UCLouvain

École polytechnique de Louvain

Investigation on EWS detection and classification through Deep Neural Networks

Author: **Adrien GUIDAT**
Supervisors: **Estelle MASSART, Pierre-Antoine ABSIL**
Reader: **Jean-Charles DELVENNE**
Academic year 2024–2025
Master [120] in Electro-mechanical Engineering

Contents

Abstract

Forecasting sudden changes in complex systems is an important but complex task. Until now multiple methods have been developped with varying accuracy. Recently, deep learning has been used for bifurcation detection and classification and has proven its capabilities to outpass the generic methods. One benchmark in this field is the work of Bury et al. which had promising results for the detection and classification of codimension-1 bifurcations, namely Hopf, fold and branch (transcritical and pitchfork) bifurcations for continuous-time systems thanks to a CNN-LSTM architecture. But transcritical and pitchfork bifurcations represent fundamentally different dynamical mechanisms and should not be grouped together. This study aims to improve the work of *Bury et al.* to classify separetly pitchfork from transcritical bifurcations.

1 Introduction

Transitions between distinct states are present in many complex systems, spanning diverse domains such as financial markets, climate dynamics , systems biology, ecological systems,... These transitions may be either catastrophic, marked by abrupt, large-scale, and often irreversible changes, or non-catastrophic, where changes are reversible. Regardless of their nature, such transitions can be triggered by slow external forcing or internal stochastic fluctuations. As systems approach critical thresholds, known as tipping or bifurcation points, fundamental changes occur in their underlying structure. This phenomenon is often referred to as a critical transition or bifurcation.

Hopefully before such a transition occur, there exist warnings that can be observed. A well studied precursor to such transitions is *critical slowing down* (CSD), which manifests as a reduced rate of recovery from perturbations. Mathematically, this is linked to the dominant eigenvalue of the system's Jacobian matrix approaching zero near the bifurcation point. The presence of CSD implies that certain statistical properties of time series, such as rising variance or autocorrelation, can serve as generic early warning signals (EWS) of incoming transitions.

However, the predictive reliability of these traditional EWS largely depend on the type of bifurcation and system studied. Their performance may be compromised by limited data availability, sensitivity to parameter choices (e.g., window size, bandwidth), and the presence of noise.

In this context, machine learning (ML) has shown to be good in identifying patterns, trends, and subtle statistical features that may be difficult to detect using traditional methods. These techniques have found widespread applications across various domains, including medicine, climate science, and nonlinear dynamics. In particular, ML has demonstrated potential in tasks such as detecting qualitative changes in system behavior, classifying different types of transitions, and providing early warnings of critical shifts in complex systems.

For instance, *Bury et al.* [1] developed a CNN-LSTM network that demonstrated strong performance in detecting early warning signals and identifying the bifurcation type across various dynamical systems, including Paleoclimate transitions and other physical systems. The network is able to detect and classify among codimension-1 bifurcations (see Section 2 on what are codimension-1 bifurcations), namely Hopf, fold and branch point bifurcations.

The primary objective in developing deep learning models in this context is to detect tipping points and classify the underlying bifurcations directly from time series data, particularly in scenarios where no prior knowledge about the system is available. In such data-driven settings, correctly identifying the type of bifurcation is crucial, as different bifurcations are associated with distinct dynamical mechanisms and post-transition behaviors. Specifically, some bifurcations may lead to abrupt and potentially irreversible shifts, which can have severe consequences for ecological systems, populations, or engineered infrastructures. Others may result in smoother, reversible transitions. Since we assume no access to the system's governing equations or parameters, this critical information must be inferred from the data alone.

In this area, pitchfork and transcritical bifurcations are usually joined into the label 'branch point bifurcations', but transcritical and pitchfork bifurcations are intrinsically different. A transcritical bifurcation involves an exchange of stability between two intersecting trajectories. While a pitchfork bifurcation implies the loss of the current branch stability, giving rise to two symmetric stable (or unstable) branches. These two bifurcation types not only represent fundamentally different dynamical mechanisms but also have distinct implications for system behavior and control. As such, they should not be treated interchangeably when studying a specific system with no prior knowledge on it.

As an example, in aircraft flight dynamics, it has been shown that specific systems may undergo transcritical, pitchfork, or other bifurcations depending on the parameters and constraints of the flight regime [2]. In such cases, accurately identifying the type of bifurcation is critical for dynamic control, as it informs how to adjust control parameters to restore or maintain stability. Therefore, the ability to distinguish between transcritical and pitchfork bifurcations has significant practical importance and can surely find applications in other domains.

Machine learning classifiers distinguishing between transcritical and pitchfork bifurcations has

only been made in discrete-time in [3], but has not yet been investigated in continuous-time, up to my knowledge. In order to fill this gap, this study aims to train a network capable of predicting bifurcations and correctly classify them between Hopf, fold, transcritical and pitchfork bifurcations.

Structure of the Report

The report is structured as follows. We begin by a theoretical overview of bifurcations, which form the foundation of our analysis. This is followed by an introduction to early warning signals (EWS), providing the necessary background and context. We then review the state of the art, with a particular focus on the contributions of Bury and collaborators. Subsequently, we present the novel elements and extensions we have developed. The next section is devoted to our results, and finally, we conclude with a discussion of these findings and potential directions for improvement.

In compliance with the guidelines of EPL and UCLouvain, ChatGPT by OpenAI was used to assist in the writing process of this master's thesis.

2 Bifurcation theory

2.1 Codimension-1 bifurcations

The behaviour of nonlinear dynamical systems is often complex and rarely yields analytical solutions. Consequently, qualitative analysis such as examining whether the system converges to an equilibrium or diverges is widely used. A system is said to exhibit *stabilizing behaviour* when its trajectory tends toward a fixed point, or equilibrium. And a *diverging behaviour* when it tends to infinity. A *bifurcation* occurs when small, continuous changes in system parameters cause a qualitative change in this long-term behaviour, such as the creation, annihilation, or destabilization of fixed points. Bifurcation points are thus critical thresholds in parameter space where such transitions arise.

These phenomena are of considerable importance in modelling sudden state transitions in various natural and engineered systems, such as species extinction, climate tipping points, or electrical circuit instabilities as they can have dramatic outcomes. This section introduces the main bifurcations studied in this work.

Specifically, we will focus the study on codimension-1 bifurcations. They refer to bifurcations occurring when a single parameter is varied (as it will be explained). In contrast, a codimension-2 bifurcation needs 2 parameters to be varied to reach the bifurcation. We present hereafter four codimension-1 bifurcations.

Saddle-Node bifurcation The saddle-node (or fold) bifurcation is among the most fundamental and well-understood. It occurs when two fixed points, a stable node and an unstable saddle, collide and annihilate as a parameter passes through a critical value. In reverse, a single unstable point splits into a stable and an unstable point.

Bifurcations are often analysed via their *normal forms*, which are simplified canonical equations capturing the local dynamics near the bifurcation. r represents the bifurcation parameter while x represents the system variable. The normal form for the saddle-node bifurcation is:

$$\frac{dx}{dt} = r + x^2.$$

Figure 1 illustrates the dynamics of this bifurcation.

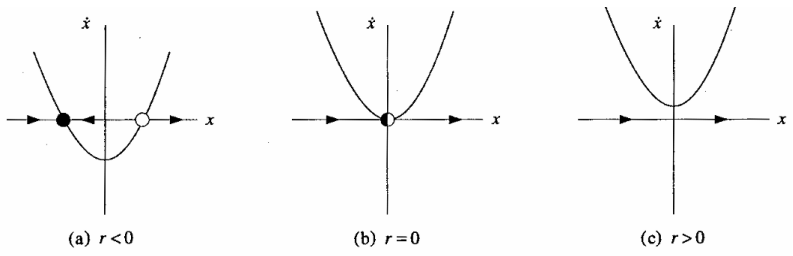

Figure 1: Illustration of the saddle-node bifurcation, reproduced from [4]. It plots the time derivative of x against x for a negative, a null and a positive bifurcation parameter. Black circles represent stable fixed point while empty circles represent unstable fixed points. The arrows show the direction of attraction for every state x

A bifurcation diagram visualizes fixed points as a function of the bifurcation parameter r. Stable branches are typically shown as solid lines, and unstable ones as Dashed lines.

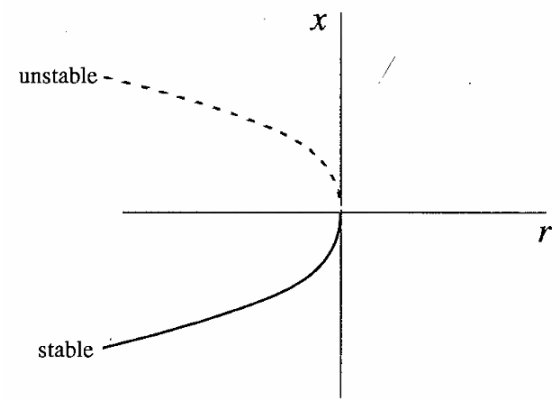

Figure 2: Bifurcation diagram of the saddle-node bifurcation, reproduced from [4]. The state variable x is plotted against the bifurcation parameter r. Bold lines represent stable branches while dashed lines represent unstable branches.

Transcritical bifurcation In a transcritical bifurcation, two fixed points exchange stability as the bifurcation parameter changes. The canonical form is:

$$\frac{dx}{dt} = rx - x^2.$$

Figure 3 shows the dynamics involved in this exchange.

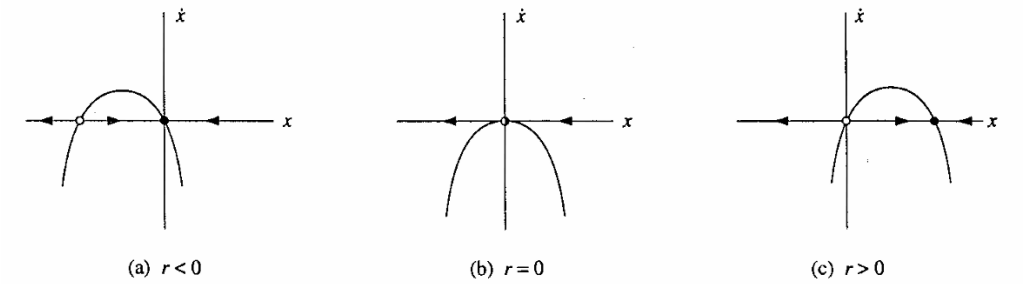

Figure 3: Dynamics of a transcritical bifurcation, reproduced from [4]. It plots the time derivative of x against x for a negative, a null and a positive bifurcation parameter. Black circles represent stable fixed point while empty circles represent unstable fixed points. The arrows show the direction of attraction for every state x

Its bifurcation diagram is depicted in Figure 4.

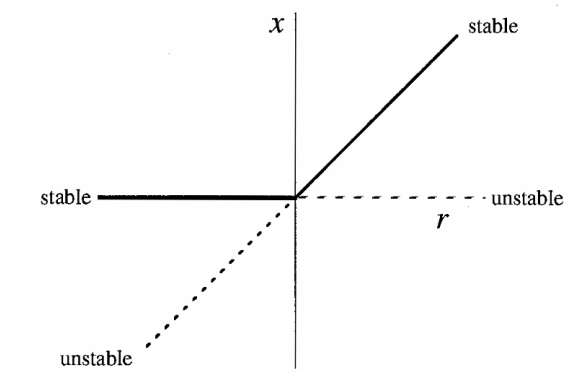

Figure 4: Bifurcation diagram of the transcritical bifurcation, reproduced from [4]. The state variable x is plotted against the bifurcation parameter r. Bold lines represent stable branches while dashed lines represent unstable branches.

Pitchfork bifurcation The pitchfork bifurcation typically occurs in systems with symmetry. There are two main types: supercritical and subcritical.

The supercritical pitchfork bifurcation has the normal form:

$$\frac{dx}{dt} = rx - x^3.$$

We note that the system is indeed symmetric around 0.

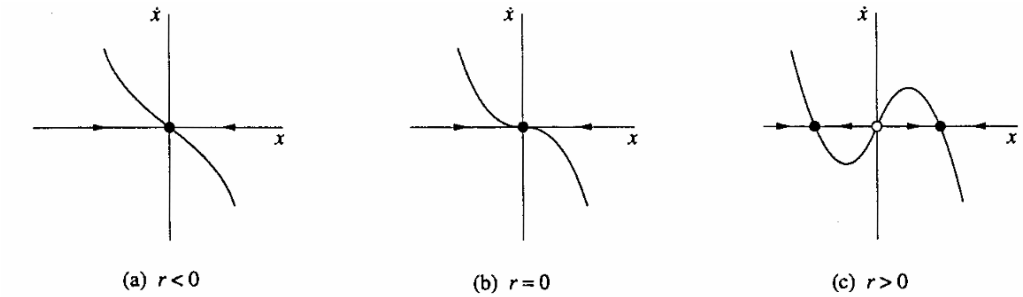

Figure 5: Supercritical pitchfork bifurcation, reproduced from [4]. It plots the time derivative of x against x for a negative, a null and a positive bifurcation parameter. Black circles represent stable fixed point while empty circles represent unstable fixed points. The arrows show the direction of attraction for every state x

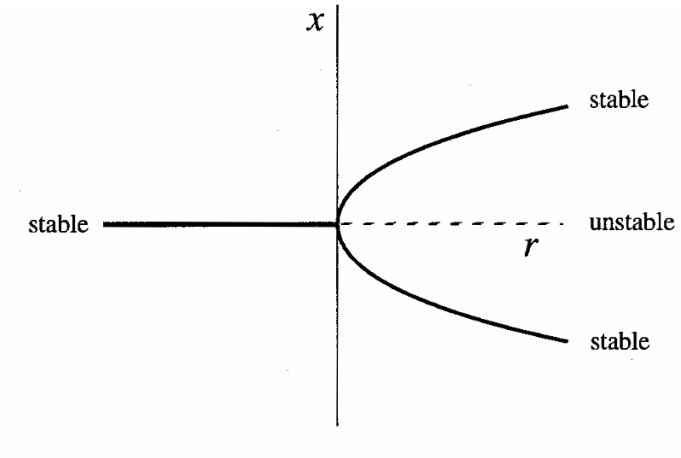

Figure 6: Bifurcation diagram of the supercritical pitchfork bifurcation, reproduced from [4]. The state variable x is plotted against the bifurcation parameter r. Bold lines represent stable branches while dashed lines represent unstable branches.

