here we train until the skill on the validation data has not increased for 5 epochs, with a maximum of 100 epochs. The layout is depicted in fig. 2. The possibility of re-tuning the network and its implications are discussed in section 3.4. For the networks targeted to create climate simulations (hereafter called climate-networks) we deviate from this

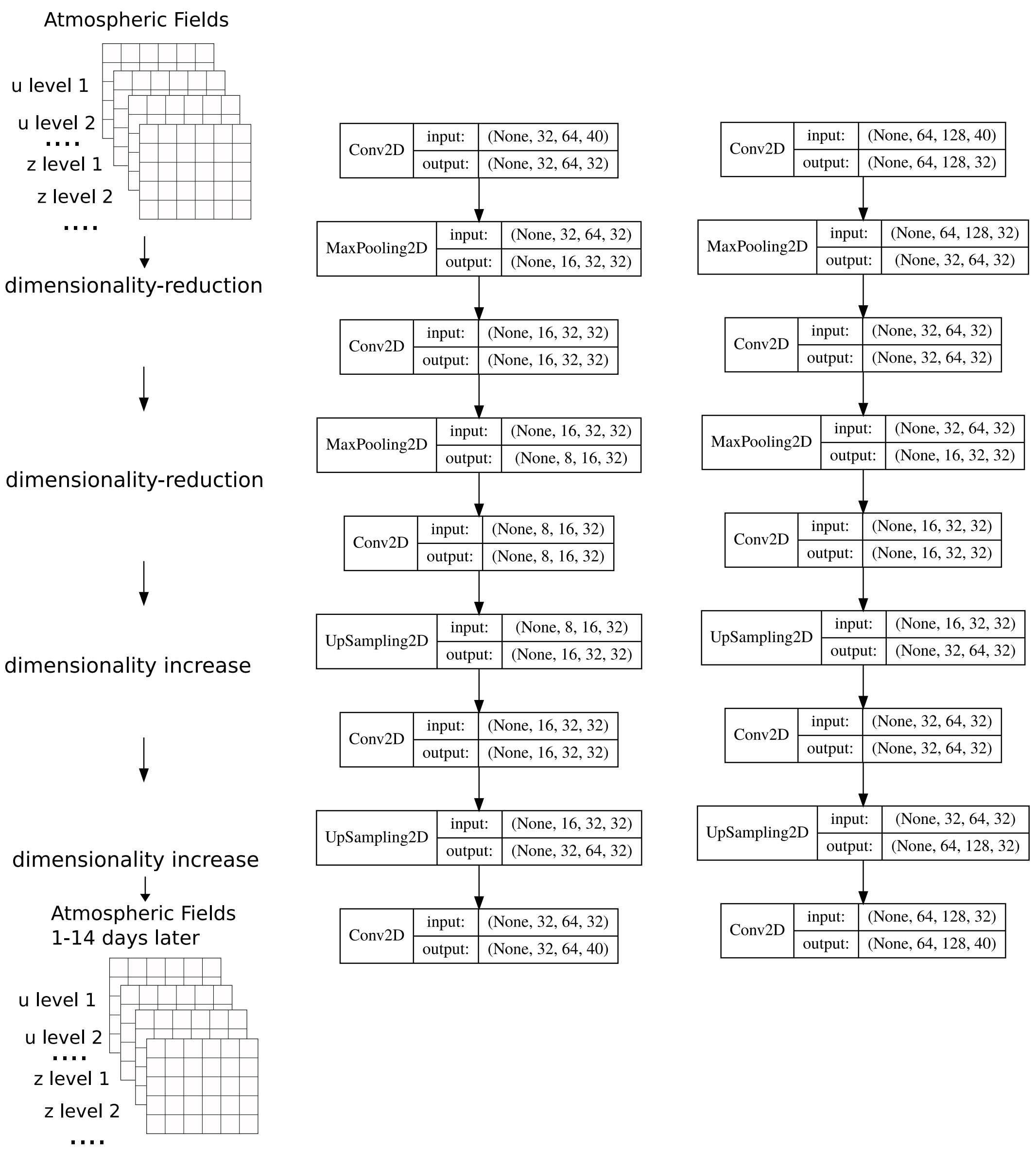
Atmospheric Fields
u level 1
u level 2
....
z level 1
z level 2
....
dimensionality-reduction
dimensionality-reduction
dimensionality increase
dimensionality increase
Atmospheric Fields
1-14 days later
u level 1
u level 2
....
z level 1
z level 2
....
Conv2D input: (None, 32, 64, 40) output: (None, 32, 64, 32)
MaxPooling2D input: (None, 32, 64, 32) output: (None, 16, 32, 32)
Conv2D input: (None, 16, 32, 32) output: (None, 16, 32, 32)
MaxPooling2D input: (None, 16, 32, 32) output: (None, 8, 16, 32)
Conv2D input: (None, 8, 16, 32) output: (None, 8, 16, 32)
UpSampling2D input: (None, 8, 16, 32) output: (None, 16, 32, 32)
Conv2D input: (None, 16, 32, 32) output: (None, 16, 32, 32)
UpSampling2D input: (None, 16, 32, 32) output: (None, 32, 64, 32)
Conv2D input: (None, 32, 64, 32) output: (None, 32, 64, 40)
Conv2D input: (None, 64, 128, 40) output: (None, 64, 128, 32)
MaxPooling2D input: (None, 64, 128, 32) output: (None, 32, 64, 32)
Conv2D input: (None, 32, 64, 32) output: (None, 32, 64, 32)
MaxPooling2D input: (None, 32, 64, 32) output: (None, 16, 32, 32)
Conv2D input: (None, 16, 32, 32) output: (None, 16, 32, 32)
UpSampling2D input: (None, 16, 32, 32) output: (None, 32, 64, 32)
Conv2D input: (None, 32, 64, 32) output: (None, 32, 64, 32)
UpSampling2D input: (None, 32, 64, 32) output: (None, 64, 128, 32)
Conv2D input: (None, 64, 128, 32) output: (None, 64, 128, 40)

Figure 2. Architecture of the neural network for the models with resolution T21 (left) and T42 (right). Each box describes a network layer, with the numbers indicating the dimension of (None, lat, lon, level). "None" refers to the time-dimension that is not relevant here, but which we include in the schematic since it is part of the basic notation of the used software library. Figure based on Fig. S1 from Scher (2018).

setup: here, we include the day of year as additional input to the network, in the form of a separate input channel. To remain consistent with the encoder-decoder setup, the output also contains the layer with the day of year. However, when producing the network climate runs, the predicted day of the year is discarded.

The last 10% samples of the training data are used for validation. This allows to monitor the training progress, control overfitting (the situation where the network works very well on the training data, but very poorly on the test data), and potentially limit the maximum number of training epochs. As input to the neural networks we use 4 variables (u, v, t and z) at 10 pressure levels; each variable at each level is represented as a separate input layer (channel). These 4 variables represent the full state of the PUMA model. PLASIM has 3 additional atmospheric variables related to the hydrological cycle (relative humidity, cloud liquid water content and cloud cover). In order to keep the architecture the same, these are not included in the standard training, and only used for a test in section 3.5.

All networks are trained to make 1-day forecasts. Longer forecasts are made by iteratively feeding back the forecast into the network. We did not train the network directly on longer lead-times, based on the finding of Dueben and Bauer (2018) that it is easier to make multiple short forecasts compared to a single long one. Due to the availability of model data and in keeping with Scher (2018), we chose 1-day forecasts as opposed to the shorter forecast step (1 hour) in Dueben and Bauer (2018). For each model, the network was trained with a set of 1, 2, 5, 10, 20, 50, 100, 200, 400 and 800 years. Since with little training data the network is less constrained, and the training success might strongly depend on exactly which short period out of the model run is chosen, the training for periods up to and including 20 years were repeated 4 times, shifting the start of the training data by 10, 20, 30 and 40 years. The impact of the exact choice of training period is discussed where appropriate. All the analyses shown in this paper are performed on the forecasts made on the first 30 years of the model run, which were never used during training and therefore provide objective scores (the ’test’ dataset).

2.4 Metrics

All network forecasts are validated against the underlying model the network was trained on (e.g. for the forecasts of the network trained on PLASIMT21 the “truth” is the evolution of the PLASIMT21 run). We specifically adopt two widely used forecast verification metrics, namely the Root Mean Square Error (RMSE) and the Anomaly Correlation Coefficient (ACC). The RMSE is defined as:

$$RMSE = \sqrt{\overline{(prediction - truth)^2}} \qquad (1)$$

where the overbar denotes a mean over space and time (for global measures), or over time only (for single gridpoints). The ACC measures the spatial correlation of the forecast anomaly fields with the true anomaly fields for a single forecast. The anomalies are computed with respect to a 30-day running climatology, computed on 30 years of model data (similar to how the European Centre for Medium Range Weather Forecasts computes its scores for forecast validation).

$$ACC_t = correlation$$
$$([truth_{1,1} - clim_{1,1}, ..., truth_{nlat,nlon} - clim_{nlat,nlon}], [prediction_{1,1} - clim_{1,1}, ..., prediction_{nlat,nlon} - clim_{nlat,nlon}]) \tag{2}$$

To compute a score over the whole period, the ACC for all individual forecasts are simply averaged:

$$ACC = \overline{[ACC_1, ..., ACC_{nforecast}[} \tag{3}$$

The ACC ranges from -1 to 1, with 1 indicating a perfect correlation of the anomaly fields, and 0 no correlation at all.

For the evaluation of the brief ERA5-test in section 3.6, we use the mean absolute error instead of RMSE to keep in line with Dueben and Bauer (2018). The mean absolute error is defined as

$$MAE = \overline{|prediction - truth|} \tag{4}$$

3 Results

3.1 Forecast skill in a hierarchy of models

We start by analyzing the RMSE and ACC of two of the most important variables: 500hPa gepotential height (hereafter zg500) and 800hPa temperature (hereafter ta800). The first is one of the most commonly used validation metrics in NWP; the second is very close to temperature at 850hPa, which is another commonly used validation metric. We focus on the networks trained with 100 years of training data, which is the same length as in Scher (2018), and is of special interest because it is roughly the same length as is available in current century-long reanalyses like ERA-20C and the NCEP/NCAR 20CR (Compo et al., 2011; Poli et al., 2016). Figure 3 shows the global mean RMSE of network forecasts at lead times of up to 14 days for all models for both zg500 and ta800. Additionally, the skill of persistence forecasts (using the initial state as forecast) is shown with dashed lines. As expected, the skill of the network forecasts decreases monotonically with lead-time. Unsurprisingly, the network for PUMAT21_noseas – the least complex model – has the highest skill (lowest error) for all lead-times for both variables, followed by the network for PUMAT21. The network for PUMAT42, which is more complex than PUMAT21, but less complex than the two PLASIM runs, lies in between. At a lead time of 1 day, the networks for both PLASIM runs have very similar skill, but the PLASIMT42 network has higher errors at longer lead-times, despite their very similar complexity. Here we have to note that at long lead-times and near-zero ACC, RMSE can be hard to interpret since it can be strongly influenced by biases in the forecasts. When looking at the ACC instead (higher values better), the picture is very similar. The network forecasts outperform the persistence baseline (dashed lines) at nearly all lead-times, except for the PLASIMT21 and PLASIMT42 cases, where the RMSE of the network forecasts is higher from around 9 days on (depending on the variable). The periodic behaviour of the

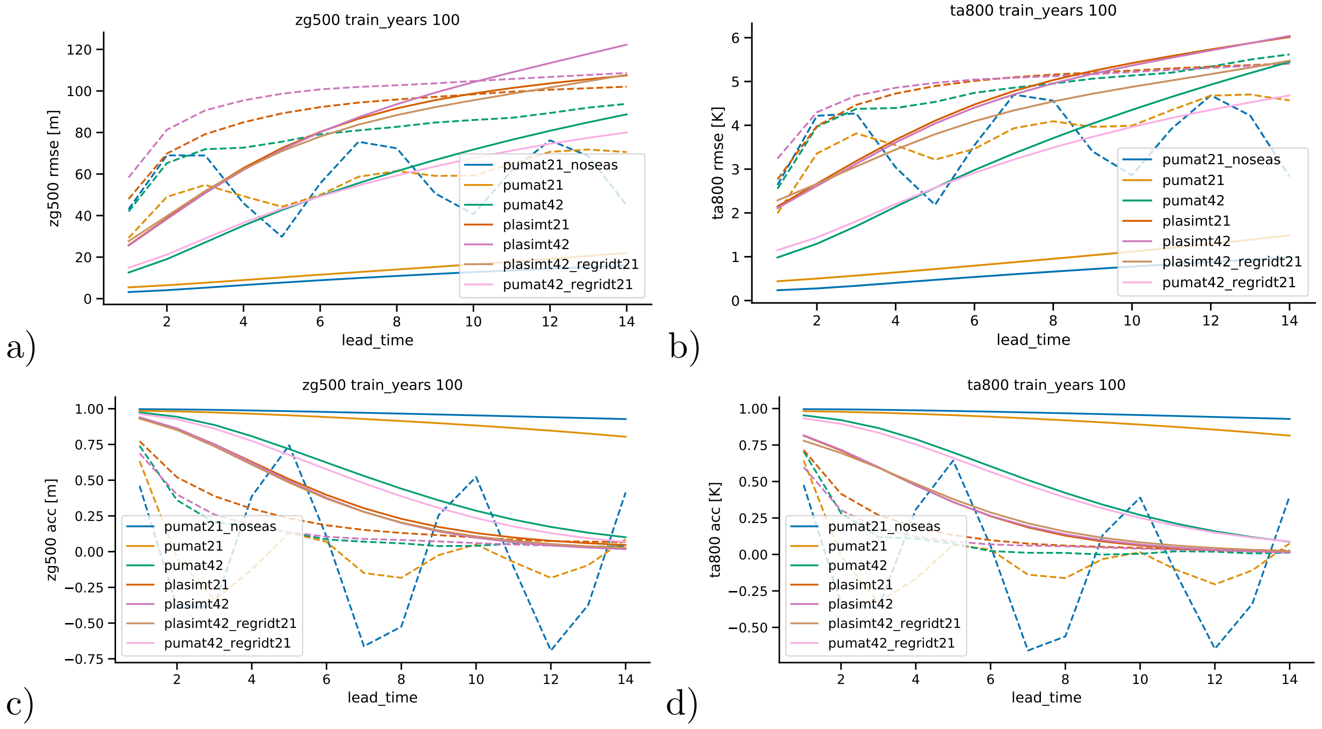

Figure 3. Root Mean Square Error (RMSE) and Anomaly Correlation Coefficient (ACC) of 500hPa gepotential height (a,c) and 800hPa temperature (b,d) for network forecasts (solid lines) and persistence forecasts (dashed lines) for all models for different lead times (in days). All forecasts are based on 100 years of training data.

persistence forecast skill for PUMA is caused by east-wards travelling rossby-waves, whose structure is relatively simple in the PUMA model. For the T42 runs that were regridded to T21 before the training the results are as follows: for PUMA, the skill of the network in predicting the regridded version of the T42 is very similar to the skill on the original T42 run. For PLASIM the skill on the regridded T42 run is comparable to both the skill on T42 and on T21 runs, albeit closer to the latter. Indeed, the skills on the original PLASIM T42 and T21 runs are much closer to each other than for PUMA. Regridding the network predictions of the two T42 runs to the T21 grid thus results in only very small changes relative to the difference between the models, especially at longer lead-times (not shown).

