Detecting Climate Patterns

A Bayesian Neural Network Approach

J.T. Arens

Delft University of Technology

Detecting climate patterns

A Bayesian neural network approach

by

J.T. Arens

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday April 26 at 2 PM.

Student number:	4206231	
Project duration:	July, 2020 – April, 2022	
Thesis committee:	Dr. Riccardo Riva,	TU Delft, supervisor
	Dr. Marco Loog	TU Delft,
	Dr. Franziska Glassmeier	TU Delft

TU Delft

Contents

1 Paper: Detecting climate patterns through a Bayesian neural network approach 1

2 Supplementary Materials 19

Preface

In front of you, you find my MSc thesis, which describes the work that was done to end my time at Geoscience and Remote Sensing at Delft University of Technology. I have decided to write this thesis work in paper format, following the guidelines for the Science Advances journal, which has been a fun challenge. I hope it reads smoother than the process to get here.

The work here can only be presented due to Riccardo Riva and Marco Loog, both of whom helped a lot through critical feedback, guidance and support on the project. I would also like to thank Franziska Glassmeier for the feedback she provided during our assessment meetings. Sandra Verhagen also deserves a spot here, for her support throughout the trials and tribulations that has been my masters. Lastly, I would like to thank all my friends and family, just for being who they are.

In loving memory,
Riet & Anton de Reuver
Jullie zijn nu voor eeuwig samen

1 Paper: Detecting climate patterns through a Bayesian neural network approach

Detecting climate patterns through a Bayesian neural network approach

Jesse T. Arens, Riccardo E.M. Riva, Marco Loog & Fransizka Glassmeier

Machine learning is becoming an increasingly important tool for climate scientists, but hampered by lacking uncertainty quantification. Here, a machine learning approach for detecting patterns indicating a changing climate is combined with probabilistic modelling to retrieve uncertainty values. We train neural networks on climate model simulations of temperature and precipitation under historical and future scenarios. We find that the resulting so-called Bayesian neural network (BNN) has similar predictive strength to an Artificial neural network (ANN), with a post-year 2000 mean absolute error of 9.00 years for temperature, but over-fits less. The BNN is able to recognise temperature change starting in 1994, which is 14 years later than the ANN. Our analysis shows that uncertainties in found climate patterns are much higher than the patterns themselves, limiting their value for further use. This work demonstrates that BNNs are a suitable tool for quantifying uncertainties of patterns indicating a changing climate.

INTRODUCTION

The Earth's climate can be described through General Circulation Models (GCMs). These climate models attempt to encapsulate the physical processes and feedbacks influencing climate and predict what effect anthropogenic activity has on it. However, as climate science suffers from an 'uncertainty monster' (*1*), projections of GCMs for future climate remain uncertain and this uncertainty will not necessarily improve quickly (*2*). For example, the range of the expected response to a doubling of CO_2 between different GCMs has increased the past decade and a large spread between GCMs remains present for mean absolute temperature simulations (*3*). While improvements are certainly made in most aspects of climate modelling, most notably in the uncertainty caused by cloud feedback processes, improvements of GCM skill remain modest (*4*). One of the main reasons for the slow progress lies in the complex feedback mechanisms which have been discovered over time. While individual uncertainties on many of these feedback mechanisms have decreased, new mechanisms have been found at the same time and the uncertainty of some of these feedback processes have proven difficult to reduce. The expectation is that for substantial reductions in projection uncertainty we would have to wait for observations that can undoubtedly be attributed to the Earth's response to anthropogenic forcing (*2*). In general, as models can only be validated with present and past observations, a model cannot be guaranteed to perform well in the future (*5*), especially as for climate models the future will be one in a regime with greenhouse gas concentrations unlike observed before. This does not mean that we should not attempt to improve GCMs, on the contrary. While improvements on projection uncertainty might be slow, the latest generation of GCMs "have a similar or even slightly higher skill in reproducing observed large-scale mean surface temperature and precipitation patterns" (Bock et al., 2020, p22, (*3*)), which indicates that these climate models might be capturing the physical processes better than previously the case.

In order to improve GCMs and achieve more certain climate projections, there is a desire to understand why the projections remain uncertain. Most notable improvements in understanding climate projection uncertainties have happened through the use of model intercomparison, which has been done for at least three decades (*6*). Combining GCMs into multi-model ensembles increases the skill, reliability and consistency of forecasts (*7*). The fifth phase of the Coupled Model Intercomparison Project (*8*), which aggregates most GCMs into an ensemble and provides outputs of experiment runs, has become the de facto source for multi-model climate sensitivity analysis the last decade and is the scientific basis of the Fifth Assessment Report of the Intergovernmental Panel on Climate Change (*9*). With the recent availability of the sixth phase of the Coupled Model Intercomparison Project (*10*), accompanied by improved model evaluation tools (*11*), new opportunities arise in model intercomparison and tackling the uncertainties.

Uncertainty in climate projections can be split between so-called scenario uncertainty, modelling uncertainty and internal climate variability (*12*). First of all, scenario uncertainty corresponds to the evolution amount of CO_2-equivalent concentrations in the future and is the most difficult to reduce. As the true emission scenario is greatly dependent on how humankind addresses climate change, scenarios such as the representative concentration pathways (RCPs) have been developed to be able to assess the impacts of different possibilities in climate action (*13–15*). Secondly, modelling uncertainty refers to the mismatch between GCM representation of the Earth and the truth. Different models provide different results to the same radiative forcing given, because they depend on assumptions that have to be made. Studies into reducing model uncertainty often pertain to biases and uncertainty within the model ensemble on how the climate changes under forcing, which is also known as the climate sensitivity. Climate sensitivity remains difficult to constrain (*16*) and so causes a significant source for the range of outcomes between individual GCMs, resulting in higher uncertainty for multi-model ensembles. A promising approach in tackling the uncertainties caused by climate sensitivity is through emergent constraints (*11*) and weighing individual models in ensembles based on independence and skill (*17, 18*). Finally, internal climate variability refers to fluctuations in the Earth's climate which happen regardless of any radiative forcing of the climate. These can be well-described phenomena, such as the El Niño – Southern Oscillation or Madden-Julian Oscillation, but there are also internal variability phenomena which remain unknown. Climate response to anthropogenic forcing will be most easy to detect in regions with a relatively small internal variability (*19*). Observations of a changing climate in such regions will likely be the first to become significant enough to be attributed to anthropogenic climate change. Through identification of the effects of internal variability on climate projections, one could isolate the forced climate response. Internal climate variability is often considered to be a noise within the climate models, however, as it proves difficult to disentangle from model errors (*20*). In order to properly attribute climate change, our knowledge of internal variability needs to increase.

One of the tools used for the analysis of climate data, is machine learning. Machine learning is becoming an increasingly important tool in Earth system science, due to bigger data availability (*21, 22*) and due to improvements and new developments in techniques for interpretable and explainable machine learning (*23*). Whereas a traditional neural network is often considered a black box, these interpretation techniques make it possible to quantify and visualise what aspects of the data it uses for its predictions. Application of machine learning techniques in Earth system science brings its own challenges and possibilities, as beautifully described by Reichstein et al. (*24*).

An interesting new approach was introduced by Barnes et al. (*25*), where they use an artificial neural network (ANN) to identify climate patterns, i.e. locations on Earth that can reliably indicate a changing climate. There is a lack of labelled datasets of climate patterns, making it difficult if not impossible to train a neural network to predict climate patterns. Instead, they opt to approach these climate patterns as something the network needs to recognise in order to perform a different prediction task, which has sufficient labelled data. Barnes et al. showcased their approach by training an ANN to predict the year of maps generated from CMIP5 climate model outputs from 1920 up to 2100, obtained by combining the CMIP5 historical and Representative Concentration Pathway of $8.5W/m^2$ (RCP8.5) experiments (*14*). After training their neural network, they extracted the information encapsulated by the model parameters. They claimed their method shows the potential for machine learning to identify climate change signals from internal variability noise. They assumed the ANN weights can be used to infer which regions show a a better signal-to-noise ratio when it comes to forced climate response than other regions. These regions could be useful in attributing observed climate change to anthropogenic forcing. Validation of their approach to extract forced climate patterns occurred through inference that the ANN performs well on the testing subset as well as on observational datasets. Like most deep learning models (*26*), uncertainty was not accounted for and therefore the extracted network weights miss important uncertainty information.

Here, we propose a combination of the approach used by Barnes et al. with a Bayesian technique to attribute uncertainties to neural network weights. The resulting model is called a Bayesian neural network (BNN) and works similar to an ANN, with the exception that scalar variables for network weights and prediction results are replaced with probability distributions. Variational inference is used for training these distribution functions (*27*). In this study, we aim to proof the feasibility of a Bayesian neural network for identifying climate patterns and assess the applicability of the approach as part of the toolbox used for evaluating climate models and its uncertainties. In order to make this assessment of the Bayesian neural network, we identify a control network through reproduction of the ANN described by Barnes et al.

Modelling model output Since we apply a neural network model on input data generated through output from climate models and therefore the word 'model' can be confusing, we will use the word 'CMIP5' when talking about the properties of the climate model ensemble, 'climate model output' or 'CMIP5 model output' when talking about the near-surface air temperature and precipitation datasets that have been generated by the CMIP5 ensemble. These CMIP5 output datasets are the input for the neural networks, which will be indicated by 'neural network' when talking about the concept in general or by their specific type. We will be using 'ANN' for an artificial neural network and the model created by it and 'BNN' for a Bayesian neural network and the model created by it.

RESULTS

Training a Bayesian neural network

We create annual-mean maps on a global 4° by 4° grid through interpolation of near-surface air temperature and precipitation output products from CMIP5 (see Materials and Methods). For temperature, outputs from 23 out of the 29 used CMIP5 models are used for training and for precipitation outputs from 18 out of 22 used CMIP5 models are used. The outputs from the other CMIP5 models are used as testing subset for validation of the neural networks. We feed the maps into a Bayesian neural network, with two hidden layers of 10 variational units each, and train it to predict the year of origin of the maps. This prediction is represented by a probability distribution (Fig. 1). The so-called reconstruction term, defined here by the mean squared error between prediction and truth, is a measure of how much the neural network's prediction deviates from the truth on average and equals to 0.114 and 0.234 for temperature and precipitation, respectively. It is relevant to note that the network is not only optimized for reconstruction, but also for minimization of the mismatch between the estimated and true posterior distribution on the weights, which is computed through a Kullback-Leibler divergence term, resulting in a total loss of 0.366 and 2.453 for temperature and precipitation, respectively. Besides the BNN, an ANN has also been set-up using a similar architecture of two hidden layers of 10 units each. This ANN is trained and tested on identical CMIP5 model output subsets and used for comparison with the BNN on its predictive qualities.

Predictive qualities of the Bayesian neural network

Predictive qualities of both the ANN and BNN are represented using scatter plots between true and predicted year (Fig. 2). In order to approximate the expected value of BNN output, averaging is applied to a Monte Carlo sampling of 10,000 predicted output years for each given input map. The results for near-surface air temperature show that the BNN is able to recognise the patterns that indicate a changing climate (*28*), on average starting from 1994 onwards. After 1997, the BNN is able to capture the changing climate in all CMIP5 temperature models used for testing. For precipitation, this 'year of climate change recognition' has an average of 2009 and after 2032 for all models used for testing.

