Stochastic Parameterizations: Better Modelling of Temporal Correlations using Probabilistic Machine Learning

Raghul Parthipan\textsuperscript{1,2}, Hannah M. Christensen\textsuperscript{3}, J. Scott Hosking\textsuperscript{2,4} and Damon J. Wischik\textsuperscript{1}

\textsuperscript{1}Department of Computer Science, University of Cambridge, Cambridge, UK
\textsuperscript{2}British Antarctic Survey, Cambridge, UK
\textsuperscript{3}Department of Physics, University of Oxford, Oxford, UK
\textsuperscript{4}The Alan Turing Institute, London, UK

Key Points:

• We use a recurrent neural network in a probabilistic framework to create a stochastic parameterization model for the Lorenz 96 system.
• Our model’s competitive performance is due to it better capturing temporal correlations than the baselines.
• The model is shown to generalise to an unseen regime.
Abstract

The modelling of small-scale processes is a major source of error in climate models, hindering the accuracy of low-cost models which must approximate such processes through parameterization. Using stochasticity and machine learning have led to better models but there is a lack of work on combining the benefits from both. We show that by using a physically-informed recurrent neural network within a probabilistic framework, our resulting model for the Lorenz 96 atmospheric simulation is competitive and often superior to both a bespoke baseline and an existing probabilistic machine-learning (GAN) one. This is due to a superior ability to model temporal correlations compared to standard first-order autoregressive schemes. The model also generalises to unseen regimes. We evaluate across a number of metrics from the literature, but also discuss how the probabilistic metric of likelihood may be a unifying choice for future probabilistic climate models.

Plain Language Summary

How can we create better climate models? Accurate climate models are essential for us to plan and adapt to climate change. We tackle this by combining probability models, which allow us to represent our uncertainty, with machine learning, a technique to learn relationships from data which are undiscoverable to humans. Our model is competitive and often superior to a both an existing machine learning baseline and a bespoke non-machine learning one when tested in a simple atmospheric simulation. This is due to our model better capturing trends through time. We need climate models that work well into the future, and we show that our methods can indeed generalise to such scenarios. We therefore suggest that the next wave of climate modelling success may come from further work combining the benefits of state-of-the-art machine learning with probabilistic modelling to create more accurate models.
1 Introduction

A major source of inaccuracies in climate models is due to ‘unresolved’ processes. These occur at scales smaller than the resolution of the climate model (‘sub-grid’ scales) but still have key effects on the overall climate. In fact, most of the inter-model spread in how much global surface temperatures increase after CO$_2$ concentrations double is due to the representation of clouds (Schneider et al., 2017; Zelinka et al., 2020). The typical approach to deal with the problem of unresolved processes has been to model the effects of these unresolved processes as a function of the resolved ones. This is known as ‘parameterization’.

In parameterization work the state of the system at timestep $t+1$, $X_{t+1}$, is obtained by a computation on $X_t$. $X$, may include, for example, the temperature and humidity in every grid cell of a specified area over Earth at time $t$. The goal is to approximate unresolved processes in such a way as to learn an update scheme which reproduces the evolution of the real atmosphere. A simple way to model the evolution would be

$$X_{k,t+1} = X_{k,t} + f_k(X_t) \tag{1}$$

where $X_{k,t}$ is the value that the state variable $X$ takes at the spatial coordinate $k$ and time point $t$ and $X_{k,t} \in \mathbb{R}^d$, $X_t \in \mathbb{R}^{dK}$, and $f$ is a function for the updating process.

A more flexible form which can be used to understand the landscape of the prior work is

$$X_{k,t+1} = X_{k,t} + f_k(X_t, h_{k,t+1}) \tag{2}$$

$$h_{k,t+1} = \beta(h_{k,t}, z_{k,t+1}) \tag{3}$$

$$z_{k,t} \sim \mathcal{N}(0, 1) \tag{4}$$

where invented hidden states (discussed below) are denoted $h_{k,t} \in \mathbb{R}^H$ and are tracked through time. $z_{k,t} \in \mathbb{R}^Z$ is a source of stochasticity.

1.1 Stochastic Schemes

Introducing randomness through (2) improved on models of the form (1). Results included better ensemble forecasts (Buizza et al., 1999; Leutbecher et al., 2017; Palmer, 2012), and improvements to model mean state (Berner et al., 2012) and climate variability (Christensen et al., 2017). The motivation for using stochasticity comes from the understanding that the effects of the unresolved (sub-grid) processes cannot be effectively predicted as a deterministic function of the resolved ones due to a lack of scale separation between them. Adding randomness allows us to capture our uncertainty about those aspects of the unresolved processes which may affect the resolved outcomes.

1.2 Hidden States

Hidden states — $h_{k,t}$ in (2) and (3) — are defined here as being states separate to the observed ones, $X_{k,t}$, which if tracked help better model $X_{k,t}$. The use of them is key to stochastic weather and climate models, allowing temporal correlations to be better modelled. Evolving them with a red noise first-order autoregressive (AR1) process such as

$$h_{k,t+1} = \phi h_{k,t} + \sigma(1 - \phi^2)^{1/2} z_{k,t+1} \tag{5}$$

is commonplace. One example is the stochastically perturbed parameterization tendencies (SPPT) scheme (Buizza et al., 1999; Palmer et al., 2009) which is widely used in forecasting models (Leutbecher et al., 2017; Palmer et al., 2009; Sanchez et al., 2016; Stockdale et al., 2011). Here, the AR1 process results in far better weather and climate forecasting skill than a simple white noise model, with Christensen et al. (2015) showing that good modelling of regime behaviour requires correlated hidden states.
There is no intrinsic reason why an AR1 process is the appropriate way to deal with these correlations though. It is simply an arbitrary model choice.

1.3 Use of High-Resolution Simulations

A popular approach to learning improved parameterization schemes is using high-resolution data as surrogates for observations (Christensen, 2020). This has several benefits over using observations, such as excellent spatio-temporal coverage and incorporation of model variables which may be poorly constrained by observations. The goal then becomes matching the accuracy of the high-resolution simulator using the lower cost, lower-resolution model. This involves learning a data-driven update scheme for the low-resolution model. However, high-resolution simulations are merely proxies for the truth and so any biases in them must be understood when evaluating learnt schemes against observational data.

1.4 Machine Learning for Parameterization

Learning the parameters of a climate model, either the simple form (1) or the general form (2)–(4), requires us to decide on a parametric form for the functions $f$ and $\beta$. For full-scale climate models, it is difficult to invent appropriate functions. The machine learning (ML) approach is to learn these from data. Various researchers have proposed ML methods for learning the deterministic no-hidden-state model (1) (Brenowitz & Bretherton, 2018, 2019; Gentile et al., 2018; Krasnopolsky et al., 2013; O’Gorman & Dwyer, 2018; Rasp et al., 2018; Yuval & O’Gorman, 2020; Yuval et al., 2021).