We next turn our attention to the spatial characteristics of the forecast error. Figure 4 shows geographical plots of the RMSE for 6-day forecasts of the networks trained with 100 years of data (the same training length as in fig. 3). In agreement with the global mean RMSE analyzed before, the network for PUMAT21_noseas has lowest errors everywhere (fig. S12 in Supplement), followed by the network for PUMAT21. The networks for PLASIMT21 and PLASIMT42 have a more complicated spatial error structure, and the mid-latitude storm-tracks emerge clearly as high-error regions. The zonally non-uniform distribution is likely

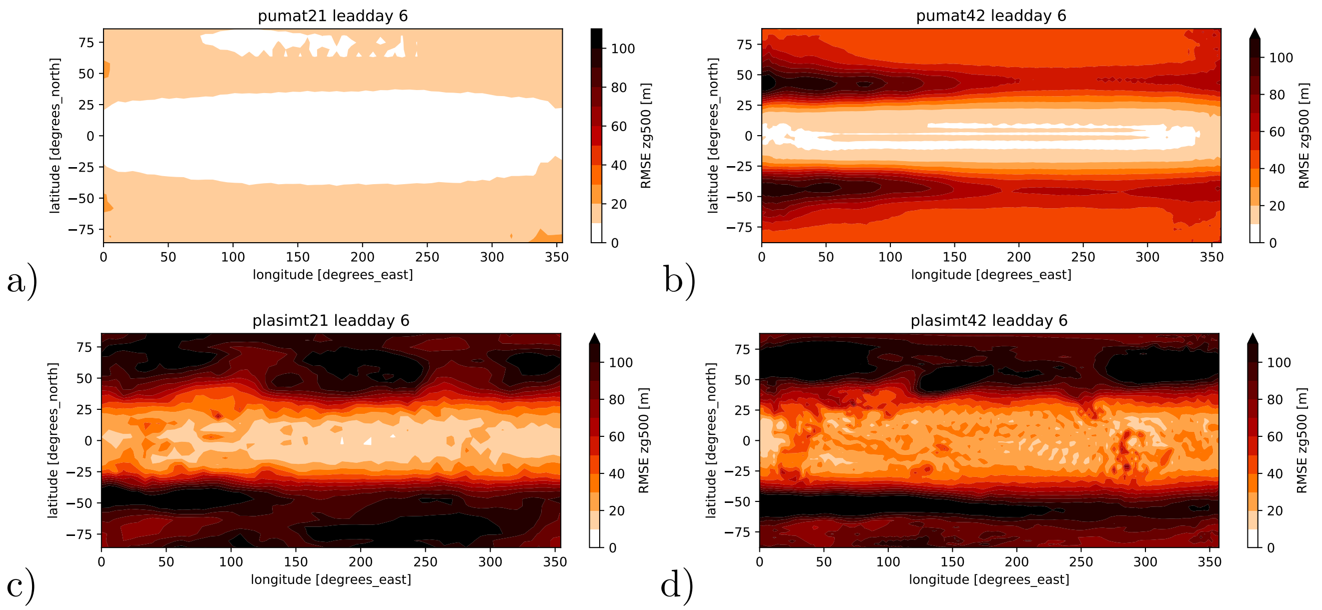

Figure 4. Maps of RMSE of the 6-day network forecasts, for the networks trained with 100 years for PUMAT21 (a), PUMAT42 (b), PLASIMT21 (c) and PLASIMT42 (d).

caused by the influence of orography (present in the PLASIM runs but not in the PUMA runs). At lead-time 1 day, the errors of the network forecasts for PUMAT42 are nearly symmetric (not shown), but at longer lead-times a zonal asymmetry emerges (fig. 4b). This is probably related to the fact that the neural network used here does not wrap around the boundaries.

3.2 Dependence of forecast skill on amount of training years

A key issue is the extent to which the above results, and more generally the skill of the network forecasts, depend on the length of the training period used. Fig. 5 shows the skill of the network forecasts for 500hPa geopotential height for different training lengths, for a lead-time of 1 day (a,c) and 6 days (b,d). As mentioned in the methods section, the networks with short training periods were trained several times with different samples from the model runs. The shading in the figure represents the uncertainty of the mean for these multi-sample networks, which is negligibly small. For the 1-day forecasts (fig. 5a, c), the results are as expected: the skill increases with an increasing number of training years, both in terms of RMSE and ACC. This increase is strongly nonlinear and, beyond ~100 years, the skill benefit of increasing the length of the training set is limited. This suggests that the complete model-space is already encompassed by around 100 years of daily data. More years will not provide new information to the network. However, it might also be the case that there is in fact more information in more years, but that the network is not able to utilize this additional information effectively. For the 6-day forecasts (fig. 5b, d), the PLASIMT21 networks display a counter-intuitive behaviour: the skill (in terms of RMSE) does not increase monotonically with increasing length of the training period, but decreases from 100 years to 200 years, while for >200 years it increases

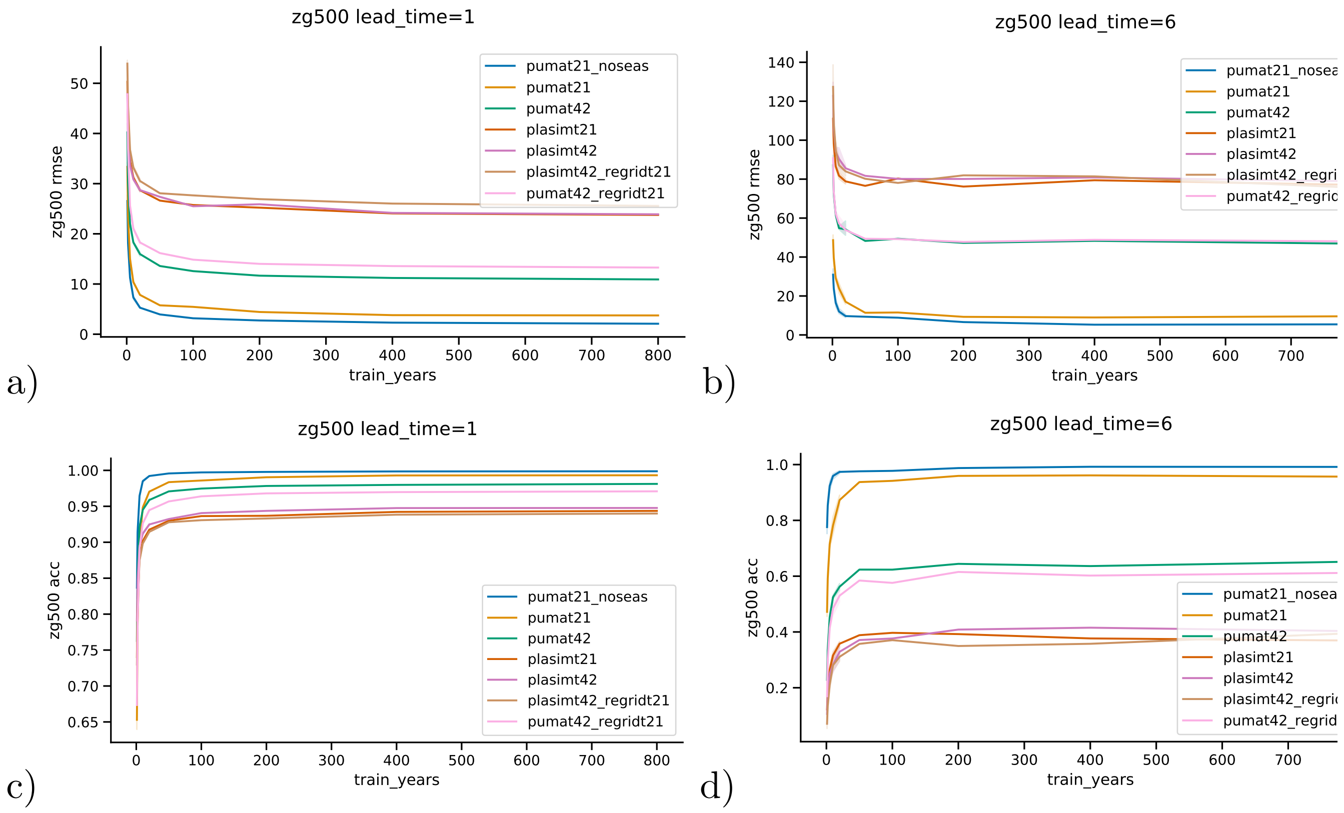

Figure 5. Dependence of network forecast skill on the length of the training period. Shown is the Root Mean Square Error (RMSE) and Anomaly Correlation Coefficient (ACC) of the network-forecasted 500hPa gepotential height for networks trained on different amounts of training years. Each line represents one model. The shading on the left side of the plots represents the 5-95 uncertainty range of the mean RMSE/ACC, estimated over networks with different training samples.

again. A similar result – albeit less pronounced – is also seen for the PLASIMT42_regridT21 networks, and also – in a slightly different form – for the skill measured via the ACC. To interpret this, one has to remember that the networks are all trained on 1-day forecasts. For 1-day forecasts the skill is indeed increasing with increasing training length, and the PUMAT21 network trained on 200 years makes better forecasts than the one trained on 100 years (fig. 5a, c). Intuitively one would assume this to translate to increased skill of the consecutive forecasts used to create the 6 day forecasts. The fact that this is not the case here might be caused by non-linear error growth. Some networks might produce slightly lower errors at lead-day 1, but those particular errors could be faster-growing than those of a network with larger day-1 errors.

3.3 Climate runs with the networks

The trained networks are not limited to forecasting the model ”weather”, but can also be used to generate a climate run starting from an arbitrary state of the climate model. For this, we use the climate networks that also include the day of year as input (see methods section). Of special interest is the question of whether the climate time-series obtained from the network is stable. In Scher (2018), the network climate for PUMAT21_noseas (using 100 years of training data) was stable and produced reasonable statistics compared to the climate model. We trained our climate networks on both 30 years and 100 years of data for all models with seasonal cycle. While the networks were all stable, they do not produce particularly realistic representations of the model climates. After some time, the storm tracks look unrealistic, and the seasonal cycle is also poorly represented (e.g. in some years, some seasons are skipped – see videos S1-S8 in the supplement that show the evolution of zg500 and zonal windspeed at 300hPa for the network climate runs). Figure 6 a) shows the evolution of ta800 at a single gridpoint at 76 °N for PUMAT21 and the climate network trained on 30 years, started from the same (randomly selected) initial state (results for different randomly selected initial states are very similar, not shown). The network is stable, but the variance is too high and some summers are left out. Surprisingly, the network for 30 years of PLASIMT21 (fig. 6 b) produced a more realistic climate. Training on 100 years instead of 30 years does not necessarily improve the quality of the climate-runs (fig. 6 c,d). For PLASIMT21, in fact, the network climate trained on 100 years is the worse performer. Interestingly, the mean climate of the PLASIMT21 network is reasonably realistic for the network trained on 100 years (fig. 7), and partly also for the network trained on 30 years (not shown), whereas the mean climates for the PLASIMT42 and PUMAT42 networks have large biases (see fig S11-17 in the Supplement).

All our networks were trained to make 1-day forecasts. The influence of the seasonal cycle on atmospheric dynamics – and especially the influence of diabatic effects – may be very small for 1-day predictions. This could make it hard for the network to learn the influence of seasonality. To test this, we repeated the training of the climate network for PLASIMT21, but now training on 5-day forecasts. This improved the seasonality of ta800 at our example gridpoint (fig. S19 in the supplement), but the spatial fields of the climate run still become unrealistic (video S9 in the supplement).