The subcritical pitchfork bifurcation differs only in the sign of the cubic term:

$$\frac{dx}{dt} = rx + x^3.$$

Here, the cubic term is destabilizing, leading to fundamentally different dynamics.

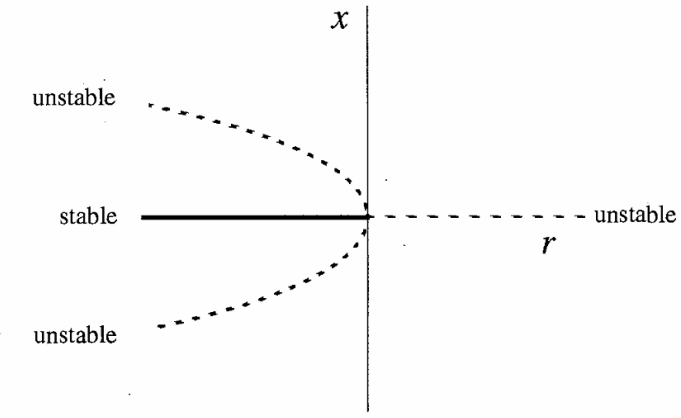

Figure 7: Bifurcation diagram of the subcritical pitchfork bifurcation, reproduced from [4]. The state variable x is plotted against the bifurcation parameter r. Bold lines represent stable branches while dashed lines represent unstable branches.

Hopf bifurcation The Hopf bifurcation occurs when a pair of complex conjugate eigenvalues of the Jacobian cross the imaginary axis, leading to the emergence or disappearance of a limit cycle, a closed trajectory in phase space.

Unlike the previous bifurcations, Hopf bifurcations require at least two-dimensional systems. They can be supercritical (creating a stable limit cycle) or subcritical (leading to unstable oscillations).

The canonical form in polar coordinates is:

$$\frac{dr}{dt} = \mu r - r^3, \quad \frac{d\theta}{dt} = \omega + br^2.$$

Here, r and θ denote the radial and angular coordinates respectively, with μ controlling stability and ω the oscillation frequency. The bifurcation parameter is here μ and not r!

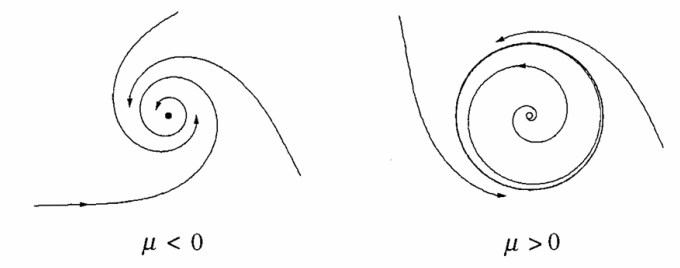

Figure 8: Supercritical Hopf bifurcation, reproduced from [4]. Black dots represent stable fixed point while empty dots represent unstable fixed points. The arrows show the direction of attraction for every state in the phase space. Dashed circles represent unstable limit cycles while bold circle represent stable limit cycles.

The subcritical variant has the radial equation:

$$\frac{dr}{dt} = \mu r + r^3.$$

The cubic term now destabilizes the dynamics, leading to abrupt transitions.

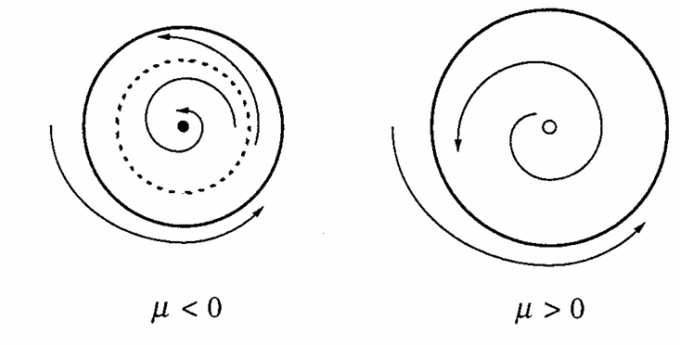

Figure 9: Subcritical Hopf bifurcation, reproduced from [4]. Black dots represent stable fixed point while empty dots represent unstable fixed points. The arrows show the direction of attraction for every state in the phase space. Dashed circles represent unstable limit cycles while bold circle represent stable limit cycles.

The Center Manifold theorem

Near a bifurcation point, the dynamics of a high-dimensional non-linear system can often be approximated by a much simpler system that captures the essential behaviour. The *Center Manifold Theorem* formalizes this idea by showing that, in a neighbourhood of a non-hyperbolic equilibrium (i.e., where the Jacobian has eigenvalues with zero real part), the system's dynamics can be reduced to a lower-dimensional surface called the *center manifold* [5].

This reduction is extremely useful because the qualitative behaviour of trajectories, such as the type of bifurcations, can be determined by analysing the system restricted to this center manifold. In many cases, the dynamics on the center manifold can be further simplified to a *normal form*. This means that a wide variety of complex systems, despite their differences, will exhibit locally the same dynamical behaviour near bifurcation points [6].

This theoretical result will later enable us to justify representing diverse and complex systems by equivalent normal forms. indeed, regardless of the system dimensionality, close to the bifurcation instability, any system can be reduced to a normal form if it exhibits such a stationary bifurcation.

2.2 AUTO

AUTO is an open-source software for continuation and bifurcation analysis for ordinary differential equations. It has been coded in Fortran around 1980 by Eusebius J. Doedel and Bart E. Oldeman among other collaborators. Its alogirthms are based on the works of *keller* [7] which will briefly be detailed in this section.

To use AUTO, one simply needs to provide the dynamical system of interest along with its governing equations. The software then performs continuation and automatically detects any potential bifurcations encountered along the solution branches. If a bifurcation is identified, AUTO reports its location (the bifurcation point), as well as its type, provided it corresponds to one of the following types: fold, Hopf, or branch point bifurcation. The category of branch points includes both transcritical and pitchfork bifurcations. Finally, AUTO offers a great freedom of use, as a lot of parameters exist (more than 50) in order to customize any part of the commputation undergone by AUTO.

Keller's mathematical foundations

We consider a non-linear system of equations depending on one parameter:

$$G(u, \lambda) = 0$$

where $u \in \mathbb{R}^n$ is the vector of unknowns and $\lambda \in \mathbb{R}$ is a scalar continuation parameter. The function $G : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is assumed to be smooth.

The goal is to trace the solution curve $(u(s), \lambda(s))$ satisfying $G(u(s), \lambda(s)) = 0$, where s is an artificial parameter. In other words, we wish to represent the equilibrium branch of u as the bifurcation parameter λ is varied.

Newton's method

Given an initial solution (u_0, λ_0) such that $G(u_0, \lambda_0) = 0$, Newton's method is typically used to find nearby solutions for slightly varied λ as long as the Jacobian matrix G_u is locally non-singular.

$$G_u(u, \lambda) = \frac{\partial G}{\partial u}$$

At each step, Newton's method solves:

$$G_u(u_k, \lambda_k)\, \delta u = -G(u_k, \lambda_k)$$

and updates $u_{k+1} = u_k + \delta u$. Recalling that $G(u_{k+1}, \lambda_k)$ must be $= 0$ [7].

Singular Points This basic approach fails when passing by a singular point. A singular point arises when the rank of G decreases, such as at a *fold*, a *branch* or a *Hopf* bifurcation point. In a formal notation a singular point happens when:

$$Rank[G_u(s_0)$$

, $\quad G_\lambda(s_0)] = N - 1$

Where G_u represents the derivative of G with respect to u while G_λ is the derivative of G with respect to λ. N is the dimension of the system.
If a singular point does not always imply a bifurcation, bifurcation points are always singular points. Indeed a bifurcation occurs when one (for fold and branch bifurcations) or more (for hopf bifurcations) eigenvalue(s) of the system crosse(s) the imaginary axis from having negative real part(s) to having positive real part(s), thus reducing the rank of the system at that point. In bifurcation analysis, one will be confronted to singular points and has to be deal with them, finding an alternative to Newton's method.

Pseudo-Arclength continuation

To overcome this difficulty, AUTO uses *pseudo-arclength continuation*, introduced by Keller [7] in 1977. Instead of fixing λ, we treat both u and λ as unknowns. Having one more unkown, we need one more equation. For this, we add a scalar constraint to parameterize the path by an arc-length, s.

Given a solution (u_0, λ_0) and its unit tangent vector $(\dot{u}_0, \dot{\lambda}_0)$, the pseudo-arclength constraint is:

$$N(u, \lambda, \Delta s) = \dot{u}_0^\top (u - u_0) + \dot{\lambda}_0(\lambda - \lambda_0) - \Delta s = 0,$$

where Δs is the arc-length step and $\dot{u}_0^\top$ stands for the transpose of $\dot{u}_0$, which we take in order to compute the dot product. This gives the equation of a plane, perpendicular to the tangent $(\dot{u}_0, \dot{\lambda}_0)$, at distance Δs from (u_0, λ_0). The trajectory of $G(u(s), \lambda(s))$ will intersect the plane under the condition that Δs is not too high, nor would be the curvature of the trajectory. We can represent ourselves the concept in Figure 10.

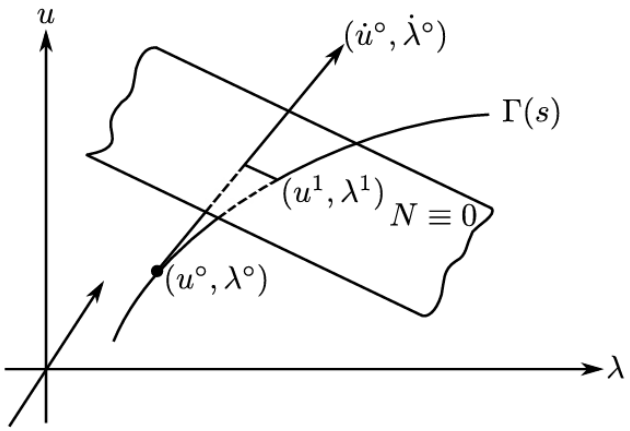

Figure 10: N-Plane visualization in the (u, λ) plane, reproduced from [7].

We now have 2 equations to solve:

$$\begin{cases} G(u, \lambda) = 0 \\ N(u, \lambda) = 0 \end{cases}$$

which has $n + 1$ equations and $n + 1$ unknowns, yielding a solvable system. Newton's method can now be applied to this regularized system, even at singular points.

Detection of bifurcations AUTO detects a potential bifurcation point when crossing a singular point (when the rank of the system is decreasing). It further identifies the bifurcation as such:

- **Fold bifurcation**: detected when the continuation direction $\dot{u}$ changes sign.
- **Hopf bifurcation**: detected when a pair of complex conjugate eigenvalues crosses the imaginary axis.
- **Branch point bifurcation**: detected when two non-trivial tangents exist at the singular point.

See Chapter 5 of *keller* [7] for more detailed explanations on how this is precisely done.

3 Early Warning Signals

The potential collapse of the Atlantic Meridional Overturning Circulation (AMOC) represents a critical threat, as it constitutes one of the key tipping elements in the Earth's climate system [8]. According to the AR6 IPCC report, a collapse within the 21st century was assessed as very unlikely (medium confidence) [9].

However, recent analysis by Ditlevsen and Ditlevsen [10] suggests a different perspective: early warning signals (EWS) derived from observed data indicate that a tipping point in the AMOC could plausibly occur between 2025 and 2095, with a 95% confidence interval. Thus taking place in the 21st century, which was not expected, significantly reducing the reaction time available. These findings highlight the importance of developing robust EWS methodologies, as they may offer critical foresight into critical transitions of any systems, including Earth's climate. But what exactly are EWS?

3.1 Introduction to tipping points

Abrupt transitions in complex systems, often referred to as tipping points, occur when the system crosses a critical threshold, shifting from one stable regime to another qualitatively distinct state. These transitions are typically fast, dramatic, and sometimes irreversible, which makes their anticipation both scientifically valuable and practically urgent. Since the publication of Scheffer et al. (2009) [11], a growing body of research has focused on identifying early indicators of such regime shifts, collectively termed *early warning signals* (EWS). These indicators provide statistical or dynamical evidence that a system is losing resilience and approaching a bifurcation point.

Despite their theoretical appeal, EWS are difficult to operationalize in real-world systems. This challenge is coming from the complexity of natural and socio-ecological dynamics, which are often noisy, high-dimensional, and only partially observable.

Mechanisms of Tipping points The theory underpinning early warning signals predominantly comes from dynamical systems analysis, particularly the loss of stability of equilibrium states. Tipping can occur through several distinct mechanisms [12]:

- **Bifurcation-induced tipping (B-tipping)**: A parameter crosses a critical value, destabilizing the current equilibrium and forcing a transition to an alternative attractor.
- **Noise-induced tipping (N-tipping)**: Random fluctuations push the system across the boundary of the basin of attraction, even if the underlying parameters are constant.
- **Rate-induced tipping (R-tipping)**: Rapid changes in system parameters prevent the system from adapting quickly enough, causing a transient loss of stability.

In real-world systems, tipping often results from a combination of mechanisms. For example, slow parameter changes may bring a system closer to a bifurcation, while stochastic perturbations expedite the transition. If N-tipping is likely to be unpredictable in its pure form, its probability to take place increases as the system is driven to the bifurcation point, and so, this increase in probability can be detected through particular EWS.

Available software Through time, many packages have been created in order to automatize the computation of EWS. Each implements different EWS or are written in different languages, depending on the objective of the package. I present here two of the existing packages:

- **'earlywarnings':** One of the earliest package created to compute EWS, coded in R in 2012 by Dakos et al. [13].
- **'ewstools':** More recently (2023) Bury developed a package in python to compute generic EWS and spectrum EWS [14]. Python packages being by the way less usual, the majority being written in R.

