Predicting AMOC Collapse in Low-Order Models with CNN-LSTM Networks

by

Stefan Vladusic

A Major Research Project
presented to the University of Waterloo
in fulfillment of the
major research paper requirement for the degree of
Masters of Mathematics
in
Computational Mathematics

Waterloo, Ontario, Canada, 2022

Author's Declaration

I hereby declare that I am the sole author of this work. This is a true copy of the work, including any required final revisions, as accepted by my examiners.

I understand that my report may be made electronically available to the public.

Abstract

The potential shutdown of Atlantic Meridional Overturning Circulation (AMOC) presents a highly significant, if not devastating, future scenario for the Earth's climate. Drawing on dynamical systems theory and machine learning, we train several neural networks to predict whether a simulated time series of a low-dimensional oceanic model precedes AMOC collapse. More specifically, we create a time series generator for the four box model of the Atlantic ocean to generate a data set of 500,000 multivariate time series. Half of the time series correspond to simulated scenarios resulting in AMOC collapse, while the other half correspond to scenarios with no such collapse. We train several hybrid Convolutional Neural Network - Long Short Term Memory (CNN-LSTM) networks on 95% of the generated data. We then compare our networks' performance to a CNN-LSTM network developed by Bury et al. in [5]. Bury et al.'s network is a classifier which predicts whether or not an input time series, representing some dynamical system, will soon transition to a new dynamical regime by way of a local bifurcation. Trained on a data set of time series given by a thousands of two-dimensional dynamical systems, Bury et al.'s classifier associates each input time series with one of four possible labels. Three of these labels infer that a specific kind of local bifurcation will induce a sudden transition, while the fourth predicts that no such transition will occur. Since one of the labels of Bury et al.'s network describes the kind of bifurcation exhibited in the four-box model — namely, a saddle-node bifurcation — this network should, a priori, accurately label four-box model data.

We find that our networks significantly outperforms one of Bury et al.'s classifiers on four-box model test data. In particular, our best CNN-LSTM network achieves an accuracy of 98.86% accuracy on test set of 5,000 multivariate four-box model time series. Meanwhile, Bury et al.'s model performs worse than a random classifier on this set with an accuray of 16.19%, though retraining the classification layer of this model model increases its accuracy to 85.90%. In order to explain the poor performance of Bury et al.'s model, we argue that the identically zero eigenvalue of the four-box model's Jacobian results in time series which Bury et al.'s model falsely associates with a transcritical bifurcation. However, other simple models of oceanic circulation, including Stommel's original two-box model, do not exhibit this identically zero eigenvalue. As a result, possible extensions of our work involve training networks on time series generated by many low-order models, and determining if Bury et al.'s network still performs poorly on data generated by these myriad models.

Acknowledgements

I would first like to thank my supervisors Chris Bauch and Chris Fletcher, whose knowledge, insight, and support were invaluable when writing this report. I also thank Tom Bury for enthusiastically answering all of my questions about his own research, and Ben Fattori, who was always willing to help me navigate the vast world of machine learning.

Completing this report would have been nearly impossible without the lovely community I found in Waterloo. As a result, I owe a debt of gratitude to all of my friends I made during my master's. This includes, but is not limited to, the other students in the CM program, members of the Bauch and Fletcher labs, all my teammates on Computer Blue, and the regulars of the MGSA pub night and Stammtisch. I also want to give special thanks to Kevin for those countless drives back home, all of those Raptors games we watched together, and our friendship over the past decade.

Finally, I must thank my family. Words simply fail to describe how much their love, support, patience, and kindness has meant to me over the years. So, I won't even try.

Table of Contents

List of Figures

List of Tables

Chapter 1

Background & Motivation

Tipping points — roughly characterized as an abrupt change in the relevant regime of a dynamical system — present a significant threat of irreversible, long-term changes to the climate system [1, 3, 13, 14]. When tipping points of the climate system were first considered in the International Panel on Climate Change's First Assessment Report, researchers concluded that a critical transition in the climate system was likely only if global warming reached 5°C above pre-industrial temperatures [14]. However, there is mounting evidence that tipping points for a wide array of climate phenomena ranging from rain forest precipitation levels, forest fire areas and ocean circulation speeds, may be surpassed even if global warming is kept below 2°C above pre-industrial temperatures [1, 14].

A notable sub-component of the earth system which could theoretically undergo a critical transition is the rate of Atlantic Meridional Overturning Circulation (AMOC) [2, 3, 6, 8, 14, 16, 26, 30]. The main circulation system of the Atlantic ocean, AMOC is, at present, largely driven by density gradients induced by thermal differences in the ocean [24]. However, paleoclimate data, and models of intermediate complexity provide considerable evidence that AMOC may suddenly transition into a considerably weaker, salinity driven circulation under sufficiently large freshwater forcings in the Atlantic [2, 3, 8, 23, 26, 30, 34].

Given the significant, potentially disastrous (see [14]), consequences of AMOC collapse, many authors have leveraged techniques from dynamical systems theory to try and predict the likelihood of such a collapse for given freshwater forcing experiments [3, 13]. In particular, many models and real life systems which pass through tipping points exhibit so-called *early warning signals* (EWS) before their abrupt shift in dynamics. Since EWS often appear as increases in rolling variances and autocorrelation in system time series, researchers have created traditional, statistical classifiers which predict if a system is approaching a

tipping point [4, 27, 28].

More recently, Bury et al. trained a state-of-the-art neural network on a wide variety of noisy data generated by low-dimensional, simple dynamical systems [5]. Although the data set used to train the network is simple, this network outperforms traditional classifiers on a set of empirical and model time series, ranging from epidemiological models to thermoacoustic data. Furthermore, the success of this network given its simple training set can be explained in part by theoretical results in dynamical systems theory [5]. These results suggest that neural networks may be leveraged to predict AMOC collapse in simulated time series with a high degree of accuracy, and possibly predict the likelihood of AMOC collapse at present day.

In this report, we present a method of generating a data set which is used to train neural networks that can detect AMOC collapse in low-dimensional models of ocean circulation. In particular, we create a data generator which returns stochastically forced time series of Zickfeld et. al.'s four-box model of Atlantic circulation (see [34]), and labels the time series depending on whether or not it approaches a tipping point. Given this data set, we train several neural networks to predict AMOC tipping points. The networks are distinguished by which dynamic variables of the model are taken as inputs, as we also investigate whether a so-called proxy variable can detect tipping points associated with one variable through time series of other variables. We compare the results of our own network to one of the networks trained by Bury et al. in [5]. By comparing network accuracy, precision and recall, we demonstrate that all of our networks outperform Bury et al.'s network when classifying four-box model data. However, we also caution that these results may not generalize to other simulated AMOC data given the simplicity and mathematical idiosyncrasies of the four-box model.

We structure our report as follows. Chapter 2 covers important background material for our analysis. Section 1.1 is a summary of AMOC and box models. Section 1.2 covers dynamical systems theory, focusing on local bifurcations, early warning signals, and the theoretical aspects which motivate Bury et al.'s own methodology. We then motivate our methodology and research goals in section 1.3. Chapter 2 is dedicated to the methods used to train our networks. In particular, section 2.1 discusses neural networks: how they are defined, trained, and the best architectures for classification tasks. Section 2.2 is then a summary of the four-box model, including our data set generating procedure. Our results are found in chapter 3. We first describe the specific features of our neural networks in section 3.1. We then compare the performance of our classifiers to one of Bury et al.'s classifiers, and tentatively explain why our models outperform Bury et al.'s models in section 3.2. Section 3.3. compares the methodology of our report to [5]. Finally, we describe some limitations and possible extensions of our work in section 3.4.

1.1 AMOC & Box Models

As the chief objective of this project is to train a state-of-the-art deep network to predict tipping points in low-order models of oceanic circulation, we briefly describe the phenomenon of *Atlantic Meridional Overturning Circulation*, or *AMOC* (see [25, 34]). AMOC describes the large-scale circulation in the Atlantic, with relatively warm surface waters rushing from the south towards the north, and relatively cold deep water flowing from the north Atlantic into the southern hemisphere[1]. While there are many contributing factors for this circulation pattern, AMOC is innately linked to so-called *Thermohaline Circulation* (henceforth THC). THC describes the circulation of the Atlantic given unequal changes in temperature and salinity of oceanic waters. In particular, unequal fluxes of heat and freshwater lead to thermal and salinity differences between different regions of the ocean. These differences then result in density gradients, which in turn lead to vigorous circulation. It is important to clarify that THC and AMOC are conceptually distinct. AMOC describes the general meridional circulation of the Atlantic, which cannot be explained by THC alone, and THC is not constrained to the meridional direction.

Although AMOC and THC are not to be conflated, the latter in large part determines the dynamics of the former [24]. So, the evidence for a bistable THC given by models of intermediate complexity [23, 24, 26], empirical data (see [2, 14], and some generalized circulation models [3, 16] suggests that that AMOC is also bistable (with the important caveat that state-of-the-art Earth System Models tend to exhibit monostable THC; see [16]). It has long been theorized that increasingly large salinity gradients between northern and tropical waters given large enough freshwater forcings in the north Atlantic, could cause the presently thermally-driven THC to "collapse" towards a weakened, reversed, salinity-driven circulation. Such investigations began with Stommel's seminal 1961 model of Atlantic circulation [24]. Described in [31], the model treats the ocean as two, connected boxes with well-defined temperatures and salinities, and specifies the rate of circulation between these two boxes as a linear function of their temperature and salinity differences. Given this model, he then demonstrates that increasing the model's freshwater forcing parameter values causes the circulation between boxes to suddenly weaken and reverse. In this way, the model provides a simple demonstration of AMOC collapse due to changes in THC.

Though Stommel's original "box model" is very simple, this simplicity is a virtue for our task. First, the model, and similar box models inspired by Stommel's work (see, for instance, [17, 34]), are much simpler and quicker to simulate when compared to more com-

[1] This paragraph is largely based on Rahmstorf 's summary of AMOC found in [24].

plicated models of AMOC. More importantly for our purposes, these box models ultimately exhibit AMOC collapse due to the presence of a so-called *local bifurcations* under variations of freshwater forcing parameters. This insight is crucial, since there already exists a large literature on training traditional and deep-learning classifiers that successfully predict sudden dynamical changes for many systems that exhibit local bifurcations (see [27, 28]). As a result, there is strong reason a priori to believe that a state-of-the-art neural network could successfully predict AMOC collapse in box models given a sufficiently large training set of simulated AMOC data. And because box models are simple to simulate, it follows that generating such a training set via box models is computationally feasible. Indeed, we describe both what a local bifurcation is, and how their corresponding properties may be leveraged to predict sudden changes in relevant dynamical systems.

1.2 Dynamical Systems & Bifurcations

1.2.1 Equilibria, Bifurcations, & EWS

The dynamics of many natural and social systems — ranging from thermal processes, to infectious diseases and disinformation — are well described by systems of ordinary differential equations (henceforth *ODE systems*) [5, 11, 27, 32]. Following much of the literature, we will also refer to ODE systems as *dynamical systems*, or just *systems*. For our purposes, dynamical systems specify how the derivative of function in $\mathbb{R}^n$ — representing the *state* of a relevant system — is related to its current state. In particular, the precise relationship between these two variables is specified by some *evolution operator* f, which in turn depends on the current system state $x(t) \in U \subseteq \mathbb{R}^n$, the time $t \in T \subseteq \mathbb{R}$, and a vector of *parameters* $\mu \in \mathbb{R}^m$:

$$\dot{x} = f(t, x; \mu)$$

Oftentimes, the evolution operator does not explicitly depend on t. In this case, the dynamical system is said to be *autonomous* [11, 32], and is written as $\dot{x} = f(x; \mu)$ or $\dot{x} = f_\mu(x)$. Furthermore, the parameters μ are often fixed. When this is the case, we will simply write $\dot{x} = f(x)$.

Given a dynamical system of the form $\dot{x} = f(x; \mu)$, we are often interested in determining the behaviour of *initial value problems* (IVPs). That is, we want to determine the behaviour of solutions $x(t, x_0)$ satisfying $\dot{x}(t) = f(x(t)),\ x(t_0) = x_0$, should they exist. Although it is guaranteed that a unique solution exists over an open subset $U \subseteq \mathbb{R}^n$ whenever f has continuous partial derivatives over an open set $U \subseteq \mathbb{R}^n$ — denoted $f \in C^1(U)$

— this solution, actually solving for $x(t)$ is generally impossible [22]. Thus, we turn to examining the generic behaviour of many systems, rather than the specific behaviour of a single system.

We begin by considering an IVP of a arbitrary dynamical system where $f \in C^1(U)$

$$\dot{x} = f(x),\ x(0) = x_0$$

As noted above, it is generally difficult to account for the dynamics of this system. However, one obvious exception to this generalization occurs when $x_0 = x^*$ such that $f(x^*) = 0$. Indeed, in this case $\dot{x} = 0 \Rightarrow x(t) = x_0$. Since these solutions to the IVP are constant in time, we say that x^* are *fixed points* or *equilibria.*

One important feature of all dynamical systems with $f_\mu \in C^1(U)$ is that we can often infer the asymptotic behaviour of solutions whose initial conditions lie x_0 in a delta neighborhood $N_\delta(x^*) := \{x \in U \mid \|x - x^*\| < \delta\}$. In particular, it turns out that the so-called *stability* of a fixed point is intimately related to it's linearization $\dot{x} = Df(x^*)[x - x^*]$ about x^*, where Df is the Jacobian of f.

Definition: Let $U \subseteq \mathbb{R}^n$ be an open set, $f \in C^1(U)$, and $\dot{x} = f(x)$ represent a dynamical system defined for all times $t \in \mathbb{R}$. A fixed point x^* is *stable* iff for all $\varepsilon > 0$, there exists $\delta > 0$, such that the solution to the IVP $\dot{x} = f(x)$, $x(0) = x_0$ satisfies $\|x(t, x_0) - x^*\| < \varepsilon$ for all $t \in \mathbb{R}$, whenever $x_0 \in N_\delta(x^*) := \{x \in U \mid \|x - x^*\| < \delta\}$. If in addition $\lim_{t\to\infty} \|x(t, x_0) - x^*\| = 0$ for all $x_0 \in N_\delta(x^*)$, then x^* is *asymptotically stable.* Finally, if x^* is not stable, it is *unstable* [22].

Furthermore, define the *dominant eigenvalue* $\lambda_D(x^*)$ of a fixed point x^* as the eigenvalue of $Df(x^*)$ with the largest real component. It can be shown that $\mathrm{Re}(\lambda_D(x^*)) \leq 0$ only if x^* is stable, and $\mathrm{Re}(\lambda_D(x^*)) > 0$ if x^* is unstable. Finally, $\mathrm{Re}(\lambda_D(x^*)) < 0$ entails that x^* is asymptotically stable [22]. In figure 1.1, we provide some examples of systems with asymptotically stable, stable and unstable fixed points at the origin.

Having defined stability, we now turn to the task of defining a bifurcation. To this end, consider an arbitrary, parameterized dynamical system:

$$\dot{x} = f(x; \mu)$$

Intuitively, a parameterized system exhibits a bifurcation when a small change in parameter values leads to a drastic change in solution behaviour. For instance, figure 1.2 shows how solutions for the system $\dot{x} = x^2 - \mu$ differ between $\mu = -1, 0, 1$. More specifically, it can be

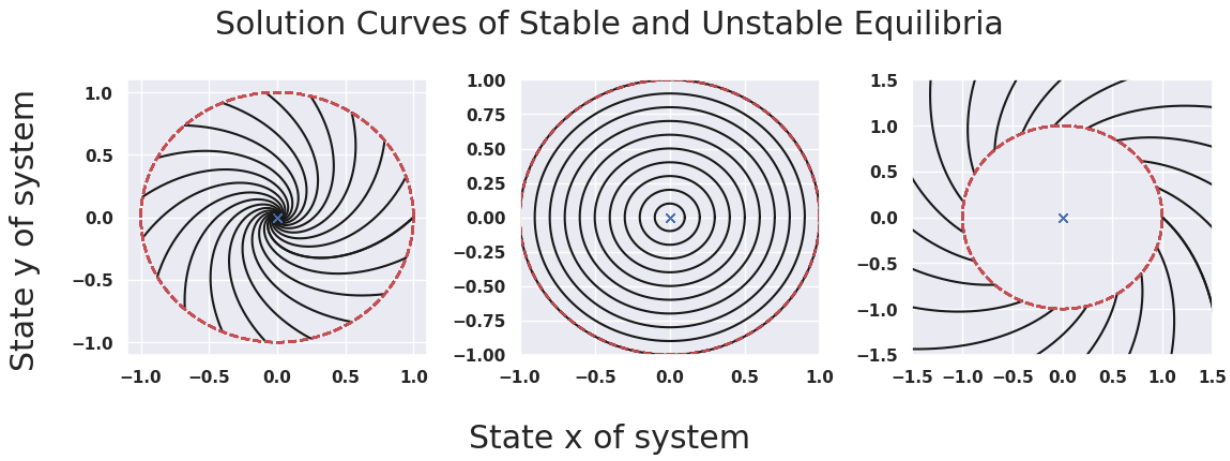

Figure 1.1: Examples of two-dimensional system behaviour near equilibria of varied stability. In particular, all initial states of the system lie on the unit circle, except for the centre plot, whose initial conditions lie on circles of radius $0.1n$, $1 \leq n \leq 10$. Fixed points are denoted by blue Xs. The left, centre, and right plots represent asymptotically stable, stable, and unstable equilibria, respectively.

shown that this system has two equilibria when $\mu < 0$, one when $\mu = 0$ and no equilibria when $\mu > 0$.