3.4 Impact of re-tuning

The design of this study was to use an already established neural network architecture – namely one tuned on a very simple model – and apply it to more complex models. However, it is of interest to know how much tuning the network architecture to the more complex models might increase forecast skill. Therefore, the same tuning procedure as in Scher (2018) for PUMAT21_noseas was repeated for PLASIMT21. Surprisingly, the resulting configuration for PLASIMT21 was exactly the same as for for PUMAT21_noseas. Thus, even with re-tuning the results would be the same. As a caveat, we note that tuning neural networks is an intricate process, and many arbitrary choices have to be made. Notably, one has to pre-define the different configurations that are tried out in the tuning (the "tuning space"). It is possible that with a different tuning space for PLASIMT21, a different configuration would be chosen than for PUMAT21_noseas. However, at least within the tuning space we used, we can conclude that a setup working well for a very simple model (PUMAT21_noseas) is also a good choice for a more complex model like PLASIMT21.

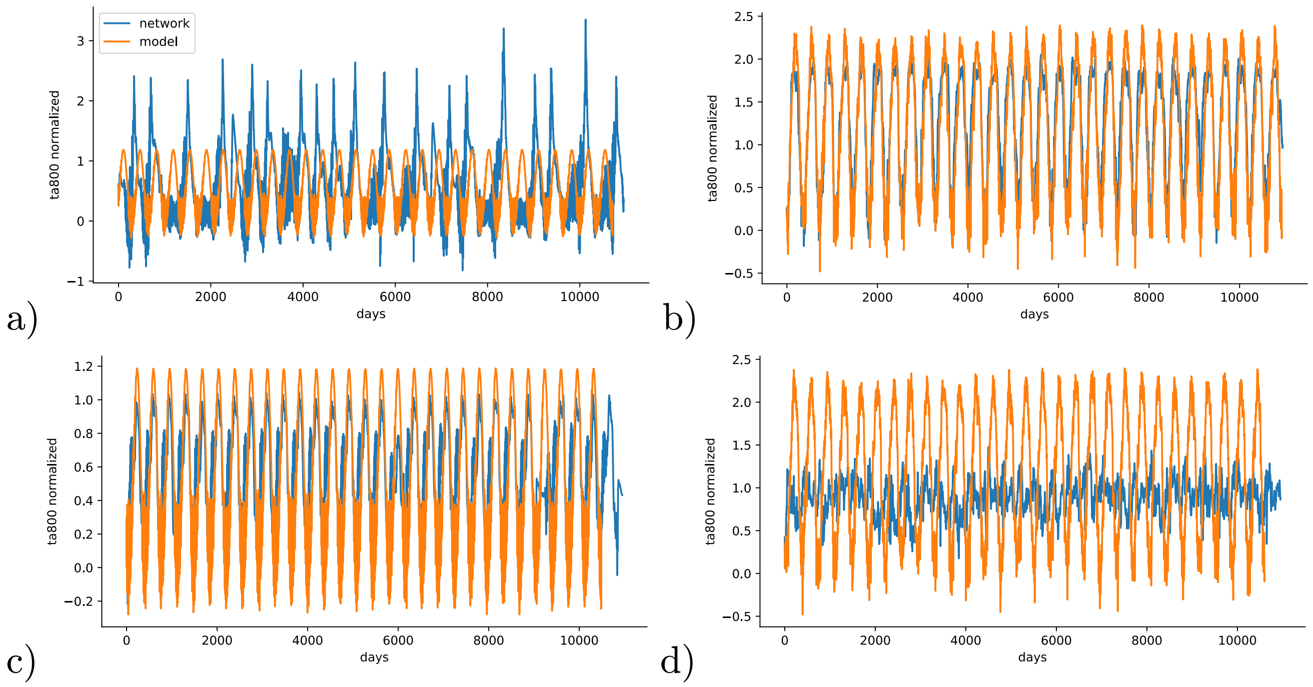

Figure 6. Evolution of daily ta800 at a single grid-point at 76°N in the GCM (orange) and in the climate network trained on the GCM (blue), started from the same initial state. The networks were trained on: 30 years of PUMAT21 (a), 30 years of PLASIMT21 (b), 100 years of PUMAT21 (c) and 100 years of PLASIMT21 (d).

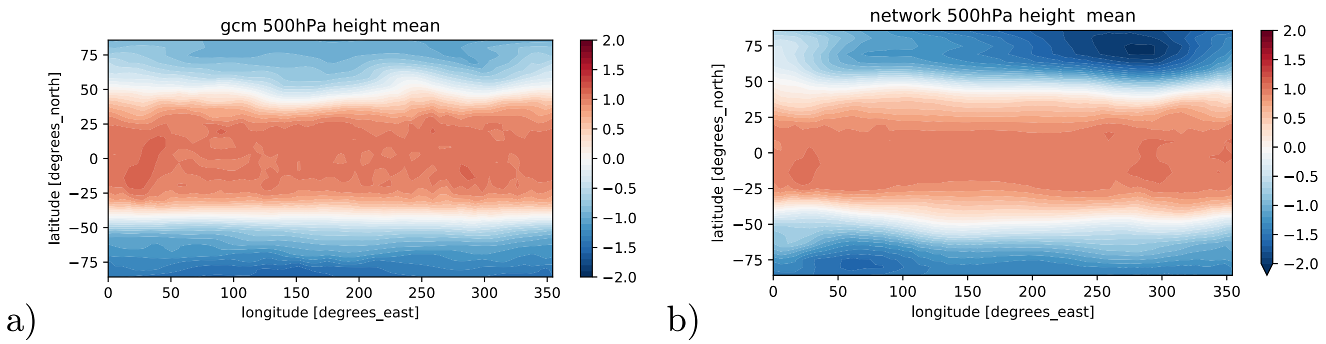

Figure 7. 30 year mean of normalized 500hPa geopotential height for PLASIMT21 (a) and the network trained on 100 years of PLASIMT21 (b).

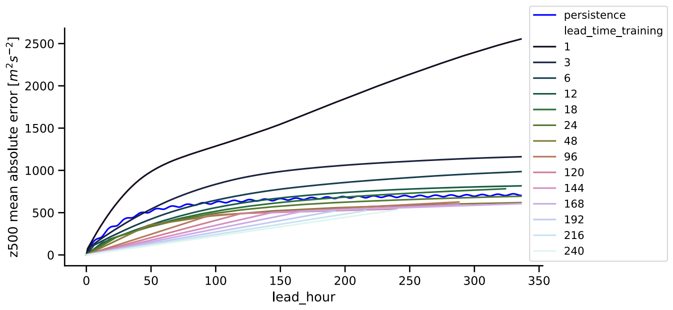

Figure 8. Mean absolute error of 500 hPa geopotential height, for networks trained on coarse-grained (T21) ERA5 data. The training was performed on different lead-times. The x-axis denotes the lead-time of the forecast in hours, the colors the lead-time used for training (in hours). Also shown is the persistence forecast (thick blue line).

3.5 Impact of including hydrological cycle as input

PLASIM, in contrast to PUMA, includes a hydrological cycle. This is represented by 3 additional 3d state variables in the model (relative humidity, cloud liquid water content and cloud cover). To the test the impact of including these variables in the neural networks, we retrained the network for 100 years of PLASIMT21 including these 3 variables (on 10 levels) as additional channels both in the input and output of the network. The network thus has 70 instead of 40 in- and output channels. Quite surprisingly, including the hydrological cycle variables slightly deteriorated the forecasts of zg500 and ta800, both in terms of RMSE and ACC, except for the ACC of ta800 from lead-times of 6 days onwards (fig. S20 in the supplement).

3.6 Performance on reanalysis data

In order to put our results in context, we also trained our network architecture on coarse-grained (regridded to T21) ERA5 (C3S, 2017) reanalysis data. We use exactly the same dataset that Dueben and Bauer (2018) used, namely 7 years (2010-2016) for training, and 8 months for evaluation (January 2017- August 2017). ERA5 is available in hourly intervals, allowing a thorough investigation of which timestep is best for training. As in Dueben and Bauer (2018), we only use zg500, and therefore deviate from the setup used in the rest of this study. We do this in order to be able to directly compare the results in the two studies. We trained networks on lead-times ranging from 1 to 240 hours, and computed the skill of the zg500 forecasts. The results are shown in fig. 8. The blue line shows the error of persistence forecasts. The network trained on 1h-forecasts is not stable and has very high forecast errors, in contrast to Dueben and Bauer who only obtained good forecasts with training on 1h lead-time. The network trained with 24h lead-time has comparable skill to the network used by Dueben and Bauer (2018) (see their fig. 3a). For longer lead-times, the networks trained on longer lead-times seem to work better. In general, it seems that if one wishes to have forecasts for a certain lead-time, it is best to train directly on that lead-time, at least for mean absolute error.

4 Discussion and Conclusions

We have tested the use of neural networks for forecasting the ‘weather’ in a range of simple climate models with different complexity. For this we have used a deep convolutional encoder-decoder architecture that Scher (2018) developed for a very simple general circulation model without seasonal cycle. The network is trained on the model in order to forecast the model state 1 day ahead. This process is then iterated to obtain forecasts at longer lead times. We also performed “climate” runs, where the network is started with a random initial state from the climate model run, and then creates a run of daily fields for several decades.

Potential improvements in the neural network architecture

In the architecture used in this paper, lat-lon grids are used to represent the fields. The convolution layers consists of a set of fixed filters that are moved across the domain. Therefore, the same filter is used close to the poles and close to the equator, even though the area represented by a single gridpoint is not the same close to the poles and close to the equator since the Earth is a globe. Using spherical convolution layers as proposed in Coors et al. (2018) would tackle this problem and may lead to improved forecasts and/or more realistic long-term simulations. The tuning of the network architecture used here tuned the depth of the convolution layers, but not the actual number of convolution layers. Therefore, it would be interesting to explore whether deeper networks (more convolution layers) could improve the performance of the networks. Other possible improvements would be to:

– Include one or more locally connected layers in addition to the convolution layers. While they can increase problems related with overfitting, locally connected layers could learn "local dynamics".

– Include spatial dependence in height. In our architecture, all variables at all heights are treated as individual channels. One could for example group variables at one height together, or use 3d-convolution.

Our results further support the idea – already proposed by Dueben and Bauer (2018) and Scher (2018) – of training a neural network on the large amount of existing climate model simulations, feeding the trained network with today’s analysis of a NWP model and using the network to make a weather forecast. The high computational efficiency of such neural network forecasts would open up new possibilities, for example of creating ensemble forecasts with much large ensemble sizes than the ones available today. Therefore, this approach would provide an interesting addition to current weather forecasting practice, and also a good example of exploiting the large amount of existing climate model data for new purposes.

Networks for weather forecasts and climate runs

One of the aims of this study was to assess whether it is possible to use a simplified reality – in this case a very simple GCM without seasonal cycle – to develop a method that also works on more complex GCMs. We showed that, for the problem of forecasting the model ’weather’, this seems to be the case: the network architecture developed by Scher (2018) also worked on the more complex models used here, albeit with lower skill. The latter point is hardly surprising, as one would expect the time-evolution of the more complex models to be harder to predict. The neural network forecast outperformed a simple persistence forecast baseline at most lead-times also for the more complex models. The fact that we can successfully forecast the weather in a range of simple GCMs a couple of days ahead is an encouraging result for the idea of weather forecasting with neural networks. We also tried to re-tune the network architecture from Scher (2018) to one of our more complex models. Surprisingly, the best network configuration that came out of the tuning procedure was exactly the same as the one obtained for the simpler model in Scher (2018). This further supports the idea that methods developed on simpler models may be fruitfully applied in more complex settings. Additionally, we tested the network architecture on coarse-grained reanalysis data, where its skill is comparable to the method proposed by Dueben and Bauer (2018). We also found that, in contrast to the findings of Dueben and Bauer (2018), it seems possible to make valid neural network forecasts of atmospheric dynamics using a wide range of timesteps (also much longer than 1 hour). This discrepancy might be caused by our use of convolutional layers, in contrast to the local deep neural network that Dueben and Bauer used. A stack of convolution layers may be interpreted as multiple layers, where each layer could possibly make a short-term forecast, and the whole stack a long-term forecast.

The second problem we addressed was using the trained networks to create "climate" runs. Scher (2018) found this generated a stable climate for the simplest model considered here, which does not have a seasonal cycle. Here, we find that this is to some extent also possible for more complex models. However, even when training on relatively long periods (100 years), the climates produced by the networks have some unrealistic properties, such as a poor seasonal cycle, significant biases in the long-term mean values and often unrealistic storm tracks. The fact that these problems don't occur for the simplest GCM without seasonal cycle, but do occur for the same GCM with seasonal cycle, indicates that seasonality considerably complicates the problem. While not a solution for creating climate runs, this suggests that for the weather forecasting problem it might be interesting to train separate networks for different times of the year (e.g. one for each month).

Code and data availability. The code developed and used for this study is available in the accompanying repository at http://doi.org/10.5281/zenodo.2572863. All external libraries used here are open source. The trained networks and the data underlying the all plots are available in the repository. The model runs can be recreated with the control files (available in the repository) and the source code of PUMA/PLASIM, which is freely available at https://www.mi.uni-hamburg.de/en/arbeitsgruppen/theoretische-meteorologie/modelle/PLASIM.html ERA-Interim data is freely avialable at https://apps.ecmwf.int/datasets/data/interim-full-daily/levtype=pl/. ERA5-data is freely availableat https://climate.copernicus.eu/climate-reanalysis

Appendix A

A1 Computation of the local dimension

Here, we outline very briefly how the local dimension d is computed. To foster easy reproducibility, we present the computation in an algorithm-like fashion, as opposed to formal mathematical notation. For a more rigorous theoretical explanation the reader is referred to Faranda et al. (2019). The code is available in the repository accompanying this paper (see Code and data availability).