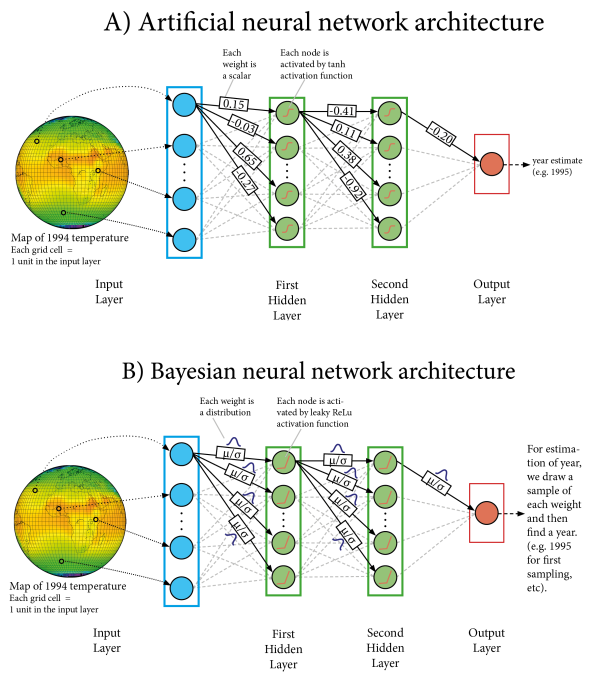

A) Artificial neural network architecture
Each weight is a scalar
Each node is activated by tanh activation function
0.15
0.03
0.65
0.72
-0.41
0.11
0.38
0.92
-0.20
year estimate (e.g. 1995)
Map of 1994 temperature
Each grid cell = 1 unit in the input layer
Input Layer
First Hidden Layer
Second Hidden Layer
Output Layer
B) Bayesian neural network architecture
Each weight is a distribution
Each node is activated by leaky ReLu activation function
μ/σ
For estimation of year, we draw a sample of each weight and then find a year. (e.g. 1995 for first sampling, etc).
Map of 1994 temperature
Each grid cell = 1 unit in the input layer
Input Layer
First Hidden Layer
Second Hidden Layer
Output Layer

Figure 1: **Neural network architectures.** (**A**) Architecture of the artificial neural network, which has two hidden layers of 10 units each. (**B**) Architecture of the Bayesian neural network, which has the same amount of hidden layers and units, but replaces the hidden layer weights and output year with distribution functions. Furthermore, the Artificial neural network uses a hyperbolic tangent as activation function, whereas the Bayesian neural network uses a leaky-ReLu activation function.

Compared to the ANN control run, the BNN is able to accurately make its predictions an average of 14 years later for temperature and 2 years later for precipitation. The BNN and shows a bigger predictive error for training and a slightly higher predictive error for testing than the ANN, but does have a smaller relative difference between training and testing error compared to the ANN, with the testing error being $2.18\times$ and $1.75\times$ the size of the training error for the ANN, compared to $1.25\times$ and $1.53\times$ for the BNN, for air temperature and precipitation respectively. A neural network that perfectly represents the complete data space would result in both errors being equal, i.e. the testing error being $1\times$ of the magnitude of training error.

We tested the robustness of the BNN by running different modes of operation. The prediction results can be found in the Supplementary Materials (Figs S1 - S3). Firstly, roughly the same results were found when using different subsets of CMIP5 models for training and testing. Secondly, as explained by Barnes et al., it might be feasible that the neural network uses the global mean of the input maps to make its predictions instead of the climate patterns we are interested in. Therefore, like Barnes et al., we removed the global mean from the input maps to test if the network really relies on spatial patterns, which provided similar but slightly worse results to our original experiments, indicating that even though the BNN also relies on global mean, it mainly relies on the spatial patterns of interest. Finally, by changing the amount of hidden layers and units in these layers, a few different model architectures have been tested. While the network performs significantly worse for an architecture with two hidden layers of 100 units each, within the margins of a few extra layers and a range between 1 and 100 units in a layer, results seem to be consistent. There are only minimal changes in predictive quality in the range between 1 and 50 units, all together indicating that a BNN is a robust model for this particular data.

Finding climate change indicator patterns

For assessment of the climate patterns found through training a neural network, we create indicator maps of the weights of the trained BNN and ANN networks (Fig. 3). In order to make this visualisation, a different neural network of only 1 hidden layer with 1 single unit has been used. This allows the neural network weights to be mapped onto the input grid in a one-on-one fashion. More sophisticated techniques for mapping weight values of complexer neural networks onto their inputs do exist, but implementation of such techniques lies beyond the scope of our work. For near-surface air temperature, both the ANN and BNN indicator maps show roughly similar patterns of higher or lower weight values, with the exception of a much higher weight in the regions of South Central and East China. The magnitude of the weight differs greatly due to different neural network architectures, as well as the sign of the weights for the precipitation maps. However, our main interest lies in the regions of relatively higher magnitudes, regardless of whether the weight is

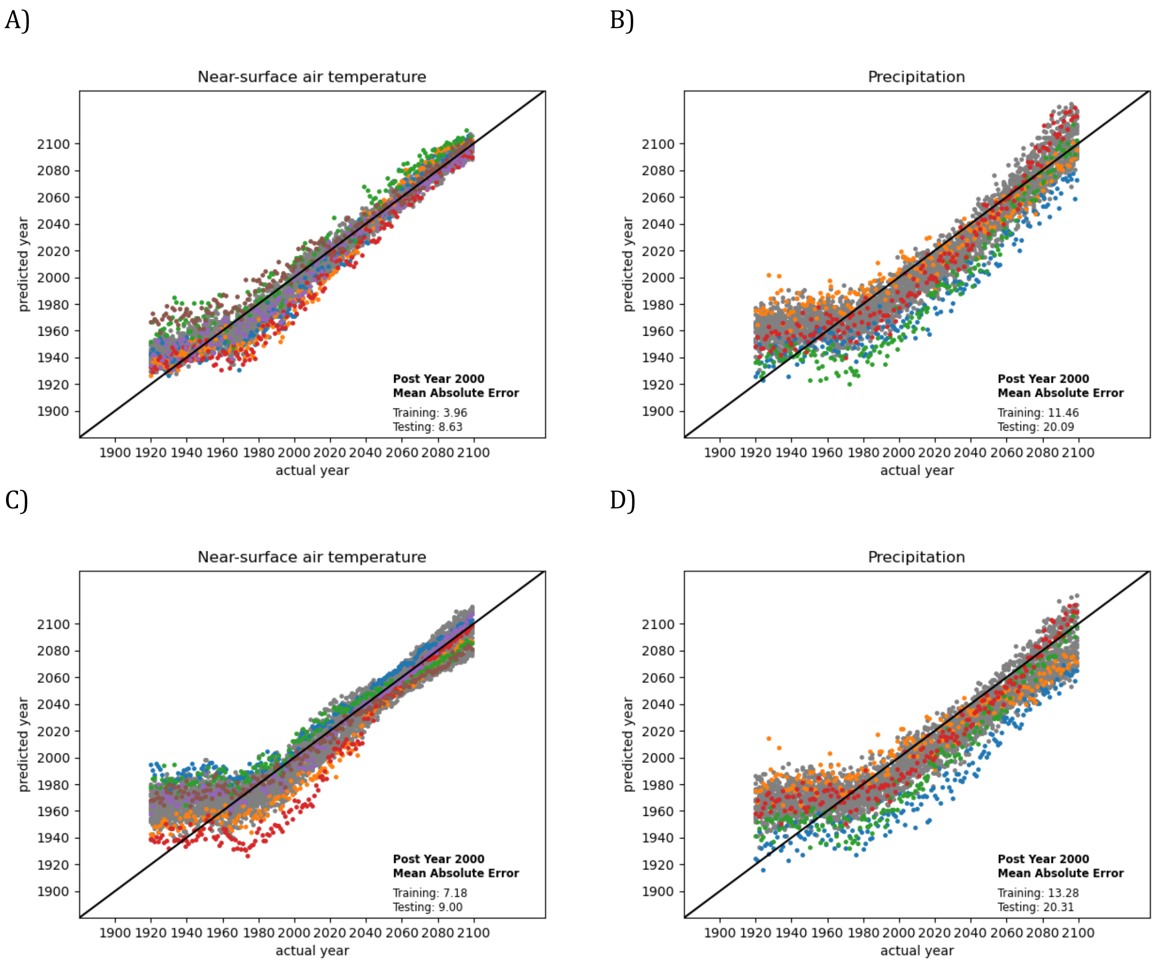


Figure 2: **Prediction results for near-surface air temperature and precipitation.** Predicted year versus actual input map year for both testing for (**A**) An artificial neural network fed with CMIP5 near-surface air temperature maps. (**B**) An artificial neural network fed with CMIP5 precipitation maps. (**C**) A Bayesian neural network fed with CMIP5 near-surface air temperature maps. (**D**) A Bayesian neural network fed with CMIP5 precipitation maps. Training results are shown in grey, testing results are shown in colors, each representing one climate model's simulation. A 1:1 line, indicating a perfect prediction, is plotted in black. Predictive post-year 2000 mean absolute errors are printed in the lower right corner of the figure and indicate how well the model performs on the second half of the time series.

positive or negative. The correlation between these maps is moderate with a value of $r = 0.44$ and found to be statistically significant based on a two-tailed p-value for testing non-correlation, based on common threshold $p < 0.05$. This indicates that both the control network and the Bayesian neural network have been trained on some of the same spatial patterns. When comparing the weight mean with the weight variance map, we find that the magnitude of the variance is generally bigger than that of the mean, but there is strong spatial variability in this. Over all grid cells, the average weight variance is 3.54 times the average absolute value of the weight mean, which means there is indeed a high level of uncertainty in the found patterns. For precipitation, results concerning weight mean and variance of the BNN are better but comparable to those of the near-surface air

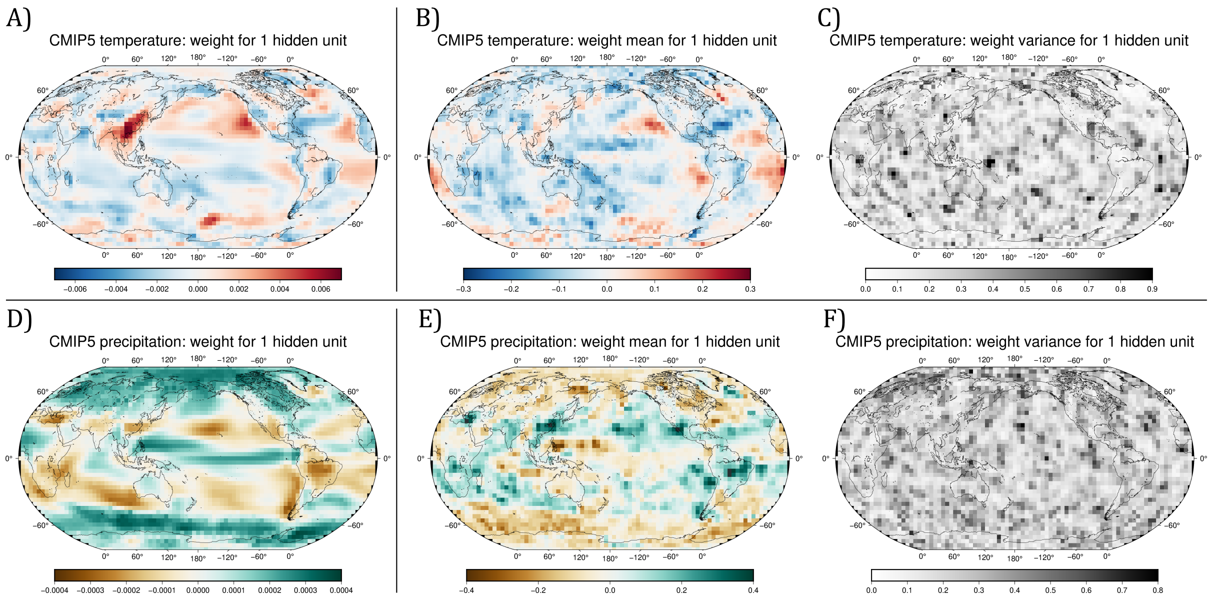

A)
CMIP5 temperature: weight for 1 hidden unit
B)
CMIP5 temperature: weight mean for 1 hidden unit
C)
CMIP5 temperature: weight variance for 1 hidden unit
D)
CMIP5 precipitation: weight for 1 hidden unit
E)
CMIP5 precipitation: weight mean for 1 hidden unit
F)
CMIP5 precipitation: weight variance for 1 hidden unit

Figure 3: **Indicator maps of patterns of change in near-surface air temperature and precipitation.** Maps are derived from weights of neural networks with 1 hidden unit. (**A and D**) Weight maps derived from the ANN control network. (**B-C and E-F**) Weight maps derived from the Bayesian neural network. Weight mean μ (B and E) and variance σ^2 (C and F) of the posterior distribution.

temperature, with the average weight variance being 2.34 times the average absolute weight mean. Compared to the ANN, found climate patterns are again quite similar, with a correlation of $r = -0.62$ and $p < 0.05$. The found precipitation patterns are also roughly similar, but the ANN shows stronger patterns at both poles than the BNN does. Again, due to the architectural difference between the neural networks, weight sign and magnitude does differ between ANN and BNN.