To capture this intuitive notion, we say a parameterized system $\dot{x} = f(x, \mu)$ *exhibits a bifurcation* when there exists a parameter value $\mu_0 \in \mathbb{R}^m$ such that any arbitrarily small change to the value of μ_0 changes the number or stability of equilibrium solutions to the relevant dynamical system. Furthermore, we say that μ_0 is a *bifurcation value* of the dynamical system. We note that this informal definition provides sufficient conditions for a local bifurcation, but that a more rigorous definition in terms of structural stability can be found in [11] or [22].

Given our definition of a local bifurcation, we note that several different kind of bifurcations exist. For instance the system given by equation $\dot{x} = x^2 - \mu$ exhibits the so-called *saddle-node* bifurcation: a bifurcation which describes two fixed points of opposite stability to appear, or coalesce and disappear, when a system parameter is varied through its bifurcation value [32]. Other kinds of bifurcations include the transcritical bifurcation, which changes the stability of a persistent fixed point. Another significant fact about bifurcations is that if the system $\dot{x} = f(x, \mu)$ a bifurcation at value μ_0, then, oftentimes, the matrix $Df(x^*, \mu_0)$ must have an eigenvalue with real component zero for at least one equi-

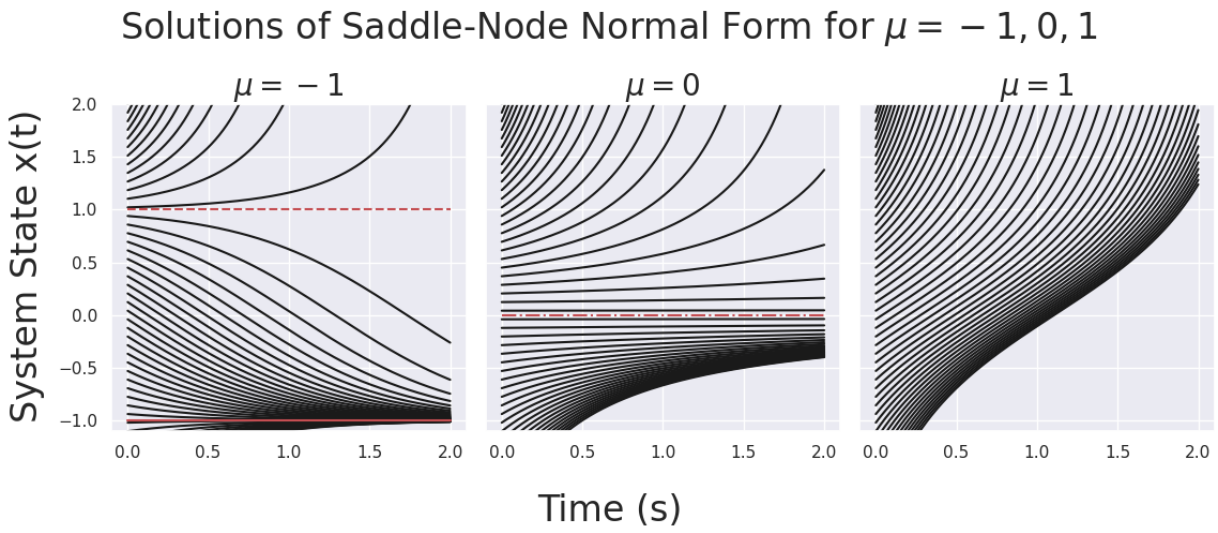

Figure 1.2: Solutions to IVPs for $\dot{x} = f(x;\mu) = x^2 - \mu,\ \mu = \pm 1, 0$. Stable, unstable, and semi-stable equilibrium solutions are given by the solid, dashed, and dot-dashed red lines, respectively.

librium x^*[2]. Indeed it is this insight which motivatves the study of *early warning signals* (henceforth EWS).

Roughly characterized, EWS denote the response of many mathematical and empirical systems as their parameters are varied towards a bifurcation. More specifically, suppose that x^* is a stable fixed point, and that a variation of system parameters towards a bifurcation point μ_0 causes $\text{Re}(\lambda_D(x^*)) \to 0^-$. In these instances, solutions near an equilibrium often exhibits two characteristic phenomena: *critical slowing down*, and what we call nonlinear emergence. The first phenomenon describes how solutions near a stable equilibrium x^* often converge more slowly towards x^* as $\text{Re}(\lambda_D(x^*)) \to 0^-$ [5, 27, 28]. Nonlinear emergence describes how nonlinear dynamics are no longer negligible for solutions close to x^* when the dominant eigenvalue of x^* approaches [5]. In some sense, these phenomena emerge from the failure of a system's linearization to adequately capture its dynamics as $\text{Re}(\lambda_D(x^*)) \to 0^-$. Indeed, consider a one-dimensional parameterized system $\dot{x} = f(x;\mu)$ at $x \in N_\delta(x^*)$ for δ small. Taking the Taylor expansion of f about (x^*, μ) fixed reveals

[2]In particular, this is a necessary condition for codimension 1 bifurcations; see §3.1 of [11] for more details).

that

$$\dot{x} = \cancel{f(x^*;\mu)}^{0} + f'(x^*)\epsilon + \frac{f''(x^*)}{2}\epsilon^2 + \mathcal{O}(\epsilon^3) \simeq \lambda_1\epsilon + \lambda_2\epsilon^2 \tag{1.1}$$

where $\epsilon := x - x^*$ and λ_i denotes the i^{th} coefficient of the relevant Taylor expansion. We see that $\lambda_D(x^*) = \lambda_1$, so as $\lambda_D(x^*) \to 0^-$ (i) the linear restorative force acting on x diminishes in magnitude, and (ii) that the quadratic term $\lambda_2\epsilon^2$ may be of comparable magnitude to the linear term if $|\lambda_2| \gg |\lambda_D(x^*)| \simeq |\epsilon|$. These two observations lead to critical slowing down and nonlinear emergence, respectively (see box 2 of [5] for more details).

1.2.2 Predicting Tipping Points with Early Warning Signals

EWS provides a fruitful framework for analysing parameterized, non-autonomous, dynamical systems. More concretely, consider the dynamical system $\dot{x} = f(x;\mu(t))$ where the model parameters continuously depend on t. We say that the system *passes through a tipping point* at t^* if and only if $\mu(t^*)$ is a bifurcation value. If the bifurcation removes or changes the stability of a stable equilibrium x^*, then the system is likely to exhibit early warning signals at times earlier than t^*. Indeed, we see this is the case for the system $\dot{x} = x^2 - (1-t)$, as shown in figure 1.3.

Since many systems model present states empirical systems as stable equilibria, including several models of ocean circulation (see [17, 31, 34]), it is worth considering whether EWS can be tracked over time to infer a system tipping point. At first glance, this appears impossible, as it would require that system states somehow move from their stable equilibria. However, systems often include noisy dynamics which perturb the system state away from its stable equilibrium x^*, and into a small neighborhood $N_\delta(x^*)$ [27]. As a system approaches a tipping point, it often exhibits critical slowing down and nonlinearities in response to noisy perturbations. These EWS then manifests themselves as statistical properties of the state's time series, particularly increasing time series variance and autocorrelation [3, 4, 5, 27]. These statistical properties are clearly visible in figure 1.4, where we have applied the same amount of noise to stable equilibrium solutions of the saddle-node normal form when $\mu = -10, -0.01$.

Given these statistical features of systems nearing tipping points, it is common to use traditional classification techniques like the Kendall-τ test to predict the occurrence or non-occurrence of a tipping point [3, 27, 28]. However, Bury et al.[3] theorized that a deep

[3]Note that unless otherwise specified, all commentary related to Bury et al's work is based on [5].

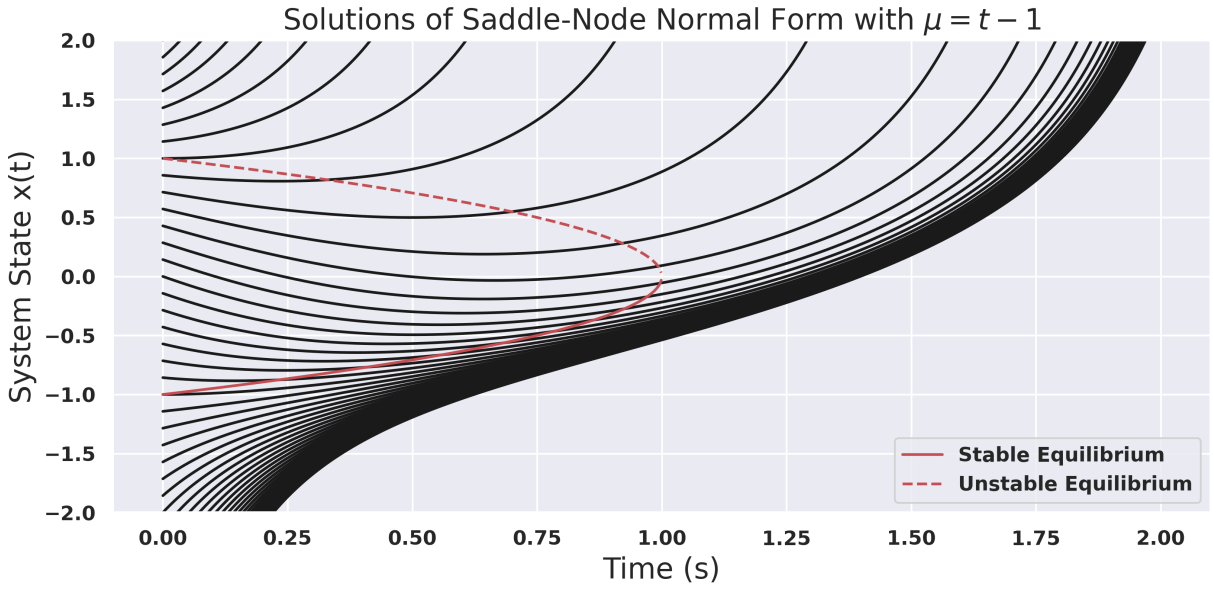

Figure 1.3: Solutions for $\mu(t) = 1 - t$. Notice for the right plot that solutions with initial conditions less than 1, are converging toward the sable equilibrium at t=0, but eventually fall out of its basin of attraction due to emergent nonlinearities. At $t = 1$, all solutions begin diverging to infinity, as expected.

neural network trained on a sufficiently large data set will be able to predict both whether or not a system will pass through a tipping point via early warning signals. Their networks, trained on generated time series from simple, two-dimensional dynamical systems, is able to predict whether or not a system will pass through a tipping point with higher accuracy and fewer false positives than traditional methods. Their network can also accurately predict what type of bifurcation the system will exhibit on many sets of generated and empirical data alike.

1.3 Motivating Questions

Motivated by the success of Bury et al.'s networks, we use their work as a sort of blueprint for our own. In particular, we aim to generate a data set of stochastically forced time series from low-order box models of AMOC, and use the data to train CNN-LSTM classifiers that predict AMOC collapse. We are primarily interested in comparing the performance of

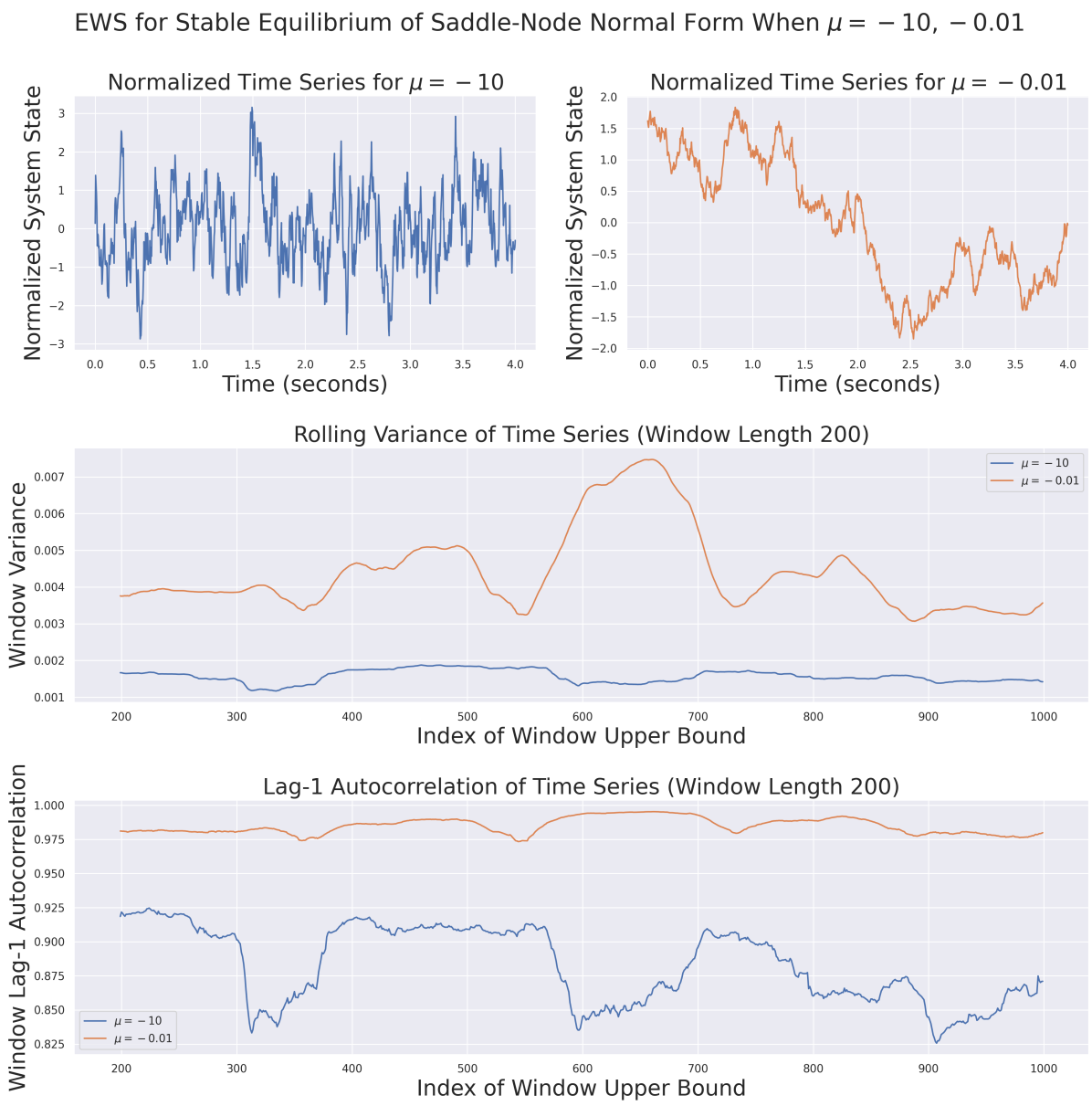

Figure 1.4: Normalized time series data of stable equilibrium solutions to the saddle-node normal form forced with equal amounts of white noise ($\sigma = 0.01$) , and their corresponding rolling variance and autocorrelation values. We note that when $\mu = -0.01$, the system exhibits much higher variance and autocorrelation compared to $\mu = -10$, as expected.

FBM-specific multivariate classifiers versus Bury et al.'s original models. In particular, we believe that AMOC dynamics generated from, and classifiers trained on box model data differs from the training procedure and data generation methods outlined by Bury et al. in three important regards. First, AMOC is not generally represented as a variable in box models, but by a linear combination of box temperatures and salinities (see, for instance, [17, 31, 34]). As a result, EWS present in the time series of the box model state may not necessarily appear in AMOC data, as the linear combination may introduce noise that either amplifies or diminishes the presence of such EWS. Second, Bury et al.'s models are univariate time series classifiers. However, box models generally have states described by no fewer than four dynamic variables. As a result, our classifiers may be able to leverage these additional variables, and their corresponding time series, to better classify AMOC data. It may even be the case that we can successfully train a classifier to detect AMOC collapse, even when it is only trained on box temperature and salinity data. Finally, we note that Bury et al.'s models predict both if a system is approaching a tipping point, and which *kind* of bifurcation induces this tipping point, resulting in four possible output values — 3 for different kinds of bifurcations, and one for no tipping point. However, we are only concerned with whether or not an AMOC time series is approaching a tipping point. To this end, it is insightful to ask if slightly modifying one of Bury et al.'s networks so that it only outputs two labels yields a better performance on our box model AMOC data when compared to the original, unmodified network. Given these concerns, we wish to concretely address the following four questions:

1. How well does one of Bury et al.'s model perform on a test set of AMOC data generated on box model data with no additional training?