We can classify EWS among multiple characteristics, the type of data used (temporal, spatial,...), the method employed (fitting models, analysing patterns,...) but the most frequent classification is whether the method is based on critical slowing down or not (CSD-based or non CSD-based).

3.2 Critical slowing down and related indicators

One of the most well-known early warning signal is called *critical slowing down* (CSD). It refers to the tendency of a system to recover more slowly from small disturbances as it gets closer to a tipping point. This happens because the stabilizing feedbacks that normally push the system back to equilibrium become weaker.

A helpful way to picture this is through the *well analogy*. Imagine the system as a well and the state of the system as a ball sitting in the well. If the well has steep sides, the ball quickly rolls back to the bottom after being pushed, meaning the system is stable. But as the sides of the well become shallower, the ball rolls back more slowly, and if the slope flattens enough, the ball might not return at all. Instead, it could escape entirely, ending up in a completely different state. This loss of restoring force leads to slower dynamics and increases the chance of a sudden shift. We can visualize this in Figure 11

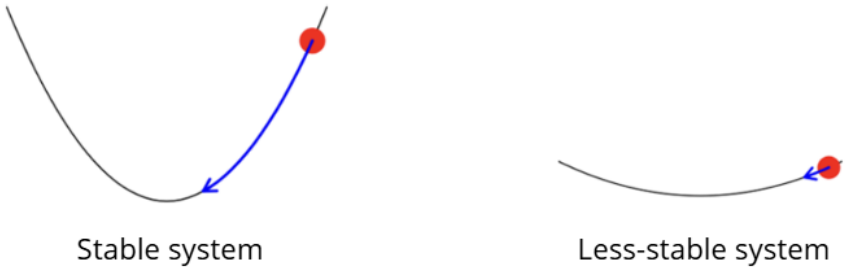

Figure 11: Representation of the well analogy. In left, a stable system (black), when the state of the system (red) is not at its equilibrium point, the restoring force is high (blue). In right, as the system is less stable, the restoring force is smaller.

Mathematically, this process is due to the dominant eigenvalue of the Jacobian matrix of the system approaching zero from below, reducing the rate at which the system recovers from perturbations [15]. As we usually don't have access to the equations driving the dynamics, we need to infer CSD from something else than the eigenvalues. Luckily, CSD manifests itself in observable features of the system's time series, for example:

- **Decreased return rate**: Perturbations dissipate more slowly as the potential well becomes shallower.
- **Increased autocorrelation (AR(1))**: The state of the system at time t becomes more correlated with its previous state at $t-1$ as we approach the bifurcation. This value tends to 1 as approaching the bifurcation.
- **Rising variance**: As the restoring forces have weakened, this allow the system to explore a broader region of state space, and so the variance increases.

Analysing the return rate requires a well-defined perturbation and knowing when the equilibrium state of the system has been reached, neither of which are easy to get, reducing its applicability.
Caution must also be taken while considering variance, as a change in its value can also be due to other factors than approaching the bifurcation, such as how the system is forced externally.

Other CSD-based methods either estimate a statistical property of the time series or try fitting a statistical model. On the other hand, there exist indicators that do not make profit of the CSD effect, the so-called non CSD-based indicators.

Non CSD-based indicators

While CSD provides a solid theoretical foundation, its assumptions, such as small perturbations around a fixed point, often fail under strong noise or fast-changing conditions. Consequently, researchers have developed *non CSD-based* indicators:

- **Skewness**: As the system approaches a tipping point, the temporal distribution of observed states may become asymmetric, skewed toward the alternative state. This metric can thus take both negative or positive values.
- **Flickering**: Strong stochasticity can temporarily push the system into the basin of attraction of an alternative state. An increasing frequency of such excursions may signal proximity to a critical transition.
- **Fitting a model**: Consists in fitting the simplest dynamical model with a tipping point (normal form) and testing its likelihood compared to a model without a tipping point.

The indicators cited here are only a small part of all the indicators that have been developed in the literature but do represent the most-used ones according to Dakos et al. [16].

As these approaches do not make the assumption of a system in equilibrium slowly driven to bifurcation, nor ask for clean time series, unlike the CSD-based methods, the range of application of EWS is quite large, with many fields using EWS in research.

Since the mid-2000s, especially following influential studies in climate and ecology like Held and Kleinen in 2004 [17], the use of EWS has expanded to various fields that we can group under 5 domains, taken by decreasing importance according to the literature review from 2004 to 2022 of *Dakos et al.* [16].

- **Ecology** : Forests, lakes, dryland ecosystems are common targets for EWS detection. See [18] for a fish population study.
- **Health sciences** : Applications range from disease emergence to cancer detection. See [19] for a study about cell decision-making processes.
- **Climate science** : Paleoclimate reconstructions and modern climate records provide data-rich contexts for testing EWS methods. See [17] for an example.
- **Economics, finance:** EWS are used to study financial crashes, bankrupts,... See [20] for a financial crisis study.
- **Physical sciences** : An heterogeneous category including critical transitions in physical and engineered systems. See [21] for the study of a thermoacoustic system.

It is of interest to know which EWS metrics are the most-used, or studied in the literature. The literature review of Dakos et al. [16] recorded in the period 2004-2022 65 different early warnings. About 75% of those were found to be CSD-based, while the remainder employed non-CSD mechanisms. It is worth noting that only 21 of these indicators were used more than once, underscoring the experimental and context-dependent nature of EWS research.
The most widely used indicators include variance, autocorrelation, and skewness. The other indicators were used depending on the domain.

Overall, EWS shows considerable promises: about 70% of empirical applications considered in [16] successfully detected an impending transition. However, this success rate must be interpreted cautiously. Many of these studies were conducted in controlled or relatively simple systems, and real-world complexity may obscure or weaken EWS signals. Moreover research on EWS, like every other research field, may be subject to publication bias and/or selective reporting.

3.3 Challenges

As I mentionned, a number of conceptual, methodological, and practical challenges complicate the effective application of EWS to real-world systems. This section synthesizes the main limitations, drawing attention to areas where further refinement and research are required.

A first complication arises when multiple interacting drivers act on the system. In many studies, a unidirectional driver is assumed to push the system toward a tipping point. However, empirical

evidence shows that when several forces operate simultaneously (potentially in opposing directions), early warning indicators may produce contradictory signals. Even when acting alone these drivers yield similar EWS trends, their interactions may create nonlinear feedbacks and ambiguity in signal interpretation [22].

A second complication comes from the fact that detecting EWS requires sufficiently long and high-resolution time series. For example, in climate science, some of the longest datasets are derived from satellite products dating back to the 1970s. Yet, these series are often composites of data from multiple sensors with differing characteristics. As sensors degrade or are upgraded, changes in radiometric sensitivity or noise levels may affect the signal-to-noise ratio (SNR), leading to misleading trends in indicators such as autocorrelation (AR(1)) or variance. These changes can mimic or obscure the signatures of impending transitions [23].

Most theoretical work on EWS has focused on univariate systems. However, real-world systems are inherently multivariate, involving complex interactions among many variables. Identifying which variable(s) to monitor is complicated, as the dynamics driving the transition may not be observable directly. Some approaches conceptualize the system as a network of interacting components, tracking changes in network topology as potential signs of reduced resilience [24].

Alternatively, dimensionality reduction methods such as Principal Component Analysis (PCA) can extract dominant patterns from multivariate datasets, summarizing system behaviour through a few leading components [17]. However, the interpretability and sensitivity of these metrics to EWS remain under investigation.

It remains unclear which early warning signals (EWS) are most appropriate for a given application, and the relationships between different indicators are still not fully understood. Exploring these interdependencies could provide valuable insights. In this context, composite metrics, combining multiple generic indicators, have been proposed to enhance predictive accuracy [25].

One of the more complex challenges in monitoring EWS is the potential of cascading tipping events, where the transition of one subsystem initiates a sequence of transitions in connected systems. Such interdependencies may be invisible when monitoring systems in isolation. For example, a regime shift in ocean circulation might trigger feedbacks in atmospheric or ecological subsystems, with little to no early warning detectable locally [26]. Effective detection would thus require networked monitoring strategies that can anticipate cross-system feedbacks.

3.4 Identifying the bifurcation type

While early warning signal indicators can help anticipate the onset of a bifurcation, they typically do not provide information about the nature of the transition. This aspect is significantly more challenging and has received less attention in the literature. To the best of my knowledge, there are currently two main approaches for inferring the bifurcation type: one based on spectral analysis, particularly through the examination of changes in the power spectrum, and the other using deep learning methods.

The power spectrum has emerged as a promising tool for identifying bifurcations in time-series data. Studies have explored its use in detecting fold bifurcations and developed spectral-based EWS, such as the spectral ratio and spectral exponent [27]. These are collectively referred to as spectral EWS.

Novel EWS In their study [15], *Bury et al.* introduce two new spectral early warning indicators derived from analytical expressions of the power spectrum:

- S_{max} This indicator grows asymptotically as $\frac{\sigma^2}{\mu^2}$, where σ represents the amplitude of external noise and μ the distance to the bifurcation. In contrast, the traditional variance scales as $\frac{\sigma^2}{\mu}$, which increases more slowly near the bifurcation. This different scaling behavior makes S_{max} more resilient to fluctuations in σ, such as those arising in systems influenced by multiplicative or temporally variable noise.

- **AIC Weights** This indicator evaluates the relative simplicity of several candidate models fitted to observed data, aiming to identify the type of bifurcation represented by the empirical power spectrum. These candidate models include the theoretical spectral forms associated with fold and Hopf bifurcations, along with a flat spectrum representing a white noise null model.

These spectral early warning signals are intended to be used jointly: S_{max} acts as an indicator of an approaching bifurcation, while the AIC weights help determine the likely type of transition by selecting the most parsimonious spectral model.

Limitations First, the spectral early warning signals introduced differentiate only between oscillatory and non-oscillatory bifurcations, without distinguishing between specific types within each category. For instance, both supercritical and subcritical Hopf bifurcations produce similar spectral signatures, despite leading to fundamentally different dynamical outcomes. A supercritical Hopf bifurcation typically results in a smooth and reversible transition to small-amplitude oscillations, whereas a subcritical Hopf bifurcation can trigger a sudden, irreversible shift to a distant limit cycle [15].

Secondly, the analysis in their work is restricted to transitions induced by local codimension-1 bifurcations. However, many complex systems undergo stability loss through global bifurcations, which do not exhibit critical slowing down and thus evade detection by conventional early warning signals. In these cases, alternative indicators based on other dynamical characteristics may be necessary.

Machine learning approaches trained on time series data offer a novel framework for classifying bifurcation types directly from observations. This study focuses on developing and extending such methods. Machine learning will thus be later described in Section 4.2.

Now will be presented some mathematical expressions for the most generic EWS indicators and spectral indicators.

3.5 Mathematical analysis

We can extract from the work of *Bury et al.* [15] the mathematical expressions of some generic EWS indicators evaluated for the codimension-1 bifurcations this study concentrates on (which are detailed in Section 2), namely the pitchfork, the transcritical, the fold and the Hopf bifurcations, for continuous-time.

Pitchfork, transcritical and fold bifurcations (continuous-time): We use σ to denote the amplitude of the additive white noise of the signal, λ represents the eigenvalues of the system, t the time and τ is the time-delay/lag parameter used to compute the autocorrelation. Finally ω is the angular frequency on which is evaluated the power spectrum. The expressions are given by:

- **Dominant eigenvalue:** $\lambda \in R, \lambda \to 0^-$
- **Variance:** $-\frac{\sigma^2}{2\lambda}$
- **Autocorrelation(τ):** $e^{\lambda|\tau|}$
- **Power spectrum(ω):** $\frac{\sigma^2}{2\pi}(\frac{1}{\omega^2+\lambda^2})$

Hopf bifurcation (continuous-time): For the Hopf bifurcation, the eigenvalues can be further described by their real part μ and their imaginary part ω_0.

- **Dominant eigenvalue:** $\lambda_{1,2} = \mu \pm i\omega_0$, $\mu \to 0^-$
- **Variance:** $-\frac{\sigma^2}{2\lambda}$
- **Autocorrelation(τ):** $e^{\mu|\tau|}cos\omega_0\tau$
- **Power spectrum(ω):** $\frac{\sigma^2}{4\pi}(\frac{1}{(\omega-\omega_0)^2+\mu^2} + \frac{1}{(\omega+\omega_0)^2+\mu^2})$

Visualization To illustrate the behaviour of early warning signals across different dynamical regimes, I computed and plotted the most widely used EWS indicators for time series simulated from the canonical forms of the four codimension-1 bifurcation types investigated in this study: fold, Hopf, pitchfork, and transcritical. These time series were generated from stochastic simulations of canonical systems using the Euler Maruyama method and noise drawn from a triangular distribution centered in 0.01 with upper and lower bounds of 0.0125 and 0.0075 respectively. (more informations on the simulations method in Section 5). To process the data, the `ewstools` Python package developed by M. Bury [14] was used, which implements the methods formulas presented above in Section 3.5.

As expected, the Akaike Information Criterion (AIC) weights, used here to assess the spectral likelihood of different bifurcation models, perform well in identifying Fold and Hopf bifurcations but surprisingly not for detecting the null bifurcation. Apart from that, the other indicators such as variance, lag-1 autocorrelation and S_{max} show a general increase as the systems approach the bifurcation point (here, fixed at $t = 500$), reflecting the phenomenon of critical slowing down. In contrast, we do not observe such trends for the null bifurcation. We also understand that we cannot a priori deduce the type of bifurcation encountered solely from these indicators (except with AIC weights), which supports our choice of investigating deep learning in the pursue of classification objectives.

Regarding the magnitude of the indicators, it is important to note that for the variance, it is not the absolute value that signals an approaching bifurcation, but rather the relative increase over time. The magnitude of the variance itself depends on the characteristics of the system, the amplitude of the noise, and other factors.