Various ML-trained stochastic models with red-noise hidden states have been proposed by Gagne et al. (2020), using a technique known as Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). Full details of the architecture are in Gagne et al. (2020) and we will refer to one of their best-performing models (which they call X-sml-r) as the GAN. A wider range of generative models can be trained using such an adversarial approach as opposed to maximum likelihood (the standard way to train models in ML, discussed in Section 3) but such methods are notoriously unstable due to the nature of the minimax loss in training. Although they used ML to learn $f$ in (2), it was not used to learn $\beta$ instead sticking to an AR1 process.

Recurrent Neural Networks (RNNs) are popular ML tools which can model temporally correlated data, removing the need for update functions to be manually specified. They have had great success in the ML literature in a variety of sequence modelling tasks, including text generation (Graves, 2013; Sutskever et al., 2011), machine translation (Sutskever et al., 2014) and music generation (Eck & Schmidhuber, 2002; Mogren, 2016). The state-of-the-art RNNs are gated ones, principally the long short-term memory (LSTM) networks (Hochreiter & Schmidhuber, 1997) and gated recurrent unit (GRU) networks (Cho et al., 2014), which provide major improvements to the standard issue of unstable gradients in vanilla RNNs.

Current parameterization work using ML to model temporal correlations has used deterministic, relatively expensive models such as deterministic RNNs and echo state networks (Chattopadhyay, Hassanzadeh, & Subramanian, 2020; Chattopadhyay, Subel, & Hassanzadeh, 2020; Vlachas et al., 2018). Although improved results were seen, many of the architectures used were deep enough such that it is not obvious they would be any cheaper than the high-resolution truth model, based on the number of mathematical operations per simulation step. Moreover, abilities to generalise where either limited or untested.

Other off-the-shelf models are not obviously suited for the parameterization task. Transformers and attention-based models (Vaswani et al., 2017) perform well for sequences but require all the previous data to be tracked for each generative step, providing a computational burden which increases the cost of simulation. Random Forests (RFs) have
been used for parameterization too (Yuval & O’Gorman, 2020) and shown to be stable at run-time (due to predicting averages from the training set) but it is not obvious how they would learn and track hidden states.

1.5 Performance in Unseen Regimes

There is limited work in the parameterization field on creating ML models that generalise to unseen regimes (e.g. future climates). This is an important part of evaluation though as we wish our models to perform well into the future if we are to make decisions based on them. Christensen et al. (2015) evaluated how well their model performed in a regime different to training, but this was not a ML model. For ML models, training data with sufficient coverage of a variety of regimes is likely needed in order to able to extrapolate to new ones (O’Gorman & Dwyer, 2018; Scher & Messori, 2019). For example, O’Gorman and Dwyer (2018) used RFs to assess performance on different climate regimes and found the model did well on a cooler climate when only trained on a warmer climate. This was suggested as being due to the ability to see samples similar to the cooler climate in the training dataset (warmer climate).

1.6 Overview

In this work, we use the Lorenz 96 model (Lorenz, 1996), henceforth the L96, as a proof-of-concept. We learn a more flexible evolution of the hidden states, \( h_{k,t} \), by casting the problem in a probabilistic framework and using ML. This combines the benefits of stochastic modelling with the power of ML. The resulting model is competitive with, and often outperforms, a bespoke baseline. Importantly, our ML model can generalise to an unseen regime. This should alleviate concerns amongst the climate science community, and scientific community in general, about the extrapolation abilities of ML models.

The L96 set-up and baselines are presented in Section 2. Our model is detailed in Section 3, with the experiments and results in Section 4. This is followed by a discussion on the use of ‘likelihood’ in evaluating probabilistic climate models (Section 5), with our conclusions in Section 6.
2 Parameterization in the Lorenz 96

We introduce the L96 model here and then present two L96 parameterization models from the literature which help clarify the above discussion and serve as our baselines.

2.1 Lorenz 96 Model

We use the two-level L96 model, a toy model for atmospheric circulation that is extensively used for stochastic parameterization studies (Arnold et al., 2013; Crommelin & Vanden-Eijnden, 2008; Gagne et al., 2020; Kwasniok, 2012; Rasp, 2020). We use the configuration described in Gagne et al. (2020). It consists of two scales of variables: a large, low-frequency variable, $X_k$, coupled to small, high-frequency variables, $Y_{j,k}$. These are dimensionless quantities, evolving as follows:

$$\frac{dX_k}{dt} = -X_k - \frac{1}{b}(X_k - 2 - X_k + 1) + F - \frac{hc}{b} \sum_{j=J(k-1)+1}^{k,l} Y_j \quad k = 1, ..., K$$

$$\frac{dY_{j,k}}{dt} = -c b Y_{j+1,k} (Y_{j+2,k} - Y_{j-1,k}) - c Y_{j,k} - \frac{hc}{b} X_k \quad j = 1, ..., J$$

where in our experiments, the number of $X_k$ variables is $K = 8$, and the number of $Y_{j,k}$ variables per $X_k$ is $J = 32$. The value of the constants are set to $h = 1$, $b = 10$ and $c = 10$. These indicate that the fast variable evolves ten times faster than the slow variable and has one-tenth the amplitude.

The L96 model is good for parameterization work as we can consider the $X_k$ to be coarse processes resolved in both low-resolution and high-resolution simulators, whilst the $Y_{j,k}$ can be considered as those that can only be resolved in high-resolution, computationally expensive simulators.

In this study the coupled set of L96 equations are treated as the ‘truth’, and the aim is to learn a good model for the evolution of $X$ using this ‘truth’ data. Success would be if the modelled evolution of $X$ matched that from the truth.

2.2 Parameterization Models in the Literature

2.2.1 Stochastic Non-ML Model with Non-ML Hidden States

Arnold et al. (2013) propose a model which we refer to as the polynomial model (in light of the polynomial in (6)) which is

$$X_{k,t+1} = X_{k,t} + \omega_k(X_t) - \Delta t \left( aX_{k,t}^3 + bX_{k,t}^2 + cX_{k,t} + d + h_{k,t+1} \right) \quad (6)$$

where $h_{k,t}$ evolves as in (5) with $h_{k,1} = \sigma z_{k,1}$, and where

$$\omega_k(X_t) = \lambda_k \left( X_t + \lambda(X_t/2) \right) \quad (7)$$

and

$$\lambda_k(X_t) = \Delta t \left( -X_{k-1,t}(X_{k-2,t} - X_{k+1,t}) - X_{k,t} + F \right) \quad (8)$$

where (7) is the implementation of a second order Runge-Kutta method and $[\lambda(a)]_k = \lambda_k(a)$.