2. How well does one of Bury et al.'s model perform on a relevant test set if it is modified to only output two labels, corresponding to whether or not AMOC collapse will occur?

3. How much better does a neural network which takes in time series for all box model variables perform when compared to a similar network which only takes in time series corresponding to AMOC?

4. Can a neural network successfully predict whether AMOC collapses even when it is not directly trained on AMOC data? That is, can time series of box temperatures and salinities be used as proxies to infer AMOC dynamics?

In order to address these questions, we must first determine which kind of neural network is best suited to our classification task, which box model we should use to generate

training data, and how we can generate this box model data efficiently to create a data set for model training. These topics are covered in §2. Given a relevant network architecture, box model and data set, we then create and train several neural network based classifiers to address questions 2-4. The performance of our networks, including comparisons to Bury et al.'s networks, are discussed in §3.

Chapter 2

Methods & Data

2.1 Deep Neural Networks & the CNN-LSTM Architecture

2.1.1 Neural Networks & Supervised Learning

Machine learning broadly concerns itself with "teaching" a computer to "learn" how to perform a specific task[1]. Given our desire to train an AMOC classifier, we are specifically interested in so-called supervised classification tasks: tasks where given some labelled input, we are interested in finding a function which predict the appropriate label from its corresponding input. In particular, inputs are represented by so-called feature vectors $x \in \mathbb{R}^n$, while labels are specified by targets $y \in \{1, ..., k\}$. The networks are then "taught" or trained to learn targets from data sets of labelled feature vectors.

A feed-forward neural network is nothing more than the composition of many *layer functions* $f : \mathbb{R}^m \to \mathbb{R}^n$ of the form $f(x; W, b) = \sigma(Wx + b)$. Here W is an $n \times m$ *weight* matrix, $b \in \mathbb{R}^n$ is a bias vector, and $\sigma : \mathbb{R}^n \to \mathbb{R}^n$ applies a nonlinear *activation function* component-wise to $Wx + b$. In principle, the activation could be any sufficiently differentiable function. But in practice only a few functions, including the sigmoid, tanh and ReLU functions shown in figure 2.1, are used as activations.

[1] This section is based on the first two chapters of [20].

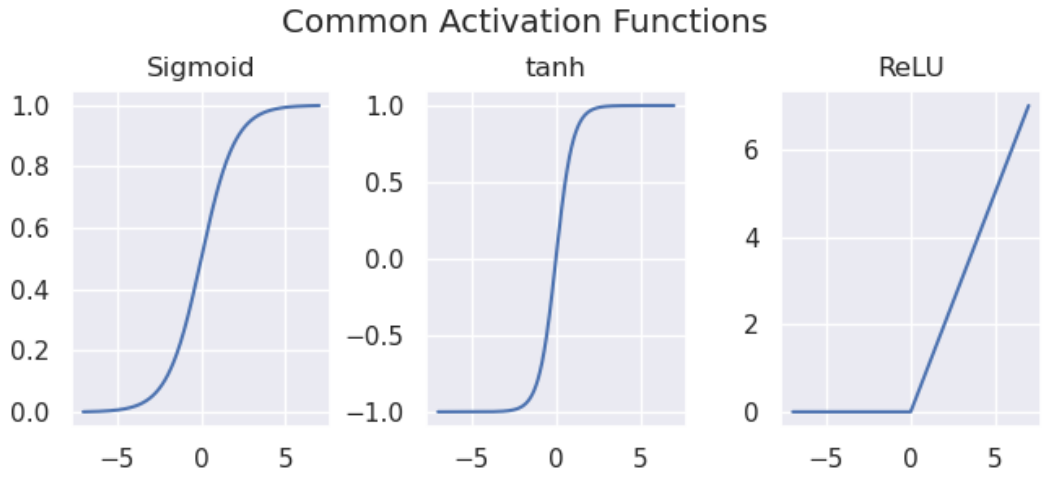

Figure 2.1: Common activation functions for networks. The sigmoid, tanh, and ReLu functions are defined as $1/(1+e^{-x})$, $(e^{2x}-1)/(e^{2x}+1)$, and $\max(0,x)$, respectively.

Given L layers, $f_1, ..., f_L$, the neural network $F_\Theta(x)$ is defined as by :

$$F_\Theta(x) := f_L(f_{L-1}(\cdots(f_1(x; W_1, b_1)\cdots; W_L, b_L) \tag{2.1}$$

$$= \sigma_L\left(W_L\sigma_{L-1}\left(\cdots\sigma_1\left(W^1x+b^1\right)\cdots+b_{L-1}\right)+b_L\right) \tag{2.2}$$

Where W_l, b_l denote the weight matrix and bias vector of layer $1 \leq l \leq L$, and Θ is the set of all weights and biases. The predicted label $\hat{y}$ of x is usually taken to be the layer activation σ_L is often the softmax function $\mathcal{S}$, as this function yields a probability distribution over all possible labels $k \in \{0, 1, ..., K-1\}$. Binary classification tasks provide an exception to this generalization. In these tasks, σ_L is usually the sigmoid function, and $\hat{y} := \mathbb{I}\{F_\Theta(x) > 0.5\}$, where $\mathbb{I}$ is an indicator function.

$$\mathcal{S}(z)_i = \frac{e^{z_i}}{\sum_{j=0}^{K-1} e^{z_j}};\ 0 \leq i \leq K-1 \tag{2.3}$$

Having specified in full generality what a neural network is, we now describe *supervised learning*. The general aim of supervised classification is to have the network accurately predict the discrete label $k \in \{0, ..., K-1\}$ of an input x that the network has not "seen" before. More specifically, let $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^n$ be a data set associated with a classification task. The concrete aim of supervised learning is to find neural network parameter values

that minimizes the average loss over a training set $\mathcal{D}_{tr} \subseteq \mathcal{D}$:

$$\hat{\Theta} = \arg\min_{\Theta} \left[\frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}_{tr}} L(F_\Theta(x_i), y_i) \right] := \arg\min_{\Theta} \left[L_{ave}\left(\mathcal{D}_{tr}, \Theta\right) \right] \tag{2.4}$$

where L is a so-called *loss function* and L_{ave} is the *average loss*. A loss function encodes some measure of inaccuracy. Common loss functions include the mean-squared error loss $L(f_\Theta(x), y) := [y_i - f_\Theta(x)]^2$, and in the case of binary classification (i.e. $k \in \{0, 1\}$) the *binary cross-entropy* loss $L(f_\Theta(x), y) := -[y \log(f_\Theta(x)) + (1-y)(1 - \log f_\Theta(x))]$. Assuming a fixed training set, define $\mathcal{L}(\Theta) = L_{ave}(\mathcal{D}_{tr}; \Theta)$. Rudimentary calculus reveals that $\nabla\mathcal{L}(\Theta)$ is the direction which causes the steepest decrease in $\mathcal{L}$.. As a result, adjusting Θ by a sufficiently small step size α in the direction $-\nabla\mathcal{L}(\Theta)$, should lead to a new parameter values $\Theta' = \Theta - \alpha\nabla\mathcal{L}(\Theta)$ which decrease $\mathcal{L}$. The procedure just described corresponds to a single iteration of the famous *gradient descent* optimizer, which at iteration $i+1$ updates network parameters as follows:

$$\Theta_{i+1} \leftarrow \Theta_i - \alpha\nabla\mathcal{L}(\Theta_i)$$

Note that $\alpha \notin \Theta$. For this reason, α is called a *hyperparameter*, as its value cannot be directly inferred or optimized by minimizing $\mathcal{L}$.

One difficulty of gradient descent is that it assumes $\nabla\mathcal{L}(\Theta)$ is computable, but it is not clear a priori that this computation is possible. However, the famous *backpropagation algorithm* exploits the particular structure of deep neural networks to find components of $\nabla\mathcal{L}(\Theta_i)$ recursively. While we do not describe the algorithm in this paper, backpropagation and gradient descent work in tandem to train networks (see §13.3 of [18]). In particular, evaluating $\mathcal{L}(\Theta_i)$, then performing backpropagation to get $\nabla\mathcal{L}(\Theta_i)$, and finally updating $\Theta_i \to \Theta_{i+1}$ defines an *epoch* of training.

Of course, the training must terminate at some point. As a result, the number of epochs is either set in advance as a hyperparameter, or is terminated according to the networks performance on a *validation set*. By artificially separating a training set $\mathcal{D}_{tr}$ into a new, smaller training set $\mathcal{T}'$, and a new validation set $\mathcal{V}$, the model's loss on the validation set can be determined after each epoch. Then training can finish when the average loss on the validation set does not decrease after a certain number of iterations (see [20]).

2.1.2 The CNN-LSTM Network Architecture

Recall that the aim of Bury et al. is to classify time series of sequential state data by discrete labels. Although we have clarified how supervised learning can train a network,

what remains unclear is how to design the neural network f so that it will be successful at this *sequence classification* task. In particular, we seek a network which tasks in time series data $X_i := \{x_i^{(1)}, ..., x_i^{(T)}\}$; $x_i^{(j)} \in \mathbb{R}^{m(j)}$, $1 \leq j \leq T$ and returns and inferred target$\hat{y}_i$, where each sequence has length $m(j) \in \mathbb{N}$. The most popular family of architectures for these classification tasks are Recurrent Neural Networks (RNN), and the Long-Short Term Memory Network (LSTM) in particular [9, 18].

Let $X_i := \{x_i^{(1)}, ..., x_i^{(T)}\}$ be a sequence in $\mathbb{R}^{m \times T}$. For the purposes of classification, a recurrent classifier consists of a recurrent network f, and a classification layer[2]. The recurrent network is a neural network that for each index $t \in \{1, ..., T\}$, takes in a single feature vector $x_i^{(t)} \in \mathbb{R}^m$, and a so-called *hidden state* $h_{t-1} \in \mathbb{R}^p$ as inputs, where $p \geq 1$ is a hyperparameter. The parameterized network $f_\Theta : \mathbb{R}^{m+p} \to \mathbb{R}^p$ then outputs a new hidden state $h_t = f(x_t, h_{t-1})$, after which $x_i^{(t+1)}$ and h_t are fed into the network, to produce $h_{t+1} = f(x_{t+1}, h_t; \Theta)$, and so on. The cycle repeats until the final feature vector $x_i^{(T)}$. Finally, the hidden state $h_T = f(x_i^{(T)}, h_{T-1})$ is fed into the single layer $\ell(\cdot) : \mathbb{R}^p \to \mathbb{R}^k$, which is usually the softmax or sigmoid function following our discussion in §2.2.1. Although it may initially appear that recurrent classifiers do not conform to neural network architectures, we note that "rolling out" an RNN, as shown in figure 2.2 yields a standard neural network. As a result, both gradient descent and backpropagation can be performed on RNNs [18]. The purpose of the hidden state h_t is to provide a *representation* of the sequence X_i at time t. The hope is that by feeding both $x_i^{(t)}$ and h_{t-1} into the recurrent network f, the hidden state h_t represents the sequence up to t by accounting for both the changes to the sequence at index t, as well as the behaviour of the sequence up to $t-1$. In practice, however, the hidden state often fails to represent more than the past few timesteps. Long-short term memory networks were specifically designed to remedy this concern. These networks enhance RNNs by augmenting the hidden state h_t with a memory state c_t, which provides additional control over how much the hidden state changes between $t-1$ and t. This modified RNN has resulted in improved performance for a wide range of sequence classification tasks, especially time series classification [15].

While LSTMs perform well in sequence classification tasks, some state-of-the art time series networks do not apply LSTMs directly on raw time series data. Instead, these classifiers first pass each sequence through a set of one-dimensional *convolutional* layers [7, 19]. A convolution extracts particular features from the input x and encodes higher-order, more abstract features of x [12]. More technically, let $x \in \mathbb{R}^n$ be a feature vector, and denote by $x_{i:j}$ the vector $(x_i, x_{i+1}, ..., x_j) \in \mathbb{R}^{j-i}$ where $1 \leq i < j \leq n$. A one-dimensional convolution between a *kernel* $w \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$ yields an *encoding* $z \in \mathbb{R}^{n-m+1}$ of x,

[2]This summary is based on [18].

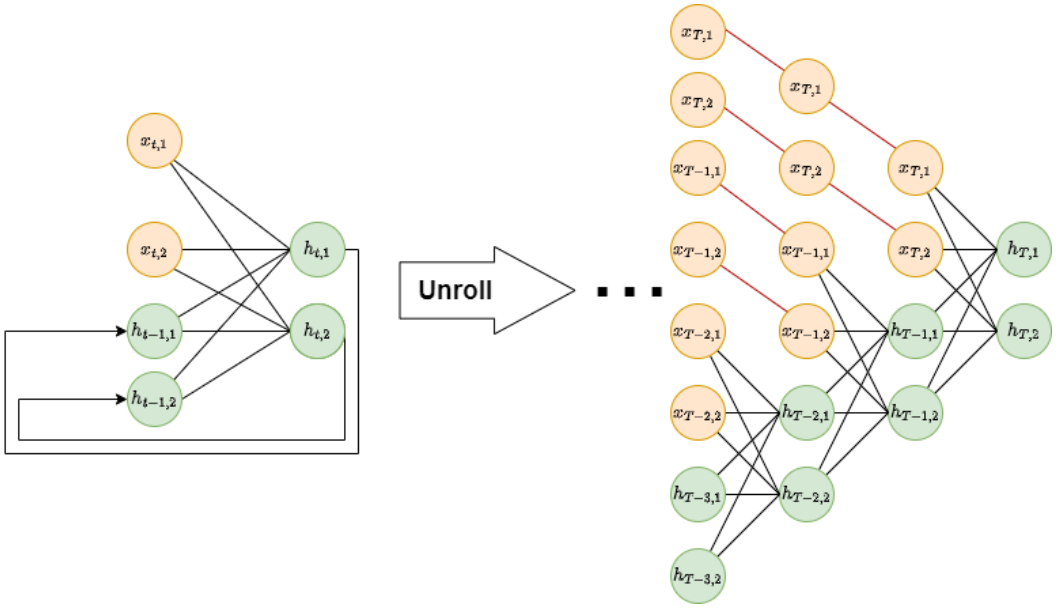

Figure 2.2: An "unrolled" LSTM network where feature vectors and the hidden state are two-dimensional. Red edges denote a weight of 1, while a missing edge denotes a weight of zero.

whose components are given by $z_k = x_{k:k+m} \cdot w$, $1 \leq k \leq n - m$. Oftentimes, the vector $x \in \mathbb{R}^n$ is *padded* to $x' = (0, ..., 0, x_1, ..., x_n)^T \in \mathbb{R}^{n+(m-1)}$ so that the encoding of x' has the same dimension as x. Since convolutions return linear dot products of features, these layers are nothing more than highly structured weight matrices. So, performing backpropagation and gradient descent, allows the network to "learn" the best kernels for a classification task.

Finally, we note that input layers are often convolved with many kernels of the same length to yield multiple encodings. These $n_z \in \mathbb{N}$ encodings, where n_z is a hyperparameter, ensure that many higher-order features are available to later layers of the network, and generally results in better classifier performance [18]. After all convolutions are performed on an input $x_i \in \mathbb{R}^n$, the encodings are concatenated together into single sequence $\overline{X}_i \in \mathbb{R}^{n_z \times n}$.

The result of incorporating convolutional layers and an LSTM into a single network is the CNN-LSTM architecture. In particular, each input sequence or time series X_i is convolved with multiple kernels to yield $\overline{X}_i$, a sequence of many high-order features. Then $\overline{X}_i$ is passed through an LSTM, which represents the dynamics of $\overline{X}_i$ into a hidden state

h_t, which is then passed through a classification layer. This ensures that h_t represents the dynamics of higher-order features of the input X_i and high-performing classifiers.

2.2 The Four-Box Model

In order to train the networks needed to address our research questions, we need to generate a large data-set consisting of AMOC time series associated with low-order box models. We focus on these models because they are explicitly described by parameterized dynamical systems, represent AMOC speeds as stable equilibrium values, and describe AMOC collapse as local bifurcations. Hence these systems should exhibit EWS. In particular, we believe that the extension of Stommel's model presented in [34] best fits our criteria. In addition to the features common to most box models, we will see in §2.2.3 that this low-order model is simple to force stochastically via the Euler-Maruyama method, and allows us to determine if the system reaches a tipping point before running any simulations. In this way the model not only captures AMOC dynamics, but allows us to label training data very efficiently.