For the lag-1 autocorrelation (AC), values are bounded between -1 and $+1$. Here again, it is the upward trend of the indicator that may signal a loss of resilience. Nevertheless, the absolute value does carry some meaning: the closer the lag-1 AC is to $+1$, the stronger the temporal correlation in the signal.

Similarly, both skewness and the spectral maximum (S_{max}) are unbounded indicators. Their exact values depend on the system and data preprocessing steps. Therefore, as with the previous indicators, it is the trend rather than the absolute magnitude that is typically used to infer proximity to a critical transition. We note however that the skewness indicator only yields a clear increasing trend for the Hopf bifurcation.

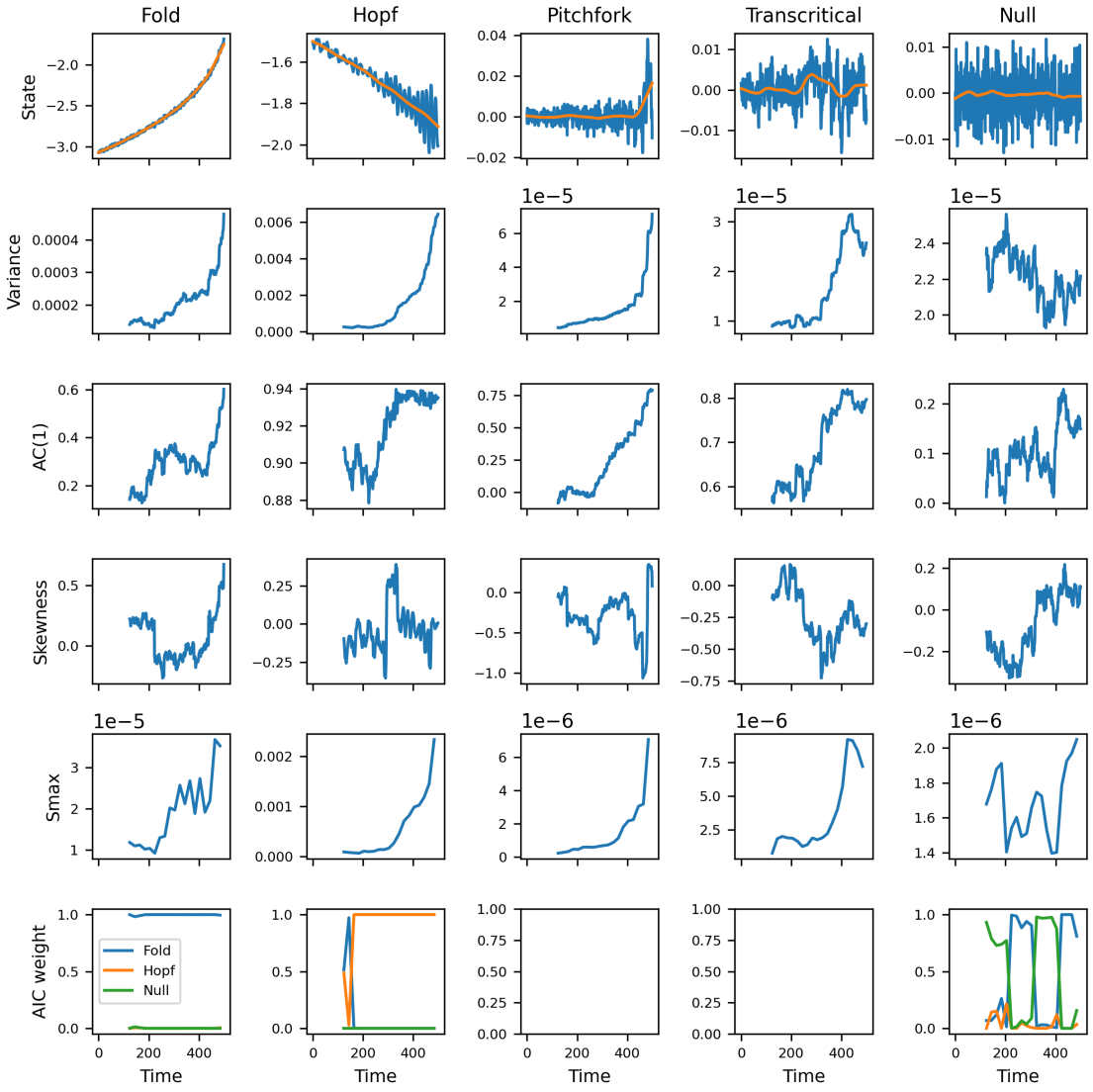

Figure 12: Time series from the canonical forms of the fold, Hopf, pitchfork and transcritical bifurcations and a null bifurcation serie and some generic EWS evaluated on them (variance, autocorrelation, skewness, Smax and AIC weights)

4 State of the Art

The recent success of neural networks in time series classification has brought interest in applying machine learning (ML) techniques to the detection of critical transitions. This approach is particularly promising because many systems approaching tipping points exhibit common early warning signals, such as critical slowing down. This universality allows deep learning models to be trained on large amounts of synthetic data and then transferred to empirical scenarios with plausible performance. [16]

Specifically, deep learning architectures that incorporate convolutional layers have been shown to outperform traditional EWS-based methods (e.g., variance, autocorrelation) in both simulated and real-world contexts [28]. In particular, these models have proven capable not only of detecting the onset of a transition but also of inferring the type of bifurcation underlying the observed dynamics [1].

A milestone in this field is the work of *Bury et al.* [1]. They developed a network capable of classifying bifurcations from time series among fold, Hopf and 'branch point' bifurcations. 'Branch point' bifurcation representing pitchfork and transcritical bifurcations grouped together. If the latter are grouped together, they are indeed not the same and represent different dynamics leading to different consequences on the next state. As a recall from Section 2, transcritical bifurcations imply an exchange of stability between two trajectories while pitchfork bifurcations imply the creation of two symmetric trajectories.

The task of distinguishing trancritical from pitchfork bifurcations is not easy as they have very similar canonical forms, see Section 2. *Bury et al.* succeeded in training a network capable of distinguishing them at better than random but only for discrete-time data [3].

The present study extends the framework introduced by *Bury et al.* [1], which presented a deep learning model capable of classifying fold, Hopf, and branch point bifurcations in continuous-time dynamical systems. Our objective is to enhance this framework by enabling the model to further discriminate between pitchfork and transcritical bifurcations. To achieve this, we incorporate the data generation methodology proposed in [3], which effectively distinguished pitchfork and transcritical bifurcations in a discrete-time setting using canonical normal forms, into the continuous-time classification framework developed in [1].

In particular, we adopt the data generation and network training pipelines from the continuous-time framework developed in [1], and augment the training set with time series generated from canonical systems exhibiting transcritical and pitchfork bifurcations. This enriched dataset allows us to train a unified model capable of distinguishing the four principal codimension-1 bifurcation types (fold, Hopf, transcritical, and pitchfork) from continuous-time data, thereby synthesizing the contributions of both [1] and [3]. Within this context, the methodological pipeline established in the original study by *Bury et al.* will be presented, followed by a brief theoretical overview of the implemented machine learning network type.

4.1 Original Architecture

The work of Bury et al. [1] which trains a network able to detect and classify incoming bifurcations on time series is divided in 3 major parts. The generation of the training data, the training of the network, and the testing of the network. All the code mentioned is available at [29].

Training set generation

General structure Since generating 500,000 time series is computationally intensive and time-consuming, the data generation process is parallelized to improve efficiency. This is achieved through two Bash scripts. The first, `run_single_batch`, executes the generation of a single batch of time series. The second, `submit_multiple_batch`, dispatches multiple batch jobs across the available GPUs and subsequently collects the generated data into a single directory, as represented in Figure 13.

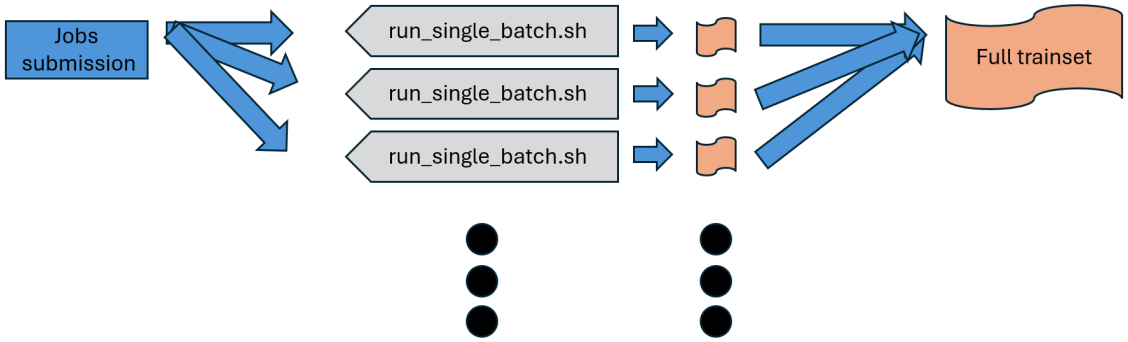

Figure 13: Multiple batches submissions and assembly. The datasets are represented in orange, the bash files in grey and the flow by blue arrows. The black dots mean "and so on".

We begin by describing the procedure used to generate a single batch of data, summarized in Figure 14. The generation process consists of several key steps, outlined below.

System generation Random two-dimensional dynamical systems are first generated and subjected to an initial simulation to test for convergence. Convergence is considered achieved when the difference between two successive steps falls below a threshold of 10^{-8}. Only converging systems proceed to the next stage. This step is implemented in `gen_model.py`.

Bifurcation identification For each system, continuation simulations are performed along its parameters using the AUTO software to detect and classify potential bifurcations. AUTO outputs the location and type of each bifurcation encountered. This step is carried out in `stoch_sim.py`.

Time series generation Using the detected bifurcation points, the system is simulated again by varying (or keeping constant) the bifurcation parameter up to the identified transition. This produces both *forced* simulations (where the system undergoes a bifurcation) and *null* simulations, where no bifurcation occurs. The resulting time series thus represent both bifurcating and non-bifurcating trajectories. This process is handled by `sim_model.py`.

Data labeling and splitting Each time series is labeled according to its bifurcation type and assigned to one of three dataset partitions: training set (95%), validation set (4%), or test set (1%). This is implemented in `to_traindata.py`.

Residual computation Finally, the residuals of the time series are computed using the `ewstools` Python package [14]. These residuals are required for training the neural network and are generated in `compute_resids.py`.

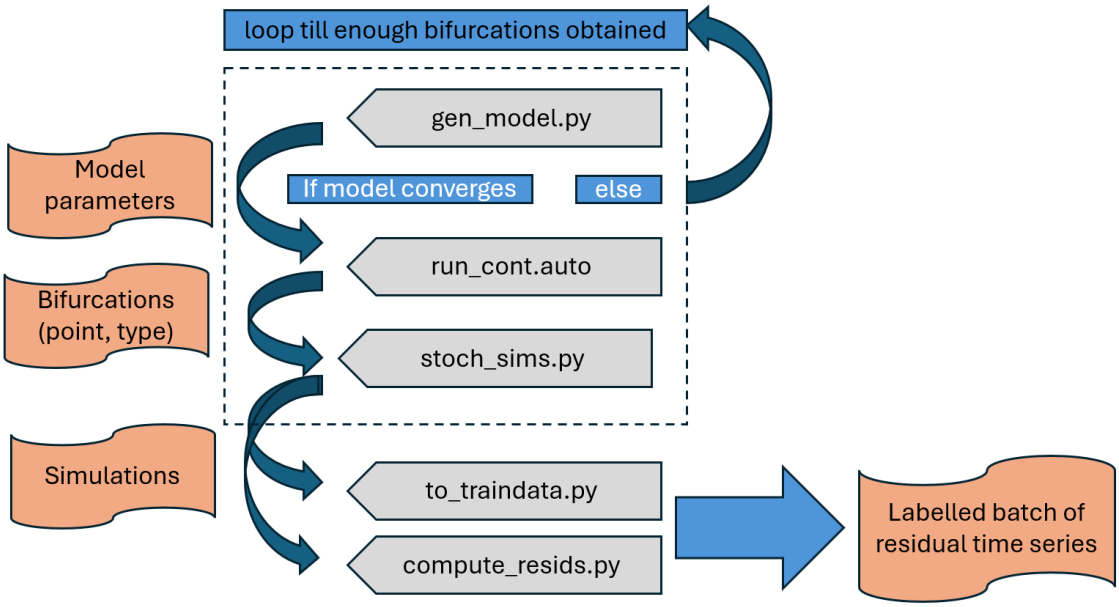

Figure 14: Single batch generation structure. The python files are represented in grey, the data exchanged in orange and the flow is showed by blue arrows.

The most important part of the described pipeline in this study is the model generation scheme, from which they further extract bifurcations and so time series. In this context, their method will be further described.

Generation scheme

To enable both the detection of approaching tipping points and the classification of their underlying nature, the training set must include representative time series for each bifurcation type. In their approach, Bury et al. [1] focus on codimension-1 bifurcations, namely fold, Hopf, and branch point bifurcations. To enable the model to also recognize the absence of critical transitions, time series from systems with no bifurcations are also included.

System generation The strategy for data generation relies on simulating randomly constructed two-dimensional dynamical systems of the form:

$$\frac{dx}{dt} = \sum_{i=1}^{10} a_i p_i(x, y),$$

$$\frac{dy}{dt} = \sum_{i=1}^{10} b_i p_i(x, y),$$

$$\text{With } p(x, y) = (1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3).$$

The parameter vector $\boldsymbol{\theta} = (a_1, \ldots, a_{10}, b_1, \ldots, b_{10}) \in \mathbb{R}^{20}$ is sampled from a standard normal distribution $\mathcal{N}(0, 1)$. To reduce computational complexity and introduce sparsity, half of the coefficients are randomly set to zero. Moreover, to promote the emergence of bounded dynamics, all coefficients corresponding to cubic terms are replaced by the negative of their absolute value:

$$\theta_i \leftarrow -|\theta_i| \quad \text{for } i \in \{7, 8, 9, 10, 17, 18, 19, 20\}.$$

Finally, the initial state vector $\mathbf{s}_0 = \begin{pmatrix} x(0) \\ y(0) \end{pmatrix}$ is sampled from a normal distribution:

$$\mathbf{s}_0 \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, 2^2 \cdot \mathbf{I}_2 \right).$$

This formulation allows for the systematic generation of diverse non-linear systems, some of which undergo bifurcations while others remain dynamically stable, thereby generating a diverse training set.