DISCUSSION

We have shown that a Bayesian neural network (BNN) is a viable method for recognition of spatial patterns indicating a changing climate. The technique works similarly to the artificial neural network (ANN) set-up introduced by Barnes et al., but enhances it by allowing for uncertainty attribution on both the neural network weights and prediction. Results for near-surface air temperature show that the BNN provides a smaller relative error between testing and training data than the ANN. This indicates that the BNN is less likely to over-fit. For precipitation this relative error is also in favour of the BNN, but the difference is less profound. The predictive results show comparable errors on the testing datasets for both the BNN and ANN, indicating they are roughly equally capable of performing correct predictions on data it was not trained on.

A downside to the Bayesian neural network is the complexity of the technique. As a minor result of this, the computational resources required are significantly higher. As the network tries to optimize for both the reconstruction loss and the Kullback-Leibler divergence, it requires more trainable parameters and computation steps than a standard ANN, resulting in it being computationally expensive. For near-surface air temperature, the average CPU processing time required for training on our system was 2398 seconds for the BNN compared to 1075 seconds for the ANN. For precipitation we found an average CPU time required of 2293 seconds for the BNN versus 678 seconds for the ANN. Due to the BNN requiring Monte Carlo sampling for computing the predictions, we find the total CPU time required to produce the plots of Fig.2 increases to 10808 seconds for temperature and 10218 seconds for precipitation, whereas the computation time for MLP predictions are near-instant. This extra computational cost is merely an annoyance for the smaller datasets used here, but could pose a limitation for more data-rich experiments. The main drawback of complexity of the technique comes forward in its implementation. As the BNN comes with more parameters to optimise, we found it more difficult to choose the right combination of settings. Settings such as a different prior and posterior on the weights and a higher or lower relative KL-weighting can have a big impact on the performance of the network. For example, a factor 10 difference in KL-weighting can result in a partial or even complete failure of the BNN to make accurate predictions. These come additional to the choice of activation function, optimization algorithm or loss function, which both a BNN and ANN have, but we found are more intuitive to optimize for the ANN. A thorough understanding of Bayesian modelling will help ensuring easier implementation of this technique, but in general for machine learning, reproducibility remains an issue (*29*). In our study, we have also struggled with reproducibility of the original experiment as published by Barnes et al.

Reproducibility issues can occur in each step of the modelling process. It starts with findable and accessible datasets and even though the CMIP5 output data has been developed with accessibility in mind (*30*), we found neither to be guaranteed. There are several data archives that provide the output data. We used the archive of the Earth System Grid Federation, which can be accessed through different gateways. For most CMIP5 models the climate model output data was available here, but some climate model output products had to be retrieved elsewhere because the desired output product was missing.

Feeding data into a neural network almost always requires some preprocessing. In order to make proper predictions, each input of the neural network needs to have the same dimensions, i.e. number of pixels. For CMIP5, the climate models work on differing spatial and temporal resolutions and therefore an annual mean value had to be calculated, as well as regridding to a global 4° by 4°. An error in data formatting or processing can result in failing of the neural network, as it did for us. Such problems can be difficult to solve since it is often difficult to identify the cause of a failing network.

Finally, as stated before, a neural network comes with a lot of adjustable hyperparameters. While we found our Bayesian neural network to be relatively robust to certain changes in the input data or neural network architecture, there are also seemingly minor changes that can cause the network to malfunction. Bayesian neural networks have more of these hyperparameters that need to be optimized, complicating the applicability of the technique by those who have limited knowledge of what each hyperparameter changes to the network.

Holistically seen, there is a lot that can complicate reproducibility or replicability of complex techniques such as the one proposed, even when data and information about processing and algorithms used are available. With the relatively complex datasets used in geosciences and climate modelling, we think reproducibility should be a core component throughout the whole modelling chain, as it can be difficult for each step in the chain. We recommend applying the FAIR data principle: Findable, Accessible, Interoperable and Reusable (*31*). All required information concerning data accessibility, retrieval and processing is provided in the Supplementary Materials (text S1, table S1 and S2). An effort has been made to make documentation of used code accessible and understandable through extensive comments in docstrings, through splitting of the code between function and script files and through availability of required toolboxes and their used version numbers through the Supplementary Materials (text S2, table S3).

The climate change indicator pattern maps of Fig. 3 are created with a neural network architecture with only one hidden unit, which allows the relevance of each grid cell of the neural network's input to be visualised through the weights. Therefore, the network allows for a simpler modelling of the CMIP5 data and we expect it might not be able to capture the patterns in the data as well as a complexer neural network. We find that the more complex neural network model of 2 hidden layers with 10 units outperform the simpler neural network models in terms of predictive accuracy for near-surface air temperature. For precipitation, we find the simpler neural network to outperform the more complex network. Similarly, we find that the Bayesian neural network with the simpler architecture outperforms the ANN in the case of precipitation, whereas the ANN performs better for near-surface air temperature, which indicates that neither model is strictly superior for all CMIP5 data (Fig.S4). This balance between ANN and BNN predictions on precipitation does shift compared to the same balance for the complexer neural networks, with particularly the BNN and ANN having similar predictive performance on the testing dataset for the complex network architecture, but the BNN performing better than the ANN for the simpler neural network architecture, based on both training and testing errors. For near-surface air temperature, this shift in balance is reversed, with the BNN under-performing the ANN in all cases, but significantly less so for the complex neural network architecture.

When looking at the maps procured by Fig.3, we find that both model types can uncover similar climate patterns in their weights, but the BNN can also indicate how certain it is of those mean weight values. Note that due to inner model working differences, it is not possible to directly compare magnitude of the weights of the ANN with those of the BNN. Due to the initialisation of the weights, also the sign of the weights can be reversed between both ANN and BNN. The main interest lies in the relative magnitude of the weights. When looking into the Bayesian neural network and the resulting uncertainties, small weight uncertainty in comparison to the mean of the weight would provide strong evidence that this method of uncovering climate patterns is reliable. Due to the nature of the prediction task, the neural network will give more importance to locations with a higher reliability of a signal found to be caused by a changing climate. Therefore, we do not expect clear patterns in the weight variance maps from the BNNs, which is reflected in the resulting maps of Fig.3. Noteworthy is the magnitude of the variance of the weights in comparison to that of the mean of the weights. The range of variances depicted in Fig.3-C and 3-F are a factor $3\times$ higher than the mean for near-surface air temperature and $2\times$ higher for precipitation. We think that if the magnitude of the weight mean would have been at least the same or a higher value than the variance, useful information could have been extracted from the found climate patterns. With these results, however, we can conclude that the usability of the found climate patterns for further application is limited. We have not investigated whether this relatively high variance is caused by the data or by the neural network architecture of only one hidden unit, as this was not the scope of our work. However, it is feasible that a more complex Bayesian neural network, such as those used for Fig.2, would be able to capture the data better, resulting in in more precise weight distributions and therefore more certainty in found climate patterns.

Techniques for computing the relevance of an input node throughout such a more complex, multi-node and multi-layer neural network do exist, but were not applied in this study. We would recommend further research into combining a Bayesian neural network with these so-called 'explainable AI' techniques. A promising technique is Layer-Wise Relevance Propagation (*32*), which allows for calculating the relevance of nodes of a trained neural network and mapping this relevance back onto the input layer. The result is a type of heatmap, indicating which pixels of an input image are relevant for the neural network to make its predictions. Combinations of Layer-Wise Relevance Propagation (LRP) with a Bayesian approach have been attempted (*33*). However, these techniques are used for neural networks tasked with classification and therefore require some alteration for application to a regression task as posed in this paper. Some strategies have been proposed for this (*34,35*), but not in combination with a Bayesian neural network.

When looking at uncertainty attribution of feature relevance metrics, a few alternatives to Bayesian neural networks exist. For example, Labe and Barnes (2021) combine their ANN architecture and LRP on a large ensemble dataset from the CESM1 climate model and identify uncertainty in their feature relevance heatmaps through a threshold (*36*). This threshold is found by a shuffling approach for generating random data and computing relevance based on the random data. This allows them to define a threshold value where output values of LRP are not likely to be found with random data, resulting in a masked relevance map showing only those locations above the threshold. Feature relevance uncertainty can also be found through the use of Monte-Carlo Dropout sampling (*37*). This technique uses dropout as a Bayesian approximation to find predictive uncertainty and then uses this together with Monte Carlo sampling on LRP outputs to find feature relevance uncertainty. A big difference between these two alternatives and a Bayesian neural network is that both techniques use a normal ANN to optimize for prediction and only afterwards attempt to attribute some uncertainty to the found weights, whereas a Bayesian neural network incorporates uncertainty as a core component of its training algorithm. These alternatives are possibly easier to implement for existing neural networks, as you do not need to change their configuration, but might perform worse in computing the uncertainty of the weights as they are not optimized for finding said uncertainty. We would recommend a comparison study between these techniques to evaluate their performance.

To conclude, Bayesian neural networks can be a useful and robust tool for uncovering uncertainty in the weights of neural networks. In the field of climate science, this uncertainty could be used to quantify the importance of climate patterns found through neural networks. Our results indicate that a Bayesian neural network is less likely to over-fit than an ANN and the Bayesian neural network can generalize almost as well as an ANN can. The resulting posterior on the weights for a simple Bayesian neural network shows a high uncertainty for most of the uncovered climate patterns, limiting their further use. More research in combining a Bayesian neural network with feature relevance techniques is recommended.

MATERIALS AND METHODS

Climate model ensemble

Neural network training and validation is performed with products from the Coupled Model Intercomparison Project phase 5 (CMIP5), which is a community effort "meant to provide a framework for coordinated climate change experiments" (Taylor et al., 2007, p1, (*38*)). They contain a set of model simulations which is run by specific climate models, allowing for easier intercomparison. In this study, long-term experiment results from the so-called historical experiment have been combined with those based on greenhouse gas concentrations derived from the RCP8.5 scenario. The historical experiments run from 1850 to 2005, after which the RCP8.5 experiments continue for 2006 - 2100. Some individual models provide extensions on their runs, e.g. historical experiments running till 2011 or RCP8.5 experiments continuing past 2100. These extensions have not been included in this study. The near-surface air temperature, abbreviated as `tas` in CMIP5 runs, describes the temperature in Kelvin at the so-called 2 metre height. This is the height above the geoid corresponding to a height 2 metres above the surface. For oceans this would correspond to 2 metres, for land this depends on the average land height over a grid cell. The precipitation, abbreviated as `pr` and also called precipitation flux, describes the amount of precipitation over time at the surface height in $kgm^{-2}s^{-1}$, for both liquid and solid phases and from all types of clouds. Retrieval of CMIP5 output data for all used models, for both near-surface air temperature and precipitation, has mainly been through the archive of the Earth System Grid Federation. More information concerning data retrieval can be found in the Supplementary Materials.