2.2.1 Overview & Dynamics

Zickfeld et al.'s *four-box model*[3], denoted *FBM* and shown schematically in figure 2.3, represents the Atlantic ocean as four well-mixed boxes with uniform temperatures and salinities. The boxes themselves represent the southern, northern, tropical and deep Atlantic oceans. For convenience's sake, these boxes are labeled 1-4, respectively. Boxes 1 and 2 are connected to boxes 3 and 4 via capillaries, which allows for the closed circulation of water and salt. The density of each box is then specified as a linear function of its temperature $T_i(t)$ and salinity $S_i(t)$:

$$\rho_i = \rho_0 \left(1 - \alpha T_i(t) + \beta S_i(t)\right)$$

where ρ_0 is a reference density for saltwater, α is a thermal expansion coefficient, β is a haline contraction coefficient. Exact values for these parameters can be found in table 2.1. The model then assumes that the volume circulation rate m of THC is, in turn, specified by the density gradient between boxes 1 and 2:

$$m(t) = \frac{k}{\rho_0}[\rho_2(t) - \rho_1(t)] = k\left(\alpha[T_2(t) - T_1(t)] - \beta[S_2(t) - S_1(t)]\right) \tag{2.5}$$

[3]Note that the ensuing discussion is based on [34].

where k is a constant linking transport volumes rates to density gradients. Notice that $m(t)$ is positive if and only if the density of box 2 is greater than box 1. Thus we may take $m > 0$ to denote a northward flow of surface waters. It is also clear from this observation that salinity and temperature drive the circulation in opposite directions, confirming that circulation is thermally driven whenever $m > 0$.

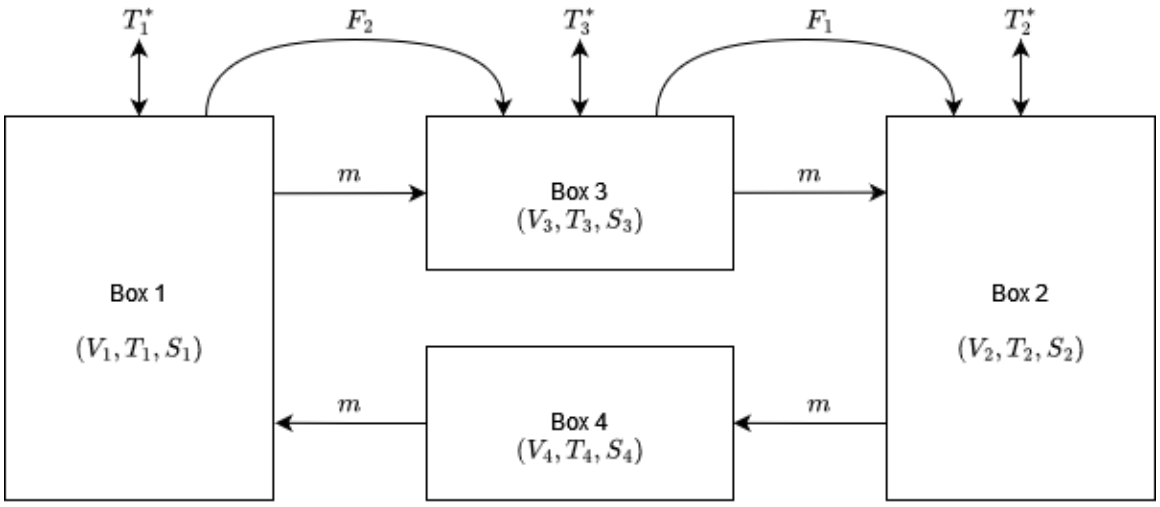

Figure 2.3: Schematic representation of the four-box model. Note that box i is defined by its fixed volume V_i, and its variable temperature and salinity given by $T_i,\ S_i$ respectively. Meanwhile, the circulation speed of the ocean is specified by m, which in turn depends on T_1, T_2, S_1, S_2. Finally, notice that the surface boxes are each forced towards a relaxation temperature $T_i^*,\ 1 \leq i \leq 3$ in the absence of internal circulation, and box salinities are forced by freshwater fluxed F_1 and F_2.

In order to fully specify m, we must account for the dynamics of box temperatures and salinities. These variables broadly depend on a *circulation term* and a *forcing term.* The circulation term describes how the temperature and salinity of box i changes due to some of the box water leaving and being replaced by water from its downstream neighbor $d(i)$. Because the amount of water leaving a box depends on $m(t)$, this term should depend on $m(t)$ and one of $T_i(t) - T_{d(i)}(t)$, $S_i(t) - S_{d(i)}(t)$. Meanwhile the forcing term captures how quickly the surface boxes are forced by the atmosphere and freshwater fluxes. In particular, surface box temperatures T_i are forced towards a relaxation temperature T_i^*. The surface box salinities are forced by two time-varying freshwater fluxes F_1, F_2 representing the freshwater leaving box 1 to box 3, and the freshwater leaving box 3 into box 2, respectively.

These considerations give rise to the following eight-dimensional dynamical system

$$\dot{T}_1 = \frac{m}{V_1}(T_4 - T_1) + \lambda_1(T_1^* - T_1) \tag{2.6}$$

$$\dot{T}_2 = \frac{m}{V_2}(T_3 - T_2) + \lambda_2(T_2^* - T_2) \tag{2.7}$$

$$\dot{T}_3 = \frac{m}{V_3}(T_1 - T_3) + \lambda_3(T_3^* - T_3) \tag{2.8}$$

$$\dot{T}_4 = \frac{m}{V_4}(T_2 - T_4) \tag{2.9}$$

$$\dot{S}_1 = \frac{m}{V_1}(S_4 - S_1) + \frac{S_0 F_1}{V_1} \tag{2.10}$$

$$\dot{S}_2 = \frac{m}{V_2}(S_3 - S_2) - \frac{S_0 F_2}{V_2} \tag{2.11}$$

$$\dot{S}_3 = \frac{m}{V_3}(S_1 - S_3) - \frac{S_0(F_1 - F_2)}{V_3} \tag{2.12}$$

$$\dot{S}_4 = \frac{m}{V_4}(S_2 - S_4) \tag{2.13}$$

We briefly note that equations 2.6-2.13 are only sensible for present day AMOC circulations. Indeed, if AMOC transport were reversed, then the downstream neighbor of each box would change, as shown in §2 of [33]. We also note that the forcing terms are themselves combinations of model parameters. In particular $\lambda_i := \Gamma/(c\rho_0 z_i)$, where each of these parameters are further described in table 2.1.

Finally, we must specify the values of the model parameters to determine m. All model parameters except the *forcing parameters* F_1, F_2, T_1^*, T_2^*, and T_3^* do not change with respect to CO_2 forcings, and are therefore *constant*. Furthermore, most parameter values can either be empirically measured or are given by the CLIMBER-2 model of intermediate complexity (see [23] for an overview of CLIMER-2). Hence, these parameters are *latent*, as their values must be inferred. In order to infer the latent constant parameters, the authors optimize them so that the FBM response to F_1 forcings match those of CLIMBER-2 [29, 34]. The resulting parameter values are shown in table 2.1.

2.2.2 Stability, Bifurcations & EWS in the FBM

Much like other box models, the FBM represents the current thermally-driven circulation rate m as a function of the model's equilibrium state. To this end, we consider equilibrium solutions to the FBM and their stability. We then see that this system exhibits something

Parameter	Description	Equilibrium Value	Constant?	Latent?
c	Specific heat capacity of seawater	$4000J$ kg^{-1} °C^{-1}	✓	✗
ρ_0	Reference density of seawater	1025 kg m^{-3}	✓	✗
α	Thermal expansion coefficient	1.7×10^{-4} °C^{-1}	✓	✗
β	Haline expansion coefficient	8×10^{-4} psu^{-1}	✓	✗
S_0	Reference salinity for seawater	35 psu	✓	✗
V_1	Volume of southern box	1.1×10^{17}m^3	✓	✗
V_2	Volume of northern box	0.4×10^{17}m^3	✓	✗
V_3	Volume of tropical box	0.68×10^{17}m^3	✓	✗
V_4	Volume of deep box	0.05×10^{17}m^3	✓	✗
z_1	Depth of southern box	3000 m	✓	✗
z_2	Depth of northern box	3000 m	✓	✗
z_3	Depth of tropical box	1000 m	✓	✗
F_1	Freshwater transport between southern and tropical boxes	0.014 Sv	✗	✗
F_2	Freshwater transport between tropical and northern boxes	0.065 Sv	✗	✗
T_1^*	Relaxation temperature of the southern box	6.6°C	✗	✓
T_2^*	Relaxation temperature of the northern box	2.7°C	✗	✓
T_3^*	Relaxation temperature of the tropical box	11.7°C	✗	✓
Γ	Thermal coupling constant	$\frac{7.3\times10^8}{3.154\times10^7}$ Js^{-1}m^{-2} °C^{-1}	✓	✓
k	Empirical flow constant	$\frac{25.4\times10^{17}}{3.154\times10^7}$ m^{3}s^{-1}	✓	✓

Table 2.1: Parameter names and descriptions for the FBM. As noted by [29, 34], the model is calibrated so that F_1 matches the equilibrium value produced by Petoukhov et al.'s CLIMBER-2 model (see [23] for a summary of CLIMBER-2).

like the saddle-node bifurcation described in figure 1.2. In particular, variations of the model's parameters cause two equilibrium circulation values of opposite stability to coalesce and eventually disappear. However, our treatment of this topic is not rigorous. We choose this informal exposition based on the rigorous work of [25] and [29], as a full treatment would be both lengthy, and is already sketched in these two papers.

Consider the equilibrium solutions of the four-box model. By definition, $\dot{T}_i = \dot{S}_i = 0,\ 1 \leq i \leq 4$, so it immediately follows from equations 2.9 and 2.13 that $T_4^{eq} = T_2^{eq};\ S_4^{eq} = S_2^{eq}$. Since equation 2.10 is zero, and $S_2^{eq} = S_4^{eq}$, we may substitute $m_{eq}(S_2^{eq} - S_1^{eq}) = S_0 F_1$ into equation 2.5 to get the following quadratic equation.

$$0 = m_{eq}^2 - k\alpha(T_2^{eq} - T_1^{eq})m_{eq} + \beta S_0 F_1 \tag{2.14}$$

We further note that equations 2.7 and 2.8 can be expressed as a function of $\Delta T := T_2^{eq} - T_1^{eq}$ and m_{eq} by re-arranging equations 2.6-2.13. Using these transformed equations, ΔT can then be expressed as a function $\Delta T(m; T_1^*, T_2^*, T_3^* \mathcal{P}_c)$. Here, $\mathcal{P}_c$ is a vector of the model's fixed parameters. Assuming further that ΔT is fixed — even if this is unrealistic — illuminates why m exhibits a saddle-node bifurcation. In this simplified scenario, equation 2.14 becomes

$$0 = m_{eq}^2 - k\alpha\Delta T m_{eq} + \beta S_0 F_1 \tag{2.15}$$

$$\Rightarrow m_{eq} = \frac{k\alpha\Delta T}{2} \pm \sqrt{\frac{1}{4}k^2\alpha^2\Delta T^2 - \beta S_0 F_1} \tag{2.16}$$

from which it is clear that $F_1^{crit} = (k\alpha\Delta T)^2/4\beta S_0$. Indeed, if $F_1 > F_1^{crit}$, then there are no real solutions for m_{eq}. Meanwhile $F_1 = F_1^{crit}$ yields a single stable solution, and $F_1 < F_1^{crit}$ yields two solutions, as expected. Further algebra reveals that one value of m_{eq} makes the derivative of equation 2.14 negative, while the other makes it positive. As a result, the two equilibria have opposite stability, as expected.

We also note that that forcing T_1^* and T_2^* can induce this saddle-node response. Indeed, now assume F_1 is fixed in time and $\frac{1}{4}k^2\alpha^2\Delta T^2 > \beta S_0 F_1$ so there are two equilibrium solutions for m, but T_1^* and T_2^* can vary. Slawig & Zickfeld show that $\Delta T(m; T_1^*, T_2^*, T_3^* \mathcal{P}_c)$ is an increasing function of T_1^* and a decreasing function of T_2^* [29]. Changing these parameters, in turn, can lower ΔT, and allows one to set $\frac{1}{4}k^2\alpha^2\Delta T^2 < \beta S_0 F_1$. This general fact is reflected in figure 2.4, where we show how linear forcings of F_1, T_1^*, T_2^* lead to negative values of m when simulating the FBM with additive noise.

The upshot is that m can undergo something like a saddle-node bifurcation under forcings of F_1, T_1^*, T_2^*, and T_3^*. The only reason we take care to specify that m undergoes

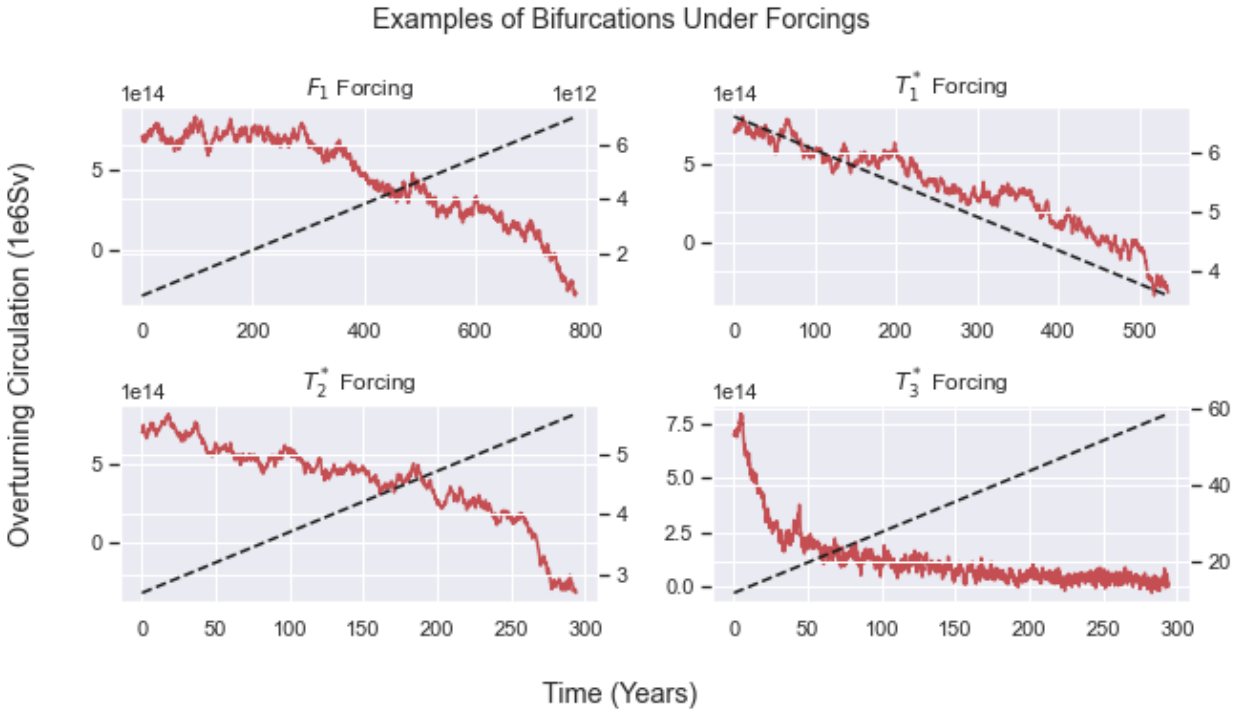

Figure 2.4: Examples of the FBM passing through a saddle-node bifurcation under individual forcings of F_1, T_1^*, T_2^* as shown in the upper left, upper right and lower left subplots, respectively. The passage of each system through a tipping point is confirmed by the fact that these subplots all result in a negative overturning circulation. Also notice that even when there is an extreme forcing of T_3^*, as shown in the lower right subplot, the overturning circulation m is always positive. Hence this extreme forcing does not result in the system undergoing a critical transition.

something "like" a bifurcation, is because m is not itself a dynamic variable of the system. However, it can be shown that the eight-dimensional system exhibits a saddle-node bifurcation only when equation 2.16 has no real solutions [25]. Since saddle-node bifurcations also necessitate that at least one eigenvalue of the FBM Jacobian approaches zero (see [11]), it follows that EWS should appear in stochastically forced time series of the system variables.

What is unclear however, is whether these EWS will appear in time series of m. Again, m is not itself a variable of the FBM, but a linear combination of the model's variables. This observation is crucially important given that the Jacobian of the FBM always has a zero

eigenvector, regardless of the variable parameter values, whose corresponding eigenspace is not spanned by any temperature variables (see appendix A.1 for more details). As a result, any perturbation along this eigenspace will always result in a nonlinear response. So if these nonlinear responses are incorporated into m, they may "mask" other early warning signals.

2.2.3 Generating Four-Box Model Training Samples

We provide a broad overview of our FBM data set generator. We pay particular attention to the ways our generator leverages analytic features of the FBM to streamline the data labelling process. Notably, our procedure utilizes the fact that local bifurcations in the FBM can be detected analytically from the time series of T_1^*, T_2^*, T_3^*, F_1. In particular, Slawig and Zickfeld show that equilibrium values of m are given by the real roots of a quartic polynomial $G(m)$, whose coefficients only depend on parameters. As a result, specifying the vector $\mathcal{P} = (F_1, T_1^*, T_2^* T_3^*)^T \in \mathbb{R}^4$, is sufficient to determine the number of equilibrium solutions $\chi(\mathcal{P})$ directly, since all other parameters are constant. Thus, it follows that a variation of $\mathcal{P}$ induces a saddle-node bifurcation just when varying $\mathcal{P}$ changes $\chi(\mathcal{P})$. This observation eliminates the need to use AUTO-OP7 or similar numerical bifurcation analysis software, greatly simplifying our generator's implementation.