First, we define the distance between the 2-D atmospheric fields at times t_1 and t_2 as:

$$dist_{t_1,t_2} = sqrt\left((x_{t_1,1} - x_{t_2,1})^2 + (x_{t_1,2} - x_{t_2,2})^2 + ... + \left(x_{t_1,N_j} - x_{t_2,N_j}\right)^2\right) \tag{A1}$$

where j is the linear gridpoint index and N_j is the total number of gridpoints. To compute d_t, namely the local dimension of a field at time t, we first take the negative natural logarithm of the distances between t and all other timesteps t_i (i.e. all times before and after t):

$$g_{t,t_i} = -ln\left(dist_{t,t_i}\right) \tag{A2}$$

and then retain only the distances that are above the 98th percentile of g_{t,t_i}:

$$exceedances = g_{t,t_i} - 98^{th} percentile(g_{t,t_i})\,\forall\, g_{t,t_i} > 98^{th} percentile(g_{t,t_i}) \tag{A3}$$

These are effectively logarithmic returns in phase space, corresponding to cases where the field x_{t_i} is very close to the field x_t. According to the Freitas-Freitas-Todd theorem (Freitas et al., 2010), modified in (Lucarini et al., 2012), the probability of such logarithmic returns (in the limit of an infinitely long timeseries) follows the exponential member of the generalized Pareto distribution (Pickands III et al., 1975). The local dimension d_t can then be obtained as the inverse of the distribution's scale parameter, which can also be expressed as the inverse of the mean of the exceedances:

$$d_t = 1/mean\left(exceedances\right) \tag{A4}$$

The local dimension is an instantaneous metric, and Faranda et al. (2019) have shown that time-averaging of the data can lead to counter-intuitive effects. The most robust approach is therefore to compute d on instantaneous fields. Here, in order to use the same data as for the machine learning, we have used daily means. We have verified that, at least in ERA-Interim, this has a negligible effect on the average d value for all variables except for geopotential at 100hPa and 200hPa (not shown).

Author contributions. SS has conceived the study and methodology, performed the analysis, produced the software and drafted the article. GM contributed to drafting and revising the article.

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

Acknowledgements. We thank Peter Dueben for providing the ERA5 data. S. Scher has been funded by the Department of Meteorology (MISU) of Stockholm University. G. Messori was partly supported by the Swedish Research Council Vetenskapsrådet (grant no. 2016-03724). The computations were done 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

Bauer, P., Thorpe, A., and Brunet, G.: The quiet revolution of numerical weather prediction, Nature, 525, 47–55, https://doi.org/10.1038/nature14956, https://www.nature.com/articles/nature14956, 2015.

Buschow, S. and Friederichs, P.: Local dimension and recurrent circulation patterns in long-term climate simulations, arXiv preprint arXiv:1803.11255, 2018.

C3S: ERA5: Fifth generation of ECMWF atmospheric reanalyses of the global climate. Copernicus Climate Change Service Climate Data Store (CDS), https://cds.climate.copernicus.eu/cdsapp#!/home, 2017.

Compo, G. P., Whitaker, J. S., Sardeshmukh, P. D., Matsui, N., Allan, R. J., Yin, X., Gleason, B. E., Vose, R. S., Rutledge, G., Bessemoulin, P., and others: The twentieth century reanalysis project, Quarterly Journal of the Royal Meteorological Society, 137, 1–28, 2011.

Coors, B., Paul Condurache, A., and Geiger, A.: Spherenet: Learning spherical representations for detection and classification in omnidirectional images, in: Proceedings of the European Conference on Computer Vision (ECCV), pp. 518–533, 2018.

Dee, D. P., Uppala, S. M., Simmons, A., Berrisford, P., Poli, P., Kobayashi, S., Andrae, U., Balmaseda, M., Balsamo, G., Bauer, d. P., et al.: The ERA-Interim reanalysis: Configuration and performance of the data assimilation system, Quarterly Journal of the royal meteorological society, 137, 553–597, 2011.

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.

Faranda, D., Messori, G., and Vannitsem, S.: Attractor dimension of time-averaged climate observables: insights from a low-order ocean-atmosphere model, Tellus A: Dynamic Meteorology and Oceanography, 71, 1–11, https://doi.org/10.1080/16000870.2018.1554413, https://doi.org/10.1080/16000870.2018.1554413, 2019.

Fraedrich, K., Jansen, H., Kirk, E., Luksch, U., and Lunkeit, F.: The Planet Simulator: Towards a user friendly model, Meteorologische Zeitschrift, 14, 299–304, https://doi.org/doi:10.1127/0941-2948/2005/0043, https://www.ingentaconnect.com/content/schweiz/mz/2005/00000014/00000003/art00001, 2005.

Freitas, A. C. M., Freitas, J. M., and Todd, M.: Hitting time statistics and extreme value theory, Probability Theory and Related Fields, 147, 675–710, 2010.

Johnson, N.: Simply complexity: A clear guide to complexity theory, Oneworld Publications, 2009.

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.

Lucarini, V., Faranda, D., and Wouters, J.: Universal behaviour of extreme value statistics for selected observables of dynamical systems, Journal of statistical physics, 147, 63–73, 2012.

McGovern, A., Elmore, K. L., Gagne, D. J., Haupt, S. E., Karstens, C. D., Lagerquist, R., Smith, T., and Williams, J. K.: Using Artificial Intelligence to Improve Real-Time Decision-Making for High-Impact Weather, Bulletin of the American Meteorological Society, 98, 2073–2090, https://doi.org/10.1175/BAMS-D-16-0123.1, https://doi.org/10.1175/BAMS-D-16-0123.1, 2017.

Nooteboom, P. D., Feng, Q. Y., López, C., Hernández-García, E., and Dijkstra, H. A.: Using network theory and machine learning to predict El Niño, Earth System Dynamics, 9, 969–983, https://doi.org/10.5194/esd-9-969-2018, https://www.earth-syst-dynam.net/9/969/2018/, 2018.

O'Gorman, P. A. and Dwyer, J. G.: Using Machine Learning to Parameterize Moist Convection: Potential for Modeling of Climate, Climate Change, and Extreme Events, Journal of Advances in Modeling Earth Systems, 10, 2548–2563, https://doi.org/10.1029/2018MS001351, https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018MS001351, 2018.

Pickands III, J. et al.: Statistical inference using extreme order statistics, the Annals of Statistics, 3, 119–131, 1975.

Poli, P., Hersbach, H., Dee, D. P., Berrisford, P., Simmons, A. J., Vitart, F., Laloyaux, P., Tan, D. G. H., Peubey, C., Thépaut, J.-N., Trémolet, Y., Hólm, E. V., Bonavita, M., Isaksen, L., and Fisher, M.: ERA-20C: An Atmospheric Reanalysis of the Twentieth Century, Journal of Climate, 29, 4083–4097, https://doi.org/10.1175/JCLI-D-15-0556.1, https://journals.ametsoc.org/doi/abs/10.1175/JCLI-D-15-0556.1, 2016.

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.

Schneider, T., Lan, S., Stuart, A., and Teixeira, J.: Earth System Modeling 2.0: A Blueprint for Models That Learn From Observations and Targeted High-Resolution Simulations, Geophysical Research Letters, 44, 12,396–12,417, https://doi.org/10.1002/2017GL076101, https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017GL076101, 2017.

Weather and climate forecasting with neural networks: using GCMs with different complexity as study-ground

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. Recently, there has been growing interest in the possibility of using neural networks for both weather forecasting and the generation of climate datasets. We use a bottom-up approach for assessing whether it should, in principle, be possible to do this. We use the relatively simple General Circulation Models (GCMs) PUMA and PLASIM as a simplified reality on which we train deep neural networks, which we then use for predicting the model weather at lead times of a few days. We specifically assess how the complexity of the climate model affects the neural network's forecast skill, and how dependent the skill is on the length of the provided training period. Additionally, we show that using the neural networks to reproduce the climate of general circulation models including a seasonal cycle remains challenging ~~-~~– in contrast to earlier promising results on a model without seasonal cycle.

1 Introduction

Synoptic weather forecasting (forecasting the weather at lead times of a few days up to 2 weeks), has for decades been dominated by computer models based on physical equations ~~-~~– the so called Numerical Weather Predictions (NWP) models. The quality of NWP forecasts has been steadily increasing since their inception (Bauer et al., 2015), and these models remain the backbone of virtually all weather forecasts. However, the fundamental nature of the weather forecasting problem can be summarized as: "starting from today's state of the atmosphere, we want to predict the state of the atmosphere x days in the future". Thus posed, the problem is a good candidate for supervised machine learning. While long thought unfeasible, the recent success of machine learning techniques in highly complex fields such as image and speech recognition warrants a review of this possibility. Machine learning techniques have already been used to improve certain components of NWP and climate models – mainly parameterization schemes (Krasnopolsky and Fox-Rabinovitz (2006); Rasp et al. (2018); Krasnopolsky et al. (2013); O'Gorman and Dwyer (2018)), to aid real-time decision making McGovern et al. (2017), to exploit observations and targeted high-resolution simulations to enhance earth system models (Schneider et al., 2017), for El-Niño predictions (Nooteboom et al., 2018) and to predict weather forecast uncertainty (Scher and Messori, 2018).

~~Recently, the ambition has shifted from~~

Recently, in addition to using machine-learning to enhance numerical models~~to using~~, there have been ambitions to use it to tackle weather forecasting itself. The holy grail is to use machine-learning, and especially "deep learning", to completely ~~replacing~~ replace NWP models, although opinions may diverge on if and when this will happen. Additionally, it is an appealing idea to use neural networks/deep learning to emulate very expensive General Circulation Models (GCMs) for climate research. Both these ideas have been tested with some success for simplified realities (Dueben and Bauer, 2018; Scher, 2018). In Scher (2018), a neural network approach was used to skillfully forecast the "weather" of a simplified climate model, as well as emulate its climate. ~~Dueben and Bauer~~Dueben and Bauer (2018), based on their success in forecasting reanalysis data regridded to very low resolution, concluded that it is "fundamentally" possible to produce deep-learning based weather forecasts.

Here, we build upon the approach from Scher and apply it to a range of climate models with different complexity. We do this in order to assess: 1) how the skill of the neural network weather forecasts depends on the available amount of training data; 2) how this skill depends on the complexity of the climate models; and 3) under which conditions it may be possible to make stable climate simulations with the trained networks, and how this depends on the amount of available training data. (1) is of special interest for the idea of using historical observations in order to train a neural network for weather forecasting. As the length of historical observations is strongly constrained (~ 100 years for long renalyses assimilating only surface observations, and ~40 years for reanalyses assimilating satellite data), it is crucial to assess how many years of past data one would need to produce meaningful forecasts. The value of (2) ~~Is of interest because it evaluates~~ lies in evaluating the feasibility of ~~using~~ climate models as a "simplified reality" for studying weather forecasting with neural networks. Finally, (3) is of interest when one wants to use neural networks not only for weather forecasting, but for the distinct yet related problem of seasonal and longer forecasts, up to climate projections.

To avoid confusion, we use the following naming conventions throughout the paper: "model" always refers to physical models (i.e. the climate models used in this study), and will never refer to a "machine learning model". The neural networks are referred to ~~as~~ by the term "network".

2 Methods

2.1 Climate models

To create long climate model runs of different complexity, we used the Planet Simulator (PLASIM) intermediate complexity GCM, and its dry dynamical core: the Portable University Model of the Atmosphere (PUMA) (Fraedrich et al., 2005). Each model was run for 830 years with two different horizontal resolutions (T21 and T42, corresponding to ~5.65 and ~2.8 degrees latitude, respectively) and 10 vertical levels. The first 30 years of each run were discarded as ~~spinup~~spin-up, leaving 800 years of daily data for each model. The runs will from now on be referred to as PLASIMT21, PLASIMT42, PUMAT21 and PUMAT42. All the model runs produce a stable climate without drift (fig. S1-S10 in the Supplement). Additionally, we regridded PLASIMT42 and PUMAT42 (with bi-linear interpolation) to a resolution of T21. These will be referred to as PLASIMT42_regridT21 and PUMAT42_regridT21.

The PUMA runs do not include ocean and orography and use Newtonian cooling for diabatic heating/cooling. The PLASIM runs include orography, but no ocean-model. The main difference between PLASIM and standard GCMs is that sub-systems of the Earth system other than the atmosphere (e.g. the ocean, sea-ice and soil) are reduced to systems of low complexity. We used the default integration timestep for all four model setups, namely 30 min for PLASIMT42, 20min for PLASIMT21, and 60 min for PUMAT21 and PUMAT42. We regrid all fields to regular lat-lon grids on pressure levels for analysis. An additional run was made with PUMA at resolution T21, but with the seasonal cycle switched off (eternal boreal winter). This is the same configuration as used in Scher (2018) and will be referred to as PUMAT21_noseas.

In order to contextualise our results relative to previous studies, we also run a brief trial of our networks on ERA5 (C3S, 2017) reanalysis data (Section 3.5 , similar to Dueben and Bauer (2018).

2.2 Complexity

Ranking climate models according to their "complexity" is a non-trivial task, as it is very hard to define what complexity actually means in this context. We note that here we use the term loosely, and do not refer to any of the very precise definitions of "complexity" that exists in various scientific fields (e.g. Johnson (2009)). Intuitively, one might simply rank the models according to their horizontal and vertical resolutions and the number of physical processes they include. However, it is not clear which effects would be more important (e.g. is a model with higher resolution but less components/processes more or less complex than a low-resolution model with a larger number of processes?). Additionally, more physical processes do not necessarily imply a more complex *output*. For example, very simple models like the Lorenz63 model (Lorenz, 1963) display chaotic behaviour, yet it is certainly possible to design a model with more parameters and a deterministic behaviour.