$h_{k,t}$ only depends on $h_{k,t-1}$ as it evolves via an AR1 process. This need not be the case, nor must (6) be a polynomial of degree three.
2.2.2 Stochastic ML Model with Non-ML Hidden States

The GAN from Gagne et al. (2020) replaces (6) and (5) by

\[ X_{k,t+1} = X_{k,t} + \omega_k(X_t) - \Delta t \ U(X_{k,t}, h_{k,t+1}) \]  \hfill (9)

\[ h_{k,t+1} = \phi h_{k,t} + (1 - \phi^2)^{1/2} z_{k,t+1} \]  \hfill (10)

where the function \( U \) is implemented by a neural network (NN), \( \omega \) is defined in (7)–(8) and \( h_{k,1} = z_{k,1} \).
3 Our Proposed RNN Model

3.1 Stochastic ML Model with ML Hidden States

Our model, henceforth denoted the RNN, follows the general form in (2)–(4) but splits the hidden state into two parts: \( h_{k,t} = (r_{k,t}, l_{k,t}) \). The model is

\[
X_{k,t+1} = X_{k,t} + \omega_k(X_t) - \Delta t \left( g_\theta(X_{k,t}) + r_{k,t+1} \right) \tag{11}
\]

\[
r_{k,t+1} = b_\theta(l_{k,t+1}) + \sigma z_{k,t+1} \tag{12}
\]

\[
l_{k,t+1} = s_\theta(l_{k,t}, r_{k,t}) \tag{13}
\]

where \( g, b \) and \( s \) are NNs with weights \( \theta \), \( z_{k,t} \) is exogenous noise as defined in (4), \( \omega \) is a function (equation (7) above) implementing a second order Runge-Kutta method and expressing physical quantities like advection, and \( \theta \) and \( \sigma \) are parameters to be learnt.

The dimensionality of the NNs are:

- \( g_\theta : \mathbb{R}^1 \to \mathbb{R}^1 \)
- \( b_\theta : \mathbb{R}^8 \to \mathbb{R}^1 \)
- \( s_\theta : \mathbb{R}^9 \to \mathbb{R}^8 \).

It is structurally based on a recurrent neural network (RNN) as it has hidden states, the same update procedure is used each step, and \( g, b \) and \( s \) are NNs.

Figure 1 shows the mechanism of generation and the NN architectures used to learn the functions. The dotted lines denote the action of deterministic functions, with the solid lines showing the sources of stochasticity. The main architectural details are that \( s \) in (13) is implemented using two GRU layers, each composed of four units, and \( b \) in (12) is represented with a dense layer of size one. \( g \) in (11) is implemented using three fully-connected layers.

The key insight here is our model allows more flexibility than the standard AR1 processes for expressing temporal correlations. Our results (Section 4) suggest that it can be trained effectively.

Here, as well as in the baselines, the parameterization models are ‘spatially local’ meaning that the full \( X_t \) vector is not taken in as input when modelling \( X_{k,t+1} \) anywhere apart from in \( \omega_k(X_t) \). This is merely a simplifying assumption and is not a requirement of our method.

3.2 Training Probability Models Using Likelihood

The RNN is probabilistic and trained using likelihood. The ‘likelihood of a sequence’ is denoted \( \Pr(x_1, x_2, ..., x_n) \) and can be interpreted as the probability assigned to a given sequence of variables \( (x_1, x_2, ..., x_n) \) by a given model.

For our RNN, the log-likelihood (the natural logarithm of the likelihood) of the sequence of \( X_t \) is given by

\[
\log \Pr(x_1, ..., x_n; x_0) = \sum_{k=1}^{K} \left( \log \Pr(r_{k,1}) + \sum_{t=2}^{n} \log \Pr(r_{k,t}|l_{k,t}) \right) - Kn \log(\Delta t)
\]

where \( r \) and \( l \) are deterministic functions of \( x_0, ..., x_n \). The derivation is in Appendix A.

A common way to train probabilistic models is to maximise the likelihood of the training data. This involves finding a set of parameters \( (\theta \) and \( \sigma \) here) which maximise the likelihood. Other enhancements to training involve including regularisation penalties such as dropout. The explicit form of the RNN’s likelihood makes training relatively easy.

3.3 RNN Training

Truth data for training was created by running the full L96 model five separate times with the following values of \( F \): \( 19, 20, 20.5, 21, 22 \). We gave our model data from different forcing scenarios so it could learn how perturbations in the forcing may affect the
Figure 1. RNN graphical model showing how each $X_t$ is generated. $X_{t+1}$ is a function of $X_t$ and $r_{t+1}$, and $r_t$ is a function of $l_t$ and $z_t$. $z_t$ is the source of stochasticity. $b_\theta$, $s_\theta$ and $g_\theta$ are NNs. $l_t$ consists of a stack of $l^1_t$ and $l^2_t$, each $\in \mathbb{R}^4$. 
evolution of \( \mathbf{X}_t \). The simulations were created by solving the two-level L96 equations using a fourth-order Runge-Kutta timestepping scheme and a fixed time step \( dt = 0.001 \) model time units (MTU), with the output saved at every 0.005 MTU. One MTU is approximately five atmospheric days when considering the time it takes for errors to double. A training set of length 2,500 MTU was assembled from the truth data, consisting of three equal 500 MTU components with \( F = (19, 20.5, 21) \) and one 1,000 MTU component with \( F = 20 \), with a \( F = 22 \) set of length 500 MTU kept as a validation holdout set.

The RNN was trained using truncated back propagation through time on sequences of length 700 time steps for 100 epochs with a batch size of 32 using Adam (Kingma & Ba, 2014), with \( \theta \) and \( \sigma \) being the learnable parameters. A variable learning rate was used, starting at 0.0001 for the first 70 epochs and decayed to 0.00003 for the remaining 30. The model parameters which gave the lowest loss on the validation set were saved.
4 Results

This section analyses the models following standard approaches in the parameterization literature. All results here are for the $F = 20$ regime unless stated otherwise. Assessing the model in the training regime allows us to verify if it can replicate the L96 attractor. Later experiments are for $F = 23$. For both regimes, 50,000 MTU long simulations (≈ 685 ‘atmospheric years’) were created for analysis.