The procedure used to generate one sample for our data set is summarized in algorithm 1. The generation procedure, which is based on a similar procedure in [5], is repeated 500,000 times to make a data set of the same size. For any given simulated time series, we first specify how many years should be simulated, what the time step Δt should be, whether a bifurcation occurs, and how many samples each training set time series should contain. We set each simulation to last a thousand years, as this is the time scale used by Zickfeld et al. when investigating the FBM response under F_1 forcings (see §5 of [34]). The time step Δt is set to $1/f$, $f \sim \mathcal{U}\{1, 10\}$ so that a wider range of lag-1 autocorrelation values are found in the training set. We ensure there are an equal number of time series that do and do not pass through a tipping point. Finally, our networks will take in 500-length time series as inputs. So, there should be 500 total timesteps in each series.

More specifically, let $k = 0, 1$ denote whether or not we wish to generate a time series where the FBM passes through a tipping point. Given k, we wish to generate the initial parameter values $\mathcal{P}_F^{init}$ and final parameter values $\mathcal{P}_F^{final}$ so that $\chi(\mathcal{P}_F^{init}) = 2$, but $\chi(\mathcal{P}_F^{final}) = 0$. In order to guarantee that this is the case, we create a sampler by fitting a linear support vector machine (henceforth LSVM) over a set of possible parameter values. This sampler, described more technically in appendix B.1., a LSVM yields a vector $w \in \mathbb{R}^4$

Algorithm 1 Four-Box Model Training Set Sample Generator.

Require: n_{iters}, n_{inp}, t_0, t_f, bif

$P_0,\ P_f \leftarrow LSVMS(\text{bif})$ ▷ Sample From Support Vector Machine

$\tau \leftarrow \{i \cdot 1000/n_{iters} \mid 0 \leq i \leq n_{iters}\}$

if bif $= 1$ **then**

$i \leftarrow \arg\min \left\{t_i \in \tau \mid \chi(P_{t_i}) = 0 \ \&\ \chi(P_{t_{i-1}}) = 2\right\}$

else

$i \leftarrow n_{iters}$

end if

$j \leftarrow i - 600$

$t_0', t_f' \leftarrow t_j, t_i$

$\sigma \leftarrow getNoise(P(t_0'))$ ▷ Find noise following [5].

$m, T, S \leftarrow solveEM(600, t_0', t_f', P(t_0'), P(t_f'), \sigma)$ ▷ Euler-Maruyama

$m \leftarrow detrendLOWESS(m)$ ▷ Pre-Processing

if $CPD(m) - j < 600$ **then** ▷ Run Changepoint Detector

BREAK

else

$j, i \leftarrow CPD(m) - 600, CPD(m)$

$m, T, S \leftarrow m_{j:i}, T_{i:j}, S_{j:i}$

end if

which infers the label $\chi(P)$ of $P \in \mathbb{R}^4$ based whether $w^T P \geq 0$ or $w^T P < 0$. Suppose without loss of generality that the LSVM infers $\chi(y) = 2$ whenever $w^T y \geq 0$. In this case, we may generate a sample P with $\chi(P) = 2$ by randomly sampling three component of P, and setting the final component so that $w^T P \geq 0$. Of course, this sampler does not guarantee $\chi(P) = 2$ due to misclassifications of the LSVM. However, in this circumstance, the sampler simply generates a new point P until $\chi(P) = 2$.

Next, we define the values of the model parameters $\mathcal{P}_F := (F_1, T_1^*, T_2^*, T_3^*)^T$ for each time $t_i = \frac{i}{f},\ i \in [0, 1000f]$. If $k = 1$, then parameters are linearly interpolated between $\mathcal{P}_F^{init}$ and $\mathcal{P}_F^{final}$over the course of the simulated millenium. Following Bury et al., the model parameters remain constant [5]. As a result of these considerations,$\mathcal{P}_{Fi} := \mathcal{P}_F(t_i)$ is given by

$$\mathcal{P}_{Fi} = \begin{cases} \mathcal{P}_F^{inital} & k = 0 \\ \frac{t_i}{1000} P_F^{final} + \left(1 - \frac{t_i}{1000}\right) \mathcal{P}_F^{inital} & k = 1 \end{cases}$$

If $k = 1$ then we compute $\{\chi(\mathcal{P}_{Fj})\}_{j=1}^{1000 \cdot f}$. It follows that a saddle-node bifurcation occurs at the first time t_b when $\chi(\mathcal{P}_{Fb}) = 0$, but $\chi(\mathcal{P}_{Fj}) = 2$ for $1 \leq j < b$. We then check that

$b > 600$. If this is the case, we retain the set $\{\mathcal{P}_{Fj}\}_{j'=b-600}^{b}$.

Next, a stochastic simulation of the four-box model is run. In particular, m, T_i, S_i $1 \leq i \leq 4$ are simulated over 600 time steps, starting at $t_{j'} = t_j - 600\Delta t$. Hence the model dynamics are simulated just until a bifurcation induced tipping point. More technically, the dynamics of the subsequent 600 time steps are simulated using the Euler-Maruyama (EM) method. This method solves for a sample path $X_t := X(t) \in \mathbb{R}^n$, $t \in [0, T]$ whose dynamics are given the stochastic differential equation $dX_t = a(X_t, t)dt + b(X_t, t)dW_t$. Here $W_t := W(t)$ denotes an $\mathbb{R}^n$ Wiener process. Given the discretization of the interval $[0, T]$ to $\{t_j\}_{j=1}^{n}$ where $t_j := T\frac{j}{n}$, $j \in \{0, ..., n\}$, the EM method determines each value $X(t_j)$, $j \in \{1, ..., n\}$ recursively:

$$X(t_j) = X(t_{j-1}) + a(X(t_j), t_j)\Delta t + b(X(t_j), t_j)\Delta W_j, \quad \Delta W_j \sim \mathcal{N}(0, \Delta t)$$

where $\Delta t := T/n$ (see chapter 9 of [10]). So, the value of the random variable X_t can be found recursively.

In their paper, Bury et al. assume that the stochastic component of each dynamical variable is Gaussian noise [5]. That is, $b(X(t_i), t_i) = \sigma$. This assumption allows the authors to determine an appropriate value of σ so that noise is present in each sample path, but does not dominate model dynamics. Our approach to determining the appropriate noise level σ for the FBM is nearly identical to this methodology. The crucial difference is that Bury et al.'s procedure sets σ according to the dominant eigenvalue of a system's stable equilibrium. However, we know that the dominant eigenvalue of the four-box model's stable equilibrium always has real component zero. This complication slightly changes how we compute the appropriate noise level σ — a topic we discuss in appendix B.2.

Once we find an appropriate value for σ, we solve for $T_i(t_j), S_i(t_j)$ $1 \leq i \leq 4$ at each time step t_j using the EM method with a stochastic component $\sigma\Delta W_j$. We also note that the model's non-constant parameters are set to $\mathcal{P}_{Fj}$ at each time-step, so that parameter forcings are incorporated in each simulated time series. Finally, $m(t_j)$ is found using equation 2.5 alongside the box temperatures and salinities at t_j.

The EM method yields a 600 length time series of m, T_i, S_i, $1 \leq i \leq 4$. In principle, these time series could be used as inputs to models, without any sort of preprocessing. However,we again follow Bury et al.'s procedure in [5], and perform a locally weighted scatter-plot smoothing regression (henceforth "LOWESS") on all time series. Intuitively, detrending removes the linear components of the original series, leaving behind a new series of residuals, as is shown in figure 2.5. So, performing LOWESS on FBM time series data should only leave behind a record of noise induced perturbations, and nonlinear responses to the noise — features which better represent the EWS in a system.

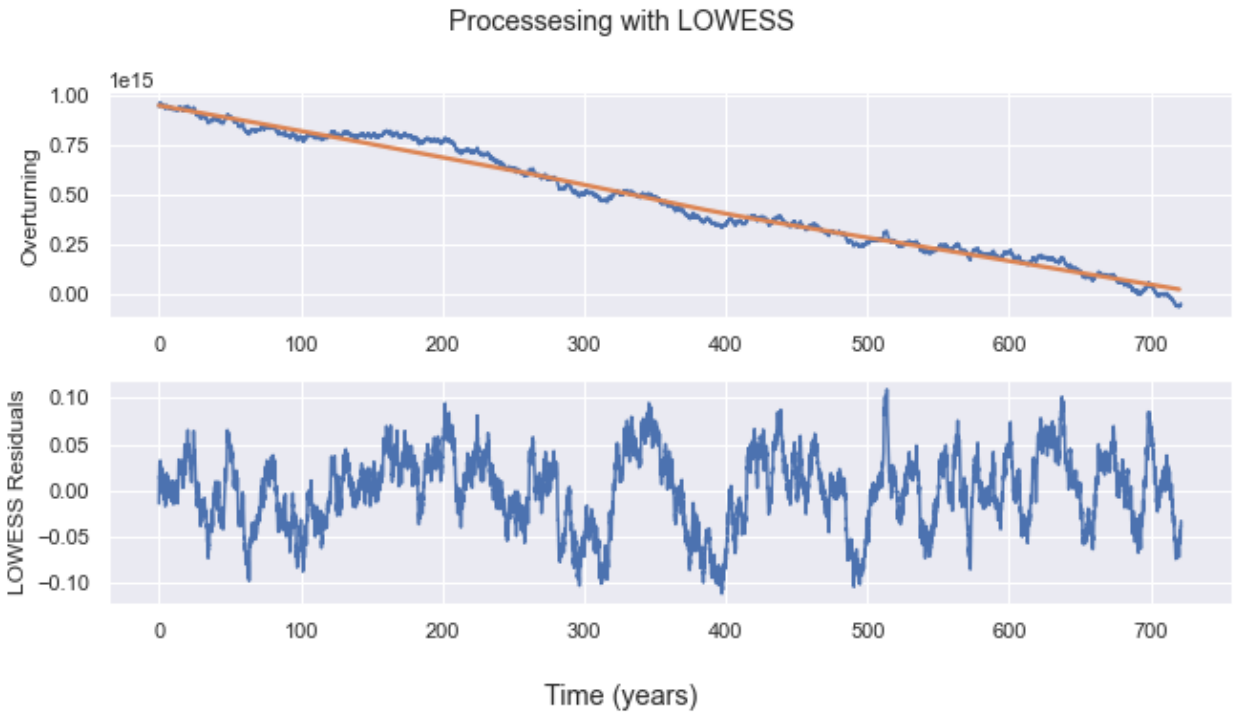

Figure 2.5: A time series resulting from performing LOWESS on a time series of m where $\chi(\mathcal{P}_F^{init}) = 2$ and $\chi(\mathcal{P}_F^{final}) = 0$. The top plot represents the unprocessed time series of m in blue, and the linear dynamics inferred by LOWESS are shown in orange. The bottom plot represents the preprocessed time series of m.

Each time series is normalized after LOWESS detrending. This is accomplished by dividing all time series entries by the sum of each entry's absolute value. Finally, we run a change point detector (henceforth CPD) on the detrended time series outputs. This detector ensures that the system has not been pushed through a tipping point by noise before the parameters reach their local bifurcation value. Consequently, if a change point is detected at index i_{cp}, the generator returns a length 500 time series from indices $i_{cp} - 500$ to i_{cp}. If $i_{cp} < 500$, the time series is discarded and the process is repeated. Finally, if no change point is detected, the output time series is just the final 500 value of the length 600 series. This series is then labelled by k and added to the data set.

In terms of implementation, we wrote an EM solver for the four-box model in python 3, making extensive use of both the `numpy` and `scipy` libraries. The LSVM fit on Ω was implemented with the `scikit-learn` library, while LOWESS detrended and CPD were

implemented with functions found in the `statsmodels` and `ruptures` libraries, respectively. The LOWESS window size was 120, while the CPD window size was 10. All relevant scripts were run on the speciality research CPU cluster provided by MFCF.

Chapter 3

Results

Given the methodology and resulting data set described in §2, we now turn to the task of evaluating CNN-LSTM networks trained on detrended FBM time series data. In order to address the questions raised in §1.3W we are not only interested in the performance of FBM-specific classifiers, but the performance of one of Bury et al.'s original models found in this GitHub repository. To this end, we consider the performance of the `best_model_1_1_len500` network — henceforth the *Bury network* — on a test set of 5,000 labelled FBM time series samples. Meanwhile, we also create and train our own *transfer learning* classifier, a *circulation only* classifier, an *all variable* classifier, and *proxy* classifier. For the transfer learning classifier, we leave the convolutional layers and LSTM components from the `best_model_1_1_len500` model (henceforth, the Bury model) intact, but train the weights and biases of a new classification layer $\ell : \mathbb{R}^p \to \mathbb{R}$ on FBM data. The other classifiers are specified what they accept as features. The circulation model takes in m as a time series feature, the proxy network takes in T_1, T_2, S_1 an S_2 as time series features, while the all variables model takes in m alongside the temperatures and salinities of every box as features.

We summarize the training procedure, specific architecture, and hyperparameter values of all of our models in §3.1. Next, we compare the performance of all of our networks and the Bury model. The metrics used to evaluate these networks are the accuracy, average loss, *precision*, *recall* and the *receiving operating characteristic* of these models. Given the similarities of our own methods and networks to the work of Bury et al., we briefly compare their methodology and data to our own in §3.3. Finally, §3.4 discusses some limitations and possible extensions of our work.

3.1 Model Architectures & Training

Although we have clarified that CNN-LSTMs are well-suited to predicting AMOC tipping points, we have not yet specify how our models are trained, nor how their hyperparameters are chosen. The precise architecture of our models, including their hyperparameter values, are again based on Bury et al. The general architecture of this model is given in figure 3.1. Meanwhile its hyperparameters are given in table 3.1, which also contains the hyperparameters of our four models as well.

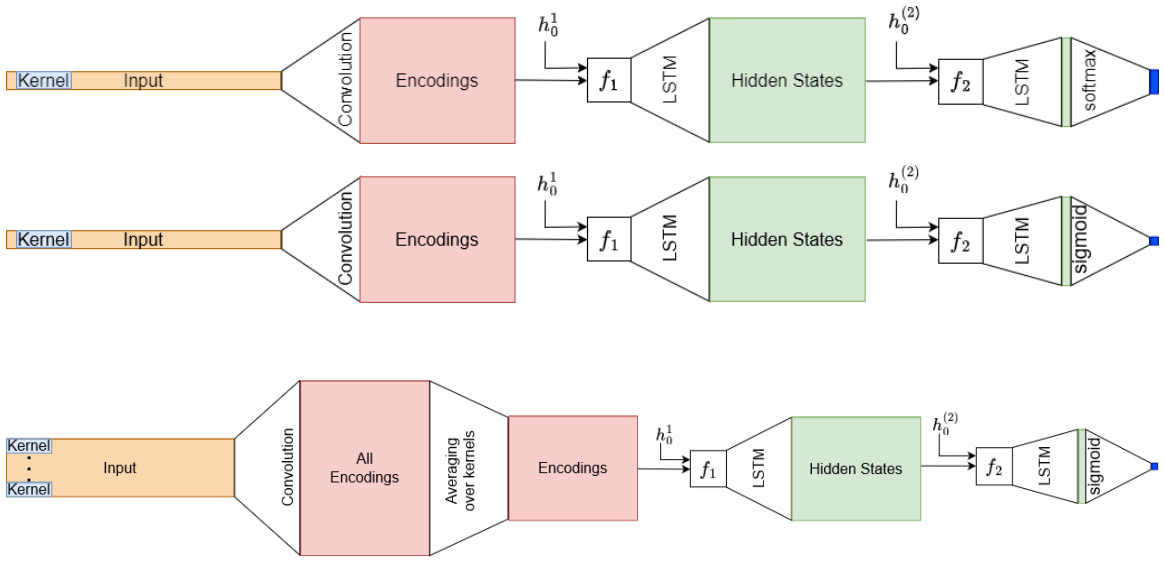

Figure 3.1: Architectures of Bury et al.'s network (top), the transfer and circulation networks (middle), and the all variable and proxy networks (bottom). The kernel (light blue) of multiple convolutions slides over the input sequence (orange), yielding a new sequence of encodings (red). For the all variable and proxy networks, these encodings are over the kernels of the convolutional layers. These encodings pass through a max-pooling layer, and then through a first LSTM f_1 which outputs all of its hidden states (green). This hidden state sequence is passed into another LSTM f_2 and outputs the final hidden state $h_{250}^{(2)}$. Finally, the components of $h_{250}^{(2)}$ pass through into a softmax layer to $\mathbb{R}^4$ for the Bury network, and a sigmoid layer for all other networks.