To circumvent this conundrum, we adopt a very pragmatic approach based solely on the output of the models, and grounded in dynamical systems theory. We quantify model complexity in terms of the the local dimension d: a measure of the number of degrees of freedom ~~needed to describe the dynamics~~ of a system ~~linearized~~ active locally around a given instantaneous state. In our case, this means that we can compute a value of d for every timestep in a given model simulation. While not a measure of complexity in the strict mathematical or computational senses of the term, d provides an objective indication of a system's dynamics around a given state and, when averaged over a long timeseries, of the system's average attractor dimension. An example of how d may be computed for climate fields is provided in Faranda et al. (2017), while for a more formal discussion and derivation of d we point the reader to Appendix A in Faranda et al. (2019). The approach is very flexible, and may be applied to individual variables of a system (which represent projections of the full phase-space dynamics onto specific sub-

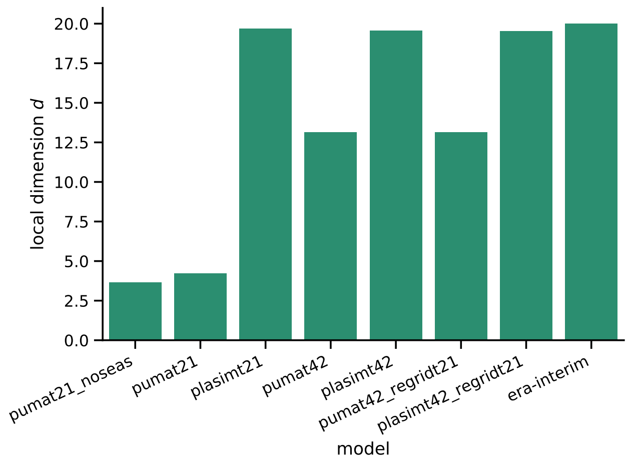

Figure 1. ~~Local~~ Averages of the local dimension d (here used as a data-based measure of "complexity") for the 500hPa geopotential height of the models used in this study and of the ERA-Interim reanalysis.

spaces, called Poincaré sections), multiple variables or, with adequate computational resources, to the whole available dataset. The exact algorithm used here is outlined in Appendix A1.

The local dimension was computed for 38 years of each model run, as well as for the ERA-Interim reanalysis on a 1x1 degree grid over 1979-2016 (Dee et al., 2011). The choice of 38 years was made because this is the amount of available years in ERA-Interim, and the length of the dataseries can affect the estimate of d (Buschow and Friederichs, 2018). Figure 1 shows the results for 500hPa geopotential height. The complexity of PUMA increases with increasing resolution, whereas both the low and the high resolution PLASIM model have a complexity approaching that of ERA-Interim. Thus ~~-~~ – at least by this measure ~~-~~ – they are comparable to the real atmosphere. The high resolution runs regridded to T21 have nearly the same complexity as the original high resolution runs. The ranking is the same for nearly all variables and levels (fig. S11 in Supplement). For the rest of the paper, the term "complexity" or "complex" always refers to the local dimension d.

2.3 Neural networks

Neural networks are in principle a series of non-linear functions with weights determined through training on data. Before the training, one has to decide the *architecture* of the network. Here, we use the architecture proposed by ~~(Scher, 2018)~~Scher (2018), which is a convolutional ~~autoencoder~~decoder-encoder, taking as input 3d model fields and outputting 3d model fields of exactly the same dimension. It was designed and tuned in order to work well on PUMAT21 without seasonality (for details see Scher (2018)). In order to ~~keep the method comparable, for~~ ease comparison with previous results, in the main part of this study ~~, no further tuning is done here, and~~ we use the same network layout and hyperparameters as in (Scher, 2018), except for the number of epochs (iterations over the training set) the network is trained over. In the original configuration only 10 epochs were used. It turned out that, especially for the more complex models, the training was not saturated after 10 epochs. Therefore, here we train until the skill on the validation data has not increased for 5 epochs, with a maximum of 100 epochs. The layout is depicted in fig. 2. The ~~implications of retuning the network~~ possibility of re-tuning the network and its implications are discussed in section 3.4~~Architecture of the neural network for the models with resolution T21 (left) and T42 (right). Figure based on Fig. S1 from (Scher, 2018)~~

. For the networks targeted ~~not to forecast the weather, but~~ to create climate simulations (hereafter called climate-networks) we deviate from this setup: here, we include the day of year as additional input to the network, in the form of a separate input channel. To remain consistent with the ~~auto-encoder~~ encoder-decoder setup, the output also contains the layer with the day of year. However, when producing the network climate runs, the predicted day of the year is discarded.

The last 10% samples of the training data are used for validation. This allows to monitor the training progress, control over-fitting (the situation where the network works very well on the training data, but very poorly on the test data), and potentially limit the maximum number of training epochs. As input to the neural networks we use 4 variables (u, v, t and z) at 10 pressure levels~~are used, whereas~~ ; each variable at each level is represented as a separate input layer (channel). These 4 variables represent the full state of the PUMA model. PLASIM has 3 additional atmospheric variables related to the hydrological cycle (relative humidity, cloud liquid water content and cloud cover). In order to keep the architecture the same, these are not included in the standard training, and only used for a test in section 3.5.

All networks are trained to make 1-day forecasts. Longer forecasts are made by iteratively feeding back the forecast into the network. We did not train the network directly on longer lead-times, based on the finding of Dueben and Bauer (2018) that it is easier to make multiple short forecasts compared to a single long one. Due to the availability of model data and in keeping with Scher (2018), we chose 1-day forecasts as opposed to the shorter forecast step (1 hour) in Dueben and Bauer (2018).

For each model, the network was trained with a set of 1, 2, 5, 10, 20, 50, 100, 200, 400 and 800 years. Since with little training data the network is less constrained, and the training success might strongly depend on exactly which short period out of the model run is chosen, the training for periods up to and including 20 years were repeated 4 times, shifting the start of the training data by 10, 20, 30 and 40 years. The impact of the exact choice of training period ~~will be~~ is discussed where appropriate.

All the analyses shown in this paper are performed on the forecasts made on the first 30 years of the model run, which were never used during training and therefore provide objective scores (the 'test' dataset).

2.4 Metrics

~~As validation for the network forecasts, we use two commonly~~ All network forecasts are validated against the underlying model the network was trained on (e.g. for the forecasts of the network trained on PLASIMT21 the “truth” is the evolution of the

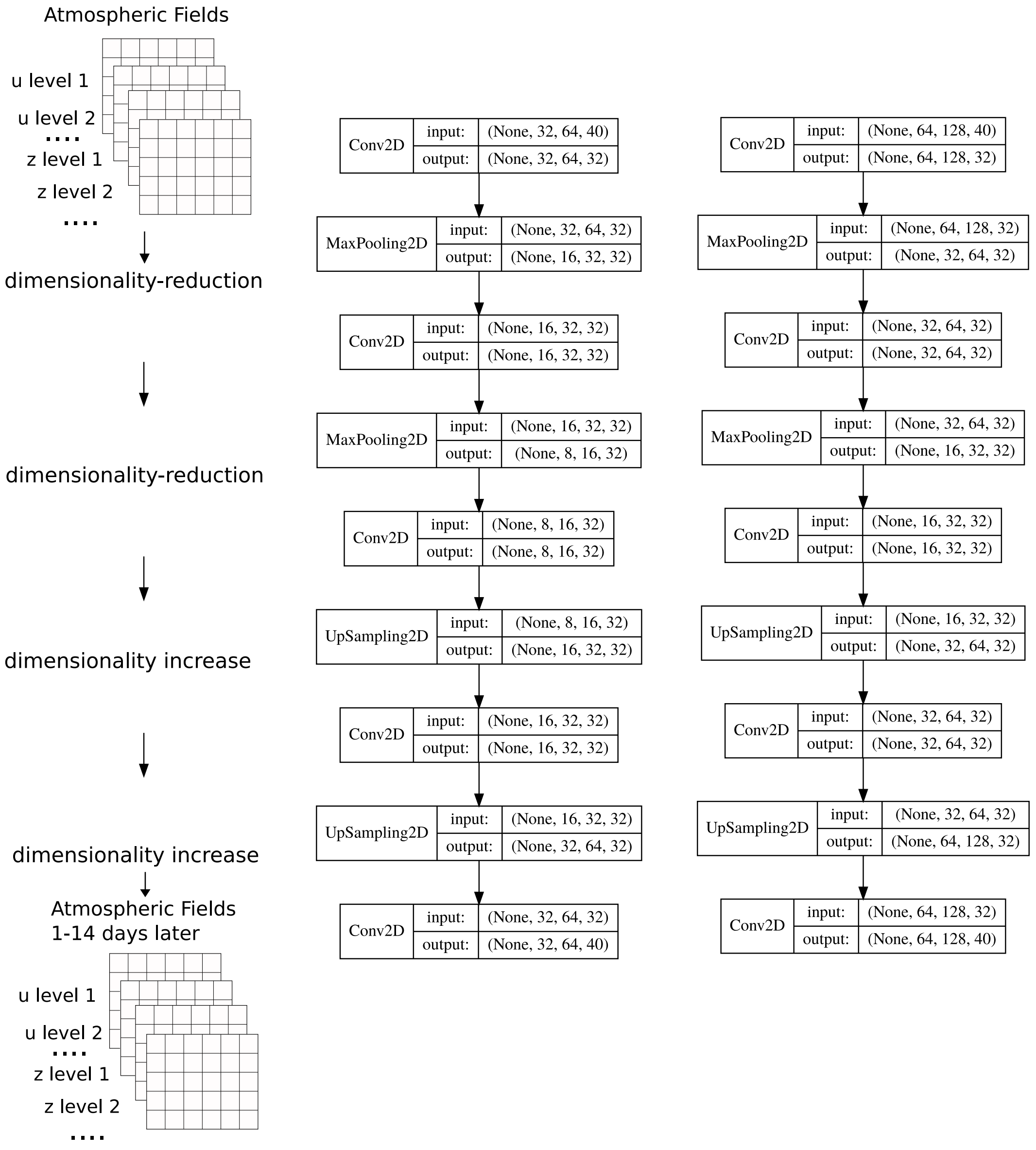
Atmospheric Fields
u level 1
u level 2
....
z level 1
z level 2
....
dimensionality-reduction
dimensionality-reduction
dimensionality increase
dimensionality increase
Atmospheric Fields
1-14 days later
u level 1
u level 2
....
z level 1
z level 2
....
Conv2D | input: (None, 32, 64, 40) | output: (None, 32, 64, 32)
MaxPooling2D | input: (None, 32, 64, 32) | output: (None, 16, 32, 32)
Conv2D | input: (None, 16, 32, 32) | output: (None, 16, 32, 32)
MaxPooling2D | input: (None, 16, 32, 32) | output: (None, 8, 16, 32)
Conv2D | input: (None, 8, 16, 32) | output: (None, 8, 16, 32)
UpSampling2D | input: (None, 8, 16, 32) | output: (None, 16, 32, 32)
Conv2D | input: (None, 16, 32, 32) | output: (None, 16, 32, 32)
UpSampling2D | input: (None, 16, 32, 32) | output: (None, 32, 64, 32)
Conv2D | input: (None, 32, 64, 32) | output: (None, 32, 64, 40)
Conv2D | input: (None, 64, 128, 40) | output: (None, 64, 128, 32)
MaxPooling2D | input: (None, 64, 128, 32) | output: (None, 32, 64, 32)
Conv2D | input: (None, 32, 64, 32) | output: (None, 32, 64, 32)
MaxPooling2D | input: (None, 32, 64, 32) | output: (None, 16, 32, 32)
Conv2D | input: (None, 16, 32, 32) | output: (None, 16, 32, 32)
UpSampling2D | input: (None, 16, 32, 32) | output: (None, 32, 64, 32)
Conv2D | input: (None, 32, 64, 32) | output: (None, 32, 64, 32)
UpSampling2D | input: (None, 32, 64, 32) | output: (None, 64, 128, 32)
Conv2D | input: (None, 64, 128, 32) | output: (None, 64, 128, 40)

Figure 2. Architecture of the neural network for the models with resolution T21 (left) and T42 (right). Each box describes a network layer, with the numbers indicating the dimension of (None, lat, lon, level). "None" refers to the time-dimension that is not relevant here, but which we include in the schematic since it is part of the basic notation of the used software library. Figure based on Fig. S1 from Scher (2018).

PLASIMT21 run). We specifically adopt two widely used forecast verification metrics, namely the Root Mean Square Error (RMSE) and the Anomaly Correlation Coefficient (ACC). The RMSE is defined as:

$$RMSE = \sqrt{\overline{(prediction - truth)^2}} \tag{1}$$

where the overbar denotes a mean over space and time (for global measures), or over time only (for single gridpoints).

The ACC measures the spatial correlation of the forecast anomaly fields with the true anomaly fields for a single forecast. The anomalies are computed with respect to a 30-day running climatology, computed on 30 years of model data (similar to how ~~ECMWF computes it's~~ the European Centre for Medium Range Weather Forecasts computes its scores for forecast validation).

$$ACC_t = correlation$$

$$\Big([truth_{1,1} - clim_{1,1}, ..., truth_{nlat,nlon} - clim_{nlat,nlon}], \Big[prediction - clim prediction_{1,1} - clim_{1,1}, ..., prediction - clim predicti$$

(2)

To compute a score over the whole period, the ACC for all individual forecasts are simply averaged:

$$ACC = \overline{[ACC_1, ..., ACC_{nforecast}[} \tag{3}$$

The ACC ranges from -1 to 1, with 1 indicating a perfect correlation of the anomaly fields, and 0 no correlation at all.