An important remark is that, although the underlying dynamical systems are two-dimensional, only a single dimension (the x-dimension) is exported as a time series. Consequently, the neural network is trained solely on one-dimensional inputs. This choice ensures that the trained model can be applied not only to one-dimensional systems but also to higher-dimensional systems by selecting the relevant dimension of interest. In contrast, a model trained on multi-dimensional inputs (e.g., 2D time series) would not be directly applicable to lower-dimensional systems. However, this approach raises a natural question: is exporting the x-dimension always a correct choice?

In fact, there is no guarantee that the bifurcation manifests itself in the x-dimension. In some systems, it may instead arise primarily in the y-dimension. In such cases, exporting only the x-dimension may result in a time series that lacks clear bifurcation signals, potentially misleading the classifier, as the time serie will still be labeled with the bifurcation arising in y-dimension. This scenario can occurs in practice. However, since both dimensions are typically coupled through cross-terms, and since we can expect at least half of the bifurcations to emerge in the x-dimension, the current strategy of exporting only the x-dimension represents a reasonable compromise.

A field for improvement would be to dynamically select the exported dimension as the one in which arises the bifurcation

Training the model

In their work on machine learning as an early warning signal, Bury *et al.* explored multiple deep learning architectures, including residual networks, functional convolutional networks, and recurrent neural networks. They discovered that the hybrid CNN-LSTM architecture yielded the best results in terms of precision and recall on their training set [1]

The model architecture is so, based on a hybrid CNN-LSTM structure, composed of one convolutional layer followed by two stacked LSTM layers. Dropout is applied to mitigate overfitting, and max pooling is used to reduce dimensionality and computational complexity. The output of these different layers is passed through a dense layer that produces class probabilities. Further details on the role and configuration of each layer are provided in Section 4.2. The overall architecture is illustrated in Figure 15.

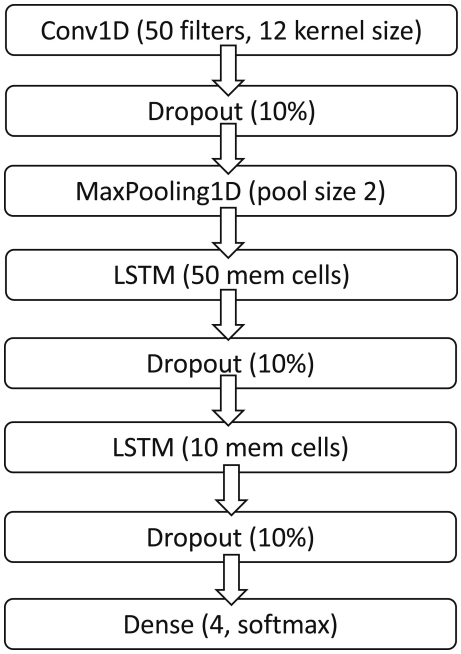

Figure 15: Neural Network Architecture of Bury's work, extracted from [1]

The hyperparameters used in the model were selected through a grid search procedure and are summarized in Table 1. The model was trained using the Adam optimizer in conjunction with the sparse categorical cross-entropy loss function. The final dense layer employs a softmax activation to produce probability distributions over the output classes.

Table 1: Model hyperparameters

Parameter	**Value**
CNN layers	1
LSTM layers	2
Dropout rate	0.10
Pooling size	2
Filters (CNN)	50
Kernel size (CNN)	12
Memory cells (LSTM layer 1)	50
Memory cells (LSTM layer 2)	10
Learning rate	0.0005
Batch size	1000
Epochs	1500

Testing the model

Empirical test set To validate the deep learning classifier on real-world cases, three sources of empirical time series were used:

Sedimentary records of the Mediterranean Sea. High-resolution records of oxygen dynamics were analyzed from three sediment cores, covering eight anoxic events. Variables such as the concentration of molybdenum (Mo) and uranium (U) served as proxies for anoxic and suboxic conditions. Data preprocessing followed Hennekam *et al.* [30].

Thermoacoustic instability experiments. Experiments were conducted in a horizontal Rijke tube, where an increase in heating voltage triggered a subcritical Hopf bifurcation leading

to thermoacoustic instability. Data included 19 forced trajectories (with voltage ramps) and 10 steady-state trajectories.

Paleoclimate transitions. Seven major paleoclimate shifts previously studied for EWS were considered. After linear interpolation to ensure equidistant spacing, EWS analysis was performed.

Overall, these empirical datasets test the classifier across a wide range of natural critical transitions, from environmental shifts to physical system instabilities. The deep learning classifier successfully predicted an upcoming bifurcation, outperforming generic EWS (variance and lag-1 ac), in two out of the three empirical cases. In the paleoclimate dataset, its performance was only comparable to lag-1 autocorrelation. Furthermore, the classifier correctly identified the bifurcation type in two out of three cases, with the only confusion occurring in the thermoacoustic system, where it slightly favored a Hopf bifurcation while still assigning significant probability to a fold bifurcation [1].

Artifical test set

They also test the network on a serie of dynamical systems perturbed with Gaussian white noise, simulated via the Euler–Maruyama method.

Fold bifurcation Detection of fold bifurcations was tested using the May harvesting model:

$$\frac{dx}{dt} = rx\left(1 - \frac{x}{k}\right) - \frac{hx^2}{s^2 + x^2} + \sigma\xi(t),$$

where x is the population biomass, r is the intrinsic growth rate, k is the carrying capacity, h is the harvesting rate, s controls the nonlinearity of harvesting, σ is the noise amplitude, and $\xi(t)$ is a Gaussian white noise process. Parameters were set to $r = 1$, $k = 1$, $s = 0.1$, $h \in [0.15, 0.27]$, $\sigma = 0.01$, with a fold bifurcation occurring around $h = 0.26$.

Hopf and transcritical bifurcations Hopf and transcritical bifurcations were tested using the Rosenzweig–MacArthur consumer–resource model:

$$\frac{dx}{dt} = rx\left(1 - \frac{x}{k}\right) - \frac{axy}{1 + ahx} + \sigma_1\xi_1(t),$$

$$\frac{dy}{dt} = \frac{eaxy}{1 + ahx} - my + \sigma_2\xi_2(t),$$

where x and y represent resource and consumer densities, and r, k, a, e, h, m are ecological parameters with noise amplitudes σ_1, σ_2. A transcritical bifurcation occurs at $a \approx 5.60$ and a Hopf bifurcation at $a \approx 15.69$. The parameter a was varied to simulate both null and forced trajectories.

Higher-dimensional model: SEIRx model A higher-dimensional example was considered with a stochastic SEIR model (S, E, I and R being the compartments of the disease), coupled to vaccination behavior dynamics:

$$\begin{aligned}
\frac{dS}{dt} &= N(1 - x) - \beta\frac{SI}{N} + \sigma_1\xi_1(t),\\
\frac{dE}{dt} &= \beta\frac{SI}{N} - (\gamma + \delta)E + \sigma_2\xi_2(t),\\
\frac{dI}{dt} &= \gamma E - (\delta + \nu)I + \sigma_3\xi_3(t),\\
\frac{dR}{dt} &= \nu I - \delta R + \sigma_4\xi_4(t),\\
\frac{dx}{dt} &= \rho x(1 - x)(\lambda + I + (2x - 1)) + \sigma_5\xi_5(t),
\end{aligned}$$

where S, E, I, R are the compartments of the disease, x is the proportion of pro-vaccine individuals, λ is the perceived vaccine risk, and $\xi_i(t)$ are independent Gaussian noise processes. A transcritical

bifurcation occurs as λ increases, leading to a collapse in vaccine sentiment and a resurgence of endemic infection.

In all models, noise perturbs the system dynamics, and the DL classifier is trained to detect the type of impending critical transition based on the behavior of residuals.

The classifier again outperformed both variance and lag-1 autocorrelation as an early warning signal across all systems, except for the SEIRx model, where all EWS exhibited poor performance. Moreover, it correctly classified all bifurcations in the three examined systems [1].

4.2 Machine Learning Architecture

This study and the study of *Bury et al.* employ a deep learning model combining convolutional neural networks (CNN) and long short-term memory (LSTM) layers to classify univariate time series.

LSTM LSTM network is regarded as the state-of-the-art time series forecasting model owing to its capability of learning long-term temporal dependencies through the design of gated units integrating activations of sigmoid and hyperbolic tangent functions. [31].

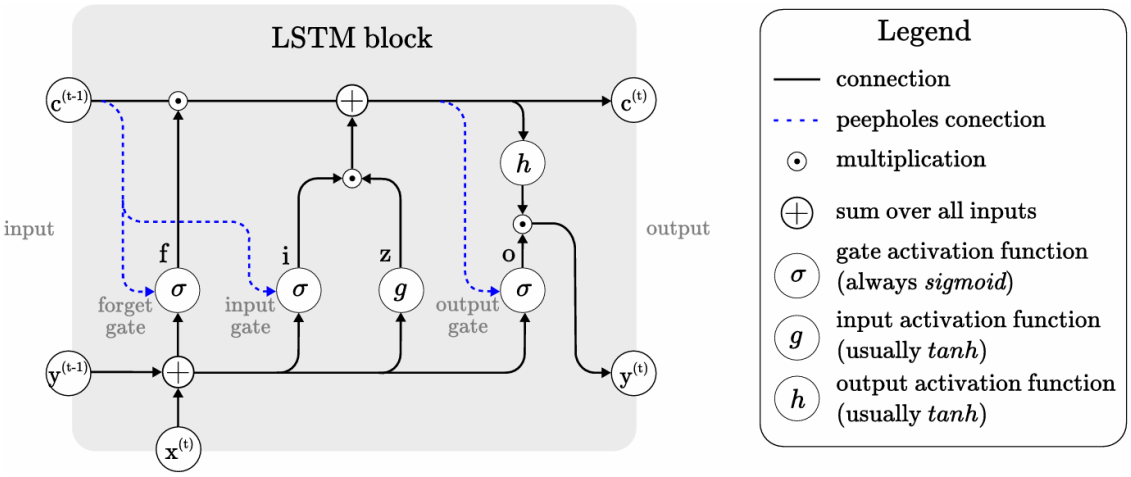

Figure 16: LSTM cell representation extracted from [32]

The LSTM model is a powerful recurrent neural system. The forward pass in this recurrent neural system is described below [32] and is depicted in Figure 16.

- **Block input** $z^{(t)} = g(W_z x^{(t)} + R_z y^{(t-1)} + b_z)$
- **Input gate** $i^{(t)} = \sigma(W_i x^{(t)} + R_i y^{(t-1)} + p_i \odot c^{(t-1)} + b_i)$
- **Forget gate** $f^{(t)} = \sigma(W_f x^{(t)} + R_f y^{(t-1)} + p_f \odot c^{(t-1)} + b_f)$
- **Cell** $c^{(t)} = z^{(t)} \odot i^{(t)} + c^{(t-1)} \odot f^{(t)}$
- **Output gate** $o^{(t)} = \sigma(W_o x^{(t)} + R_o y^{(t-1)} + p_o \odot c^{(t)} + b_o)$
- **Block output** $y^{(t)} = h(c^{(t)}) \odot o^{(t)}$

Where W_a, R_a and p_a are the weights associated respectively with $x^{(t)}$, $y^{(t-1)}$, $c^{(t-1)}$, while b_a stands for the bias weight vector for $a \in [z, i, f, o]$; z standing for the block input, i the input gate, f the forget gate and o the output gate. $\odot$ denotes point-wise multiplication.

In the above equations, σ, g and h denote point-wise non-linear activation functions. The logistic sigmoid $\sigma(x) = \frac{1}{1+e^{1-x}}$ is used as a gate activation function, while the hyperbolic tangent $g(x) = h(x) = \tanh(x)$ is often used as the block input and output activation function.

CNN Since AlexNet *(Krizhevsky et al.)* [33] won the ImageNet competition in 2012, deep CNNs have seen a lot of successful applications in many different domains. Motivated by the success of these CNN architectures in these various domains, researchers have started adopting them for time series analysis [34]. A convolution can be seen as applying and sliding a filter over the time series.

A general form of applying the convolution for a centered time stamp t is given in the following equation:

$$C_t = f(\omega * X_{t-L/2:t+L/2} + b)| \ \forall \ t \in [1, T]$$

where C denotes the result of a convolution (dotproduct $*$) applied on a univariate time series X of length T with a filter (kernel) ω of length L (the kernel size), a bias parameter b and a final non-linear function f such as the Rectified Linear Unit (ReLU) for example. Multiple filters (M) can be applied in a CNN layer. [34].

Convolutional Neural Networks (CNNs) are hybridized with LSTM to extract the fundamental features from the input sequence automatically and generate memory to recognize the same patterns appearing at different times. As such, CNN-LSTM architectures excel at pattern recognition and sequence prediction [1], which is exactly what we seek for bifurcation prediction and classification through time series analysis.

Implementation example

Entry Let's consider the architecture depicted in Figure 17, composed of 1 CNN layer, 1 LSTM layer, with Dropout, Pooling and a final Dense layer. We will go through the flow to get an idea of how the network is trained. The input signal x, which represents a time serie, is initially processed through a one-dimensional convolutional layer.

Dropout Dropout layers are placed after the CNN and the LSTM layers to drop a certain portion of the trained neurons (E%) in order to avoid overfitting the training data.

Pooling Pooling layers are used to reduce the complexity of the problem by reducing the size of the problem, but thus reducing the resolution. It works by sliding a dynamical window of size S and applying a mathematical function to it. It can be averaging, taking the maximum value, the minimum,...