Given a length 500 univariate time series, Bury et al.'s networks convolve the padded

input with 50 kernels of size 12, followed by a max pooling layer of size 2. The resulting 250×50 series is fed into a first LSTM. This LSTM outputs its hidden state $h_t^{(1)} \in \mathbb{R}^{50}$ for each index between 1 and 250 to make a new 250×50 sequence. The resulting sequence is then fed into a second LSTM with hidden states $h_t^{(2)} \in \mathbb{R}^{10}$, and only outputs its last hidden state h_{250}. Then $h_{250} \in \mathbb{R}^{10}$ is passed into a sofmax classification layer with four outputs corresponding to the four possible labels available in their model.

Due to time constraints, and the fact that Bury et al. performed a grid search to find optimal hyperparameters (see [5]), all of our networks have a very similar architecture to Bury et al.'s networks. Indeed, these similarities are clear when comparing the architectures in figure 3.1. In fact, there are only two substantial differences between Bury et al.'s architecture and our own. First, because we are only interested in whether or not the input time series is approaching a tipping point, the classification layers in our models are sigmoid layers rather than softmax layers. Second, our all variable classifier takes in time series whose nine-dimensional feature vector x_t at index $1 \leq t \leq 500$ encodes the values of $m, T_i, S_i,\ 1 \leq i \leq 4$. Similarly, the feature vector $x_t \in \mathbb{R}^4$ at index $1 \leq t \leq 500$ of our proxy network encodes the values of $T_1,\ T_2,\ S_1,\ S_2$. As a result, applying 50 kernels and max pooling to each of the feature series, then concatenating the results yields an output sequence with shape 250×450 or 250×200, respectively. We, therefore, further reduce these outputs by summing over all encoded features given by a particular kernel. This averaging results in a 250×50 sequence. Hence, the output of the convolution layer in these models is the same shape as the output of the Bury model, allowing us to leave the LSTM architecture intact. For the transfer learning and circulation models, the architecture of the convolutional layers are identical to the Bury model.

Despite the near identical network architectures, our training procedure differs significantly from Bury et al.'s in several important regards. First, we do not replace any feature vectors in our time series by zeros. So, unlike the networks trained by Bury and colleagues, our networks only take length 500 time series as inputs. Second, we train most of our models for only 150 epochs rather than 1500 epochs. Although this was not our intention, each epoch for the circulation, all variables and proxy models took approximately 5 minutes to run. Furthermore, there were negligible changes to training and validation accuracy for each model after around 100 epochs. As a result, we terminated training early. However, the total training time for the transfer learning network was little over six minutes. So, we saw little reason to end training early. Third, we set the learning rate of the all variables and proxy models to 0.005 rather than 0.0005 for quicker parameter convergence. Finally, the loss function of our models was just the binary cross-entropy function. Obviously, Bury et al.'s networks use a different loss function (a generalization of the binary cross-entropy function called the sparse categorical cross-entropy; see this Keras document), as the bi-

	Models	Bury Model	Transfer Learning	Circulation	All Variables	Proxy
Model Specifications	Input shape	(1, 500)	(1, 500)	(1, 500)	(9, 500)	(4, 500)
	Kernel Size	12	12	12	12	12
	Number of Kernels	50	50	50	50	50
	LSTM 1 Hidden Space Dimension	50	50	50	50	50
	LSTM 2 Hidden Space Dimension	10	10	10	10	10
	Non Output Activation	ReLU	ReLU	ReLU	ReLU	ReLU
	Output Layer Activation Function	Softmax	Sigmoid	Sigmoid	Sigmoid	Sigmoid
	Epochs	1500	1500	1500	150	150
	Learning Rate	0.0005	0.0005	0.0005	0.005	0.005
	Cost Function	Categorical Sparse Entropy	Binary Cross-Entropy	Binary Cross-Entropy	Binary Cross-Entropy	Binary Cross-Entropy
	Optimizer	Adam	AdamW	AdamW	AdamW	AdamW

Table 3.1: Hyperparameter and other constants associated with network architecture/learning for all models relevant to our analysis.

nary cross-entropy is only applicable to binary classification problems. As a last remark, our time series data set was split into training, validation and test sets each containing 95%, 5% and 1% of our data set respectively. In this regard, our training process is similar to Bury et al.'s process.

Given these architectures and hyperparameters, all models were trained using the `keras` interface of the `TensorFlow` library in anaconda3. We further note that all of our models were optimized using the `AdamW` optimizer, with all parameters initialized with the `lecunNormal` initializer. All networks were trained on the `gpu-pr1-02` cluster provided by MFCF. The resulting training and validation accuracy and losses are shown in figure 3.2.

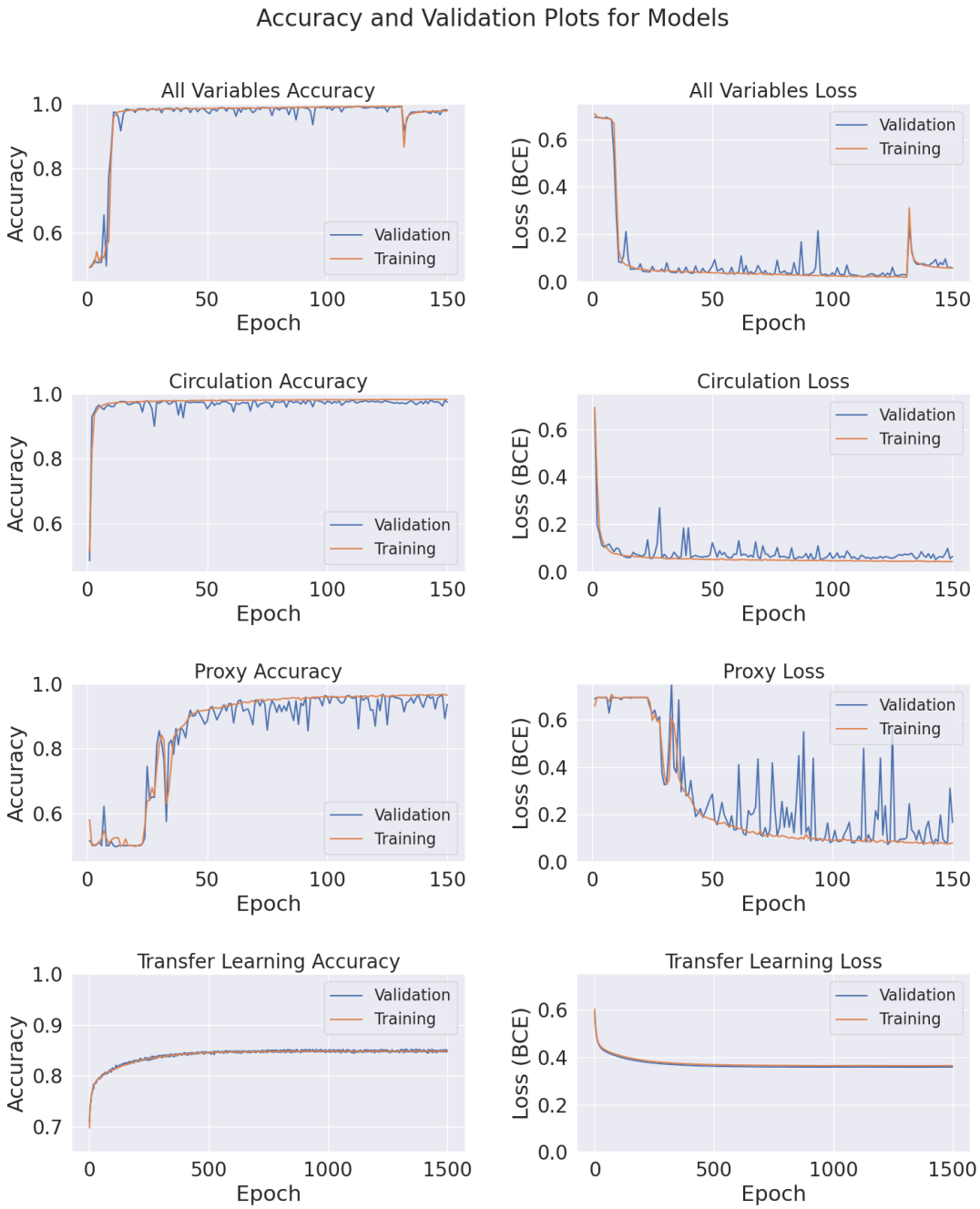

Figure 3.2: Accuracy and loss plots of the training and validation sets for the transfer learning, proxy, circulation, and all data models.

3.2 Network Performance

The network accuracy and losses given in figure 3.2 suggest that our classifiers perform very well on our FBM data set. However, to better understand how our networks perform in general, we must investigate their performance on our test set. To this end, we are interested in the accuracy, *recall* and *precision* our networks over these sets. The *recall* of a classifier with respect to a data set is just the ratio of true negatives compared to number of true and false negatives [18]. Meanwhile, the *precision* of this classifier is the ratio of true positives to the number of true and false positives. Intuitively, a classifier has high recall just when it does not underestimate the number of true positives in a data set, while it has a high precision when it avoids false positives. The accuracy, recall and precision of all our networks can be found in table 3.2.

One way of further visualizing the recall and precision of classifiers is to use a receiver operating characteristic curve (or *ROC curve*). For a binary classification network with a sigmoid output layer, we define the *discrimination threshold* τ to be the value between 0 and 1 such that the inferred label $\hat{y}$ of an input X is 1 iff $F_\Theta(X) > \tau$. In practice, $\tau = 0.5$. The ROC curve is just the curve created by plotting the true positive vs the false positives of a classifier as an implicit function of τ. The area under the ROC curve, oftentimes called the *area under the curve* or *AUC*, determines how well a classifier performs with respect to recall and sensitivity. The AUC of a classifier is 1 if it is a perfect classifier, and less that 0.5 if it is no better than randomly guessing a label. The ROC of our classifiers are shown in figure 3.4. We also plot some examples of time series which are misclassified by all of our networks in figure 3.5, to better visualize which time series "trick" our networks.

To evaluate the Bury model on our test data, we plot the *confusion matrix* of the inferred test labels in figure 3.3. The confusion matrix C of a K-label classifier is a $K \times K$ matrix such that $C_{i,j}$ entry is the ratio of how many inputs with label i had the inferred label j compared to the total number of inputs with label i [18]. We chose the confusion matrix as the Bury classifier has more labels than our own. So, a direct comparison of test set losses, accuracy, or ROC may be misleading. However, in figure 3.6 we provide plots of some (detrended) FBM time series which are correctly classified by our own networks, but not the Bury network. We also include the averaged recall and precision for the Bury model in table 3.2.

Given these results, we address our motivating questions in reverse order. First, figure 3.2 and table 3.2, it is abundantly clear that a proxy network can, in fact, determine if m will pass through a tipping point. In one sense, this result is not surprising, as it is the variables which determine whether m passes through a tipping point. Furthermore,

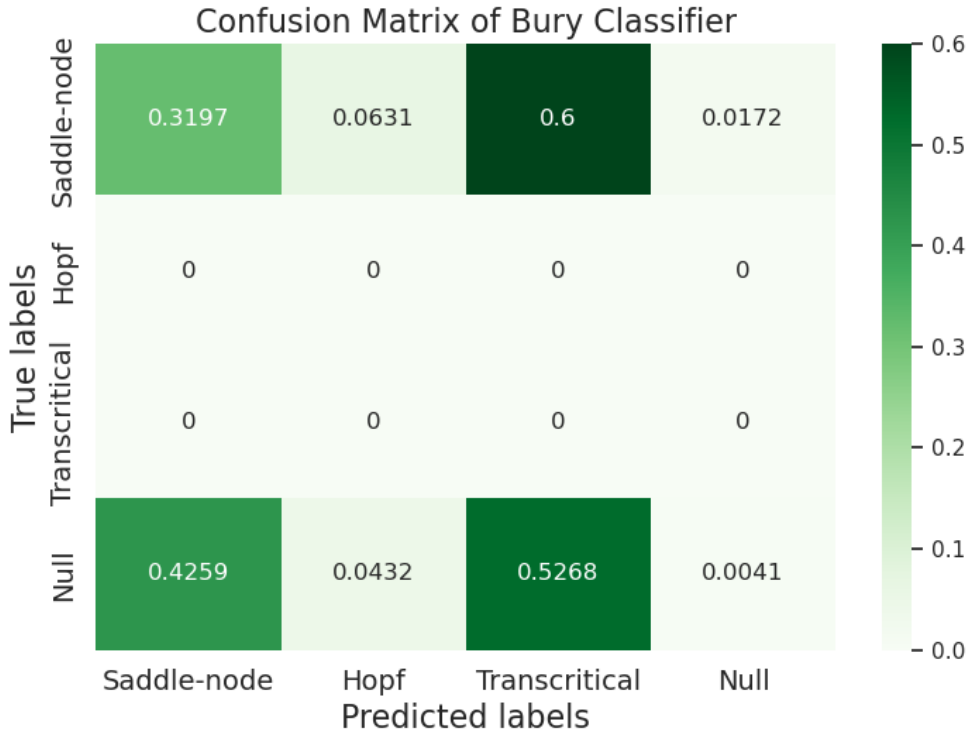

Figure 3.3: Heat map of Bury Model confusion matrix when evaluated on our FBM test set.

	Metric	Accuracy	Precision	Recall
Network	Bury Model	∼16.19%	∼ 32.25%	∼ 19.66%
	Transfer Learning	∼ 85.90%	∼ 87.04%	∼ 80.14%
	Circulation	∼98.22%	∼ 98.25%	∼ 97.75%
	All Variables	∼98.86%	∼ 98.46%	∼ 98.97%
	Proxy	∼97.72%	∼ 97.62%	∼ 97.72%

Table 3.2: Accuracy, precision and recall of our four models.

the relationship between the proxy variables and m are linear. However, the eigenspaces associated with the stable equilibrium's linearization do not directly correspond to the FBM's dynamic variables, and the proxy model does not "see" the entire FBM state. So, the result is still promising, if expected.

As for our third question, the results suggest that training a classifier on all simulated

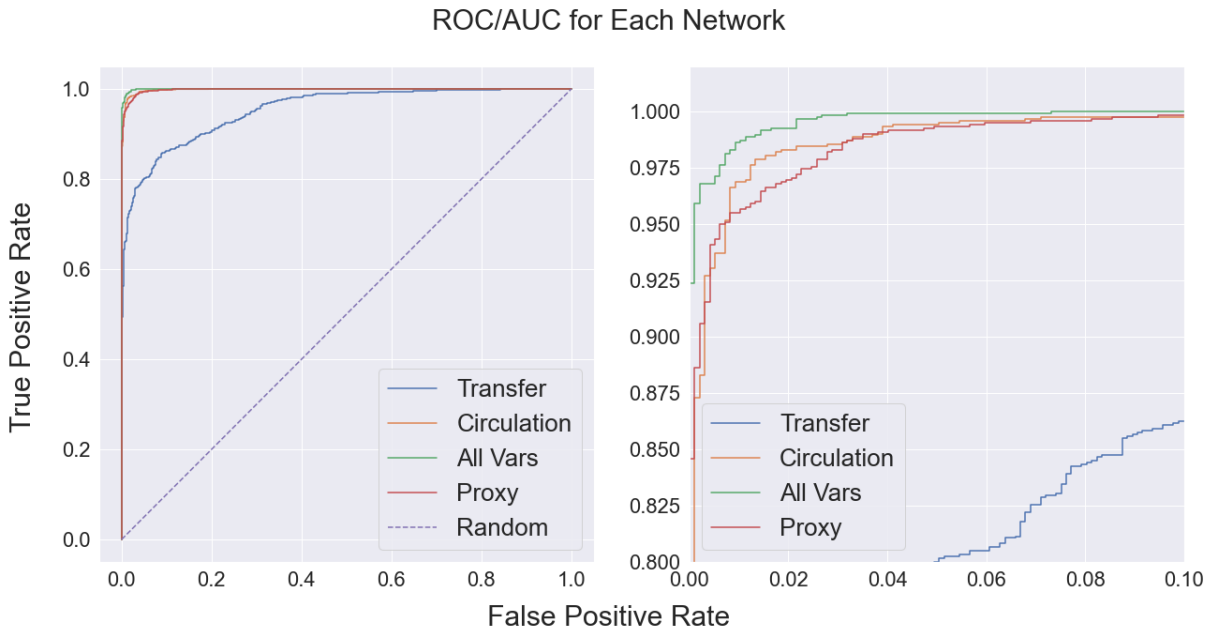

Figure 3.4: ROC curves for the transfer, circulation, all variables, and proxy networks. These networks have an AUC of Note that the left subplot is a restriction of the

FBM data provides a clear, but marginal, increase on model performance. However, this interpretation of the results misses that the circulation classifier all on its own performs tremendously well. That the all variables classifier outperforms the circulation classifier is significant, just because there are scant performance increases that could be made in the first place.