4.1 Weather Evaluation

The models are evaluated in a weather forecast framework. 745 initial conditions were randomly selected from the truth attractor and an ensemble of 40 forecasts each lasting 3.5 MTU were produced from each initial condition. Figure 2 shows the spread and error terms for these experiments over time. The error is defined as

$$\text{error}(t) = \sqrt{\frac{1}{M} \sum_{m=1}^{M} (X_{m}^{O}(t) - X_{m}^{\text{sample}}(t))^2}$$

where there are $M$ different initial conditions, $X_{m}^{O}(t)$ is the observed state at time $t$ for the $m^{th}$ initial condition, and $X_{m}^{\text{sample}}(t)$ is the ensemble mean forecast at time $t$, initialized at $t_{\text{init}}$, such that $t = t_{\text{init}} + \tau$.

The spread is defined as

$$\text{spread}(t) = \sqrt{\frac{1}{M} \sum_{m=1}^{M} \frac{1}{N} \sum_{n=1}^{N} (X_{m,n}(t) - X_{m}^{\text{sample}}(t))^2}$$

where $N$ is the number of ensemble members and $X_{m,n}(t)$ is the state of the $n^{th}$ member at time $t$ for the $m^{th}$ initial condition.

As noted in Leutbecher and Palmer (2008), for a perfectly reliable forecast (defined as one where $X_{m,n}(t)$ and $X_{m}^{O}(t)$ are independent samples from the same distribution), for large $N$ and $M$ the error should be equal to the spread. A reduction in error indicates an ensemble forecast that better tracks observations, and a spread/error ratio close to one indicates a reliable forecast. Figure 2 shows that the polynomial and RNN have the smallest errors, but the RNN has the better spread/error ratio after 0.75 MTU.

4.2 Climate Evaluation

The ability of each model to simulate the climate of the L96 was evaluated using the 50,000 MTU simulations. The histograms in Figure 3a are approximations of each model’s marginal distribution of $X_{k,t}$. Successful reproduction of the L96 climate would result in a model’s histogram matching the truth model’s. Qualitatively, all the models are rather similar to the truth.

We can quantitatively compare how well histograms match the truth using KL-divergence. We compute this as follows: we discretize our modelled continuous distribution of a variable by binning into a histogram, and do the same for the truth model. We then evaluate the KL-divergence between $q(a)$, the distribution described by the histogram of a variable $A$ in the true L96 model, and $p(a)$, the distribution described by the histogram of that variable in our model

$$\text{KL}(q(a) || p(a)) = \sum_{a} q(a) \log \frac{q(a)}{p(a)}$$

where the sum is over all the discrete values which $A$ takes. The smaller this measure, the better the goodness-of-fit. The results (Table 1 column one) confirm that the RNN best matches the truth model here.
Figure 2. Error and spread for weather experiments. We expect a smaller error for a more accurate forecast model. The spread/error ratio would be close to one in a ‘perfectly reliable’ forecast.

Figure 3. Histograms of $X_{k,t}$. (a) Comparisons for the full simulation data. (b) Depiction which shows sampling error. The truth histogram is shown in green and bold. The closer a model’s histogram to the truth, the better it models the probability density of $X_{k,t}$. 
Table 1. KL-divergence (goodness-of-fit) between 1) the truth and 2) the distributions shown in the noted Figures. The smaller the KL-divergence, the better the match between the true and modelled distributions. The best model in each case is shown in **bold**.

| Model   | Figure 3a | Figure 4 | Figure 5a | Figure 5b | Figure 6 | Figure 7a | Figure 7b |
|---------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Polynomial | 0.004    | 3.9       | 0.03      | 0.040     | 1.3       | **0.002** | 0.026     |
| RNN      | **0.001** | **1.8**   | **0.02**  | **0.004** | **1.0**   | 0.007     | 0.009     |
| GAN      | 0.010     | 11.6      | 0.14      | 0.050     | 2.5       | 0.017     | **0.007** |

It is also important to check whether the histograms are composed of enough data such that they are good representations of each model’s actual distribution of $X_{k,t}$. Figure 3b seeks to answer this by accounting for sampling error. By splitting a particular model’s simulation run into five equally sized sets and plotting these histograms on top of each other, variability can be made apparent. Having done this, there is no noticeable difference between Figure 3a and b, suggesting that the error from the sampling process is small relative to the differences between each model.

The L96 model used here displays two distinct regimes as shown in Christensen et al. (2015), where regimes were determined using a metric based on temporally-local spatial covariances. We take a similar approach to examine the regimes. Briefly, the time series are temporally smoothed with a running average over 0.4 MTU, and in order to reduce the dimensionality of the system Principal Component Analysis is used with four components to decompose the time series. For the truth time series, the components $PC_1$ and $PC_2$ are degenerate and are in phase quadrature, representing wavenumber two oscillations. $PC_3$ and $PC_4$ are also degenerate and in phase quadrature, representing wavenumber one oscillations. All model runs are projected onto the truth model’s components. Given the degeneracies, the magnitude of the principal component vectors, $||[PC_1, PC_2]||$ and $||[PC_3, PC_4]||$ are computed and histograms of the system are plotted in this space.

The presence of two regimes is apparent in Figure 4a where there is a large maximum corresponding to the major regime, A, but also a smaller peak corresponding to the minor regime, B. The polynomial fails to capture the two regimes, whereas the ML models do. The RNN puts an appropriate amount of density in Regime B, whilst the GAN puts too much. Comparison is made easier by decomposing the 2D Figure 4 into two, 1D density plots (Figure 5). We can use the KL-divergence to measure how well our models match the truth across all these three different histograms. Table 1 shows the RNN best matches the truth across all three histograms.

4.3 Generalisation Experiment

The results below are for where the L96 forcing is set to $F = 23$. This is different to the $F$ values in the training data so allows us to test generalisation. The ability for models to simulate this climate was explored in a similar manner to the $F = 20$ one by looking at the Principal Component projections. The same components as determined from the $F = 20$ truth data set are used. Figure 6 is created in an analogous manner to Figure 4. The RNN has the smallest KL-divergence (Table 1).

As before, further comparison can be made by examining the two, 1D density plots in Figure 7 which are analogous to Figure 5. The RNN is competitive in both cases according to the KL-divergence (Table 1).
Figure 4. Regime characteristics of the L96. 2D histograms show the magnitudes of the projections of $X_{k,t}$ on to the principle components from the truth series.

Figure 5. Density plots for each regime for $F = 20$. The truth is shown in green and bold. (a) Magnitude of $[PC1, PC2]$. (b) Magnitude of $[PC3, PC4]$. 
Figure 6. Regime characteristics of the L96 for $F = 23$. 2D histograms show the magnitudes of the projections of $X_{k,t}$ on to the principal components from the $F = 20$ truth series.