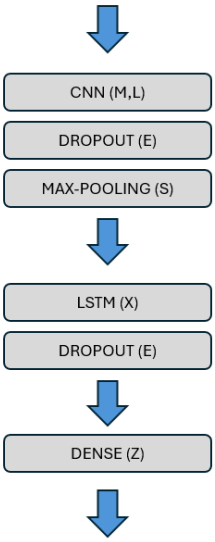

Figure 17: Architecture diagram

LSTM The output of the pooling layer is then passed into a LSTM layer comprising X memory units. At each time step t of the serie, the LSTM units update.

Dense layer The final classification is carried out by a fully connected dense layer comprising Z output units, each corresponding to one of the Z target classes. This layer requires an activation function to convert raw outputs into class probabilities. A common choice is the softmax activation function, which computes the class probabilities as follows:

$$p_i = softmax(b_i + \sum w_{ij} x_j)$$

where w_i and b_i are the weights and bias terms associated with class i, and x denotes the output from the LSTM layer. j loops on the size of x.

Training Finally, the network needs to be optimized. A generic optimizer would be the Adam one. The latter needs a loss function which quantifies the error between the predicted output of a model and the actual output. Categorical cross entropy is an example, which is evaluated as:

$$L = \sum t_i log(y_i)$$

With y_i the probability vector output from the model and t_i the one-hot encoded truth label. [19] The training process is governed by three key hyperparameters: the learning rate, the batch size, and the number of epochs. The learning rate determines the magnitude of the parameter updates at each iteration of the optimization algorithm. The batch size specifies the number of training samples used per iteration. Finally, training proceeds for a fixed number of epochs, where each epoch corresponds to one complete pass through the entire training dataset, so consisting of multiple iterations.

5 Extension of Bury's approach

To be able to distinguish transcritical from pitchfork bifurcations, which is the objective of this study, while still accurately identifying fold and Hopf bifurcations, we chose to partially reuse the existing training set from Bury, specifically the time series containing fold, Hopf, and null bifurcations. To this dataset, we added time series exhibiting transcritical and pitchfork bifurcations.

For the generation of these additional time series, we employed the pipeline developed by Bury but, rather than generating random two-dimensional systems, we generated canonical-form systems to enable precise assessment of the bifurcation type. Indeed, since AUTO does not differentiate pitchfork from transcritical bifurcations, grouping them instead under the general label of branch point bifurcations, it can no more be used to detect and classify these bifurcations. Consequently, the approach inspired by Bury's discrete-time framework [3] was adopted, where systems are generated from canonical forms to ensure accurate identification of both bifurcation points and types.

System generation Following the approach of *Bury et al.*, I simulated two-dimensional dynamical systems characterized by polynomial terms up to third order. This choice is motivated by the fact that the minimal system capable of representing the pitchfork, the transcritical, the fold and the Hopf bifurcations is this one. Indeed, the normal form for the Hopf bifurcation in Cartesian coordinates is given by:

$$\frac{dx}{dt} = \mu x - y - x(x^2 + y^2),$$
$$\frac{dy}{dt} = x + \mu y - y(x^2 + y^2).$$

In this study, we focus exclusively on the x-dimension of the system as it is the only one which will be exported to later generate the corresponding time series.

The ODE in x is given as a reminder by:

$$\frac{dx}{dt} = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy + a_6 y^2 + a_7 x^3 + a_8 x^2 y + a_9 xy^2 + a_{10} y^3.$$

Transcritical bifurcations generation I remind the canonical form of the transcritical bifurcation:

$$\frac{dx}{dt} = rx - x^2. \tag{1}$$

I therefore impose: $a_1 = a, a_2 = \mu, a_7 = 0$ (to differentiate from pitchfork), $a_7 = rnd$ rnd being a random number drawn from the interval $[0.2; 2]$.

Pitchfork bifurcations generation I also remind the canonical form of the pitch bifurcation

$$\frac{dx}{dt} = rx \pm x^3. \tag{2}$$

I therefore impose: $a_1 = a, a_2 = \mu, a_4 = 0$ (to differentiate from transcritical), $a_7 = \pm rnd$

The bifurcation parameter μ, the constant a and the constant rnd are randomly drawn as such:

$$\mu \sim \mathcal{U}(-2,\ 2), a \sim \mathcal{U}(-1,\ 1), \texttt{rnd} \sim \mathcal{U}(0.2,\ 2.0)$$

Class separability To ensure better class separability, I explicitly disabled shared terms that might blur the distinction between the two bifurcation types. Specifically, in pitchfork systems, the x^2 term was set to zero, while in transcritical systems, it is the x^3 term that was set to zero.

Sparsity To set a significant portion of the non-canonical parameters (the ones not directly involved in the canonical form of the system) to zero, I randomly assigns 60% of these non-canonical terms to zero.

Three generation schemes were elaborated for this study, which are presented here:

- **Model 1**: Canonical forms are implemented without class separability nor with sparsity.
- **Model 2**: Canonical forms are implemented, with class separability and sparsity.
- **Pure canonical forms**: Canonical forms are implemented, but every other terms are set to 0, leading to pure canonical systems."

Each generated system was first simulated to verify its numerical stability and convergence. If it passed this test, it was then submitted to `AUTO` for bifurcation analysis. I retained only systems that exhibited a bifurcation point identified by AUTO as a *branch point* (i.e., pitchfork or transcritical), and extracted the exact bifurcation point. The reason is twofold. First, as the systems generated in Model 1 and Model 2 are not pure canonical forms, we do not always obtain a branch point bifurcation, justifying the use of AUTO to only retain branch point bifurcations.

It is important to note that nothing guarantees that an expected transcritical bifurcation (expected from its canonical form), will not be a pitchfork bifurcation or vice-versa. This issue will be later addressed in Section 7.

Secondly, AUTO yields the exact bifurcation point, which we could only suppose to be at 0, which would have been the case if the system was a pure canonical form.

Labelling and splitting Finally, the newly generated time series exhibiting pitchfork and transcritical bifurcations were labelled accordingly and appended to the existing dataset from *Bury*, which contains time series corresponding to fold, Hopf, null, and branch point bifurcations. The combined dataset was then split into training, validation, and test sets using the same proportions as before: 0.94/0.04/0.01.

During the data loading phase for network training, the original time series labelled as branch point bifurcations were excluded to prevent redundancy, which could otherwise bias the model or hinder its ability to distinguish between pitchfork and transcritical bifurcations.

Data labelling and partitioning were handled using two CSV files:

- `labels.csv`: assigns each time series a bifurcation class label.
- `groups.csv`: assigns each time series to a set between the training, the validation, or the test set.

Unlike the original setup where `groups.csv` was generated during data generation, I opted to create this file during data loading. This architecture provided greater flexibility, allowing us to dynamically select how many examples of each bifurcation type to load, thus controlling the class balance during training.

The code described in this section and on which this study relies is available at [35].

5.1 Training the model

Bury *et al.* explored multiple deep learning architectures, including residual networks, functional convolutional networks, and recurrent neural networks for their classification objectives. They found that the CNN-LSTM architecture yielded the best results in terms of precision and recall on their training set [1].

In this study, although the objective shifts slightly from classifying among null, fold, Hopf, and branch point bifurcations to further distinguishing between pitchfork and transcritical bifurcations, I initially chose to retain the CNN-LSTM architecture.

The reason is that, although my training set differs from that of Bury, the broader modelling context remains similar enough to justify reusing the original architecture as a strong baseline.

Nevertheless, we cannot guarantee that an other architecture would not yield better performances at distinguishing pitchfork bifurcations from transcritical ones. This leaves ground for further research.

Hyperparameters Bury *et al.* optimized their hyperparameters (including convolutional filter sizes, LSTM memory units, dropout rates,...) via grid-sweep methods [1].

In this study, I retained Bury's original parameters as a reasonable starting point, considering the similitude of the two tasks and sets. But again, there is no guarantee that those hyperparameters would be optimal for this specific task.

A promising area of improvement lies in optimizing these parameters with a focus on maximizing classification accuracy specifically for pitchfork and transcritical bifurcations. This could be achieved not only through traditional grid search but also via more advanced metaheuristic techniques such as for example, the Grey Wolf Optimizer (GWO), which has shown competitive performance in neural architecture optimization [31].

5.2 Testing the model

Two specific models were used to evaluate the network's ability to distinguish between pitchfork and transcritical bifurcations:

Rotating hoop (Pitchfork) Consider a rotating circular hoop of radius r with a bead of mass m sliding along it, as shown in Figures 18 and 19.

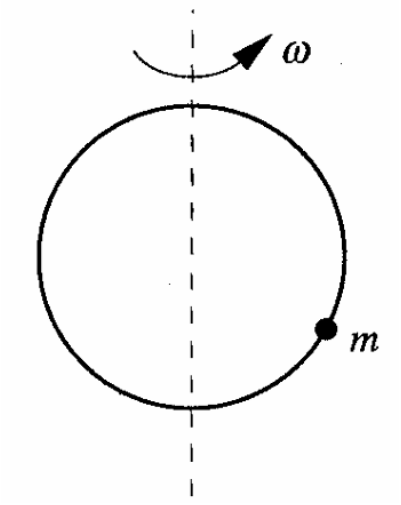

Figure 18: Rotating hoop (black circle) with bead (with mass m) reproduced from [4]. The hoop rotates at speed ω.

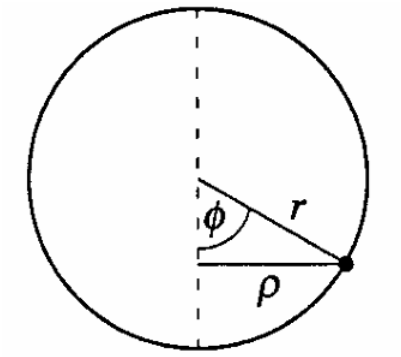

Figure 19: Rotating hoop (black circle) with bead (with mass m) reproduced from [4]. r denotes the radius, ρ the distance between the bead and the vertical axis of the hoop.

Under classical mechanical assumptions detailed in page 61 and 62 of *Strogatz* [4], the system's dynamics reduce to:

$$b\dot{\theta} = mg\sin(\theta)\left(\frac{r\omega^2}{g}\cos(\theta) - 1\right)$$

As shown in Strogatz [4], this system undergoes a supercritical pitchfork bifurcation at $\frac{r\omega^2}{g} = 1$. Physically, this corresponds to the appearance of two symmetric equilibrium positions once the hoop rotates fast enough.

Two types of simulations were conducted on this system. Forced simulations, where the bifurcation parameter ω was varied from an initial condition $\omega = 2$ to a value behind the bifurcation $\omega = 6$ so that we force the system to transition, the bifurcation taking place at $\omega = 4.43$. The other type of simulations undertaken were null simulations, where ω was fixed at a value before the bifurcation here $\omega = 4$, so that no bifurcation arised. These simulations were both of importance in order to have positive cases (with bifurcation) and negative cases (without bifurcation) to later establish metrics capturing the accuracy of the predictions.

Rozenzweig MacArthur consumer resource model To test transcritical bifurcation detection, I used a consumer resource model from Rozenzweig MacArthur [36].

$$\frac{dx}{dt} = rx\left(1 - \frac{x}{k}\right) - \frac{axy}{1 + ahx} + \sigma_1 \xi_1(t), \tag{3}$$

$$\frac{dy}{dt} = \frac{eaxy}{1 + ahx} - my + \sigma_2 \xi_2(t), \tag{4}$$

In this model, r represents the intrinsic growth rate per capita of the resource population x, while k denotes its environmental carrying capacity. The parameter a captures the consumer y's attack rate on the resource, and e defines the efficiency with which consumed resources are converted into consumer biomass. The term h corresponds to the handling time, and m is the mortality rate per capita of the consumer. Parameters σ_1 and σ_2 specify the intensities of stochastic fluctuations, with $\xi_1(t)$ and $\xi_2(t)$ denoting independent Gaussian white noise processes.

The system is investigated using the fixed parameter values $r = 4$, $k = 17$, $e = 0.5$, $h = 0.15$, $m = 2$, $\sigma_1 = 0.01$, and $\sigma_2 = 0.01$. Under these settings, the system undergoes a transcritical bifurcation at $a = 5.60$, and a Hopf bifurcation at $a = 15.69$. For simulations involving the transcritical transition, we consider a baseline scenario (null trajectories) with $a = 2$, and explore perturbed dynamics (forced trajectories) for values of a ranging from 2 to 6.

5.3 Computational resources

The generation of the training dataset and the training of deep learning models required access to high-performance computational resources, particularly GPUs with sufficient memory and processing capabilities.

Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region.

System architecture More specifically, computations were carried out on the *Manneback* cluster, hosted at UCLouvain and managed under the CÉCI consortium. The cluster is composed of an heterogeneous architecture including:

- **CPUs**: AMD EPYC Rome and Milan, as well as Intel Xeon Cascade Lake processors.
- **GPUs**: NVIDIA Tesla V100, Tesla A100, and GeForce RTX 2080 Ti.

Job management The submission and scheduling of computational jobs were handled using the SLURM workload manager. SLURM scripts were employed to distribute tasks across nodes, manage GPU resources, and monitor job execution status. The job submission script used for this study is available in the repository of the code at [35].

Documentation and access The CISM provides extensive documentation and user support to facilitate access and efficient use of its computing infrastructure. Relevant resources can be found at: [37].

6 Results

Overview In this study, the primary focus was placed on generating robust and diverse training datasets rather than extensively tuning or optimizing the neural network architecture. To ensure comparability between experiments, all models were trained using the same neural architecture and fixed training hyperparameters, regardless of the dataset used. This methodological choice allows us to evaluate the effect of dataset generation strategies on model performance in a controlled manner, even if the configurations were thus probably not optimal for the specific training sets.

Evaluation metrics

To assess the performance of the network on the test set, we rely on standard classification metrics: *precision*, *recall*, and the *F1-score*. These metrics provide complementary information about the model's ability to make correct predictions.

Precision measures the proportion of predicted positive cases that are truly positive:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

where TP (true positives) denotes the number of correctly predicted positive instances, and FP (false positives) is the number of incorrectly predicted positive instances.