For the evaluation of the brief ERA5-test in section 3.6, we use the mean absolute error instead of RMSE to keep in line with Dueben and Bauer (2018). The mean absolute error is defined as

$$MAE = \overline{|prediction - truth|} \tag{4}$$

3 Results

3.1 Forecast skill in a hierarchy of models

We start by analyzing the RMSE and ACC of two of the most important variables: 500hPa gepotential height (hereafter ~~named~~ zg500) and 800hPa temperature (hereafter ~~called~~ ta800). The first is one of the most commonly used validation metrics in NWP; the second is very close to temperature at 850hPa, which is another commonly used validation metric. We focus on the networks trained with 100 years of training data, which is the same length as in Scher (2018), and is of special interest because it is roughly the same length as is available in current century-long reanalyses like ERA-20C and the NCEP/NCAR 20CR (Compo et al., 2011; Poli et al., 2016). Figure 3 shows the global mean RMSE of network forecasts at lead times of up to 14 days for all models for both zg500 and ta800. Additionally, the skill of persistence forecasts (using the initial state as forecast) is shown with dashed lines. As expected, the skill of the network forecasts decreases monotonically with lead-time. Unsurprisingly, the network for PUMAT21_noseas ~~-~~ – the least complex model ~~-~~ – has the highest skill (lowest error) for all lead-times for both variables, followed by the network for PUMAT21. The network for PUMAT42, which is more complex than PUMAT21, but less complex than the two PLASIM runs, lies ~~as expected~~ in between. ~~Interestingly, at~~ At a lead time of 1 day, the networks for both PLASIM runs have very similar skill, but ~~PLASIM42~~ the PLASIMT42 network has higher errors at longer lead-times, despite their very similar complexity. Here we have to note that at long lead-times and near-zero ACC, RMSE can be hard to interpret since it can be strongly influenced by biases in the forecasts. When looking at the ACC instead (higher values better), the picture is very similar~~, except that for zg500, PLASIMT42 has slightly lower skill than~~ . The network forecasts outperform the persistence baseline (dashed lines) at nearly all lead-times, except for the PLASIMT21 and PLASIMT42 cases, where the RMSE of the network forecasts is higher from around 9 days on (depending on the variable). The periodic behaviour of the persistence forecast skill for PUMA is caused by east-wards travelling rossby-waves, whose structure is relatively simple in the PUMA model. For the T42 runs that were regridded to T21 before the training the results are as follows: for PUMA, the skill of the network in predicting the regridded version of the T42 is very similar to the skill on the original T42 run. For PLASIM the skill on the regridded T42 run is comparable to both the skill on T42 and on T21 runs, albeit closer to the latter. Indeed, the skills on the original PLASIM T42 and T21 runs are much closer to each other than for PUMA. Regridding the network predictions of the two T42 runs to the T21 grid thus results in only very small changes relative to the difference between the models, especially at longer lead-times (not shown).

We next turn our attention to the spatial characteristics of the forecast error. Figure 4 shows geographical plots of the RMSE for 6-day forecasts of the networks trained with 100 years of data (the same training length as in fig. 3). In agreement with the global mean RMSE analyzed before, the network for PUMAT21_noseas has lowest errors everywhere (fig. S12 in Supplement), followed by the network for PUMAT21. The networks for PLASIMT21 and PLASIMT42 have a more complicated spatial error structure, and the mid-latitude storm-tracks emerge clearly as high-error regions. The zonally non-uniform distribution is likely caused by the influence of orography (present in the PLASIM runs but not in the PUMA runs). ~~The smaller but sill present zonal non-uniformity in~~ At lead-time 1 day, the errors of the network forecasts for PUMAT42 are nearly symmetric (not shown), but at longer lead-times a zonal asymmetry emerges (fig. 4b). This is probably related to the fact that the neural network used here does not wrap around the boundaries.

3.2 Dependence of forecast skill on amount of training years

A key issue is the extent to which the above results, and more generally the skill of the network forecasts, depend on the length of the training period used. Fig. 5 shows the skill of the network forecasts for 500hPa geopotential height for different training lengths, for a lead-time of 1 day (a,c) and 6 days (b,d). As mentioned in the methods section, the networks with short training periods were trained several times with different samples from the model runs. ~~Figure 5 displays the mean skill; the shading~~ The shading in the figure represents the uncertainty of the mean for these multi-sample networks, which is negligibly small. For

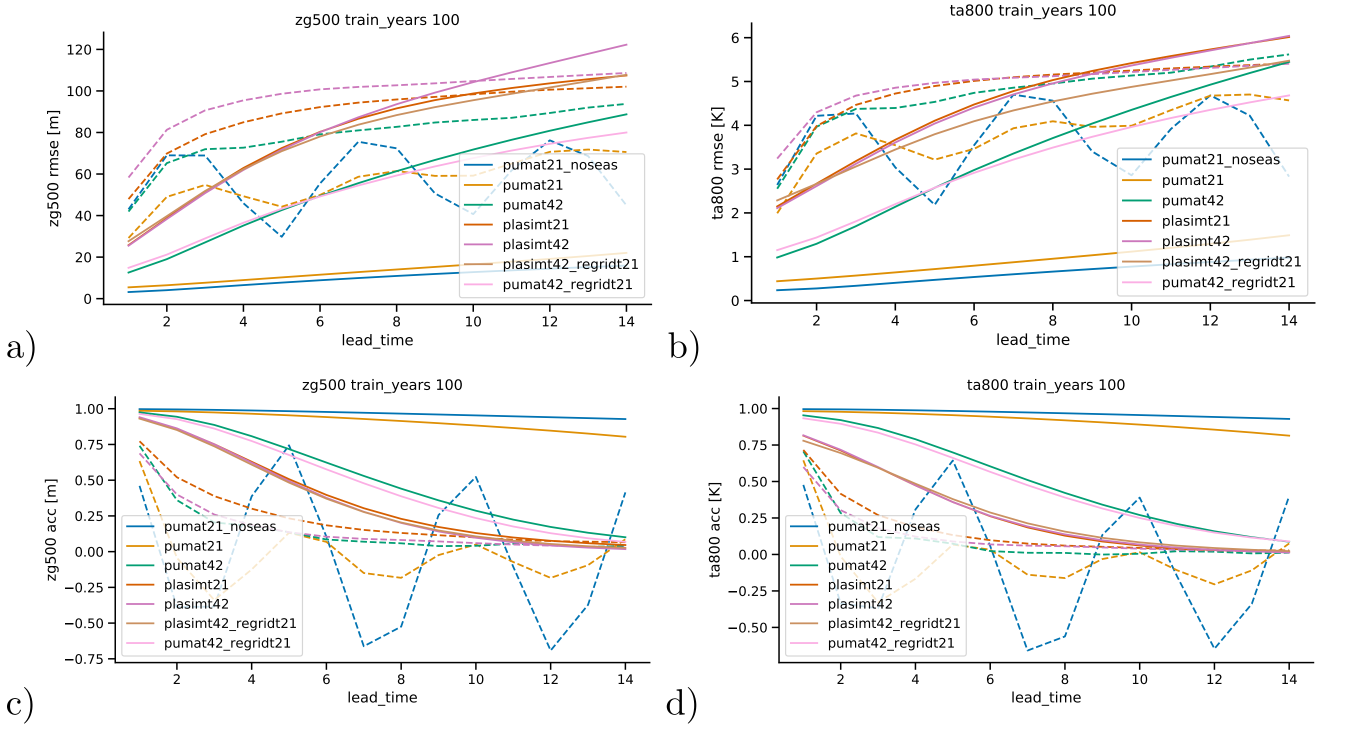

Figure 3. Root Mean Square Error (RMSE) and Anomaly Correlation Coefficient (ACC) of 500hPa gepotential height (a,c) and 800hPa temperature (b,d) for network forecasts (solid lines) and persistence forecasts (dashed lines) for all models for different lead times (in days). All forecasts are based on 100 years of training data.

the 1-day forecasts (fig. 5a, c), the results are as expected: the skill increases with an increasing number of training years, both in terms of RMSE and ACC. This increase is strongly nonlinear and, beyond ∼~~200~~ 100 years, the skill benefit of increasing the length of the training set is limited. This suggests that the complete model-space is already encompassed by around ~~200~~ 100 years of daily data. More years will not provide new information to the network. However, it might also be the case that there is in fact more information in more years, but that the network is not able to utilize this additional information effectively. For the 6-day ~~predictions~~forecasts (fig. 5b, d), the PLASIMT21 networks display a ~~counterintuitive~~ counter-intuitive behaviour: the skill (in terms of RMSE) does not ~~monotonically increase~~ increase monotonically with increasing length of the training period, but decreases from 100 years to 200 years, while for >200 years it increases again. A similar result ~~–~~– albeit less pronounced ~~–~~– is also seen for the PLASIMT42_regridT21 networks, and also ~~–~~– in a slightly different form ~~–~~– for the skill measured via the ACC. To interpret this, one has to remember that the networks are all trained on 1-day forecasts. For 1-day forecasts ~~– as mentioned above –~~the skill is indeed increasing with increasing training length, and the PUMAT21 network trained on 200 years makes better forecasts than the one trained on 100 years (fig. 5a, c). Intuitively one would assume this to translate to increased skill of the consecutive forecasts used to create the 6 day forecasts. The fact that this is not the case here might be

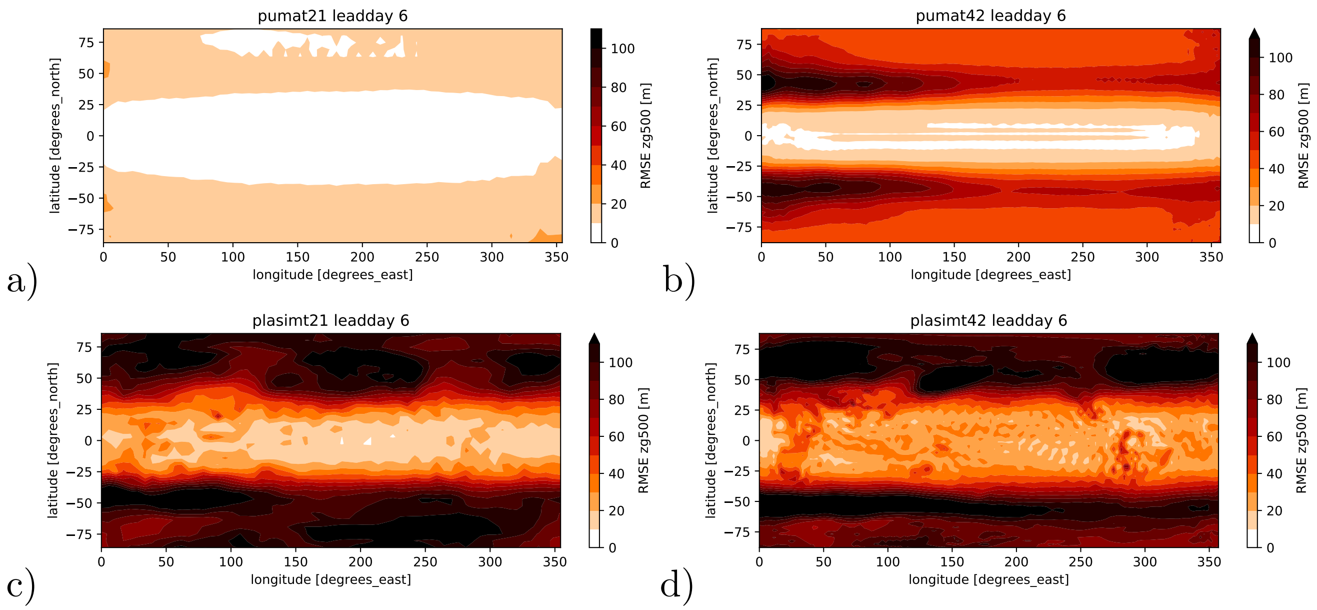

Figure 4. Maps of RMSE of the 6-day network forecasts, for the networks trained with 100 years for PUMAT21 (a), PUMAT42 (b), PLASIMT21 (c) and PLASIMT42 (d).

caused by non-linear error growth. Some networks might produce slightly lower errors at lead-day 1, but ~~the particular errors they have~~ those particular errors could be faster-growing than those of a network with larger day-1 errors.