Figure 7. Density plots for each regime for $F = 23$. The truth is shown in green and bold. (a) Magnitude of $[PC1,PC2]$. (b) Magnitude of $[PC3,PC4]$. 
4.4 Computational Cost

We wish our models to have a lower cost than the full L96. The computational cost is calculated by considering the number of operations carried out per simulated time step ($dt = 0.005$). The RNN (8,682) is notably cheaper than the truth model (88,000) and the GAN (14,074), so meets the computational cost objective of parameterization work. The polynomial (334) is the cheapest.
5 Evaluation Using Likelihood

5.1 Hold-out Likelihood

Likelihood can be used to evaluate models, alongside existing metrics. This is done by evaluating the likelihood (explained in Section 3.2) of hold-out data. The concept of having separate training and hold-out sets is common to the ML literature and using hold-out likelihood for evaluation is a standard approach. Here, the hold-out sets for $F = 20$ and $F = 23$ are created by taking a 10,000 MTU subset from the 50,000 MTU sets created in Section 4. The hold-out log-likelihoods are shown in Table 2. Further detail on these calculations is given below in Section 5.3.

5.2 Why Likelihood is Useful

Likelihood can capture things about a model which other metrics alone cannot, as it is a composite measure which assesses a model’s full joint distribution. For example, in the univariate case, the mean-squared-error and variance of a model tell two separate things about its performance, with implicit assumptions being made about variables being normally distributed (whenever MSE is used). The likelihood captures both of these, and without such assumptions. It also captures information pertaining to more complex, joint distributions, saving the need for custom metrics to be invented to assess specific features. To illustrate this, consider we wish to assess a model’s temporal correlation. The likelihood already contains this information. Although custom metrics could be invented to assess this, the likelihood is an off-the-shelf metric which is already available. We suggest it is wasteful not to use it.

The greater hold-out likelihood of our RNN is consistent with it doing better than the other models across a number of assessments in Section 4.

The composite nature of likelihood also makes it a useful diagnostic tool. Just like KL-divergence, the further away a model is — in any manner — from the data in the hold-out set, the worse the likelihood will be. Therefore, if a model has a poor likelihood despite performing well on a range of standard metrics, this suggests there are still notable deficiencies in the model which need investigating.

Being a composite metric brings challenges though. Poor performance in certain aspects may be overshadowed by good performance elsewhere. This can result in cases where increased hold-out likelihoods do not always correspond to better sample quality, as noted in the ML literature (Goodfellow et al., 2014; Grover et al., 2018; Theis et al., 2015; Zhao et al., 2020). In our work, although the RNN has the greatest likelihood for $F = 23$, it does not have the smallest KL-divergence for the 1D histograms in Figure 7. We also found that when certain RNN models were trained far longer, even though their hold-out log-likelihoods improved, sometimes the simulations became unstable.

Poor sample quality with high likelihood is also a useful diagnostic which can lead to improved models. If we have certain metrics which we believe are important to sample quality then we can incorporate them into our probability model. This automatically leads to better likelihood scores (as we will remove probability mass from ‘unrealistic’ regions in sample space) and increased realism. Here is a simple illustrative example: suppose a model for precipitation was generating sequences of data, some of which were negative. We know that precipitation cannot be negative — that would be unrealistic. By altering our probability model to account for precipitation only taking positive values our simulation data would be more realistic (no negative values would be simulated anymore) and our model would have a greater log-likelihood on hold-out data (by removing its ability to generate negative precipitation values, it will put more probability mass around positive values — which the hold-out set is composed of). If we could set up the problem of unstable simulations in a similar manner, perhaps it too could be rectified.
Table 2. Log-likelihood on hold-out data. The RNN has the best likelihood in both cases.

| Model     | F = 20  | F = 23  |
|-----------|---------|---------|
| Polynomial| 4.98    | 4.73    |
| RNN       | **6.53**| **5.89**|
| GAN       | $\approx −37.69 \pm 0.01$ | $\approx −44.86 \pm 0.00$ |

5.3 Likelihood Calculations

For the polynomial model, the log-likelihood of the $X_t$ sequence is

$$
\log \Pr(x_1, \ldots, x_n; x_0) = \sum_{k=1}^{K} \left( \log \Pr(h_{k,1}) + \sum_{t=1}^{n-1} \log \Pr(h_{k,t+1}|h_{k,t}) \right) - Kn \log(\Delta t) \tag{14}
$$

with a derivation in Appendix A.

We present an approach to approximate the GAN likelihood. This is despite its full form being intractable (involving integrals which cannot be efficiently approximated using Monte Carlo sampling) and so it not being typically used to evaluate GANs. We calculate the likelihood of a model which functions and gives results almost identical to the GAN (one with a small amount of white noise added) using importance sampling and the reparameterization method as in the Variational Autoencoder (Kingma & Welling, 2013). Here, (9) is altered to

$$
X_{k,t+1} = X_{k,t} + \omega_k(X_t) - \Delta t \ u(X_{k,t}, h_{k,t+1}) \tag{15}
$$

where $u(X_{k,t}, h_{k,t+1}) = U(X_{k,t}, h_{k,t+1}) + N_{k,t+1}$ and $N_{k,t} \sim N(0,0.001^2)$. The log-likelihood is

$$
\log \Pr(x_1, \ldots, x_n; x_0) = \log \mathbb{E}_{h \sim \Pr_H} \frac{\Pr(u|h, x_0)\Pr_H(h)}{\Pr_H(h)} - Kn \log(\Delta t)
$$

where the expectation can now be approximated by a large sum if there is a suitable importance sampler $\Pr_H(h)$. A derivation and further decomposition is in Appendix A along with details on how the importance sampler is constructed and trained. 50 samples were used for the importance sampling, and this was repeated 50 times to give 95% confidence intervals. We note that given the finite number of samples taken it is of course possible that the GAN likelihood is larger, but a well-trained importance sampler should minimise the chance of this.
6 Discussion and Conclusion

In this work, even though we used ML to model $g_\theta(X_{k,t})$ (often called the ‘sub-grid tendency’) in (11), the real benefit comes from using ML to model the hidden state evolution. For example, on setting $g_\theta(X_{k,t})$ in our model equal to $aX_{k,t}^3 + bX_{k,t}^2 + cX_{k,t} + d$ from the polynomial baseline, our model performance hardly suffered (and the cost halved).

Using physical knowledge to structure ML models can help with learning. We leveraged many elements from the polynomial baseline (such as the form of the Runge-Kutta updating and the physical features) and this gave better results than if we had instead modelled the system like

\begin{align}
X_{k,t+1} &= X_{k,t} + f_\theta(X_t, h_{k,t+1}) \\
\beta_\theta(h_{k,t}, z_{k,t+1}) &= h_{k,t+1} \\
\mathcal{N}(0, 1) &= z_{k,t+1}
\end{align}

at a cost similar to our presented RNN, where $f_\theta$ and $\beta_\theta$ are NNs which we hope would learn important features such as advection. Although in theory a NN can learn to create whatever features are useful, giving useful features can make learning easier especially when architectures are limited due to cost constraints. The NN does not need to learn the known, useful physical relationships and instead can focus on learning other helpful ones.