Recall (also known as sensitivity) quantifies the proportion of actual positives that are correctly identified by the model:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where FN (false negatives) represents the number of positive instances incorrectly classified as negative.

F1-score is the harmonic mean of precision and recall, providing a balanced metric when both false positives and false negatives are important:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Macro and Weighted Averages In multiclass classification settings, precision, recall, and F1-score can be averaged across classes to yield global performance measures:

- **Macro-average**: Computes the unweighted mean of the metric for each class, treating all classes equally:

  $$\text{Macro-F1} = \frac{1}{C} \sum_{i=1}^{C} \text{F1}_i$$

  where C is the number of classes and F1_i is the F1-score for class i. This average is sensitive to poor performance on minority classes.

- **Weighted-average**: Computes the mean by weighting each class's metric by its support (number of true instances):

  $$\text{Weighted-F1} = \frac{1}{N} \sum_{i=1}^{C} n_i \cdot \text{F1}_i$$

  where n_i is the number of true instances of class i, and $N = \sum_{i=1}^{C} n_i$ is the total number of samples. This approach reflects class imbalance in the final score.

These metrics are particularly useful for summarizing model performance across multiple classes and guiding model selection in real-world applications, plausibly with imbalanced datasets.

Training performance and class confusion Across all models, we observed significant difficulty in correctly distinguishing between pitchfork and transcritical bifurcations. The model showed particularly poor performance on the test set, with F1-scores often below 0.5. On artifical models'time series, it has shown a complete incapability to predict the correct class.

Model 1 – Canonical extension of Bury's dataset

The first model builds upon the original dataset of Bury et al., which includes null, fold, and Hopf bifurcations. To this, we appended synthetically generated *pitchfork* and *transcritical* bifurcations, derived from canonical forms with additional random perturbation terms. The idea was to know the bifurcation type from its canonical squeleton while introducing diversity. More infos on the system generation can be found in 5. However, this approach led to mediocre results for the newly added classes. We can observe the performances of the network on the test set in Table 2.

Class	Precision	Recall	F1-score	Support
Fold	0.66	0.55	0.60	250
Hopf	0.77	0.73	0.75	250
Trans	0.45	0.73	0.56	250
Null	0.70	0.68	0.69	250
Pitch	0.47	0.29	0.36	250
Accuracy			0.60	1250
Macro avg	0.61	0.60	0.59	1250
Weighted avg	0.61	0.60	0.59	1250

Table 2: 1st model performances on the test set

Discussion The F1-score of 0.56 obtained for the transcritical class may appear moderate, but a closer inspection reveals that it results from an imbalanced classification behaviour. Specifically, the model exhibits a high recall of 0.73, indicating that most transcritical cases are correctly identified. However, the precision is low (0.45), meaning that many predictions labelled as transcritical actually correspond to other bifurcation types, in particular, pitchfork bifurcations.

This suggests that the model tends to classify most branch point bifurcations as transcritical, which inflates recall at the expense of precision. Consistently, the pitchfork class suffers from a very low recall of 0.29, confirming that many of its instances are misclassified as transcritical.

In summary, the 0.56 F1-score does not reflect genuinely good classification performance, but rather an unbalanced prediction strategy that favours one class at the cost of another.

Model 2 – Sparse parameterization

In this version, we increased the sparsity of the models by limiting the number of non-zero parameters. Additionally, mutual exclusion constraints were imposed on pitchfork and transcritical parameters, aiming to increase their separability as explainded in Section 5. However, this adjustment yielded only minor improvements, suggesting that parameter-level disentanglement is insufficient when the resulting time series remain structurally similar. We can once again see the performances of this model in Table 3

Class	Precision	Recall	F1-score	Support
Fold	0.66	0.55	0.60	250
Hopf	0.80	0.66	0.72	250
Trans	0.41	0.51	0.46	250
Null	0.66	0.65	0.65	250
Pitch	0.40	0.44	0.42	250
Accuracy			0.56	1250
Macro avg	0.59	0.56	0.57	1250
Weighted avg	0.59	0.56	0.57	1250

Table 3: Second model performances on the test set

Discussion The performance of the second model is notably poor, with F1-scores of 0.42 for the pitchfork class and 0.46 for the transcritical class. They are both under the expected performance of a total random guessing in a balanced setting. These results indicate that the model fails to capture meaningful patterns to distinguish these bifurcations and highlight the need for a deeper investigation into both the data and the model design.

Hopf and Fold Performance It is of interest to compare the performance of our classifier on Hopf and fold bifurcations with the results reported by Bury in [1] as we basically retrieved Hopf and fold data directly from their training set. In that study, *Bury* reported an F1-score of 84% on the test set. In contrast, our results yield F1-scores ranging between 0.6 and 0.7. How can we explain this difference?

First, Bury's reported score is the average over 20 independently trained networks, while our results are based on a single trained network. Ensemble averaging is known to improve generalization and classification performance which partly explains the difference.

Secondly, *Bury* trained his models on a dataset that was approximately ten times larger than the one used in our study, which significantly enhances model performance by providing greater variability and more robust learning.

Despite these differences, our results still show that the network is capable of correctly identifying Hopf and fold bifurcations. If one wished to increase the accuracy and recall, I suppose this could be achieved relatively easily by three means: enlarging the training dataset, optimizing the training parameters and training multiple networks to later average the predictions obtained.

6.1 Model data test

We illustrate the performance of our network against 2 generic EWS (variance and lag-1 AC) on an artificially generated system exhibiting a pitchfork bifurcation. The quality of binary predictions is calculated using receiver operating characteristic (ROC) curves.

For the classical EWS, predictions are derived from their respective Kendall's τ values, positive values of τ indicating increasing trends. A bifurcation is predicted when this trend statistic surpasses a decision threshold (which is dynamically computed). In contrast, the deep learning model directly outputs a probability distribution over classes, and a bifurcation is predicted when the assigned probability exceeds a threshold (also dynamically computed).

Kendall's τ coefficient. Kendall's τ is a statistical metric named in tribute of its author, measuring the correlation between two rankings. A value of 1 indicates strong agreement, while a value of -1 indicates strong disagreement [38].
It is mathematically defined as:

$$\tau = \frac{\sum}{\frac{n(n-1)}{2}},$$

with n the size of the serie, and $\sum$ the score of the actual ranking. The score can be computed in different ways, one being: $\sum = n_c - n_d$. With n_c and n_d respectively the number of concordant and discordant pairs:

$$\text{the pair } (x_i, x_j) \text{ is considered concordant if } (x_j - x_i)(j - i) > 0 \text{ for } j > i.$$

$$\text{the pair } (x_i, x_j) \text{ is considered discordant if } (x_j - x_i)(j - i) < 0 \text{ for } j > i.$$

As such, the Kendall value is a natural choice to characterize the trend of EWS variables.

Receiver Operating Characteristic The Area Under the Receiver Operating Characteristic (ROC) Curve, or AUC, is a widely used metric for evaluating the performance of binary classifiers. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) [39].

The true positive rate and false positive rate are defined as:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$
$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}},$$

With TN the number of true negatives, TP the number of true positives, FP the number of false positives and FN the number of false negatives. The AUC is then computed taking the area under the curve:

$$\text{AUC} = \int_0^1 \text{TPR}(x)\, dx.$$

Rotating Hoop System (Pitchfork bifurcation)

The rotating hoop system, and the simluation methodology are described in Section 5.

Performance As shown in Figure 20, the ML-based classifier achieves an AUC of 1.0, indicating perfect prediction. This result was expected as *Bury et al.* obtained similar results [1].

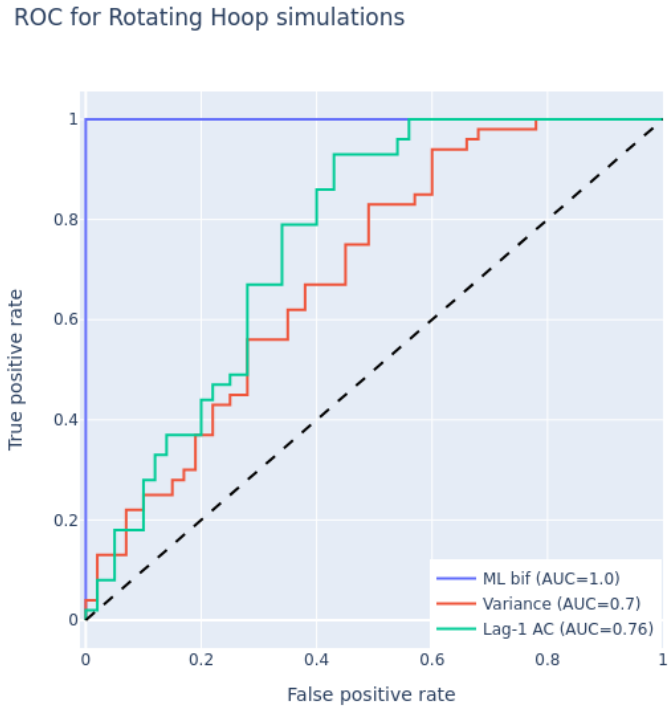

Figure 20: ROC curves for the prediction of a bifurcation in the rotating hoop system using machine learning (ML), variance, and lag-1 autocorrelation (AC)

In the meantime, the lag-1 autocorrelation (AC) performs moderately well with an AUC of 0.76, capturing the transition but with less confidence than our neural network.
The same observation can be made for the Variance with an AUC of 0.70.

Classification Accuracy While assuring ourselves we did not lose the ability to predict a bifurcation , the objective of this study was to classify correctly pitchfork and transcritical bifurcations. As such, we interest ourselves in the classification results depicted in Figure 21. Despite correctly detecting that a bifurcation is imminent, the model assigns only 5% probability to the pitchfork class (the correct category). Instead, the Hopf class is most strongly predicted (85%), followed by fold (10%).

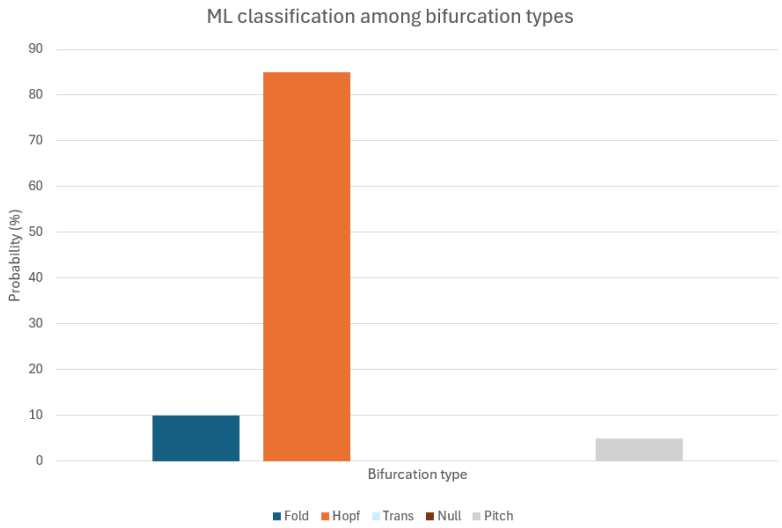

Figure 21: Predicted class probabilities for each bifurcation type from the ML model applied to the rotating hoop system (pitchfork bifurcation).

This misclassification was expected: the model had already failed to correctly classify pitchfork bifurcations during training and among the test set. As such, it was not expected that it could generalize.

I suppose that testing the network on a model exhibiting transcritical bifurcations would produce results similar to those obtained with pitchfork bifurcations: the model would likely detect the presence of a bifurcation but fail to correctly classify it as transcritical. Conducting such an experiment would serve two purposes. First, it would allow us to validate this expected result. Second, in the expected case where the model misclassifies the bifurcation, it would be informative to observe which class is predicted instead. This could provide useful insight into the network's internal representation and decision boundaries, and opens the door for further testing and analysis. An example of model from which we could extract such testing data would be the Rozenzweig MacArthur consumer resource model, exhibiting a transcritical bifurcation and further explainded in Section 5.

7 Discussion

The overall classification results obtained on the test set and application-data, reveal a clear unbalance in performance between the different bifurcation types, which is true for both models. While the models perform reasonably well for fold and Hopf bifurcations, for which the time series were directly took from *Bury*, they struggle to distinguish between pitchfork and transcritical cases. This underperformance is unlikely to be caused by the size of the training set, as all bifurcation classes were trained with the same number of samples, yet Hopf and fold yield satisfactory results. Several hypotheses can be proposed to explain this phenomenon.

First, about the bad predictions on the model test data, it is more than plausible that the training data lacks diversity in the pitchfork and transcritical classes. Indeed, while the Hopf and fold bifurcations instances were generated from total random systems, assuring the maximum reachable diversity, pitchfork and transcritical were generated from canonical forms upgraded with extra terms. Especially, the bifurcations could only take place on one special parameter, the bifurcation parameter from the canonical form, thus reducing the diversity of the set. If the variety of dynamical behaviors or parameter regimes within these classes is too narrow, the model may fail to generalize to unseen instances, thus explaining the bad performance.

As a reference, *Bury et al.* in [3], where they developed a network capable of distinguishing pitchfork from transcritical bifurcations in discrete time, generated systems with polynomial terms up to the tenth order. In contrast, our work considers only up to third-order terms.

This difference arises because *Bury et al.* generated one-dimensional systems, whereas we analyse two-dimensional systems, which inherently increases both the complexity of the generated dynamics and of the computations. Nevertheless, investigating the effect of incorporating higher-order terms in our generated systems remains an interesting direction for future work.

It is important to note that, although we used two-dimensional systems to generate transcritical and pitchfork bifurcations, as it was a reasonable starting point for this study, leveraging Bury's work, these bifurcations do not require a two-dimensional framework, unlike Hopf bifurcations. Therefore, we retain the flexibility to generate pitchfork and transcritical bifurcations using simple one-dimensional systems.