Climate modelling groups provide data for monthly mean atmospheric variables, split between the historical and RCP8.5 experiments. The spatial resolution of these models can differ. As the desired input format for our neural networks are yearly-mean, global 4° by 4° maps, processing has been done using the Climate Data Operators command line toolbox (*39*). In order to select the correct years for concatenation of both experiments, we use functions `-selyear` to select $1920 - 2005$ from the historical data and $2006 - 2100$ from the RCP8.5 data and `-mergetime` to merge both files based on their timestamps. Afterwards, `-yearmonmean` and `-remapcon` are used to compute a yearly mean based on the monthly data and remap the data to a 90 cells longitude by 45 cells latitude grid, which corresponds to 4° by 4°. The remapping algorithm used (*40*) is first-order conservative, which means that global average temperature and precipitation values remain constant. The resulting data that is used as input for the neural networks consist of 180 yearly-mean maps per climate model, each consisting of 4050 input units. For the near-surface air temperature, 29 different climate models are used, of which 80% (23 models) are used for training and the remaining 20% (6 models) for testing, resulting in $16.912.800$ data points for training and $4.228.200$ data point for testing. For precipitation, 22 different climate models are used, resulting in 18 models with a total of $12.830.400$ data points for training and 4 models with a total of $3.207.600$ data points for testing.

Artificial neural networks

Artificial neural network modelling is performed using the Tensorflow (*41*) and Keras (*42*) APIs, which allow for easier model building by providing functions that perform most of the complex math for neural networks. An artificial neural network, in the form used here also called a multi-layer perceptron, is simply a set of weighted equations, which are optimized to make predictions. An ANN consists of an input layer vector $a^{[0]}$, an output layer $\hat{y}$ and L hidden layers, indexed by l. Each layer consists out of n units (also called nodes or neurons), which can vary per layer. In order to correctly model the data, a neural network needs to be trained. Training happens through a repeated cycle in which the weights of the equations between each nodes are updated. A single training cycle consists of three phases: forward propagation to compute the prediction with the current weights, backward propagation in which the first derivative of the loss $\mathcal{L}$, that is the mismatch between truth and prediction, is computed to identify in which direction a change of the weights would reduce the loss and finally a weight update phase, in which the weights are changed slightly to improve the ANN model's predictive accuracy. Often, a specific optimizer algorithm is used to determine exactly which weights are updated and by how much.

In forward propagation, nodes are densely connected with each other, meaning that every node's value in layer $l-1$ is used to compute the value of every node in the next layer l. For each node, the value is propagated forward using a linear propagation term,

$$z^{[l]} = w^{[l]}a^{[l-1]} + b^{[l]}, \tag{1}$$

where $z^{[l]}$ is an intermediate value after the linear propagation term, $w^{[l]}$ are the weights between the current node and all nodes in the previous layer, $a^{[l-1]}$ the values of the nodes in the previous layer and $b^{[l]}$ a bias term. The linear term is followed by a non-linear activation function,

$$a^{[l]} = g^{[l]}(z^{[l]}), \tag{2}$$

where $g^{[l]}$ is the activation function for layer l, resulting in a value of the node $a^{[l]}$. Note that this is a vectorized equation and so the value at a specific node is the summation of the forward propagation from each individual node in the previous layer connected to this specific node. The non-linear activation function can vary per hidden layer and needs to be chosen based on the type of prediction to be made. For a regression task, the function for computing the output layer typically does not have a non-linear activation. We use a hyperbolic tangent, that is

$$a^{[l]} = \tanh(z^{[l]}) = \frac{\mathrm{e}^{z^{[l]}} - \mathrm{e}^{-z^{[l]}}}{\mathrm{e}^{z^{[l]}} + \mathrm{e}^{-z^{[l]}}}, \tag{3}$$

as activation function between input layer and hidden layers. Furthermore, no activation function between the last hidden layer and the output layer is used. When in training, the model is initialised with zero bias and random weights using the Glorot Uniform distribution (*43*), denoted by

$$w \sim U\left[-\frac{\sqrt{6}}{\sqrt{n^{[l]} + n^{[l+1]}}}, \frac{\sqrt{6}}{\sqrt{n^{[l]} + n^{[l+1]}}}\right]. \tag{4}$$

After initialisation, the input maps are propagated forward through the neural network. The resulting output $\hat{y}$ is used to evaluate the loss function $\mathcal{L}$. For the ANN we use the mean square error loss,

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2, \tag{5}$$

where N is the total number of input samples used for training and i the index of each specific sample. As the loss is minimized, the division by the constant N presented here is not computed in our implementation. Multicollinearity, which is the occurrence of dependencies among the input variables of a model, is a common problem with regression models. For the CMIP5 datasets used, this could manifest through spatial correlation across the grid points and might result in a neural network that relies on only a few grid cells to make its predictions. As we desire a model that finds large scale climate patterns, a L2-norm regularization is applied to the weights on the input units. This norm, also called ridge regression, is implemented by adding a penalty to the loss function, based on the magnitude of the weights. This reduces the chance of a few very big weights dominating the neural network. The L2-norm is defined as

$$L2 = \lambda \frac{\sum(W^{[1]})^2}{2n^{[0]}n^{[1]}}, \tag{6}$$

where ridge parameter λ is a chosen constant which determines how strongly the weights are regulated, $W^{[1]}$ denotes all weights between the input layer and the first hidden layer and is normalized by the total amount of these weights, which corresponds to $n^{[0]}n^{[1]}$ as the nodes are densely connected. The resulting loss function,

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 + \lambda \frac{\sum(W^{[1]})^2}{2n^{[0]}n^{[1]}}, \tag{7}$$

can resolve the issues caused by multicollinearity. The chosen ridge parameter λ depends on the severity of the multicollinearity. We have experimented with different values for λ and based on predictive testing error performance, the values used are $\lambda = 10^4$ for near-surface air temperature and $\lambda = 10^7$ for precipitation. These values are the same as used by Barnes et al. and so we confirm their choice, based on our own results.

After evaluation of the loss function, the neural network's weights and biases are updated in order to minimize the loss. This is done by changing them, often with a predefined step size, in the direction that corresponds to the biggest reduction in loss function. In order to find this direction, a process called back-propagation is used, in which the gradient of the loss function is evaluated on each weight and bias term. We consider it outside the scope of this paper to discuss how these gradients are computed. When the gradient is known, the weights are updated by a small step in the direction opposite of the gradient,

$$w^{[l]} \leftarrow w^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial w^{[l]}}, \tag{8}$$

where step size α is a constant influencing how much you update the weight. Similarly, the bias is also updated based on its gradient and step size, as

$$b^{[l]} \leftarrow b^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial b^{[l]}}. \tag{9}$$

If α is too big, it might become difficult to find a minimum loss, whereas if α is too small it can take very long to approach the minimum. The computation of the gradient through back-propagation, weight update and choice of α is often performed by a so-called optimizer algorithm which has lower computational costs due to making certain approximations and usually converges faster to a minimum loss. From a wide range of optimizer algorithms and their hyperparameters applied on training the ANNs, we found the AdaDelta optimizer algorithm (*44*) with a learning rate of $\alpha = 0.001$ to perform well for the experiment with two hidden layers of 10 units each. For the experiment used for making the weight maps of Fig. 3, with a neural network model of only one hidden layer with one hidden unit, we found the Adam optimizer algorithm (*45*) with an exponentially decaying learning rate, where the initial learning rate of $\alpha = 0.1$ is multiplied by 0.98 every 100 training batch updates, to provide good performance. A batch is the amount of input samples that are considered in a single gradient computation phase. We have used a common default batch size of 32, which means that 32 input samples are considered for each weight update. A full iteration over the entire input data provided is called an epoch and the total amount of epochs for training has been set to a value of 2000. Both the AdaDelta and Adam optimizer algorithms allow for such a batch-based computation of the gradient. This means that the gradient for all samples in the dataset is estimated by taking a randomly selected subset of the data, which allows for reduced computational costs in the back-propagation and weight update phases, but comes with a drawback of slower convergence to the true minimum loss as it does not consider all gradient information at the same time.

Bayesian modelling and neural networks

A Bayesian neural network is structured very similarly to that of the ANN explained above, with structures of hidden layers, units, forward and backward propagation functions and optimizer algorithms. However, a Bayesian neural network can be seen as a superimposition of a probabilistic model onto a conventional neural networks such as an ANN (*46*) and this causes a fundamental difference in the set-up and training process of the neural network model.

The process used in Bayesian statistics is one of updating your belief based on new evidence presented, while simultaneously taking in account prior knowledge about the event taking place. The backbone for this process is Bayes' theorem. Bayes' theorem describes how evidence E can be used to update your belief on the hypothesis H. This is expressed in the posterior probability $P(H|E)$, which describes how probable it is that the hypothesis H is true given the observed evidence E. Bayes' theorem can be expressed as

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}, \tag{10}$$

where $P(E|H)$ is the likelihood or probability of observing the evidence given the hypothesis is true, $P(H)$ is the prior probability and describes the probability of the hypothesis before the new data E is observed and $P(E)$ is the probability of the evidence happening regardless of hypothesis and also called the marginal likelihood. This marginal likelihood can also be expressed as

$$P(E) = P(E|H) \cdot P(H) + P(E|\neg H) \cdot P(\neg H), \tag{11}$$

where $\neg H$ is the logic negation meaning `not` H. This expression includes the hypothesis but is easier to compute and more intuitive to reason with. In the case of a probabilistic model, we are interested in the distributions on the weights w based on presented data D and rewrite Bayes' theorem into

$$p(w|D) = \frac{p(D|w)p(w)}{\int_w p(D|w')p(w')dw'}, \tag{12}$$

where the denominator, often called the normalization constant or evidence integral, is equated to evidence $p(D)$ and generally easier to calculate than the evidence itself. This evidence integral can be derived from the fact that the sum over all possible hypotheses should be equal to 1. In this equation, the prior distribution $p(w)$ corresponds to the initialisation of the weights. Based on observed data, we can compute the probability density of the likelihood $p(D|w)$ of the data to be represented by the model. If one knows the normalization constant, the posterior distribution $p(w|D)$ could be used to update the models' posterior distribution on the weights. This process of updating your model based on provided data is called Bayesian inference. Even though this might sound simple in theory, in practice solving the marginal likelihood through the integral presented in Eq. 12 is often intractable, meaning that it could be solved but it would not be possible to do so efficiently and would take too many resources to solve. A workaround is found through methods that can be used to approximate the posterior probability.

The approximation method used in the BNN is called variational inference. With this technique, we assume the true posterior $p(w|D)$ to be approximated by a variational posterior $q(w|\theta)$, where θ is a set of parameters that are to be optimized in order to make the best approximation for the true posterior. Variational inference is considered faster and easier to apply on large datasets than the alternative, Markov chain Monte Carlo (MCMC) sampling (*47*). Finding these optimal parameters of q is done through the use of the Kullback-Leibler divergence (KL-divergence). For two distributions with densities $f(x)$ and $g(x)$, one can express the KL-divergence $\mathcal{D}_{KL}$ between these distributions as

$$\mathcal{D}_{KL}[f(x)||g(x)] = \int f(x) \log \frac{f(x)}{g(x)} dx. \tag{13}$$

For our model with distributions introduced in Eq. 12, the KL-divergence between $q(w|\theta)$ and $p(w|D)$ can be simplified and written as

$$\begin{aligned} \mathcal{D}_{KL}[q(w|\theta)||p(w|D)] = \log p(D) \\ + \mathcal{D}_{KL}[q(w|\theta)||p(w)] \\ - \mathbb{E}_{q(w|\theta)}[\log p(D|w)]. \end{aligned} \tag{14}$$

Finding the best variational posterior then equates to finding an an optimal parameter set θ which minimizes the KL-divergence, as denoted by

$$\theta^* = \arg\min_\theta \mathcal{D}_{KL}[q(w|\theta)||p(w|D)], \tag{15}$$

where the $\log p(D)$-term can be ignored as the data can be considered constant. For neural network modelling, this minimization problem is introduced in the form of a loss function,

$$\begin{aligned} \mathcal{L}(\theta|D) = \mathcal{D}_{KL}[q(w|\theta)||p(w)] \\ - \mathbb{E}_{q(w|\theta)}[\log p(D|w)]. \end{aligned} \tag{16}$$

The KL-divergence term between variational posterior and prior in this equation is called the complexity cost, as it incentivises the variational posterior to be close to an often relatively simple prior. The second term is often called the likelihood cost or reconstruction term, as it forces the variational posterior to have a high likelihood with the data, in other words to reconstruct or model the input data well. Note that the complexity cost is completely independent of data and therefore Eq.16 describes a trade-off between modelling the complexity of the data and keeping close to the simplicity prior. In literature and implementations for variational inference, one often will encounter the idea of 'maximizing the Evidence Lower Bound (ELBO)'. This ELBO can be easily derived from Eq. 14 using the property that KL-divergence between two distributions is nonnegative. The resulting inequality,

$$\log p(D) \geq \mathbb{E}_{q(w|\theta)}[\log P(D|w)] - \mathcal{D}_{KL}[q(w|\theta)||p(w)], \tag{17}$$

results in a right-hand side being a 'lower bound on the log-evidence' and maximizing this is equivalent to minimizing the loss function of Eq. 16.