There are evidently more sophisticated ways to model the system. Instead of simply adding on the hidden state as in (11) it could be the input of a complex function as in (9) of the GAN. We noticed that in some cases the RNN struggled to overfit the data, regardless of the complexity of the network. This could be due to difficulties in learning the evolution of the hidden states. Further sophistication in the model is required as is to be expected when modelling a chaotic system like the L96. Forms like (16)–(17) are more flexible and there may be a cost threshold beyond which they are better than their physically-informed counterparts.

Creating architectures that permit better hidden states to be learnt (ones which model long-term correlations better) would give better models. Despite our use of GRU cells, which along with the LSTM were great achievements in sequence modelling, we still faced issues with the modelling of long-term trends. Certain models (mainly those using the flexible form (16)–(17)) resulted in unstable simulations. Using hierarchical models which learn to model trends at different temporal resolutions (Chung et al., 2016; Liu et al., 2020) may provide improvement.

Learning from all the high-resolution data whilst still being computationally efficient at simulation time is an interesting idea. Whilst we cannot keep all the high-resolution dimensions in our schemes (as otherwise we might as well just run the high-resolution model outright — which is not possible as it is too slow for the desired timescales and is a reason why this field of work exists), only keeping the average (as in existing work which coarse-grains data) means much data is thrown away. Our preliminary investigations used the graphical model in Figure 8. We designed this so that in training, $I_t$ is learnt such that it contains useful information about the high-resolution data, $Y_t$, whereas during generation $Y_t$ is not required to be simulated. For the L96, this showed no clear improvement over our RNN though.

Further work with GANs could result in better models. There may be aspects of realism which we either may not be aware of or may not be able to quantify, so are more difficult to include explicitly in our probability models. The GAN discriminator could learn what features constitute ‘realism’ and so the generator may learn to create sequences which contain these. However, preliminary work using GANs and Wasserstein-GANs (Arjovsky et al., 2017; Gulrajani et al., 2017) to train our RNN (instead of by likelihood) did not show benefit. This may be due to the difficulty of training GANs on longer sequences.
Figure 8. Model which allows learning from high-resolution data without requiring it at simulation time, where \( N_t \sim N(0, I) \).

They have been used to train RNNs on medical timeseries (Esteban et al., 2017) and to model music (Mogren, 2016) so applying this approach to climatic sequences may be something for future work to reconcile.

The challenges of using GANs are seen by how we have not been able to reproduce the results shown in Gagne et al. (2020) despite the same set-up. This points to the instability of GAN training as noted by them too. Mode collapse is a common issue encountered when training GANs, where the generator fails to produce samples that explore certain modes of the distribution, and could explain why Regime B in Figure 4 has more density than desired.

6.1 Conclusion

We have shown that a probabilistic framework permits ML tools to be easily deployed for the parameterization task. Our model is competitive with a bespoke baseline, likely due to the marrying of a stochastic framework with the RNN’s superior ability to model temporal correlations. The approach now needs testing on real climate data — both to specifically improve on AR1 hidden state processes, and to learn new, more flexible overall models of \( X_t \). The field is ripe for other probabilistic ML tools to be used and we suspect that further customisation of these will lead to many improved parameterization models.

We believe that likelihood would be a universal metric for evaluating all probabilistic climate models in the future. It can often be fairly easily calculated, and we propose that where this be the case the climate community also evaluate the likelihood on holdout sets for any devised probabilistic model (ML or otherwise). Doing this would provide a consistent evaluation metric across the climate modelling literature once the issues with likelihood have been further understood.

Finally, we have shown that ML models can generalise to unseen scenarios. The hesitancy in the climate science community about ML models’ abilities to generalise is understandable given that in much ML work testing regimes are created by shuffling datasets, testing interpolation (a far easier test to pass) not extrapolation. Here, by using a challenging test we have shown our model does generalise to an unseen regime (\( F = 23 \)).
By borrowing from the scientific community’s approach to making and using such tests, we subject our models to the same scrutiny that has been used to assess numerous classical scientific models in the past.
Appendix A  Likelihood Derivations

In all cases, the log-likelihood of the sequence of $X_t$ is given by

$$\log \Pr(x_1, \ldots, x_n; x_0) = \log \Pr(x_1; x_0) + \log \Pr(x_2|x_1; x_0) + \log \Pr(x_3|x_2, x_1; x_0) + \ldots + \log \Pr(x_n|x_{n-1}, \ldots, x_1; x_0)$$  \hspace{1cm} (A1)

which follows from the laws of probability.

A1  RNN

Now from (11), we can write

$$(X_t|X_{t-1}=x_{t-1}, \ldots, X_1=x_1; X_0=x_0) = f(x_{t-1}) - \Delta t(r_t|X_{t-1}=x_{t-1}, \ldots, X_1=x_1; X_0=x_0)$$  \hspace{1cm} (A2)

and given this relationship between $X_t$ and $r_t$, a change of variables for the likelihood gives

$$\log \Pr(x_t|x_{t-1}, \ldots, x_1; x_0) = \log \Pr(r_t|r_{t-1}, \ldots, r_1) - K \log(\Delta t)$$  \hspace{1cm} (A3)

(A1) is therefore

$$= \log \Pr(r_1) + \log \Pr(r_2|r_1) + \log \Pr(r_3|r_2, r_1) + \ldots + \log \Pr(r_n|r_{n-1};) - K n \log(\Delta t)$$  \hspace{1cm} (A4)

$$= \log \Pr(r_1) + \sum_{i=2}^{n} \log \Pr(r_i|l_i) - K n \log(\Delta t)$$  \hspace{1cm} (A5)

$$= \sum_{k=1}^{K} \left( \log \Pr(r_{k,1}) + \sum_{i=2}^{n} \log \Pr(r_{k,i}|l_{k,i}) \right) - K n \log(\Delta t)$$  \hspace{1cm} (A6)

where (A5)–(A6) follow from the independencies in the graphical model (Figure 1).