3.3 Climate runs with the networks

The trained networks are not limited to forecasting the model "weather", but can also be used to generate a climate run starting from an arbitrary state of the climate model. For this, we use the climate networks that also include the day of year as input (see methods section). Of special interest is the question of whether the climate time-series obtained ~~by~~ from the network is stable. In Scher (2018), the network climate for PUMAT21_noseas (~~trained on~~ using 100 years of training data) was stable and produced reasonable statistics compared to the climate model. We trained our climate networks ~~both on~~ on both 30 years and 100 years of data for all models with seasonal cycle. While the networks were all stable, they do not produce particularly realistic representations of the model climates. After some time, the storm tracks look unrealistic, and the seasonal cycle is also poorly represented (e.g. in some years, some seasons are skipped – see videos S1-S8 in the supplement that show the evolution of zg500 and zonal windspeed at 300hPa for the network climate runs). Figure 6 a) shows the evolution of ta800 at a single gridpoint at 76 °N for PUMAT21 and the climate network trained on 30 years, started from the same (randomly selected) initial state (results for different randomly selected initial states are very similar, not shown). The network is stable, but the variance is too high and some summers are left out. Surprisingly, the network for 30 years of PLASIMT21 (~~panel~~ fig. 6 b) produced a

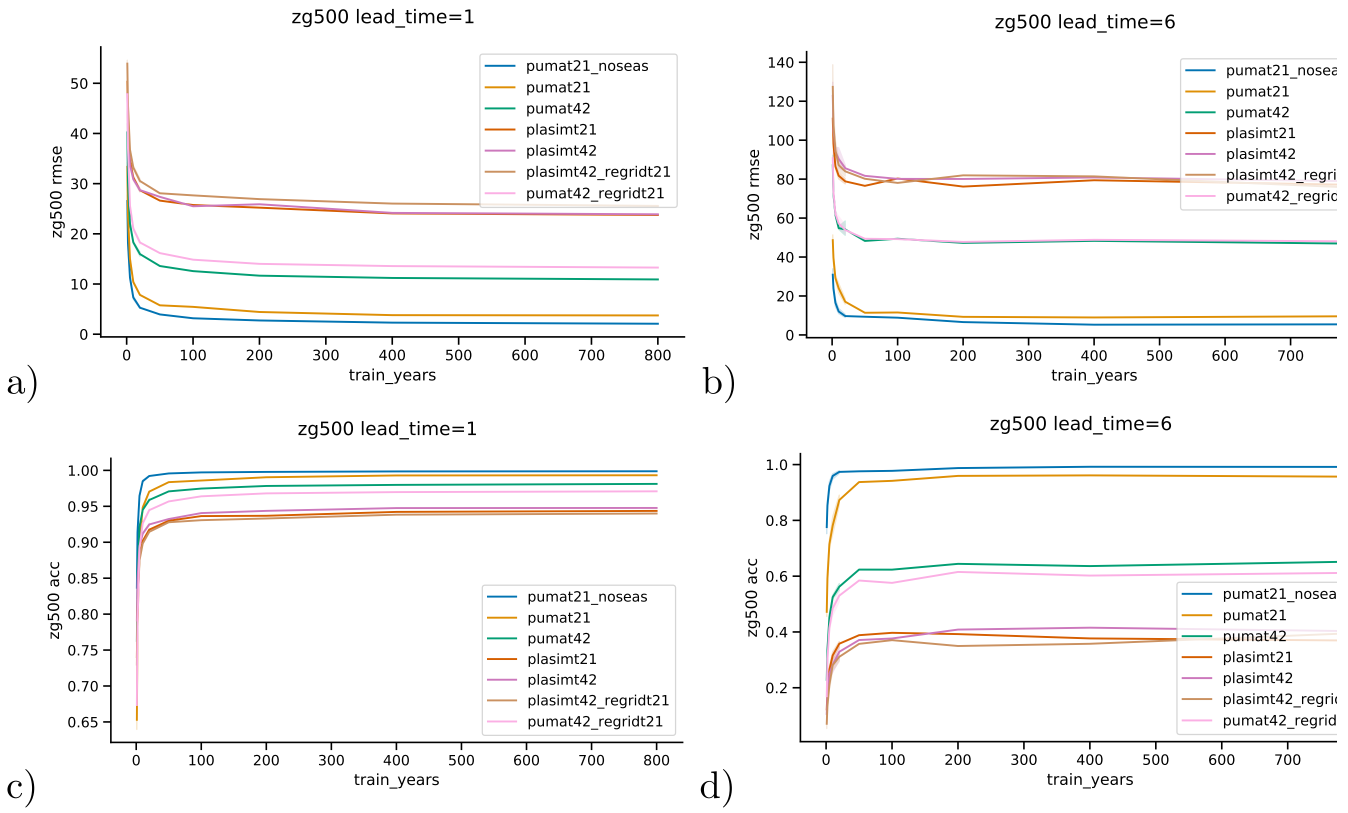

Figure 5. Dependence of network forecast skill on the length of the training period. Shown is the Root Mean Square Error (RMSE) and Anomaly Correlation Coefficient (ACC) of the network-forecasted 500hPa gepotential height for networks trained on different amounts of training years. Each line represents one model. The shading on the left side of the plots represents the 5-95 uncertainty range of the mean RMSE/ACC, estimated over networks with different training samples.~~)~~

more realistic climate. Training on 100 years instead of 30 years does not necessarily improve the quality of the climate-runs (~~panels~~ fig. 6 c,d). For PLASIMT21, in fact, the network climate trained on 100 years is ~~even more problematic.~~

~~Evolution of daily ta800 at a single grid-point at 76°N in the GCM (orange) and in the climate network trained on the GCM (blue), started from the same initial state. The networks were trained on: 30 years of PUMAT21 (a), 30 years of PLASIMT21 (b), 100 years of PUMAT21 (c) and 100 years of PLASIMT21 (d)~~

the worse performer. Interestingly, the mean climate of the PLASIMT21 network is reasonably realistic for the network trained on 100 years (fig. 7), and partly also for the network trained on 30 years (not shown), whereas the mean climates for the PLASIMT42 and PUMAT42 networks have large biases (see fig S11-17 in the Supplement).

All our networks were trained to make 1-day forecasts. The influence of the seasonal cycle on atmospheric dynamics – and especially the influence of diabatic effects – may be very small for 1-day predictions. This could make it hard for the network

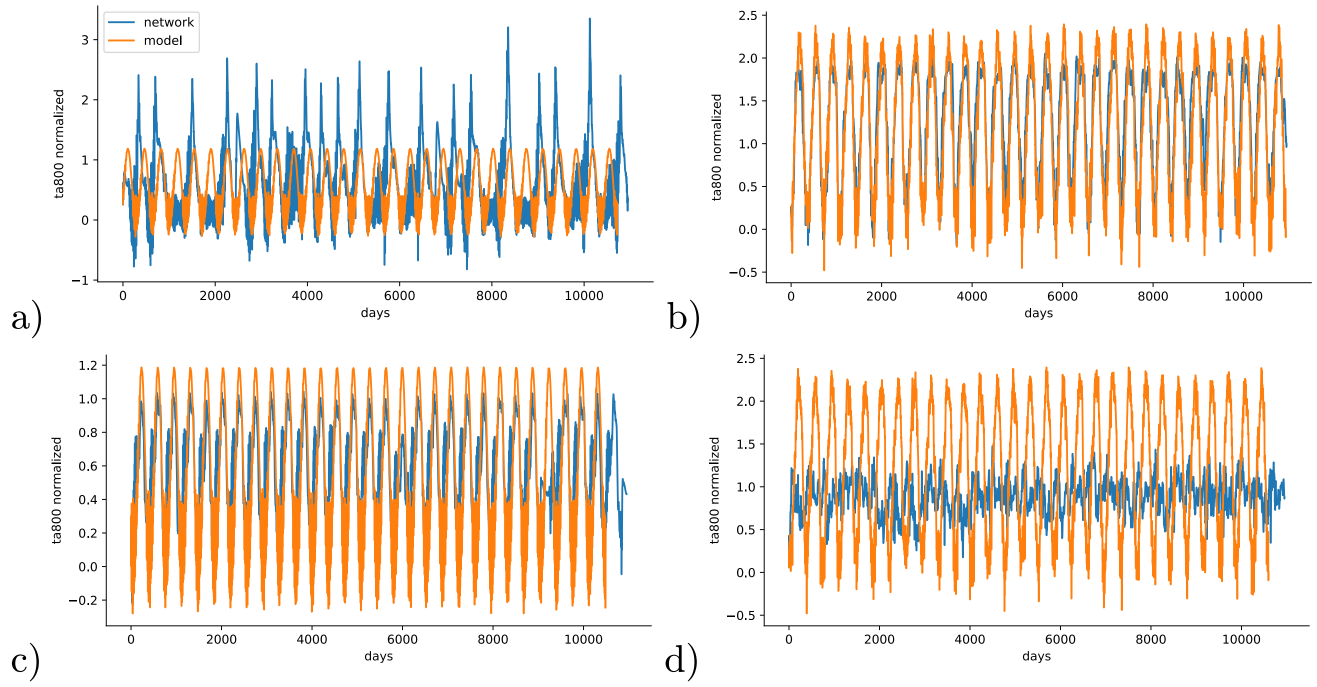

Figure 6. Evolution of daily ta800 at a single grid-point at 76°N in the GCM (orange) and in the climate network trained on the GCM (blue), started from the same initial state. The networks were trained on: 30 years of PUMAT21 (a), 30 years of PLASIMT21 (b), 100 years of PUMAT21 (c) and 100 years of PLASIMT21 (d).

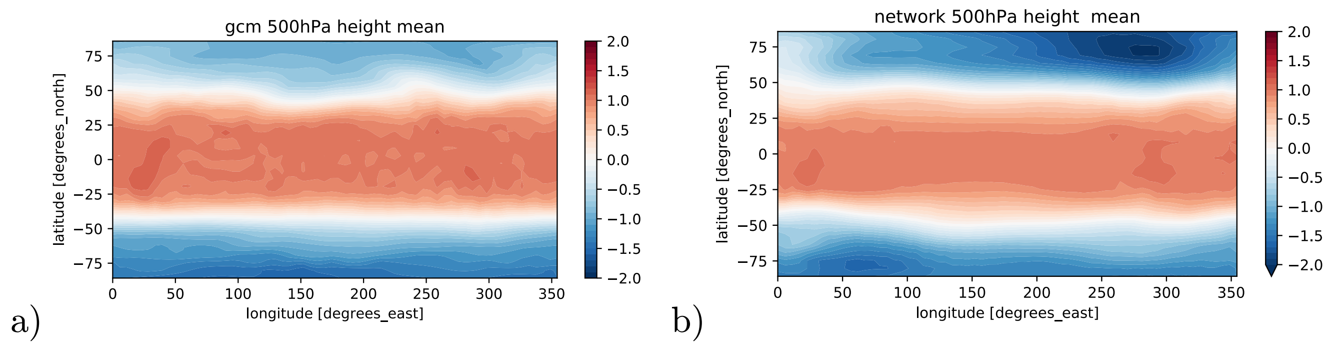

Figure 7. 30 year mean of normalized 500hPa geopotential height for PLASIMT21 (a) and the network trained on 100 years of PLASIMT21 (b).

to learn the influence of seasonality. To test this, we repeated the training of the climate network for PLASIMT21, but now training on 5-day forecasts. This improved the seasonality of ta800 at our example gridpoint (fig. S19 in the supplement), but the spatial fields of the climate run still become unrealistic (video S9 in the supplement).

3.4 Impact of re-tuning

The design of this study was to use an already established neural network architecture ~~-~~ – namely one tuned on a very simple model ~~-~~ – and apply it to ~~to~~ more complex models. However, it is of interest to know how much tuning the network architecture to the more complex models might increase forecast skill. Therefore, the same tuning procedure as in Scher (2018) for PUMAT21_noseas was repeated for PLASIMT21. ~~Interestingly~~Surprisingly, the resulting configuration for PLASIMT21 was exactly the same as for for PUMAT21_noseas. Thus, even with re-tuning the results would be the same. As a caveat, we note that tuning neural networks is an intricate process, and many arbitrary choices have to be made. Notably, one has to pre-define the different configurations that are tried out in the tuning (the "tuning space"). It is possible that with a different tuning space for PLASIMT21, a different configuration would be chosen than for PUMAT21_noseas. However, at least within the tuning space we used, we can conclude that a setup working well for a very simple model (PUMAT21_noseas) is also a good choice for a more complex model like PLASIMT21.

3.5 Impact of including hydrological cycle as input

PLASIM, in contrast to PUMA, includes a hydrological cycle. This is represented by 3 additional 3d state variables in the model (relative humidity, cloud liquid water content and cloud cover). To the test the impact of including these variables in the neural networks, we retrained the network for 100 years of PLASIMT21 including these 3 variables (on 10 levels) as additional channels both in the input and output of the network. The network thus has 70 instead of 40 in- and output channels. Quite surprisingly, including the hydrological cycle variables slightly deteriorated the forecasts of zg500 and ta800, both in terms of RMSE and ACC, except for the ACC of ta800 from lead-times of 6 days onwards (fig. S20 in the supplement).

3.6 Performance on reanalysis data

In order to put our results in context, we also trained our network architecture on coarse-grained (regridded to T21) ERA5 (C3S, 2017) reanalysis data. We use exactly the same dataset that Dueben and Bauer (2018) used, namely 7 years (2010-2016) for training, and 8 months for evaluation (January 2017- August 2017). ERA5 is available in hourly intervals, allowing a thorough investigation of which timestep is best for training. As in Dueben and Bauer (2018), we only use zg500, and therefore deviate from the setup used in the rest of this study. We do this in order to be able to directly compare the results in the two studies. We trained networks on lead-times ranging from 1 to 240 hours, and computed the skill of the zg500 forecasts. The results are shown in fig. 8. The blue line shows the error of persistence forecasts. The network trained on 1h-forecasts is not stable and has very high forecast errors, in contrast to Dueben and Bauer who only obtained good forecasts with training on 1h lead-time. The network trained with 24h lead-time has comparable skill to the network used by Dueben and Bauer (2018) (see

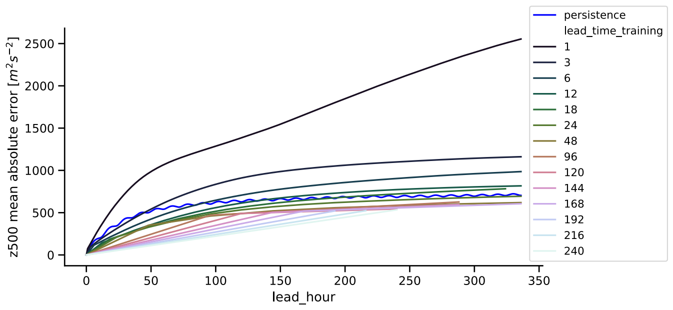

Figure 8. Mean absolute error of 500 hPa geopotential height, for networks trained on coarse-grained (T21) ERA5 data. The training was performed on different lead-times. The x-axis denotes the lead-time of the forecast in hours, the colors the lead-time used for training (in hours). Also shown is the persistence forecast (thick blue line).

their fig. 3a). For longer lead-times, the networks trained on longer lead-times seem to work better. In general, it seems that if one wishes to have forecasts for a certain lead-time, it is best to train directly on that lead-time, at least for mean absolute error.