With the provided knowledge on variational inference, a probabilistic model can be superimposed onto a neural network. The approximated KL-divergence between each weight's true and variational posterior, $\mathcal{D}_{KL}[q(w|\theta)||p(w|D)]$, is added as a penalty term to the loss of the neural network and θ can be updated through gradient descent, using a technique called Bayes by Backprop (*27*). The term 'Bayes by Backprop' refers to applying and optimizing a Bayesian probability model through backpropagation on a neural network. Blundell et al. (*27*) show that the loss function can be approximated through Monte Carlo sampling of the variational posterior and in the limit can be considered an integral,

$$\mathcal{L}(\theta|D) \approx \int f(w,\theta)dw, \tag{18}$$

where $f(w,\theta)$ is a function based on Eq.16 and defined by

$$f(w,\theta) = \log q(w|\theta) - \log p(w)p(D|w). \tag{19}$$

Through `Proposition 1` of Blundell et al., they show that, for a variational Gaussian posterior, this loss function can be used to compute gradients in the neural network and subsequently be optimized through normal backpropagation. The approach they use for this is based on a scaling and shifting of a unit Gaussian into the variational Gaussian. The standard deviation σ is parameterised by $\sigma = \log(1 + \exp(\rho))$, resulting in $\sigma = (\mu, \rho)$ and a posterior sample of the weights being computed by means of a draw from a unit Gaussian $\epsilon \sim \mathcal{N}(0, I)$ through $w = \mu + \log(1 + \exp(\rho)) \circ \epsilon$. With $f(w,\theta)$ as defined by Eq. 18, the gradient with respect to the mean can be calculated by

$$\Delta_\mu = \frac{\partial f(w,\theta)}{\partial w} + \frac{\partial f(w,\theta)}{\partial \mu} \tag{20}$$

and the gradient with respect to the standard deviation parameter ρ can be calculated with

$$\Delta_\rho = \frac{\partial f(w,\theta)}{\partial w}\frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(w,\theta)}{\partial \rho}. \tag{21}$$

These gradients are remarkably similar to the gradients found through backpropagation of an ANN. The term $\frac{\partial f(w,\theta)}{\partial w}$ is equivalent to the ANN gradients and so in order to find the optimal mean and standard deviation for the weights, only a scaling and shifting of the normal gradients need to be computed. The eventual update phase on the posterior,

$$\mu \leftarrow \mu - \alpha\Delta_\mu \tag{22}$$

$$\rho \leftarrow \rho - \alpha\Delta_\rho, \tag{23}$$

is also similar to that of the ANN in Eq.8, just for two parameters instead of one. The learning rate α and whole backpropagation process can also be optimized using the exact same optimization algorithms, which is one of the main strengths of this Bayes by Backprop method. For a more detailed discussion on the Bayes by Backprop method, we would like to kindly refer you to the introductory paper by Blundell et al (*27*).

Bayesian neural network modelling is performed using the Tensorflow Probability package (*48*), supplemented with the aforementioned Tensorflow and Keras API's. Weights in the neural network are not represented by draws from a initialisation distribution, but by a prior weight distribution $p(w)$. Choice of the prior can have significant impact om the performance of a Bayesian neural network. While use of a unit Gaussian distrubution, $p(w) = \mathcal{N}(0, 1)$, or similar isotropic Gaussian distributions with a different mean and variance are the standard for BNN priors, improvements might exist in the form of distributions with heavier tail, such as the Laplace or Student's t distributions (*49*). Another option is to learn the prior by training several iterations of an ANN and using the found distribution on the weights as prior for the BNN. We have tested these different options and based on an expected predictive log-likelihood scoring statistic, a Laplace distribution as described by

$$p(x;\mu,b)=\frac{1}{2b}\,\text{exp}\left(-\frac{|x-\mu|}{b}\right), \qquad (24)$$

found through training an ANN, provided the best results. A Laplace distribution was fitted to the weights of each individual ANN layer and then used as the prior for the corresponding layer in the BNN. For near-surface air temperature, the prior distributions used are

$$p(w^{[0]})=\text{Laplace}(-7\times10^{-5},2.8\times10^{-1}),$$
$$p(w^{[1]})=\text{Laplace}(-6.5\times10^{-3},3.8)\text{ and}$$
$$p(w^{[2]})=\text{Laplace}(1.2\times10^{-1},1.4)$$

for the three corresponding layers of the BNN. For precipitation, the prior distributions used are

$$p(w^{[0]})=\text{Laplace}(-2.2\times10^{-5},1.8\times10^{-1}),$$
$$p(w^{[1]})=\text{Laplace}(-4\times10^{-2},2.5)\text{ and}$$
$$p(w^{[2]})=\text{Laplace}(-2.4\times10^{-2},8.8\times10^{-1}).$$

For the 1-unit neural networks used to create Fig.3, a Student's t prior distribution performed better than the Laplace distributions used for the complexer networks. The Student's t prior is defined by

$$p(y;\nu,\mu,\hat{\sigma})=\frac{\Gamma((\nu+1)/2)}{\sqrt{\nu\pi}\Gamma(\nu/2)}(1+y^2/\nu)^{-(\nu+1)/2}, \qquad (25)$$

where ν denotes the degrees of freedom, Γ is the gamma function and $y=(x-\mu)/\hat{\sigma}$ a function for shifting the location μ and scale $\hat{\sigma}$ of the distribution. The used prior distributions used are

$$p(w)=\text{T}(2,-8.3\times10^{-5},1.3\times10^{-3})\text{ and}$$
$$p(w)=\text{T}(2,-4.1\times10^{-5},1.3\times10^{-3})$$

for near-surface air temperature and precipitation, respectively.

Besides a prior, weights are also assigned a variational posterior $q(w|\theta)$ to approximate the posterior distribution on the weights through training of θ. We assume the posterior to be independent between weights and of a Gaussian distribution, so $\theta=(\mu,\sigma)$. For this specific choice of distributions, Tensorflow Probability does not provide built-in solutions for computing the KL-divergence between variational posterior and prior $\mathcal{D}_{KL}[q(w|\theta)||p(w)]$, meaning the approximation introduced in Eq.18 has to be used instead of the complexity cost term of Eq. 16. The BNN is implemented with the reconstruction term being embodied by the mean square error, identical to that of the ANN, as it can be shown that this is equivalent to minimizing the likelihood for this specific regression problem.

In training, identical batch sizes and epochs to those of the ANN have been used. A different non-linear activation function has been used for the BNN, as the leaky Rectified Linear Unit (leaky-RELU), denoted by

$$g^{[l]}(z^{[l]})=\begin{cases} z^{[l]} & \text{if } z^{[l]}>0 \\ 0.01z^{[l]} & \text{otherwise.}\end{cases} \qquad (26)$$

performed better than the hyperbolic tangent of Eq. 3. The Adam optimizer algorithm has been used for all BNN models, with a fixed learning rate of $\alpha = 0.001$ for the experiments with two hidden layers of 10 units each and an exponentially decaying learning rate for the experiments with a neural network model of only one hidden layer with one hidden unit, identical to the ANN implementation for these simpler neural network architectures. No ridge regression is applied, as a BNN's prior on the weights already yields L2-regularisation (*27*). Finally, when using batches smaller than the total amount of data points for training, Tensorflow automatically corrects the loss by normalizing for the batch size. However, this only happens for the reconstruction term and not for the complexity term. A weighting for the complexity term can be added to prevent it from becoming relatively more important than the reconstruction term. What weight is applied on the KL-divergence penalty in the loss term can be tuned by the user, depending on desired balance between reconstruction versus complexity. Based mostly on predictive test error and on expected predictive log-likelihood, we have found a scaling factor of 10^{-4} for temperature and 10^{-3} for precipitation to provide good results for the multi-layered 10 unit by 10 unit neural networks. For the BNN architecture of one hidden layer with 1 hidden unit, the scaling factor for near-surface air temperature chosen was $1/4050$, based on the input size of 4050 nodes, while for precipitation we found an optimal scaling of 10^{-4}.

Additional statistics

Predictive test scoring is computed based on the mean absolute error between prediction $\hat{y}_i$ and truth y_i, defined by

$$S_{\text{MAE}} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|, \tag{27}$$

for each map i out of the total of N maps used for testing. While the testing error can be used as an approximation for the generalization error, it does not take into account the quality of the found posterior uncertainties. A proper scoring rule that provides a trade-off between the predictive skill and the posterior found is the predictive log-likelihood (*49*). This scoring rule is computed as the average predictive log-likelihood, computed with a predictive posterior distribution, found through Monte-Carlo sampling of BNN predictions on the testing dataset, and the true years given by the testing dataset. It is defined by

$$S_\ell = \frac{1}{M}\sum_{i=1}^{M}\ell(q(\hat{y}_i), y_i), \tag{28}$$

where ℓ is the log-likelihood between the predictive posterior distribution $q(\hat{y}_i)$ and truth value y_i, with M as the amount of sampled predictions to compute the average over.

For quantification of the year at which the trained neural networks are able to recognise patterns indicating a changing climate, we use the method proposed by Mora et al. (*28*), which computes which years predicted by the neural network exceeds the bounds of historical variability in a specified baseline period. They state a baseline period of 20 years is sufficient and there is only small difference between the year found for the first few consecutive years outside the bounds and the year found when considering all subsequent years to lie outside the bounds of historical variability. Therefore, we have defined a baseline period from 1920 - 1940, evaluated the years predicted by the test set from that period and defined the ’year of changing climate recognition’ to be the first year after 1940 for which all subsequent years the neural network predicts a later year than the latest predicted year in the baseline period. In order to get a more robust answer, we computed the year of changing climate recognition for 25 iterations of training the neural networks. Other statistics presented in figures and text, such as the required runtime and the predictive errors, have also been compared with the statistics found through 25 iterations in order to validate for robustness.

SUPPLEMENTARY MATERIALS

Supplementary material for this article is available after the references and notes.
Text S1. Extra information on data retrieval and processing
Text S2. More specifics on neural network implementation and accessibility
Fig. S1. Predictive results of neural networks with different subsets of training and testing data.
Fig. S2. Predictive results of neural networks where the global mean is removed.
Fig. S3. Predictive results of neural networks with altered architectures.
Fig. S4. Predictive results of neural networks with 1 layer and 1 hidden unit, used for finding indicator patterns.
Table S1. Properties of CMIP5 model output products for near-surface air temperature, including retrieval locations.
Table S2. Properties of CMIP5 model output product for precipitation, including retrieval locations.
Table S3. Information on used programmes and packages.

DATA AND MATERIAL AVAILABILITY

Information on how to use the provided code or retrieve CMIP5 data is provided through the Supplementary Materials. All data and code used in this paper and required to validate our conclusions can be found online through Github at `https://github.com/jessearens/Bayesian-climate-patterns`.

REFERENCES

1. J.A. Curry and P.J. Webster, Climate science and the uncertainty monster, *Bulletin of the American Meteorological Society* **92**, 1667 (2011).

2. A.J. Watson, Certainty and uncertainty in climate change predictions: What use are climate models?,

Environmental and Resource Economics **39**, 37 (2008).