A2  Polynomial Model

For the polynomial model, the graphical model is shown in Figure A1a. From (6), we can write

$$(X_t|X_{t-1}=x_{t-1}, \ldots, X_1=x_1; X_0=x_0) = f(x_{t-1}) - \Delta t(h_t|X_{t-1}=x_{t-1}, \ldots, X_1=x_1; X_0=x_0)$$  \hspace{1cm} (A7)

and given this relationship between $X_t$ and $h_t$, a change of variables for the likelihood gives

$$\log \Pr(x_t|x_{t-1}, \ldots, x_1; x_0) = \log \Pr(h_t|h_{t-1}, \ldots, h_1) - K \log(\Delta t)$$  \hspace{1cm} (A8)

(A1) is therefore

$$= \log \Pr(h_1) + \log \Pr(h_2|h_1) + \log \Pr(h_3|h_2, h_1) + \ldots + \log \Pr(h_n|h_{n-1};) - K n \log(\Delta t)$$  \hspace{1cm} (A9)

$$= \log \Pr(h_1) + \sum_{i=1}^{n-1} \log \Pr(h_{i+1}|h_i) - K n \log(\Delta t)$$  \hspace{1cm} (A10)

$$= \sum_{k=1}^{K} \left( \log \Pr(h_{k,1}) + \sum_{i=1}^{n-1} \log \Pr(h_{k,i+1}|h_{k,i}) \right) - K n \log(\Delta t)$$  \hspace{1cm} (A11)

where (A10)–(A11) follow from the independencies described in (5).
Figure A1. Graphical model for (a) Polynomial and (b) GAN with added white noise.
A3 GAN

The graphical model of the GAN with a small amount of white noise added is shown in Figure A1b. $X_t$ is related to $u_t$ as in (15) so

$$(X_t | X_{t-1} = x_{t-1}, ..., X_1 = x_1; X_0 = x_0) = f(x_{t-1}) - \Delta t(u_t | X_{t-1} = x_{t-1}, ..., X_1 = x_1; X_0 = x_0)$$

(A12)

and a change of variables for likelihood gives

$$\log \Pr(x_t | x_{t-1}, ..., x_1; x_0) = \log \Pr(u_t | u_{t-1}, ..., u_1; x_0) - K \log(\Delta t)$$

(A13)

(A1) is therefore

$$= \log \Pr(u_1 | x_0) + \log \Pr(u_2 | u_1, x_0) + \log \Pr(u_3 | u_2, u_1, x_0) + ... + \log \Pr(u_n | u_{n-1:0}, x_0) - Kn \log(\Delta t)$$

$$= \log \Pr(u | x_0) - Kn \log(\Delta t)$$

and just decomposing the first term below:

$$\log \Pr(u | x_0) = \log \mathbb{E}_{h \sim \Pr_H} \Pr(u | h, x_0)$$

$$= \log \mathbb{E}_{h \sim \Pr_H} \frac{\Pr(u | h, x_0) \Pr_H(h)}{\Pr_H(h)}$$

(A14)

where (A14) can be approximated with a sufficiently large sum given a good enough encoder $\Pr_H(h)$.

For training purposes, the lower-bound is used:

$$\geq \mathbb{E}_{h \sim \Pr_H} \log \frac{\Pr(u | h, x_0) \Pr_H(h)}{\Pr_H(h)}$$

(A15)

The terms in the numerator decompose as follows, using the independencies from the graphical model and associated equations:

$$\log \Pr(u | h, x_0) = \sum_{t=1}^{n} \log \Pr(u_t | h_t, x_{t-1})$$

$$= \sum_{t=1}^{n} \sum_{k=1}^{K} \log \Pr(u_{k,t} | h_{k,t}, x_{k,t-1})$$

and

$$\log \Pr_H(h) = \log \Pr_H(h_1) + \sum_{t=2}^{n} \log \Pr_H(h_t | h_{t-1})$$

$$= \sum_{k=1}^{K} \left( \log \Pr_H(h_{k,1}) + \sum_{t=2}^{n} \log \Pr_H(h_{k,t} | h_{k,t-1}) \right)$$

The perfect importance sampling distribution would be proportional to $\Pr(h | u, x_0)$. Therefore we want $\Pr_H(h)$ to allow for the same dependencies. We choose it by carrying out the following steps:

$$\log \Pr_H(h) \propto \log \Pr(h | u, x_0)$$

$$= \log \Pr_H(h_1 | u, x_0) + \sum_{t=2}^{n} \log \Pr_H(h_t | h_{t-1}, u, x_0)$$

(A16)
where (A16) follows from using the laws of probability to decompose the term above, followed by applications of the independencies from the graphical model. Note that although the process is Markov it is not necessarily time-homogeneous. To deal with this, an RNN is used to model \( \Pr_H(h_t|h_{t-1}, u, x_0) \) as \( \Pr_H(h_t|h_{t-1}, u, x_0, s_t) \) where the state \( s_t \) can in principle account for the non-homogeneous updating procedure of \( h_t \). The log-likelihood is therefore

\[
\sum_{k=1}^{K} \left( \log \Pr_H(h_{k,1}|u_k, x_{k,0}) + \sum_{t=2}^{n} \log \Pr_H(h_{k,t}|h_{k,t-1}, u_k, x_{k,t}, s_{k,t}) \right) \tag{A17}
\]

where the spatial independencies introduced in (A17) follow from the graphical model and \( u_{k,.} = u_{k,1:n} \) and \( x_{k,.} = x_{k,0:n-1} \). The only simplifying assumption used here is that the same update model for \( h_{k,t} \) is used for each component \( k \). Finally, the sequence of \( (u_k, x_k) \) is summarised using a bidirectional RNN to give \( w_{k,t} \). Therefore, the final form of the log-likelihood of our importance sampler is

\[
\sum_{k=1}^{K} \left( \log \Pr_H(h_{k,1}|w_k, x_{k,0}) + \sum_{t=2}^{n} \log \Pr_H(h_{k,t}|h_{k,t-1}, w_k, x_{k,t-1}, s_{k,t}) \right) \tag{A18}
\]

The training of the importance sampler is done by maximising (A15), with the learnable parameters being those of the model used to learn \( h_{k,1} \), the RNN used to model the update of \( h_{k,t} \), and the bidirectional RNN.
Acknowledgments
The source code can be accessed at https://github.com/raghul-parthipan/l96_rnn.
R.P. was funded by UK Research and Innovation. H.M.C. was funded by Natural Environment Research Council grant number NE/P018238/1. J.S.H. was supported by the British Antarctic Survey, Natural Environment Research Council (NERC) National Capability funding. D.J.W. was funded by the University of Cambridge.