The performance of the transfer network indicates that even though the Bury model itself performs poorly on our FBM test set, minimal changes to its architecture result in a well-performing classifier. This, in turn, suggests that the EWS and nonlinearities in stochastically forced FBM time series are not unfamiliar to the Bury model, but instead appear to represent a transcritical bifurcation induced tipping point. We also note that even though the transfer network is notably less accurate than the other networks we trained, it is computationally much cheaper than training a network from scratch. As a result, we suspect there are multiple practical instances where simply replacing the classification layer of the models already trained by Bury et al. will yield a cheap and relatively accurate classifier.

Finally, we investigate the performance of the original Bury model in some detail.

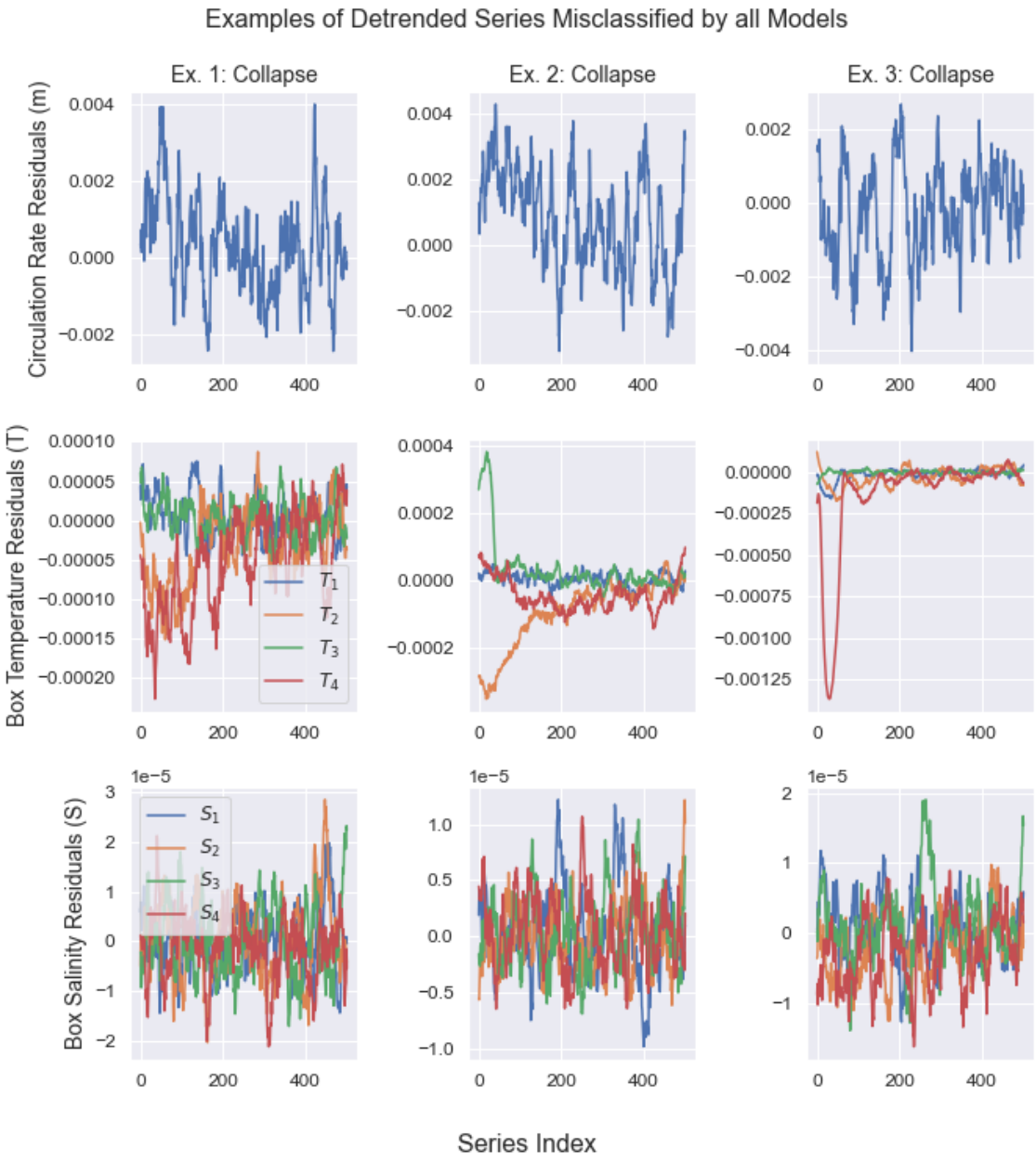

Figure 3.5: Examples of time series incorrectly classified by our own classifiers. In each case, the time series precede AMOC collapse.

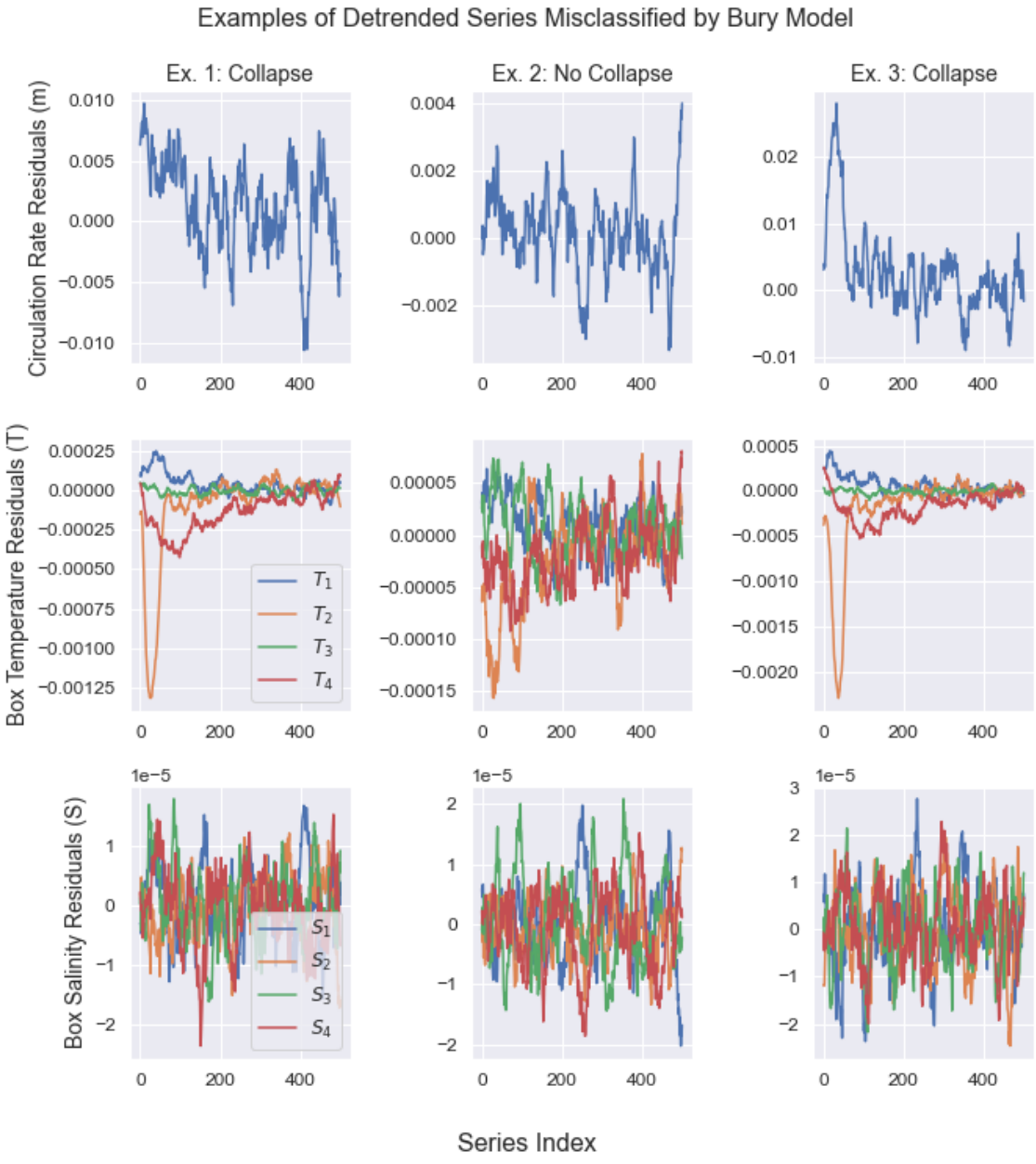

Figure 3.6: Examples of time series correctly classified by our own classifiers, but not the Bury network. Notice that the time series of m with higher variance correspond to collapse scenarios. We can also see that box temperatures have larger variance for the samples approaching a tipping point, while the variance of salinity time series are not obviously related to the sample labels.

Although it was expected that our models outperform the Bury model, the latter classifier performs worse than random. Fundamentally, we believe that much of the classifier's poor performance is explained by the identically zero eigenvalue of the FBM Jacobian. Our belief is based on the results of a small experiment we ran on the Bury model. In particular, consider the following dynamical system:

$$\dot{x} = -x^3 \tag{3.1}$$
$$\dot{y} = y^2 - \mu \tag{3.2}$$

It is clear that, like the FBM, this system exhibits a saddle-node bifurcation and that its Jacobian at the fixed points $(0, \pm\sqrt{-\mu})$ always has an identically zero eigenvalue. As a result, this system has no linear restorative component in the Jacobian's nullspace (i.e. the x-axis).

Given this system, we define $z(t) = y(t) - 5x(t)$, where the coefficient -5 was selected as the FBM parameters $\beta/\alpha \simeq -5$. We then simulated two hundred length-500 time series of $z(t) = -5x(t) + y(t)$. Here μ is randomly initialized between -20 and -1, and was either linearly forced to 0 over the 500 timesteps, or remained constant. We then forced each sample stochastically and detrended it, following the general procedure outlined in §3.1.1.

The results of this experiment, shown in table 3.2, clearly demonstrate that the model successfully infers the correct type of tipping point (or lack thereof) when given time series of y. However, the model is clearly biased towards the transcritical label when given z. So, simply taking linear combinations of system variables leads to poor model performance. We explicitly constructed system 3.2 to have similar properties to the FBM. So, these results, alongside the comparatively high performance of the transfer model, suggests that the Bury model performs poorly on the FBM data because the EWS in m is more similar to a transcritical induced tipping point, than it is a saddle-node tipping point.

	Saddle-node	Hopf	Transcritical	Null
y forced	98	0	2	0
z forced	3	0	96	1
y null	1	0	25	74
z null	1	0	95	4

Table 3.3: Counts for each type of bifurcation inferred by the Bury model. Note that the underlying system exhibits a saddle-node bifurcation

3.3 Comparing our Methods to Bury et al.

Given that our work is ultimately an extension of Bury et al.'s work, we summarize the key differences between our methods and data to their own. These differences — summarized in table 3.4 — ultimately underline that Bury et al.'s methodology considers simple parameter forcings for a very large number of simple dynamical systems. Meanwhile, our methodology considers comparatively robust parameter forcings over a single, higher-dimensional system.

We first compare the data generation procedures given by the different methodologies. Bury et al. generate their data set by simulate a large number of two-dimensional dynamical systems. In particular, the systems are third-order polynomials with randomly sampled coefficients. Given such a system, the authors first determine if the relevant system exhibits a bifurcation under variations of a single model parameter between -5 and 5. This bifurcation, should it exist, is inferred using the numerical software AUTO-OP7 (see [21].

If the system exhibits a bifurcation, the authors simulate the relevant system with additive white noise twice: once with no changes to its parameters, and once with its relevant parameter forced between -5 and 5. As described in appendix B.2, the amplitude of the additive white noise is based on the dominant eigenvalue of the relevant dynamical system about its equilibrium, with the assumption that this eigenvalue is strictly negative. For each simulation, one of resulting time series given by the two-dimensional system is detrended using LOWESS and shortened to length 500 time series. The time series is then labelled according to the type of bifurcation detected by AUTO-OP7. In this way, Bury et al.'s data generation procedure involves a very large number of very low-dimensional dynamical systems, whose dynamics under a variation of a single parameter are inferred numerically, and results in a univariate time series with many possible labels.

In contrast, our data generation procedure only involves a single, comparatively higher-dimensional dynamical system, with many set coefficients. This specificity in the dynamical model, however, leads to a wide variety of different parameter forcings, and the ability to deduce the (non-)existence of a bifurcation directly from these parameter forcings. In particular, our data set generation method involves forcing multiple parameters simultaneously, with a wide range of initial and final conditions. Like Bury et al., we run one simulation of the FBM with additive noise where the model parameters remain equal to their initial values, and one where the parameters are forced towards a local bifurcation. However, unlike Bury et al., we do not require the use of AUTO-OP7 or similar software to label our time series, as the appropriate labels can be directly deduced from the FBM. Furthermore, our white noise amplitude is not given by the dominant eigenvalue of the FBM's

stable equilibrium, as this is always 0 (see appendices B.2 and A.1, respectively). Finally, the resulting, length 500, detrended time series are multivariate rather than univariate. In this way, our our data generation procedure involves a single, higher-dimensional dynamical systems, whose dynamics under many different parameter variations are deduced analytically, and results in a multivariate time series with only two possible labels.

In terms of network training, our methodology is much more similar to Bury et al.'s own methodology. In particular, networks are CNN-LSTMs networks with one set of convolutions layers, two LSTMs, and a final, dense layer for classification. However, Bury et al.'s training procedure is more complicate and robust than our own. First, Bury et al. develop variants of their classifiers where each length 500 time series is padded by somewhere between 0 and 450 zeros. This was done so that their networks can infer whether or not time series as short as 50 timesteps precede a tipping point. Our training procedure on the other hand does not pad any inputs prior to training. Bury et al.'s procedure also trains each model for 1500 epochs, and finds optimal hyperparameters for their architecture using a grid search. Meanwhile, all of our models trained from scratch are only trained for 150 epochs, and simply use the optimal hyperparameters found by Bury et al.

3.4 Limitations & Extensions

While our results are a promising proof-of-concept about using machine learning to detect AMOC collapse, they ultimately suffer from a lack of generalizability. In a more concrete sense, our networks are trained, validated and tested entirely on four box model data. However, this provides little reason a priori to expect that our classifiers will perform well on test sets of data generated from models of intermediate complexity, or empirical data. Consequently, an important extension of our work would be to test our networks on more varied data sets. This, however, proves to be a formidable task as, in our experience, it is difficult to find data from models of intermediate complexity, or the computational resources to generate such data. Furthermore, it is unclear how the data from these models could be associated with the variables of the FBM. This limits the applicability of the proxy and all variables models, which require variable data to infer labels.

There is also a more specific sense in our results may not generalize to other AMOC data. First, the FBM has a persistent, identically zero dominant eigenvalue assuming that equilibria states exist. However, this feature is not shared by Stommel's box model, let alone all box models. Indeed, we show in appendix A.2 that Stommel's original two-box model (see [31]) does not have an identically zero eigenvalue. As a result, even if the

	Bury et al. Methodology	**Our Methodology**
Number of Systems Represented in Data Set	Thousands	One
Number of Parameters Forced in Data Generation	One	Four
Parameter Forcing	Between -5 and 5	Given by thousands of sampled initial and final conditions
Bifurcation Location	Inferred Numerically	Deduced analytically
White Noise Amplitude	Based on dominant eigenvalue	Appendix B.2
Length of Time Series Inputs	Between 50 and 500	500
Number of Time Series Features	One	Up to Nine
Number of Possible Time Series Labels	Four	Two
Number of Training Epochs	1500	150 (except transfer network)
Hyperparameter Values	Found with grid search	Set in advance

Table 3.4: Differences between our methodology, and the methodology of Bury et al.

identically zero eigenvalue of the FBM Jacobian is what "tricks" the Bury model, this property may simply reflect an idiosyncrasy of the FBM, rather than an important feature of AMOC models. As a result, future networks based on our own would benefit from being trained on a wide array of box model data. It would also be interesting to see how the Bury model would perform on two-box model data, given that the two-box model may have negative dominant eigenvalues.

Chapter 4

Conclusions

In this report, we have demonstrated that CNN-LSTM neural networks can be used to detect impending tipping points given time series of the four-box model. We have also described and implemented a four-box model time series generator, which was used to train our models. Our results show that early warning signals are contained in the time series of the circulation volume transport m alone, as well as the temperatures and salinities of the box alone. As a result, classifiers which are not directly given time series data of m can nevertheless infer whether AMOC collapse will occur with high accuracy. We also demonstrated that the Bury model performs poorly on the data set generated by the four-box model, and argued that this poor performance is ultimately related to an identically zero eigenvalue of the four-box model's Jacobian matrix.

While our result provide a good proof-of-concept for an AMOC tipping point classifier, we caution that our networks will not necessarily perform well on other simulated AMOC data. In particular, the idiosyncrasies of the four-box model may result in particular early warning signals which are not present in other box models. Future extensions of our work should therefore incorporate data from other box models when training classifiers. Furthermore, we believe that our networks should be tested on AMOC data given by models of intermediate complexity, or on nonlinear parameter forcings before making general statements about their performance. Finally, further investigations into the performance of the Bury model on other box networks may clarify why the model performed poorly on our own data set.