4 Discussion and Conclusions

We have tested the use of neural networks for forecasting the ~~’~~‘weather’ in a range of simple climate models with different complexity. For this we have used a deep convolutional ~~auto-encoder~~ encoder-decoder architecture that Scher (2018) developed for a very simple general circulation model without seasonal cycle. The network is trained on the model in order to forecast the model state 1 day ahead. This process is then iterated to obtain forecasts at longer lead times. We also performed “climate” runs, where the network is started with a random initial state from the climate model run, and then creates a run of daily fields for several decades.

Potential improvements in the neural network architecture

In the architecture used in this paper, lat-lon grids are used to represent the fields. The convolution layers consists of a set of fixed filters that are moved across the domain. Therefore, the same filter is used close to the poles and close to the equator, even though the area represented by a single gridpoint is not the same close to the poles and close to the equator since the Earth is a globe. Using spherical convolution layers as proposed in Coors et al. (2018) would tackle this problem and may lead to improved forecasts and/or more realistic long-term simulations. The tuning of the network architecture used here tuned the depth of the convolution layers, but not the actual number of convolution layers. Therefore, it would be interesting to explore whether deeper networks (more convolution layers) could improve the performance of the networks. Other possible improvements would be to:

– Include one or more locally connected layers in addition to the convolution layers. While they can increase problems related with overfitting, locally connected layers could learn "local dynamics".

– Include spatial dependence in height. In our architecture, all variables at all heights are treated as individual channels. One could for example group variables at one height together, or use 3d-convolution.

Our results further support the idea – already proposed by Dueben and Bauer (2018) and Scher (2018) – of training a neural network on the large amount of existing climate model simulations, feeding the trained network with today's analysis of a NWP model and using the network to make a weather forecast. The high computational efficiency of such neural network forecasts would open up new possibilities, for example of creating ensemble forecasts with much large ensemble sizes than the ones available today. Therefore, this approach would provide an interesting addition to current weather forecasting practice, and also a good example of exploiting the large amount of existing climate model data for new purposes.

Networks for weather forecasts and climate runs

One of the ~~major~~ aims of this study was to assess whether it is possible to use a simplified reality ~~-~~ – in this case ~~the most~~ a very simple GCM without seasonal cycle ~~-~~ – to develop a method that also works on more complex GCMs. We showed that, for the problem of forecasting the model 'weather', this seems to be the case: the network architecture developed by Scher (2018) also worked on the more complex models used here, albeit with lower skill. The latter point is hardly surprising, as one would expect the time-evolution of the more complex models to be harder to predict. The neural network forecast outperformed a simple persistence forecast baseline at most lead-times also for the more complex models. The fact that we can successfully forecast the weather in a range of simple GCMs a couple of days ahead is an encouraging result for the idea of weather forecasting with neural networks. We also tried to re-tune the network architecture from Scher (2018) to one of our more complex models. Surprisingly, the best network configuration that came out of the tuning procedure was exactly the same as the one obtained for the simpler model in ~~(Scher, 2018)~~Scher (2018). This further supports the idea that methods developed on simpler models may be fruitfully applied in more complex settings. Additionally, we tested the network architecture on coarse-grained reanalysis data, where its skill is comparable to the method proposed by Dueben and Bauer (2018). We also found that, in contrast to the findings of Dueben and Bauer (2018), it seems possible to make valid neural network forecasts of atmospheric dynamics using a wide range of timesteps (also much longer than 1 hour). This discrepancy might be caused by our use of convolutional layers, in contrast to the local deep neural network that Dueben and Bauer used. A stack of convolution layers may be interpreted as multiple layers, where each layer could possibly make a short-term forecast, and the whole stack a long-term forecast.

The second problem we addressed was using the trained networks to create ~~very long integrations of model weather, namely a~~ "climate" ~~run. For this, the network is started with a random initial state from the climate model run, and then creates a run of daily fields for several decades~~runs. Scher (2018) found this generated a stable climate for the simplest model considered here, which does not have a seasonal cycle. ~~We further find that it~~ Here, we find that this is to some extent also possible for more complex models. However, even when training on relatively long periods (100 years), the climates produced by the networks have some unrealistic properties, such as ~~reproducing the seasonal cyclesomewhat poorly, having~~ a poor seasonal cycle, significant biases in the long-term mean values and often unrealistic storm tracks. The fact that these problems don't occur for the simplest GCM without seasonal cycle, but do occur for the same GCM with seasonal cycle, indicates that seasonality considerably complicates the problem. While not a solution for creating climate runs, this suggests that for the weather forecasting problem it might be ~~wise~~ interesting to train separate networks for different times of the year (e.g. one for each month).

~~Our results further strengthen the idea - already proposed by Dueben and Bauer (2018) and Scher (2018) - of training a neural network on the large amounts of existing climate model simulations, feeding the trained network with today's analysis of a NWP model and using the network in order to make a weather forecast. The high computational efficiency of such neural network forecasts would open up new possibilities, for example of creating ensemble forecasts with much large ensemble sizes than the ones available today. Therefore, this approach would provide an interesting addition to current weather forecasting practice, and also a good example of exploiting the large amount of existing climate model data for new purposes.~~

Code and data availability. The code developed and used for this study is available in the accompanying repository at http://doi.org/10.5281/zenodo.2572863. All external libraries used here are open source. The trained networks and the data underlying the all plots are available in the repository. The model runs can be recreated with the control files (available in the repository) and the source code of PUMA/PLASIM, which is freely available at https://www.mi.uni-hamburg.de/en/arbeitsgruppen/theoretische-meteorologie/modelle/PLASIM.html ERA-Interim data is freely avialable at https://apps.ecmwf.int/datasets/data/interim-full-daily/levtype=pl/. ERA5-data is freely availableat https://climate.copernicus.eu/climate-reanalysis

Appendix A

A1 Computation of the local dimension

Here, we outline very briefly how the local dimension d is computed. To foster easy reproducibility, we present the computation in an algorithm-like fashion, as opposed to formal mathematical notation. For a more rigorous theoretical explanation the reader is referred to Faranda et al. (2019). The code is available in the repository accompanying this paper (see Code and data availability).

First, we define the distance between the 2-D atmospheric fields at times t_1 and t_2 as:

$$dist_{t1,t2\,t_1,t_2} = sqrt\left((x_{t_1,1} - x_{t_2,1})^2 + (x_{t_1,2} - x_{t_2,2})^2 + ... + (x_{t_1,N_j} - x_{t_2,N_j})^2\right) \quad \text{(A1)}$$

where ~~j~~ j is the linear gridpoint index and N_j is the total number of gridpoints. To compute d_t, namely the local dimension of a field at time t, we first take the negative natural logarithm of the distances between t and all other timesteps t_i (i.e. all times before and after t):

$$g_{t,t_i} = -ln(dist_{t,t_i}) \tag{A2}$$

and then retain only the distances that are above the 98th percentile of g_{t,t_i}:

$$exceedances = g_{t,t_i} - 98^{th} percentile(g_{t,t_i}) \,\forall\, g_{t,t_i} > 98^{th} percentile(g_{t,t_i}) \tag{A3}$$

These are effectively logarithmic returns in phase space, corresponding to cases where the field ~~x_t~~ x_{t_i} is very close to the field ~~x_{t_i}~~ x_t. According to the Freitas-Freitas-Todd theorem (Freitas et al., 2010), modified in (Lucarini et al., 2012), the probability of such logarithmic returns ~~is a~~ (in the limit of an infinitely long timeseries) follows the exponential member of the generalized Pareto distribution (Pickands III et al., 1975). The local dimension d_t can then be obtained as the inverse of the distribution's scale parameter, which can also be expressed as the inverse of the mean of the exceedances:

$$d_t = 1/mean(exceedances) \tag{A4}$$

The local dimension is an instantaneous metric, and Faranda et al. (2019) have shown that time-averaging of the data can lead to counter-intuitive effects. The most robust approach is therefore to compute d on instantaneous fields. Here, in order to use the same data as for the machine learning, we have used daily means. We have verified that, at least in ERA-Interim, this has a negligible effect on the average d value for all variables except for geopotential at 100hPa and 200hPa (not shown).

Author contributions. SS has conceived the study and methodology, performed the analysis, produced the software and drafted the article. GM contributed to drafting and revising the article.

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

Acknowledgements. ~~G. Messori have~~ We thank Peter Dueben for providing the ERA5 data. S. Scher has been funded by the Department of Meteorology (MISU) of Stockholm University. G. Messori was partly supported by the Swedish Research Council Vetenskapsrådet (grant no. 2016-03724). The computations were done 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

Bauer, P., Thorpe, A., and Brunet, G.: The quiet revolution of numerical weather prediction, Nature, 525, 47–55, https://doi.org/10.1038/nature14956, https://www.nature.com/articles/nature14956, 2015.

Buschow, S. and Friederichs, P.: Local dimension and recurrent circulation patterns in long-term climate simulations, arXiv preprint arXiv:1803.11255, 2018.

C3S: ERA5: Fifth generation of ECMWF atmospheric reanalyses of the global climate. Copernicus Climate Change Service Climate Data Store (CDS), https://cds.climate.copernicus.eu/cdsapp#!/home, 2017.

Compo, G. P., Whitaker, J. S., Sardeshmukh, P. D., Matsui, N., Allan, R. J., Yin, X., Gleason, B. E., Vose, R. S., Rutledge, G., Bessemoulin, P., and others: The twentieth century reanalysis project, Quarterly Journal of the Royal Meteorological Society, 137, 1–28, 2011.

Coors, B., Paul Condurache, A., and Geiger, A.: Spherenet: Learning spherical representations for detection and classification in omnidirectional images, in: Proceedings of the European Conference on Computer Vision (ECCV), pp. 518–533, 2018.

Dee, D. P., Uppala, S. M., Simmons, A., Berrisford, P., Poli, P., Kobayashi, S., Andrae, U., Balmaseda, M., Balsamo, G., Bauer, d. P., et al.: The ERA-Interim reanalysis: Configuration and performance of the data assimilation system, Quarterly Journal of the royal meteorological society, 137, 553–597, 2011.

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.

Faranda, D., Messori, G., and Vannitsem, S.: Attractor dimension of time-averaged climate observables: insights from a low-order ocean-atmosphere model, Tellus A: Dynamic Meteorology and Oceanography, 71, 1–11, https://doi.org/10.1080/16000870.2018.1554413, https://doi.org/10.1080/16000870.2018.1554413, 2019.

Fraedrich, K., Jansen, H., Kirk, E., Luksch, U., and Lunkeit, F.: The Planet Simulator: Towards a user friendly model, Meteorologische Zeitschrift, 14, 299–304, https://doi.org/doi:10.1127/0941-2948/2005/0043, https://www.ingentaconnect.com/content/schweiz/mz/2005/00000014/00000003/art00001, 2005.

Freitas, A. C. M., Freitas, J. M., and Todd, M.: Hitting time statistics and extreme value theory, Probability Theory and Related Fields, 147, 675–710, 2010.

Johnson, N.: Simply complexity: A clear guide to complexity theory, Oneworld Publications, 2009.

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.

Lucarini, V., Faranda, D., and Wouters, J.: Universal behaviour of extreme value statistics for selected observables of dynamical systems, Journal of statistical physics, 147, 63–73, 2012.

McGovern, A., Elmore, K. L., Gagne, D. J., Haupt, S. E., Karstens, C. D., Lagerquist, R., Smith, T., and Williams, J. K.: Using Artificial Intelligence to Improve Real-Time Decision-Making for High-Impact Weather, Bulletin of the American Meteorological Society, 98, 2073–2090, https://doi.org/10.1175/BAMS-D-16-0123.1, https://doi.org/10.1175/BAMS-D-16-0123.1, 2017.

Nooteboom, P. D., Feng, Q. Y., López, C., Hernández-García, E., and Dijkstra, H. A.: Using network theory and machine learning to predict El Niño, Earth System Dynamics, 9, 969–983, https://doi.org/10.5194/esd-9-969-2018, https://www.earth-syst-dynam.net/9/969/2018/, 2018.

O'Gorman, P. A. and Dwyer, J. G.: Using Machine Learning to Parameterize Moist Convection: Potential for Modeling of Climate, Climate Change, and Extreme Events, Journal of Advances in Modeling Earth Systems, 10, 2548–2563, https://doi.org/10.1029/2018MS001351, https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018MS001351, 2018.

Pickands III, J. et al.: Statistical inference using extreme order statistics, the Annals of Statistics, 3, 119–131, 1975.

Poli, P., Hersbach, H., Dee, D. P., Berrisford, P., Simmons, A. J., Vitart, F., Laloyaux, P., Tan, D. G. H., Peubey, C., Thépaut, J.-N., Trémolet, Y., Hólm, E. V., Bonavita, M., Isaksen, L., and Fisher, M.: ERA-20C: An Atmospheric Reanalysis of the Twentieth Century, Journal of Climate, 29, 4083–4097, https://doi.org/10.1175/JCLI-D-15-0556.1, https://journals.ametsoc.org/doi/abs/10.1175/JCLI-D-15-0556.1, 2016.

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.

Schneider, T., Lan, S., Stuart, A., and Teixeira, J.: Earth System Modeling 2.0: A Blueprint for Models That Learn From Observations and Targeted High-Resolution Simulations, Geophysical Research Letters, 44, 12,396–12,417, https://doi.org/10.1002/2017GL076101, https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017GL076101, 2017.