References
Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In International conference on machine learning (pp. 214–223).
Arnold, H. M., Moroz, I. M., & Palmer, T. N. (2013). Stochastic parametrizations and model uncertainty in the lorenz'96 system. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 371(1991), 20110479.
Berner, J., Jung, T., & Palmer, T. (2012). Systematic model error: The impact of increased horizontal resolution versus improved stochastic and deterministic parameterizations. Journal of Climate, 25(14), 4946–4962.
Brenowitz, N. D., & Bretherton, C. S. (2018). Prognostic validation of a neural network unified physics parameterization. Geophysical Research Letters, 45(12), 6289–6298.
Brenowitz, N. D., & Bretherton, C. S. (2019). Spatially extended tests of a neural network parametrization trained by coarse-graining. Journal of Advances in Modeling Earth Systems, 11(8), 2728–2744.
Buizza, R., Milleer, M., & Palmer, T. N. (1999). Stochastic representation of model uncertainties in the ecmwf ensemble prediction system. Quarterly Journal of the Royal Meteorological Society, 125(560), 2887–2908.
Chattopadhyay, A., Hassanzadeh, P., & Subramanian, D. (2020). Data-driven predictions of a multiscale lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network. Nonlinear Processes in Geophysics, 27(3), 373–389.
Chattopadhyay, A., Subel, A., & Hassanzadeh, P. (2020). Data-driven super-parameterization using deep learning: Experimentation with multiscale lorenz 96 systems and transfer learning. Journal of Advances in Modeling Earth Systems, 12(11), e2020MS002084.
Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.
Christensen, H. M. (2020). Constraining stochastic parametrisation schemes using high-resolution simulations. Quarterly Journal of the Royal Meteorological Society, 146(727), 938–962.
Christensen, H. M., Berner, J., Coleman, D. R., & Palmer, T. (2017). Stochastic parameterization and el niño–southern oscillation. Journal of Climate, 30(1), 17–38.
Christensen, H. M., Moroz, I. M., & Palmer, T. N. (2015). Simulating weather regimes: Impact of stochastic and perturbed parameter schemes in a simple atmospheric model. Climate Dynamics, 44(7), 2195–2214.
Chung, J., Ahn, S., & Bengio, Y. (2016). Hierarchical multiscale recurrent neural networks. arXiv preprint arXiv:1609.01704.
Crommelin, D., & Vanden-Eijnden, E. (2008). Subgrid-scale parameterization with conditional markov chains. Journal of the Atmospheric Sciences, 65(8), 2661–2675.
Eck, D., & Schmidhuber, J. (2002). A first look at music composition using lstm recurrent neural networks. Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale, 103, 48.
Esteban, C., Hyland, S. L., & Rätsch, G. (2017). Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*.

Gagne, D. J., Christensen, H. M., Subramanian, A. C., & Monahan, A. H. (2020). Machine learning for stochastic parameterization: Generative adversarial networks in the Lorenz’96 model. *Journal of Advances in Modeling Earth Systems, 12*(3), e2019MS001896.

Gentine, P., Pritchard, M., Rasp, S., Reinaudi, G., & Yacalis, G. (2018). Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters, 45*(11), 5742–5751.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Grover, A., Dhar, M., & Ermon, S. (2018). Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 32).

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735–1780.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Krasnopolsky, V. M., Fox-Rabinovitz, M. S., & Belochitski, A. A. (2013). 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*.

Kwasniok, F. (2012). Data-based stochastic subgrid-scale parametrization: an approach using cluster-weighted modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 370*(1962), 1061–1086.

Leutbecher, M., Lock, S.-J., Ollinaho, P., Lang, S. T., Balsamo, G., Bechtold, P., … others (2017). Stochastic representations of model uncertainties at ecmwf: State of the art and future vision. *Quarterly Journal of the Royal Meteorological Society, 143*(707), 2315–2339.

Leutbecher, M., & Palmer, T. N. (2008). Ensemble forecasting. *Journal of computational physics, 227*(7), 3515–3539.

Liu, Y., Kutz, J. N., & Brunton, S. L. (2020). Hierarchical deep learning of multiscale differential equation time-steppers. *arXiv preprint arXiv:2008.09768*.

Lorenz, E. N. (1996). Predictability: A problem partly solved. In *Proc. seminar on predictability* (Vol. 1).

Mogren, O. (2016). C-run-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*.

O’Gorman, P. A., & Dwyer, J. G. (2018). 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*(10), 2548–2563.

Palmer, T. N. (2012). Towards the probabilistic earth-system simulator: A vision for the future of climate and weather prediction. *Quarterly Journal of the Royal Meteorological Society, 138*(665), 841–861.

Palmer, T. N., Buizza, R., Doblas-Reyes, F., Jung, T., Leutbecher, M., Shutts, G. J., … Weisheimer, A. (2009). *Stochastic parametrization and model uncertainty*. ECMWF Reading, UK.
Rasp, S. (2020). Coupled online learning as a way to tackle instabilities and biases in neural network parameterizations: general algorithms and Lorenz 96 case study (v1.0). *Geoscientific Model Development*, *13*(5), 2185–2196.

Rasp, S., Pritchard, M. S., & Gentine, P. (2018). Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, *115*(39), 9684–9689.

Sanchez, C., Williams, K. D., & Collins, M. (2016). Improved stochastic physics schemes for global weather and climate models. *Quarterly Journal of the Royal Meteorological Society*, *142*(694), 147–159.

Scher, S., & Messeri, G. (2019). Generalization properties of feed-forward neural networks trained on Lorenz systems. *Nonlinear processes in geophysics*, *26*(4), 381–399.

Schneider, T., Teixeira, J., Bretherton, C. S., Brient, F., Pressel, K. G., Schä, C., & Siebesma, A. P. (2017). Climate goals and computing the future of clouds. *Nature Climate Change*, *7*(1), 3–5.

Stockdale, T. N., Anderson, D. L., Balmaseda, M. A., Doblas-Reyes, F., Ferranti, L., Mogensen, K., . . . Vitart, F. (2011). Ecmwf seasonal forecast system 3 and its prediction of sea surface temperature. *Climate Dynamics*, *37*(3), 455–471.

Sutskever, I., Martens, J., & Hinton, G. E. (2011). Generating text with recurrent neural networks. In *ICML*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

Theis, L., Oord, A. v. d., & Bethge, M. (2015). A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Vlachas, P. R., Byeon, W., Wan, Z. Y., Sapsis, T. P., & Koumoutsakos, P. (2018). Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *474*(2213), 20170844.

Yuval, J., O’Gorman, P. A., & Hill, C. N. (2021). Use of neural networks for stable, accurate and physically consistent parameterization of subgrid atmospheric processes with good performance at reduced precision. *Geophysical Research Letters*, *48*(6), e2020GL091363.

Yuval, J., & O’Gorman, P. A. (2020). Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions. *Nature communications*, *11*(1), 1–10.

Zelinka, M. D., Myers, T. A., McCoy, D. T., Po-Chedley, S., Caldwell, P. M., Ceppi, P., . . . Taylor, K. E. (2020). Causes of higher climate sensitivity in CMIP6 models. *Geophysical Research Letters*, *47*(1), e2019GL085782.

Zhao, M., Cong, Y., Dai, S., & Carin, L. (2020). Bridging maximum likelihood and adversarial learning via α-divergence. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, pp. 6901–6908).