3. L. Bock, A. Lauer, M. Schlund, M. Barreiro, N. Bellouin, C. Jones, G. A. Meehl, V. Predoi, M. J. Roberts, and V. Eyring, Quantifying Progress Across Different CMIP Phases With the ESMValTool, *Journal of Geophysical Research: Atmospheres* **125**, 1 (2020).
4. V. Masson-Delmotte, P. Zhai, A. Pirani, S.L. Connors, C. Péan, S. Berger, N. Caud, Y. Chen, L. Goldfarb, M.I. Gomis, M. Huang, K. Leitzell, E. Lonnoy, J.B.R. Matthews, T.K. Maycock, T. Waterfield, O. Yelekçi, R. Yu, and B. Zhou, *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change* (Cambridge University Press, Cambridge, United Kingdom, 2021).
5. N. Oreskes, K. Shrader-Frechette, and K. Belitz, Verification, validation, and confirmation of numerical models in the earth sciences, *Science* **263**, 641 (1994).
6. R.D. Cess, G.L. Potter, J.P. Blanchet, G.J. Boer, S.J. Ghan, J.T. Kiehl, H. Le Treut, X.-Z. Liang, J.F.B. Mitchell, J.-J. Morcrette, D.A. Randall, M.R. Riches, E. Roeckner, U. Schlese, A. Slingo, K.E. Taylor, W.M. Washington, R.T. Wetherald, and I. Yagai, Interpretation of Cloud-Climate Feedback as Produced by 14 Atmospheric General Circulation Models, *Science* **245**, 513 (1989).
7. C. Tebaldi and R. Knutti, The use of the multi-model ensemble in probabilistic climate projections, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **365**, 2053 (2007).
8. K.E. Taylor, R.J. Stouffer, and G.A. Meehl, An overview of CMIP5 and the experiment design, *Bulletin of the American Meteorological Society* **93**, 485 (2012).
9. T.F. Stocker, D. Qin, G.-K. Plattner, M. Tignor, S.K. Allen, J. Boschung, A. Nauels, Y. Xia, V. Bex, and P.M. Midgley, *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change* (Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, 2013).
10. V. Eyring, S. Bony, G.A. Meehl, C.A. Senior, B. Stevens, R.J. Stouffer, and K.E. Taylor, Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization, *Geoscientific Model Development* **9**, 1937 (2016).
11. V. Eyring, P.M. Cox, G.M. Flato, P.J. Gleckler, G. Abramowitz, P. Caldwell, W.D. Collins, B.K. Gier, A.D. Hall, F.M. Hoffman, G.C. Hurtt, A. Jahn, C.D. Jones, S.A. Klein, J.P. Krasting, L. Kwiatkowski, R. Lorenz, E. Maloney, G.A. Meehl, A.G. Pendergrass, R. Pincus, A.C. Ruane, J.L. Russell, B.M. Sanderson, B.D. Santer, S.C. Sherwood, I.R. Simpson, R.J. Stouffer, and M.S. Williamson, Taking climate model evaluation to the next level, *Nature Climate Change* **9**, 102 (2019).
12. E. Hawkins and R. Sutton, The potential to narrow uncertainty in regional climate predictions, *Bulletin of the American Meteorological Society* **90**, 1095 (2009).
13. N. Nakicenovic and Rob Swart, Emission Scenarios, *(Intergovernmental Panel on Climate Change) Cambridge University Press* **1**, 570 (2000).
14. D.P. van Vuuren, J. Edmonds, M. Kainuma, K. Riahi, A. Thomson, K. Hibbard, G.C. Hurtt, T. Kram, V. Krey, J-F. Lamarque, T. Masui, M. Meinshausen, N. Nakicenovic, S.J. Smith, and S.K. Rose, The representative concentration pathways: An overview, *Climatic Change* **109**, 5 (2011).
15. K. Riahi, D.P. van Vuuren, E. Kriegler, B.C. Edmonds, J.and O'Neill, S. Fujimori, N. Bauer, K. Calvin, R. Dellink, O. Fricko, W. Lutz, A. Popp, J.C. Cuaresma, S. KC, M. Leimbach, L. Jiang, T. Kram, S. Rao, J. Emmerling, K. Ebi, T. Hasegawa, P. Havlik, F. Humpenöder, L.A. Da Silva, S. Smith, E. Stehfest, V. Bosetti, J. Eom, D. Gernaat, T. Masui, J. Rogelj, J. Strefler, L. Drouet, V. Krey, G. Luderer, M. Harmsen, K. Takahashi, L. Baumstark, J.C. Doelman, M. Kainuma, Z. Klimont, G. Marangoni, H. Lotze-Campen, M. Obersteiner, A. Tabeau, and M. Tavoni, The Shared Socioeconomic Pathways and their energy, land use, and greenhouse gas emissions implications: An overview, *Global Environmental Change* **42**, 153 (2017).
16. R. Knutti and G.C. Hegerl, The equilibrium sensitivity of the Earth's temperature to radiation changes, *Nature Geoscience* **1**, 735 (2008).
17. B.M. Sanderson, M. Wehner, and R. Knutti, Skill and independence weighting for multi-model assessments, *Geoscientific Model Development* **10**, 2379 (2017).
18. P.G. Sansom, D.B. Stephenson, and T.J. Bracegirdle, On Constraining Projections of Future Climate Using Observations and Simulations From Multiple Climate Models, *Journal of the American Statistical Association* (2021).
19. D.W.J. Thompson, E.A. Barnes, C. Deser, W.E. Foust, and A.S. Phillips, Quantifying the role of internal climate variability in future climate trends, *Journal of Climate* **28**, 6443 (2015).

20. J.E. Kay, C. Deser, A. Phillips, A. Mai, C. Hannay, G. Strand, J. M. Arblaster, S.C. Bates, G. Danabasoglu, J. Edwards, M. Holland, P. Kushner, J-F. Lamarque, D. Lawrence, K. Lindsay, A. Middleton, E. Munoz, R. Neale, K. Oleson, L. Polvani, and M. Vertenstein, The community earth system model (CESM) large ensemble project : A community resource for studying climate change in the presence of internal climate variability, *Bulletin of the American Meteorological Society* **96**, 1333 (2015).

21. N. Jones, How machine learning could help to improve climate forecasts, *Nature* **548**, 379 (2017).

22. A. Karpatne, I. Ebert-Uphoff, S. Ravela, H.A. Babaie, and V. Kumar, Machine Learning for the Geosciences: Challenges and Opportunities, *IEEE Transactions on Knowledge and Data Engineering* **31**, 1544 (2019).

23. A. McGovern, R. Lagerquist, D.J. Gagne, G.E. Jergensen, K.L. Elmore, C.R. Homeyer, and T. Smith, Making the black box more transparent: Understanding the physical implications of machine learning, *Bulletin of the American Meteorological Society* **100**, 2175 (2019).

24. M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais, and Prabhat, Deep learning and process understanding for data-driven Earth system science, *Nature* **566**, 195 (2019).

25. E. A. Barnes, J.W. Hurrell, I. Ebert-Uphoff, C. Anderson, and D. Anderson, Viewing Forced Climate Patterns Through an AI Lens, *Geophysical Research Letters* **46**, 13389 (2019).

26. Y. Gal, Uncertainty in Deep Learning, Ph.D. thesis, University of Cambridge (2016).

27. C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, Weight uncertainty in neural networks, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015* **37**, 1613 (2015).

28. C. Mora, A.G. Frazier, R.J. Longman, R.S. Dacks, M.M. Walton, E.J. Tong, J.J. Sanchez, L.R. Kaiser, Y.O. Stender, J.M. Anderson, C.M. Ambrosino, I. Fernandez-Silva, L.M. Giuseffi, and T.W. Giambelluca, The projected timing of climate departure from recent variability, *Nature* **502**, 183 (2013).

29. M. Hutson, Artificial intelligence faces reproducibility crisis Unpublished code and sensitivity to training conditions make many claims hard to verify, *Science* **359**, 725 (2018).

30. D.N. Williams, R. Ananthakrishnan, D. E. Bernholdt, S. Bharathi, D. Brown, M. Chen, A. L. Chervenak, L. Cinquini, R. Drach, I. T. Foster, P. Fox, D. Fraser, J. Garcia, S. Hankin, P. Jones, D. E. Middleton, J. Schwidder, R. Schweitzer, R. Schuler, A. Shoshani, F. Siebenlist, A. Sim, W. G. Strand, M. Su, and N. Wilhelmi, The earth system grid: Enabling access to multimodel climate simulation data, *Bulletin of the American Meteorological Society* **90**, 195 (2009).

31. M.D. Wilkinson, M. Dumontier, IJ.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.W. Boiten, P.E. da Silva Santos, L.B.and Bourne, J. Bouwman, A.J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C.T. Evelo, R. Finkers, A. Gonzalez-Beltran, A.J.G. Gray, P. Groth, C. Goble, J.S. Grethe, J. Heringa, P.A.C. t Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S.J. Lusher, M.E. Martone, A. Mons, A.L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M.A. Swertz, M. Thompson, J. Van Der Lei, E. Van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, and B. Mons, Comment: The FAIR Guiding Principles for scientific data management and stewardship, *Scientific Data* **3**, 1 (2016).

32. S. Bach, A. Binder, G. Montavon, F. Klauschen, K.R. Müller, and W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, *PLoS ONE* **10**, 1 (2015).

33. K. Bykov, M.M.-C. Höhne, K.-R. Müller, S. Nakajima, and M. Kloft, How Much Can I Trust You? – Quantifying Uncertainties in Explaining Neural Networks, *arXiv Preprint, v1* (2020).

34. B.A. Toms, E.A. Barnes, and I. Ebert-Uphoff, Physically Interpretable Neural Networks for the Geosciences: Applications to Earth System Variability, *Journal of Advances in Modeling Earth Systems* **12** (2020).

35. E.A. Barnes, B.A. Toms, J.W. Hurrell, I. Ebert-Uphoff, C. Anderson, and D. Anderson, Indicator Patterns of Forced Change Learned by an Artificial Neural Network, *Journal of Advances in Modeling Earth Systems* **12**, 1 (2020).

36. Z.M. Labe and E.A. Barnes, Detecting climate signals using explainable AI with single-forcing large ensembles, *ESSOAr* pp. 1–40 (2021).

37. K. Fabi and J. Schneider, On Feature Relevance Uncertainty: A Monte Carlo Dropout Sampling Approach, *ArXiv Preprint, v1* (2020).

38. K. Taylor, S.r Ronald, and G. Meehl, A summary of the cmip5 experiment design, *PCDMI Rep.* **4** (2007).

39. U. Schulzweida, CDO User Guide (2019).

40. P.W. Jones, First- and second-order conservative remapping schemes for grids in spherical coordinates, *Monthly Weather Review* **127**, 2204 (1999).

41. M. Abadi, P. Agarwal, A.and Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems (2015). Software available from tensorflow.org.

42. F. Chollet et al., Keras (2015). Software available from keras.io.

43. X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, *Journal of Machine Learning Research* **9**, 249 (2010).

44. M.D. Zeiler, Adadelta: An adaptive learning rate method, *ArXiv Preprint, v1* (2012).

45. D.P. Kingma and J. Ba, Adam: A method for stochastic optimization, *ArXiv Preprint, v9* (2017).

46. V. Mullachery, A. Khera, and A. Husain, Bayesian Neural Networks, *CoRR* (2018).

47. D.M. Blei, A. Kucukelbir, and J.D. McAuliffe, Variational Inference: A Review for Statisticians, *Journal of the American Statistical Association* **112**, 859 (2017).

48. J.V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R.A. Saurous, TensorFlow Distributions, *ArXiv Preprint, v1* (2017).

49. V. Fortuin, A. Garriga-Alonso, S.W. Ober, F. Wenzel, G. Rätsch, R.E. Turner, M. van der Wilk, and L. Aitchison, Bayesian Neural Network Priors Revisited, *ArXiv Preprint, v3* pp. 1–17 (2021).