References

[1] David Armstrong McKay, Arie Staal, Jesse F Abrams, Ricarda Winkelmann, Boris Sakschewski, Sina Loriani, Ingo Fetzer, Sarah E Cornell, Johan Rockström, and Timothy M Lenton. Updated assessment suggests 1.5°c global warming could trigger multiple climate tipping points. *Earth and Space Science Open Archive*, page 80, 2021.

[2] Niklas Boers. Observation-based early-warning signals for a collapse of the Atlantic meridional overturning circulation, 8 2021.

[3] Chris A. Boulton, Lesley C. Allison, and Timothy M. Lenton. Early warning signals of atlantic meridional overturning circulation collapse in a fully coupled climate model. *Nature Communications*, 5(1):5752, Dec 2014.

[4] T. M. Bury, C. T. Bauch, and M. Anand. Detecting and distinguishing tipping points using spectral early warning signals. *Journal of The Royal Society Interface*, 17(170):20200482, 2020.

[5] Thomas M. Bury, R. I. Sujith, Induja Pavithran, Marten Scheffer, Timothy M. Lenton, Madhur Anand, and Chris T. Bauch. Deep learning for early warning signals of tipping points. *Proceedings of the National Academy of Sciences*, 118(39):e2106140118, 2021.

[6] Wei Cheng, John C. H. Chiang, and Dongxiao Zhang. Atlantic meridional overturning circulation (AMOC) in CMIP5 models: RCP and historical simulations. *Journal of Climate*, 26(18):7187 – 7197, 2013.

[7] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691, 2017.

[8] E. Hawkins, R. S. Smith, L. C. Allison, J. M. Gregory, T. J. Woollings, H. Pohlmann, and B. de Cuevas. Bistability of the Atlantic overturning circulation in a global climate model and links to ocean freshwater transport. *Geophysical Research Letters*, 38(10):L10605, 2011.

[9] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review - data mining and knowledge discovery, 3 2019.

[10] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, 2010.

[11] Yuri A. Kuznetsov. *Elements of Applied Bifurcation Theory*. Springer, 3 edition, 2011.

[12] Yann LeCun, Bernhard E. Boser, John S. Denker, D. Henderson, Richard Howard, W. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[13] Timothy M. Lenton, Hermann Held, Elmar Kriegler, Jim W. Hall, Wolfgang Lucht, Stefan Rahmstorf, and Hans Joachim Schellnhuber. Tipping elements in the Earth's climate system. *Proceedings of the National Academy of Sciences*, 105(6):1786–1793, 2008.

[14] Timothy M. Lenton, Johan Rockström, Owen Gaffney, Stefan Rahmstorf, Katherine Richardson, Will Steffen, and Hans Joachim Schellnhuber. Climate tipping points - too risky to bet against. *Nature*, 575(7784):592–595, 2019.

[15] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200209, 2021.

[16] Wei Liu, Zhengyu Liu, and Esther C. Brady. Why is the AMOC monostable in coupled general circulation models? *Journal of Climate*, 27(6):2427 – 2443, 2014.

[17] Valerio Lucarini and Peter H. Stone. Thermohaline circulation stability: a box model study. part I: Uncoupled model. *Journal of Climate*, 18(4):501 – 513, 2005.

[18] Kevin P. Murphy. *Probabilistic Machine Learning: An Introduction*. MIT Press, 2022.

[19] Ronald Mutegeki and Dong Seog Han. A CNN-LSTM approach to human activity recognition. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 362–366, 2020.

[20] Michael A. Nielsen. Neural networks and deep learning, 2018.

[21] Livia Owen and Eric Harjanto. A basic manual for AUTO-07P in computing bifurcation diagrams of a predator-prey model. In Mohd Hafiz Mohd, Norazrizal Aswad Abdul Rahman, Nur Nadiah Abd Hamid, and Yazariah Mohd Yatim, editors, *Dynamical Systems, Bifurcation Analysis and Applications*, pages 205–224, Singapore, 2019. Springer Singapore.

[22] Lawrence Perko. *Differential Equations and Dynamical Systems: With 241 Illustrations*. Springer, 2014.

[23] V. Petoukhov, A. Ganopolski, V. Brovkin, M. Claussen, A. Eliseev, C. Kubatzki, and S. Rahmstorf. CLIMBER-2: a climate system model of intermediate complexity, 1 2000.

[24] S. Rahmstorf. Thermohaline circulation. In Scott A. Elias, editor, *Encyclopedia of Quaternary Science*, pages 739–750. Elsevier, Oxford, 2007.

[25] Stefan Rahmstorf. On the freshwater forcing and transport of the Atlantic thermohaline circulation. *Climate Dynamics*, 12(12):799–811, May 1996.

[26] Stefan Rahmstorf, Michel Crucifix, Andrey Ganopolski, Hugues Goosse, Igor Kamenkovich, Reto Knutti, Gerrit Lohmann, Robert Marsh, Lawrence A. Mysak, Zhaomin Wang, and Andrew J. Weaver. Thermohaline circulation hysteresis: A model intercomparison. *Geophysical Research Letters*, 32(23):L23605, 2005.

[27] Marten Scheffer, Jordi Bascompte, William A. Brock, Victor Brovkin, Stephen R. Carpenter, Vasilis Dakos, Hermann Held, Egbert H. van Nes, Max Rietkerk, George Sugihara, and et al. Early-warning signals for critical transitions. *Nature*, 461(7260):53–59, 2009.

[28] Marten Scheffer, Stephen R. Carpenter, Timothy M. Lenton, Jordi Bascompte, William Brock, Vasilis Dakos, Johan van de Koppel, Ingrid A. van de Leemput, Simon A. Levin, Egbert H. van Nes, and et al. Anticipating critical transitions, 10 2012.

[29] Thomas Slawig and Kirsten Zickfeld. Parameter optimization using algorithmic differentiation in a reduced-form model of the Atlantic thermohaline circulation. *Nonlinear Analysis: Real World Applications*, 5(3):501–518, 2004.

[30] D. A. Smeed, G. D. McCarthy, S. A. Cunningham, E. Frajka-Williams, D. Rayner, W. E. Johns, C. S. Meinen, M. O. Baringer, B. I. Moat, A. Duchez, and H. L. Bryden. Observed decline of the Atlantic meridional overturning circulation 2004-2012. *Ocean Science*, 10(1):29–38, 2014.

[31] Henry Stommel. Thermohaline convection with two stable regimes of flow. *Tellus*, 13(2):224–230, 1961.

[32] Steven Henry Strogatz. *Nonlinear Dynamics and Chaos : with Applications to Physics, Biology, Chemistry, and Engineering*. CRC Press, second edition edition, 2018.

[33] Sven Titz, Till Kuhlbrodt, Stefan Rahmstorf, and Ulrike Feudel. On freshwater-dependent bifurcations in box models of the interhemispheric thermohaline circulation. *Tellus A: Dynamic Meteorology and Oceanography*, 54(1):89–98, 2002.

[34] Kirsten Zickfeld, Thomas Slawig, and Stefan Rahmstorf. A low-order model for the response of the Atlantic thermohaline circulation to climate change.

APPENDICES

Appendix A

Box Model Jacobians

A.1 Four-Box Model

Using MATLAB's symbolic Math Toolbox, we find the Jacobian $Df(x)$ of the FBM is:

$$\begin{pmatrix}
\sigma_8 - \frac{m}{V_1} - \lambda_1 & -\sigma_8 & 0 & \frac{m}{V_1} & -\sigma_4 & \sigma_4 & 0 & 0 \\
\sigma_6 & -\lambda_2 - \frac{m}{V_2} - \sigma_6 & \frac{m}{V_2} & 0 & -\sigma_2 & \sigma_2 & 0 & 0 \\
\frac{m}{V_3} - \sigma_7 & \sigma_7 & -\lambda_2 - \frac{m}{V_3} & 0 & \sigma_3 & -\sigma_3 & 0 & 0 \\
-\sigma_5 & \frac{m}{V_4} + \sigma_5 & 0 & -\frac{m}{V_4} & \sigma_1 & -\sigma_1 & 0 & 0 \\
\sigma_{16} & -\sigma_{16} & 0 & 0 & -\frac{m}{V_1} - \sigma_{12} & \sigma_{12} & 0 & \frac{m}{V_1} \\
\sigma_{14} & -\sigma_{14} & 0 & 0 & -\sigma_{10} & \sigma_{10} - \frac{m}{V_2} & \frac{m}{V_2} & 0 \\
-\sigma_{15} & \sigma_{15} & 0 & 0 & \frac{m}{V_3} + \sigma_{11} & -\sigma_{11} & -\frac{m}{V_3} & 0 \\
-\sigma_{13} & \sigma_{13} & 0 & 0 & \sigma_9 & \frac{m}{V_4} - \sigma_9 & 0 & -\frac{m}{V_4}
\end{pmatrix}$$

where

$$\sigma_1 = \frac{\beta\, k\, (T_2 - T_4)}{V_4} \qquad \sigma_2 = \frac{\beta\, k\, (T_2 - T_3)}{V_2} \qquad \sigma_3 = \frac{\beta\, k\, (T_1 - T_3)}{V_3} \qquad \sigma_4 = \frac{\beta\, k\, (T_1 - T_4)}{V_1}$$

$$\sigma_5 = \frac{\alpha\, k\, (T_2 - T_4)}{V_4} \qquad \sigma_6 = \frac{\alpha\, k\, (T_2 - T_3)}{V_2} \qquad \sigma_7 = \frac{\alpha\, k\, (T_1 - T_3)}{V_3} \qquad \sigma_8 = \frac{\alpha\, k\, (T_1 - T_4)}{V_1}$$

$$\sigma_9 = \frac{\beta\, k\, (S_2 - S_4)}{V_4} \qquad \sigma_{10} = \frac{\beta\, k\, (S_2 - S_3)}{V_2} \qquad \sigma_{11} = \frac{\beta\, k\, (S_1 - S_3)}{V_3} \qquad \sigma_{12} = \frac{\beta\, k\, (S_1 - S_4)}{V_1}$$

$$\sigma_{13} = \frac{\alpha\, k\, (S_2 - S_4)}{V_4} \qquad \sigma_{14} = \frac{\alpha\, k\, (S_2 - S_3)}{V_2} \qquad \sigma_{15} = \frac{\alpha\, k\, (S_1 - S_3)}{V_3} \qquad \sigma_{16} = \frac{\alpha\, k\, (S_1 - S_4)}{V_1}$$

where we assume that the state vector of the FBM is specified by $T_i,\ 1 \leq i \leq 4$, then $S_i,\ 1 \leq i \leq 4$, respectively. Notice immediately that the vector $(0,0,0,0,1,1,1,1)^T$ is in the nullspace of the Jacobian, and therefore an eigenvector with eigenvalue zero. It follows that the FBM Jacobian always has an identically zero eigenvalue.

A.2 Two-Box Model

Stommel's original box model is defined as follows (see [31]):

$$\begin{aligned}
\dot{T}_1 =& c(T_1^* - T_1) + |q|\,(T_2 - T_1) \\
\dot{T}_2 =& c(T_2^* - T_2) + |q|\,(T_1 - T_2) \\
\dot{S}_1 =& d(S_1^* - S_1) + |q|\,(S_2 - S_1) \\
\dot{S}_2 =& d(S_2^* - S_2) + |q|\,(S_1 - S_2)
\end{aligned}$$

Where $q = k_0(\alpha(T_2 - T_1) - \beta(S_1 - S_2))$ is the overturning rate between the two boxes, d, S_1^* and S_2^* are salinity forcing parameters, and all other variables and parameters are defined analogously to the FBM. Assuming thermally-driven AMOC (i.e $q > 0$) the Jacobian of this system is

$$\begin{pmatrix}
\sigma_3 & \sigma_5 & -\sigma_1 & \sigma_1 \\
\sigma_5 & \sigma_3 & \sigma_1 & -\sigma_1 \\
\sigma_2 & -\sigma_2 & \sigma_4 & \sigma_6 \\
-\sigma_2 & \sigma_2 & \sigma_6 & \sigma_4
\end{pmatrix}$$

where

$$\begin{aligned}
&\sigma_1 = \beta\, k_0\,(T_1 - T_2) && \sigma_2 = \alpha\, k_0\,(S_1 - S_2) && \sigma_3 = \alpha\, k_0\,(T_1 - T_2) - q - c \\
&\sigma_4 = -d - q - \beta\, k_0\,(S_1 - S_2) && \sigma_5 = q - \alpha\, k_0\,(T_1 - T_2) && \sigma_6 = q + \beta\, k_0\,(S_1 - S_2)
\end{aligned}$$

which MATLAB confirms is rank 4 in general. It immediately follows that the Jacobian does not therefore have a zero eigenvalue in general.

Appendix B

Supplementary Notes on Data Generation

B.1 The Linear Support Vector Machine Sampler

Given a data set $\mathcal{D}$ with features $x \in \mathbb{R}^n$ and binary labels, linear support vector machines (LSVMs) describe classifiers of the form

$$h(x; w) := \text{sign}(w^T x + w_0) \tag{B.1}$$

where $w_0 \in \mathbb{R}$. The classifier is optimized to minimize the hinge-loss:

$$L(\mathcal{D}; w) := \frac{1}{n} \sum_{(x,y)\in\mathcal{D}} (1 - y \cdot h(x; w))$$

We do not describe how optimal values are found — interested readers can consult §17.5 of [18]. However, we do note that given a trained LSVM, it is easy to transform the classifier into a sampler. Given a label l, randomly sample the first $n-1$ components of a sample s. It follows that $\text{sign}(w^T s + w_0) = -1$ iff:

$$\begin{aligned} & w_0 + \sum_{i=1}^{n} w_i x_i < 0 \\ \Rightarrow\ & s_n < -\frac{w_0 + \sum_{i=1}^{n-1} w_i x_i}{w_n} := \overline{s_n} \end{aligned}$$

where we assume without loss of generality that $w_n \neq 0$. As a result, if we would like to sample a point with (inferred) label ± 1, we can simply set s_n to a random value greater than or less than $\overline{s_n}$, respectively.

We use this insight to construct our sampler for parameter values. In particular, our data generating procedure requires a method for sampling $\mathcal{P}_F^{init}$ and $\mathcal{P}_F^{final}$ which guarantees that a critical transition will occur or not occur, given our desired outcome. This first requires that we specify the possible range of all components of an arbitrary vector $P = (T_1^*, T_2^*, T_3^*, F_1)^T \in \mathbb{R}^4$.

To this end, we set the range of all components to be between one fifth and five times their equilibrium values given in [34]. This choice of boundary values is not principled, but proves to to be sufficient for our purposes. We then define the grid $\Omega \in \mathbb{R}^4$ so that it has 50^4 equally spaced grid points between our parameter boundaries. After fitting a LSVM to Ω, we save the optimal w, w_0. We can then generate points by uniformly sampling F_1, T_1^*, T_2^* between their specified ranges. Then, we sample T_3^* uniformly between the decision boundary given by w, and its upper or lower bound, depending on the label passed to the sampler.

B.2 Determining White Noise Amplitude for EM Method

In the appendix of [5], Bury et al. compute white noise level for a given stochastic simulation as a function of its dominant eigenvalue. More concretely, consider the dynamical system $\dot{x} = f(x; \mu)$ with stable equilibrium x^*. The authors take the white noise amplitude σ to be used in the EM method as follows:

$$\sigma = \frac{\sqrt{2|\mathrm{Re}(\lambda_D(x^*))|}}{100} \cdot T \tag{B.2}$$

where $T \sim \mathcal{T}(0.75, 1, 1.25)$ and $\mathcal{T}$ denotes the triangular distribution. The reason the authors choose amplitudes with equation B.2 is that the variance of simulated time series with white noise amplitude σ is approximately $0.01 \cdot T$.

Notice that equation B.2 is non-zero iff $|\lambda_D(x^*)| \neq 0$. However, we know from appendix A.1 that the FBM always has $\lambda_D(x^*) = 0$, so we must modify equation B.2. To this end, suppose that $L(x^*)$ is the set of eigenvalues of $Df(x^*)$. We then define

$$\lambda_D^-(x^*) := \arg\max\{\mathrm{Re}(\lambda) \mid \lambda \in L(x^*),\ \mathrm{Re}(\lambda) \neq 0\} \tag{B.3}$$

and set

$$\sigma = \frac{\sqrt{2|\mathrm{Re}(\lambda_D^-(x^*))|}}{1000} \cdot T \tag{B.4}$$

where, again, $T \sim \mathcal{T}(0.75, 1, 1.25)$. We note that the denominator of equation B.4 is 1000, as this was the first power of 10 in which fewer than 25% of FBM simulations forced towards a local bifurcation did not pass through a tipping point before timestep 500.