2

Supplementary Materials

Supplementary Texts

Text S1: Extra information on data retrieval and processing

CMIP5 output data is retrieved through the Earth System Grid Federation. We found the archives provided by the Department of Energy/Lawrence Livermore National Laboratory (DOE/LLNL) to be the most complete and easiest access to the data. They can be accessed through their CMIP5 node. The easiest way to download is through Globus Connect on your personal device. In order to do this, you need an account at DOE/LLNL and an account at Globus. Unfortunately, not all used CMIP5 data was accessible through the DOE/LLNL node at Globus. Supplementary Tables S1 and S2 provide information about which climate models have been included in this study and where they have been retrieved.

Data processing is performed using Climate Data Operators (CDO). CDO is a command-line tool allowing for operations on geospatial datasets. With the used datasets as provided in Tables S1 and S2, there were a few problems in computing the $4°$ by $4°$ global grid, annual mean maps. Combining of historical with rcp8.5 experiments for GFDL-CM3, GFDL-ESM2G and GFDL-ESM-2M resulted in two different parameters for monthly average temperature. In the end, besides a warning, this posed no problems. The precipitation model for HadGEM2-ES contained a double month at the end of the rcp8.5 dataset, which was solved by specifically selecting only up till the first entry. Furthermore, HadGEM2-ES also posed a problem with the historical experiment running a month shorter than the other models, whereas the rcp8.5 experiment started a month earlier. This was solved by selecting 2005 as the start year for the rcp8.5 experiment's processing, which was possible because no other experiments had 2005 in their rcp8.5 data. Similarly, NorESM1-ME had extended historical data, resulting in double months when merging the historical with rcp8.5 experiment. This was solved by explicitly selecting only up to and in including 2005 as data for the historical experiments. The CDO shell script and output `.npz` files are available through the Github repository found at `https://github.com/jessearens/Bayesian-climate-patterns`.

Text S2: More specifics on neural network implementation

Neural network implementation is mainly performed using python with the Tensorflow and Tensorflow Probability packages. The code can be found on Github through `https://github.com/jessearens/Bayesian-climate-patterns`. Required packages to run the code and their version types are provided in Table S3. The repository is split into two folders: `Core` and `Tests`, with the first containing the main functions and scripts to produce the main results presented in this thesis. The latter contains supplementary scripts used for finding optimal hyper-parameters and testing results on robustness. Detailed code description is provided for the `Core` code and is meant to provide sufficient information for replicability and reproducibility of the main experiment.

Supplementary Figures

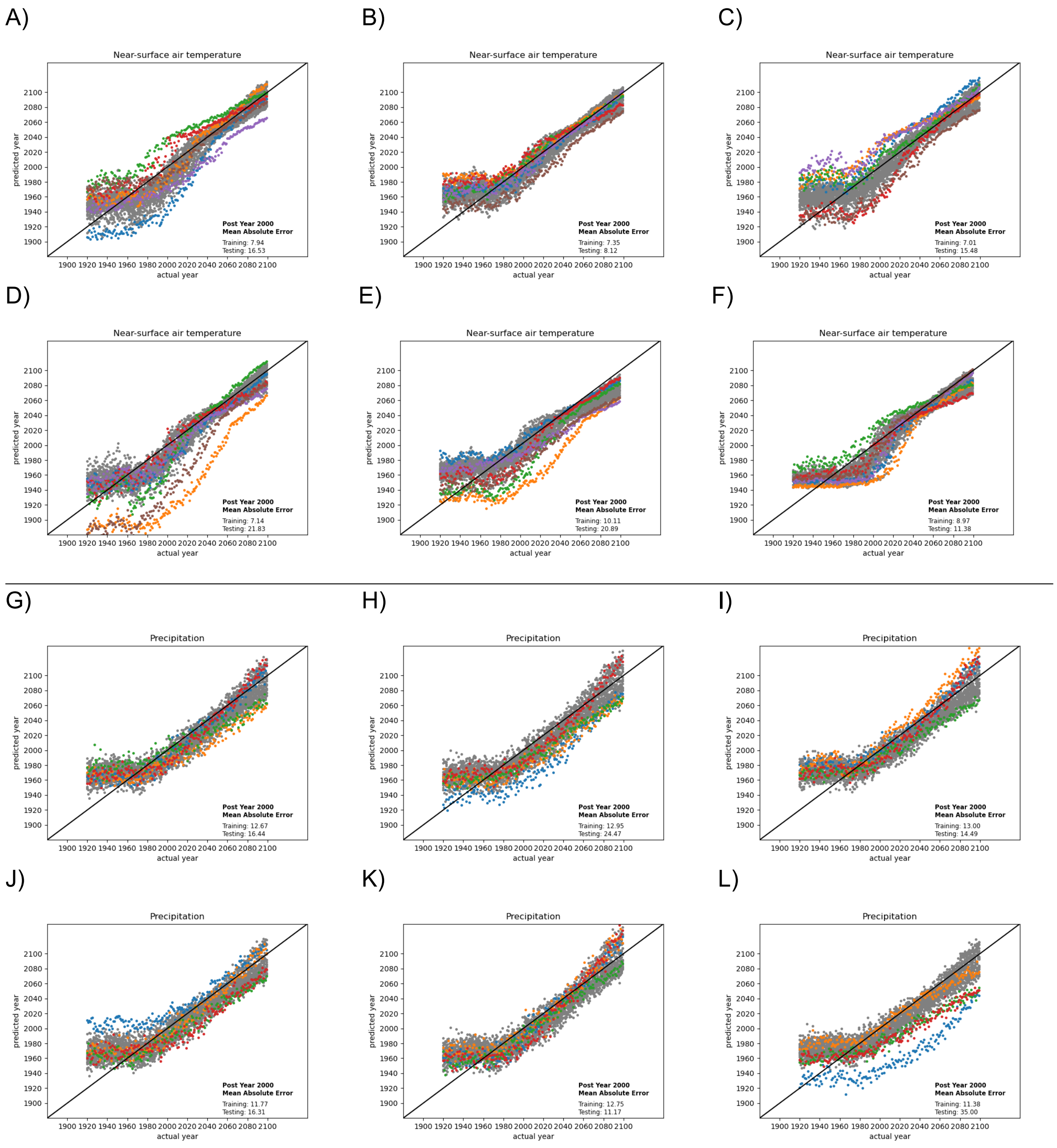


Figure S1: **Prediction results of a Bayesian neural network trained on near-surface air temperature and precipitation input maps, for different subsets of training and testing data.** Predicted year versus actual input map year for both testing and training for (**A-F**) Bayesian neural networks trained on CMIP5 near-surface air temperature maps and for (**G-L**) Bayesian neural networks trained on CMIP5 precipitation maps. Training results are shown in grey, testing results are shown in colors, each representing one climate model's simulation. A 1:1 line is plotted in black. Predictive post-year 2000 mean absolute errors are printed in the lower right corner of the figure and indicate how well the model performs on the second half of the time series.

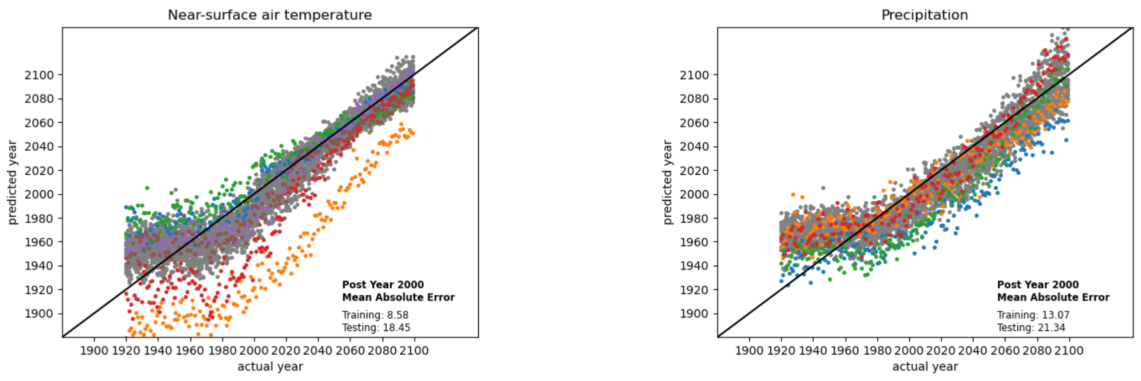
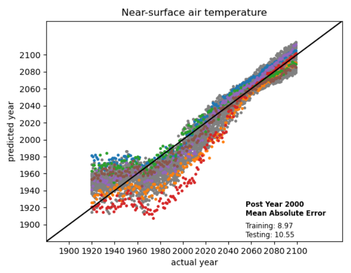
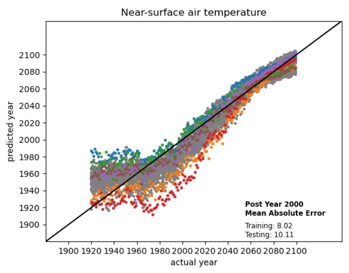


Figure S2: **Prediction results of a Bayesian neural network trained on near-surface air temperature and precipitation input maps, with global mean removed.** Predicted year versus actual input map year for both testing and training for (**A**) a Bayesian neural network fed with CMIP5 near-surface air temperature maps and (**B**) a Bayesian neural network fed with CMIP5 precipitation maps. The global mean value has been removed from all maps. Training results are shown in grey, testing results are shown in colors, each representing one climate model's simulation. A 1:1 line is plotted in black. Predictive post-year 2000 mean absolute errors are printed in the lower right corner of the figure and indicate how well the model performs on the second half of the time series.

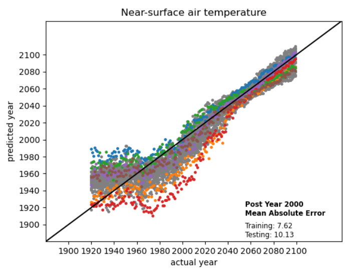
A)



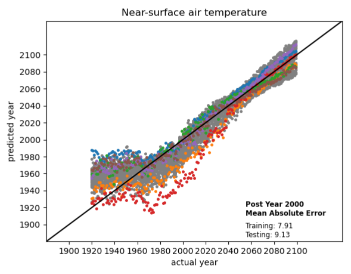
B)



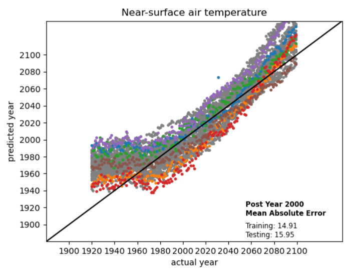
C)



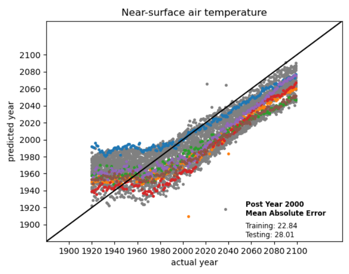
D)



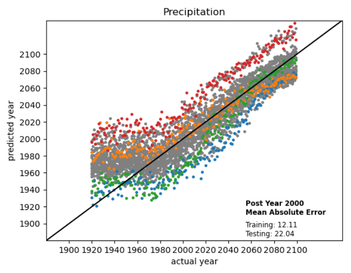
E)



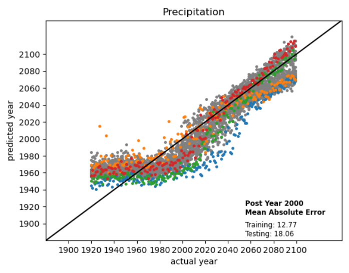
F)



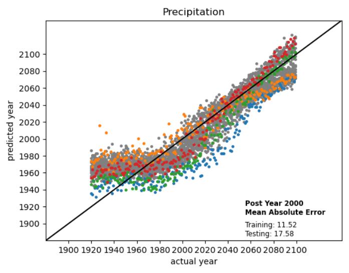
G)



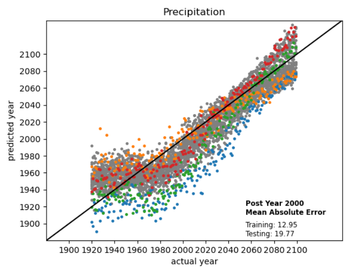
H)



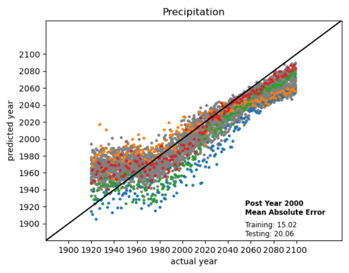
I)



J)



K)



L)

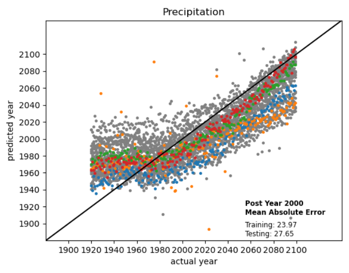
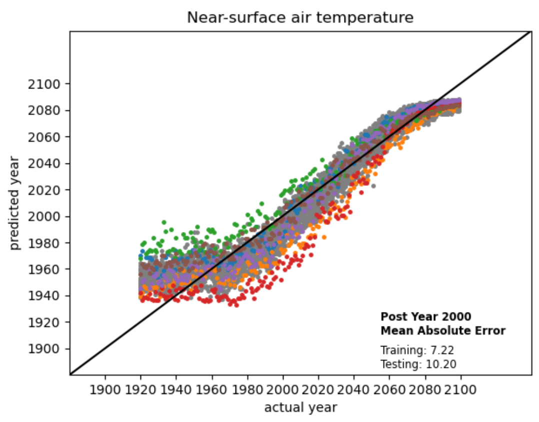
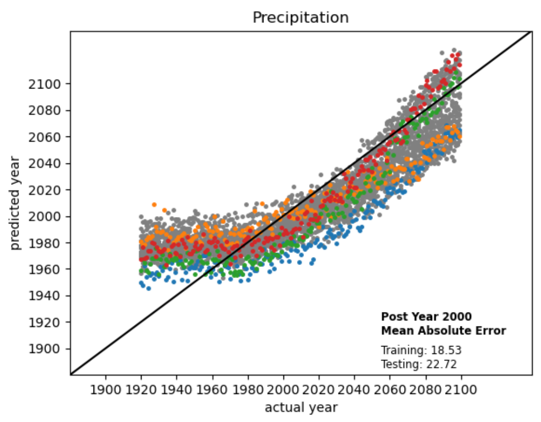


Figure S3: **Prediction results of a Bayesian neural network trained on near-surface air temperature and precipitation input maps, for different architectures of the neural networks.** Predicted year versus actual input map year for both testing and training for (**A-F**) a Bayesian neural network fed with CMIP5 near-surface air temperature maps and (**G-L**) a Bayesian neural network fed with CMIP5 precipitation maps. Different neural network architectures used are: (**A and G**) 3 layers of 10 units each, (**B and H**) 4 layers of 10 units each, (**C and I**) 2 layers of 5 units each, (**D and J**) 2 layers of 20 units each, (**E and K**) 2 layers of 50 units each and finally (**F and L**) 2 layers of 100 units each. Training results are shown in gray, testing results are shown in colors, each representing one climate model's simulation. A 1:1 line is plotted in black. Predictive post-year 2000 mean absolute errors are printed in the lower right corner of the figure and indicate how well the model performs on the second half of the time series.

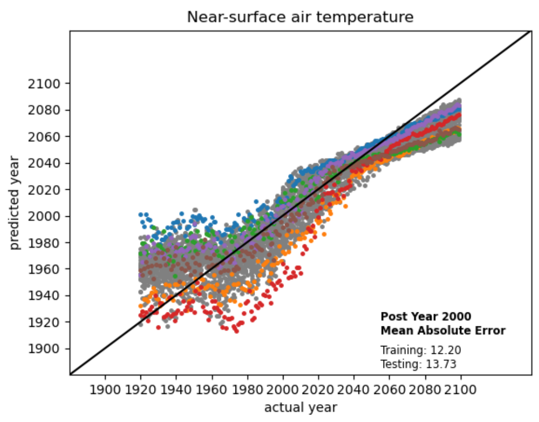
A)



B)



C)



D)

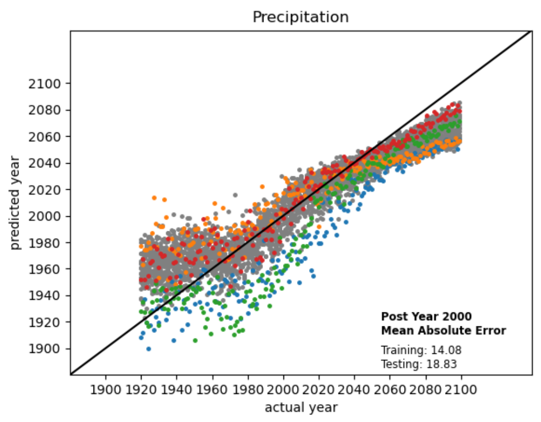


Figure S4: **Prediction results for near-surface air temperature and precipitation.** Predicted year versus actual input map year for both testing for (**A**) An 1-unit artificial neural network fed with CMIP5 near-surface air temperature maps. (**B**) An 1-unit artificial neural network fed with CMIP5 precipitation maps. (**C**) An 1-unit Bayesian neural network fed with CMIP5 near-surface air temperature maps. (**D**) An 1-unit Bayesian neural network fed with CMIP5 precipitation maps. Training results are shown in grey, testing results are shown in colors, each representing one climate model's simulation. A 1:1 line, indicating a perfect prediction, is plotted in black. Predictive post-year 2000 mean absolute errors are printed in the lower right corner of the figure and indicate how well the model performs on the second half of the time series.

Supplementary Tables

Table S1: **Properties of CMIP5 model output products for near-surface air temperature, including retrieval locations.**

Name	Modeling center	Retrieved from	Extra information
ACCESS1-0	CSIRO and BOM	ESGF (DOE/LLNL)	-
ACCESS1-3	CSIRO and BOM	ESGF (DOE/LLNL)	-
CCSM4	NCAR	ESGF (DOE/LLNL)	RCP8.5 forcing through Earth System Grid
CESM1-BGC	CESM-Contributors	ESGF (DOE/LLNL)	-
CMCC-CMS	CMCC	CMCC data services	-
CNRM-CM5	CNRM & CERFACS	ESGF (DOE/LLNL)	two versions available, we used versions v20110901 (historical) and v20110930 (rcp8.5)
CSIRO-Mk3-6-0	CSIRO & QCCCE	ESGF (DOE/LLNL)	-
CanESM2	CCCma	CCCma	esmHistorical and esm-RCP8.5 versions
GFDL-CM3	NOAA-GFDL	ESGF (DOE/LLNL)	-
GFDL-ESM2G	NOAA-GFDL	ESGF (DOE/LLNL)	-
GFDL-ESM2M	NOAA-GFDL	NOAA-GFDL data	esmHistorical and esmRCP8.5 versions, v20110601 -
GISS-E2-H-CC	NASA GISS	ESGF (DOE/LLNL)	rcp8.5 forcing through NASA NCCS data services, v20160512
GISS-E2-H	NASA GISS	ESGF (DOE/LLNL)	-
GISS-E2-R-CC	NASA GISS	ESGF (DOE/LLNL)	rcp8.5 forcing through NASA NCCS data services, v20160512
GISS-E2-R	NASA GISS	ESGF (DOE/LLNL)	-
HadGEM2-AO	METRI KMA	ESGF (DOE/LLNL)	-
HadGEM2-CC	Met Office Hadley Centre	ESGF (DOE/LLNL)	-
HadGEM2-ES	Met Office Hadley Centre	ESGF (DOE/LLNL)	-
inmcm4	INM	ESGF (DOE/LLNL)	-
IPSL-CM5A-LR	IPSL	ESGF (DOE/LLNL)	-
IPSL-CM5A-MR	IPSL	ESGF (DOE/LLNL)	-
MIROC-ESM-CHEM	JAMSTEC & CCSR-NIES	ESGF (DOE/LLNL)	-
MIROC-ESM	JAMSTEC & CCSR-NIES	ESGF (DOE/LLNL)	-
MIROC5	JAMSTEC & NIES & AORI	ESGF (DOE/LLNL)	-
MPI-ESM-MR	MPI-M	ESGF (DOE/LLNL)	-
MRI-CGCM3	MRI	ESGF (DOE/LLNL)	historical forcing through Woods Hole Oceanographic Institution, v20110831
NorESM1-M	NCC	ESGF (DOE/LLNL)	-
NorESM1-ME	NCC	ESGF (DOE/LLNL)	-

Table S2: **Properties of CMIP5 model output product for precipitation, including retrieval locations.**

Name	Modeling center	Retrieved from	Extra information
ACCESS1-0	CSIRO & BOM	ESGF (DOE/LLNL)	-
ACCESS1-3	CSIRO & BOM	ESGF (DOE/LLNL)	-
CMCC-CMS	CMCC	CMCC data services	-
CNRM-CM5	CNRM & CERFACS	ESGF (DOE/LLNL)	two versions available, we used versions v20110901 (historical) and v20110930 (rcp8.5)
CSIRO-Mk3-6-0	CSIRO & QCCCE	ESGF (DOE/LLNL)	-
CanESM2	CCCma	CCCma	esmHistorical and esmRCP8.5 versions
GFDL-CM3	NOAA-GFDL	ESGF (DOE/LLNL)	-
GFDL-ESM2G	NOAA-GFDL	ESGF (DOE/LLNL)	-
GFDL-ESM2M	NOAA-GFDL	NOAA-GFDL data	esmHistorical and esmRCP8.5 versions, v20110601
GISS-E2-H-CC	NASA GISS	ESGF (DOE/LLNL)	rcp8.5 forcing through NASA NCCS data services, v20160512
GISS-E2-H	NASA GISS	ESGF (DOE/LLNL)	-
GISS-E2-R-CC	NASA GISS	ESGF (DOE/LLNL)	rcp8.5 forcing through NASA NCCS data services, v20160512
GISS-E2-R	NASA GISS	ESGF (DOE/LLNL)	-
HadGEM2-CC	Met Office Hadley Centre	ESGF (DOE/LLNL)	-
HadGEM2-ES	Met Office Hadley Centre	ESGF (DOE/LLNL)	-
inmcm4	INM	ESGF (DOE/LLNL)	-
MIROC-ESM-CHEM	JAMSTEC & CCSR-NIES	ESGF (DOE/LLNL)	-
MIROC-ESM	JAMSTEC & CCSR-NIES	ESGF (DOE/LLNL)	-
MIROC5	JAMSTEC & NIES & AORI	ESGF (DOE/LLNL)	-
MRI-CGCM3	MRI	ESGF (DOE/LLNL)	historical forcing through Woods Hole Oceanographic Institution, v20110831
NorESM1-M	NCC	ESGF (DOE/LLNL)	-
NorESM1-ME	NCC	ESGF (DOE/LLNL)	-

Table S3: **Information on used programmes and packages.**

Name	Version	Project location
CUDA Toolkit	11.0	`https://developer.nvidia.com/cuda-toolkit-archive`
CuDNN SDK	8.1.0	`https://developer.nvidia.com/cudnn`
Climate Data Operators	1.9.9rc1	`https://code.mpimet.mpg.de/projects/cdo`
Generic Mapping Tools	6.1.1	`https://www.generic-mapping-tools.org/`
Python	3.8.10	`https://www.python.org`
- matplotlib	3.4.3	`https://matplotlib.org/`
- netCDF4	1.5.8	`http://unidata.github.io/netcdf4-python/`
- numpy	1.20.3	`https://numpy.org/`
- pygmt	0.2.1	`https://www.pygmt.org/`
- scipy	1.7.3	`https://scipy.org/`
- TensorFlow	2.8.0	`https://www.tensorflow.org/`
- TensorFlow GPU support	2.6.0	`https://www.tensorflow.org/install/gpu`
- TensorFlow Probability	0.13.0	`https://www.tensorflow.org/probability/`