Second, a more fundamental issue may come from the labelling procedure. Indeed, if the assumption that a system generated from the canonical form of a pitchfork bifurcation, and identified as a branch point bifurcation by AUTO, does yield a pitchfork bifurcation, is not verified, transcritical bifurcations can be labelled as pitchfork, and vice-versa. Such labelling errors would prevent the model to learn meaningful decision boundaries between bifurcation types.

Alternatively, the classification task itself may be inherently difficult. In particular, supercritical pitchfork and transcritical bifurcations have both the same stable pre-bifurcation trajectory. Which is a straight line up to the bifurcation. See Figure 22 to picture the pre-bifurcation dynamics of both cases. It is only at the branch point that they distinguish themselves. So we can only wish that they exhibit different reactions to noise and perturbations, which we could detect and assess to make a prediction. Also, they have quite similar canonical forms, see Section 2.

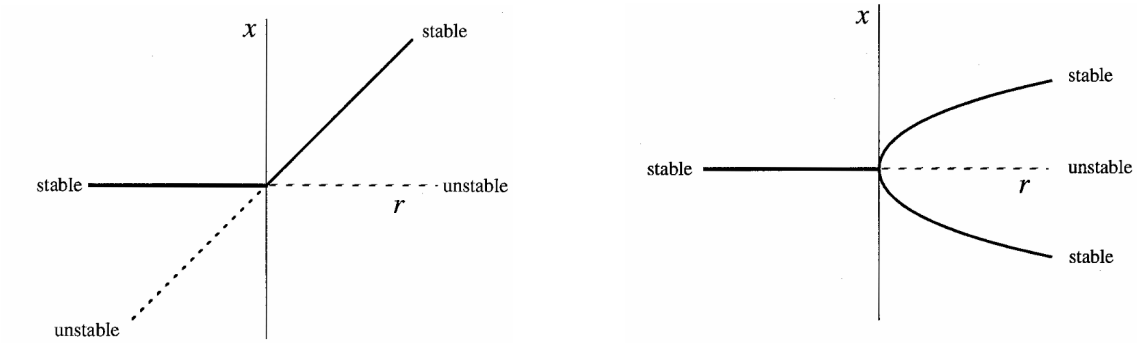

Figure 22: Transcritical (left) and supercritical pitchfork (right) bifurcation diagrams reproduced from [4]

To test these hypotheses, we propose a series of targeted experiments. First, we will train a model on pure canonical forms (without extra terms), so that we are sure labelling is done correctly. If classification performance improves significantly, this would suggest that bad labelling in the extended training set is a major source of error, and that lack of diversity may not be the primary issue.

Second, we will generate fold time series in the same manner as the pitchfork or transcritical ones. First, to verify the integrity of the model pipeline and rule out implementation errors. Secondly, to assess whether the canonical-form generating scheme is inherently bad or not.

These experimental validations will be presented in the next section, and aim to identify the contributing factors to the poor performance observed, thereby guiding improvements in the training data and model architecture.

7.1 Investigation

Canonical Fold comparison

As an exploratory analysis, we generated fold bifurcations using the same canonical method as for the pitchfork and transcritical bifurcations.

The performances on the test set are presented in Table 4. Notably, the F1-score for the fold class increased from approximately 0.60 (as observed in the two previous models) to 0.79 in this configuration, a substantial improvement. This result was anticipated, as the reduced diversity and higher structural similarity among samples in this dataset likely make the classification task easier. However, such gains in accuracy are often accompanied by diminished generalization capabilities, particularly when applied to more heterogeneous or empirical data, which we may see further.

Class	Precision	Recall	F1-score	Support
Fold	0.82	0.77	0.79	250
Hopf	0.88	0.78	0.83	250
Trans	0.45	0.52	0.48	250
Null	0.76	0.70	0.73	250
Pitch	0.42	0.46	0.44	250
Accuracy			0.64	1250
Macro avg	0.67	0.64	0.65	1250
Weighted avg	0.67	0.64	0.65	1250

Table 4: Third model (with fold time series generated from upgraded canonical forms) performances on the test set

We evaluated both the 1st model and this model on their ability to predict bifurcations in the May harvesting model, which is known to exhibit fold bifurcations. See more informations in section 4.1.

The performance outcomes were as follows:

- **Original network**: Demonstrated difficulties in accurately identifying fold bifurcations.
- **Actual network**: Failed to recognize the presence of fold bifurcations entirely.

As expected, the network struggle to generalize to real-world data, thus confirming the fact that the canonical-like generation scheme lack diversity.

While the center manifold theory says that, near a bifurcation, the system's dynamics can be approximated by a normal form on a reduced-dimensional manifold (which is the concept grounding the idea for using canonical forms for generating training datasets), the practical application of

this theory in machine learning contexts may require more nuanced approaches to data generation and model training.

As the new network resulted in better training accuracy but lower generalization performance, consistent with the hypothesis that structurally simpler datasets are easier to fit but less robust for unseen data, this may inform us on at least one of the several weaknesses of our generation scheme of the data, namely, the lack of diversity. This also informs us that the pipeline implemented from the generation of systems to the prediction of the network is done correctly and is not the source of bad results.

Experiment on pure canonical forms

To further investigate the hypothesis that mislabeling may have contributed to the poor classification performance, we conducted an experiment using only pure canonical forms of each bifurcation type, deliberately excluding any perturbation terms (but keeping additive noise). Details on the generation of these systems can be found in Section 5. The rationale behind this experiment comes from the recognition that, in general, we lack formal guarantees that a system intended to exhibit a transcritical bifurcation will not, due to the extra terms, instead exhibit a pitchfork bifurcation, and vice versa. This uncertainty raises concerns about the correctness of the labels in the original training set as a potential source of error.

Class	**Precision**	**Recall**	**F1-score**	**Support**
Fold	0.66	0.54	0.59	250
Hopf	0.81	0.62	0.70	250
Trans	0.43	0.44	0.43	250
Null	0.62	0.66	0.64	250
Pitch	0.40	0.54	0.46	250
Accuracy			0.56	1250
Macro avg	0.58	0.56	0.57	1250
Weighted avg	0.58	0.56	0.57	1250

Table 5: Fourth model (with systems generated from pure canonical forms without extra terms) performances on the test set

By restricting the training data to pure canonical forms, for which the bifurcation type is known with certainty, we ensured that label correctness is no longer in question. If the model showed improved performance in this setting, it would have supported the hypothesis that label noise was a primary cause for bad performances, under a certain remark.

Since the systems are restricted to pure canonical forms, it raises the question of whether the network is thereby able to classify these bifurcations more easily or not. On one hand, the reduction in diversity could simplify the learning task, allowing the model to fit the data more readily. On the other hand, this loss of diversity might prevent the network from encountering data that exhibit the distinctive trends necessary to differentiate transcritical bifurcations from others, leaving only less informative samples.

If the model indeed fits more easily, any observed improvement in performance might stem not from the elimination of label noise but rather from this decreased diversity in the training set. Conversely, if the model encounters greater difficulty in fitting the data, a lack of performance improvement should not be interpreted as an absence of label noise in the original set, but instead it may result from other factors such as the diminished diversity of the training data.

The results, showed in Table 5 remain poor even under these controlled conditions (with f-1 score of 0.46 for the pitchfork class and 0.43 for the transcritical one). This outcome suggests that mislabeling (if there is) alone does not fully account for the classification difficulties. But

considering the remark made above, this test is in reality subjected to caution, and we cannot a priori deduce anything from these results without further theoretical research.

7.2 Alternative Approach

The core challenge addressed in this study is the development of a network capable of distinguishing transcritical bifurcations from pitchfork bifurcations. To achieve this, it was necessary to generate a labeled training set containing time series representative of each bifurcation type. To label the data, our approach was based on the assumption that generating time series from systems defined by the canonical forms of these bifurcations, augmented with additional terms, would result in bifurcation of the looked-after type. In other words, that the extra-terms would not alter the bifurcation intrinsic type.

However, this assumption is difficult to verify. While we can confirm the presence of a bifurcation thanks to AUTO, specifically, a branch point bifurcation (including pitchfork and transcritical bifurcation), we cannot rigorously guarantee that the system's behavior adheres strictly to the canonical form. Indeed, a system designed as a canonical pitchfork bifurcation might exhibit characteristics resembling a transcritical bifurcation, and vice versa due to the extra terms. As the trained model is not capable of distinguishing transcritical from pitchfork bifurcations, we should consider the fact this could be due to a bad labelling in the trainset. In this sense, finding an other way to label the training data is defenitly a field of research.

One promising direction involves taking profit of the branch continuation simulation capabilities of AUTO. The idea is the following; when detecting a branch bifurcation, AUTO would continue each branch emerging. With these trajectories, that we can visualize in a graph, we would aim to distinguish transcritical from pitchfork bifurcations thanks to some algorithm we would develop.

Indeed, the canonical forms of these bifurcations yield clearly distinguishable graphical structures: Figure 23 shows a typical pitchfork bifurcation, whereas Figure 24 illustrates the transcritical case. These differences are visually striking in their pure form and offer a strong motivation to pursue this line of research.

Nevertheless, we cannot rely on pure canonical forms: training a neural network solely on time series generated from pure canonical bifurcation forms proved ineffective in generalizing to more complex systems. Despite the theoretical foundation provided by the center manifold theorem, the network failed to accurately classify bifurcation types when applied to systems outside the narrow scope of canonical dynamics.

To overcome this, it is necessary to generate a broader and more diverse set of dynamical systems. However, moving away from the canonical forms introduces complexity in interpreting the resulting bifurcation diagrams. For example, Figure 25 depicts a case that looks, at first sight, like a pitchfork bifurcation, but watching the trajectory near the bifurcation point, we observe a transcritical bifurcation, which would have folded on itself later on.
A more complex case is given by Figure 26, where it is quite impossible to assert if the trajectory is the one from a transcritical bifurcation near the bifurcation point or a pitchfork one.

Ultimately, this raises an important question: on what basis can we define the bifurcation type, and which algorithm would be capable of achieving a sufficiently accurate classification rate? These questions remain open and could form the foundation for future research. If successfully addressed, such a method would allow for the generation of completely random dynamical systems while still enabling accurate and automatic labeling of their bifurcation types. Thus, solving the lack of diversity and the bad labelling issues.

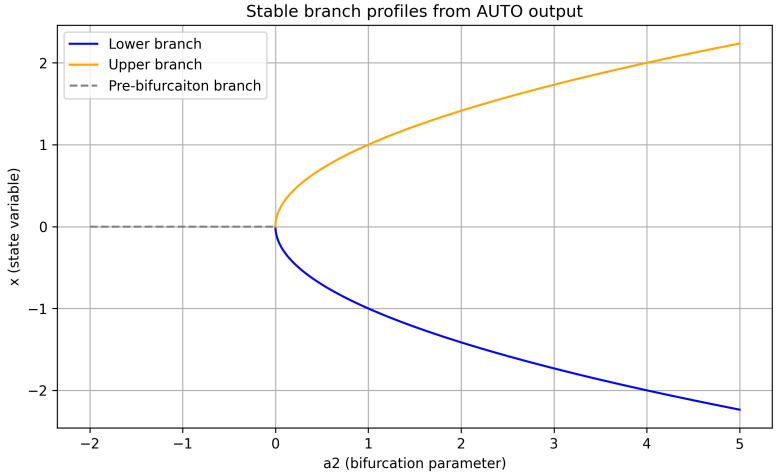

Figure 23: AUTO Branch continuation simulation from the canoncial form of the pitchfork bifurcation

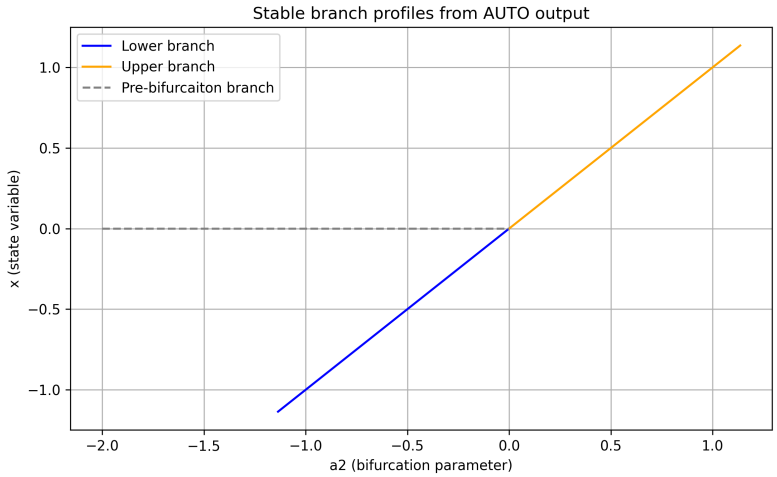

Figure 24: AUTO Branch continuation simulation from the canonical form of the transcritical bifurcation

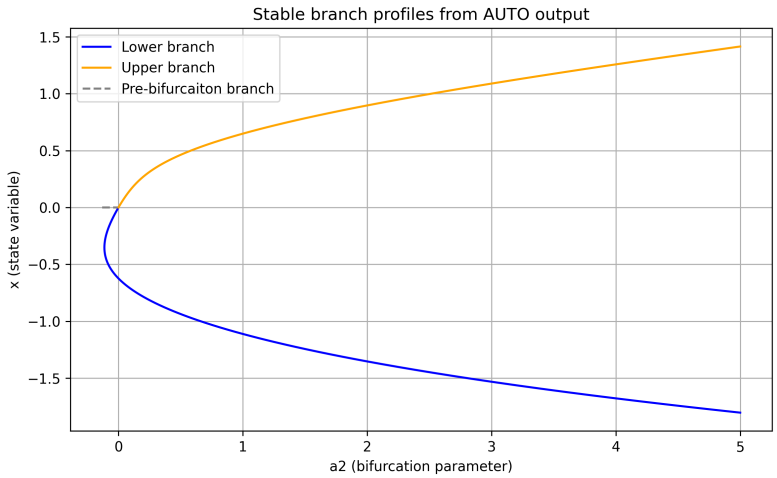

Figure 25: AUTO Branch continuation simulation from a random system, with the pre-bifurcation trajectory, not intuitive case

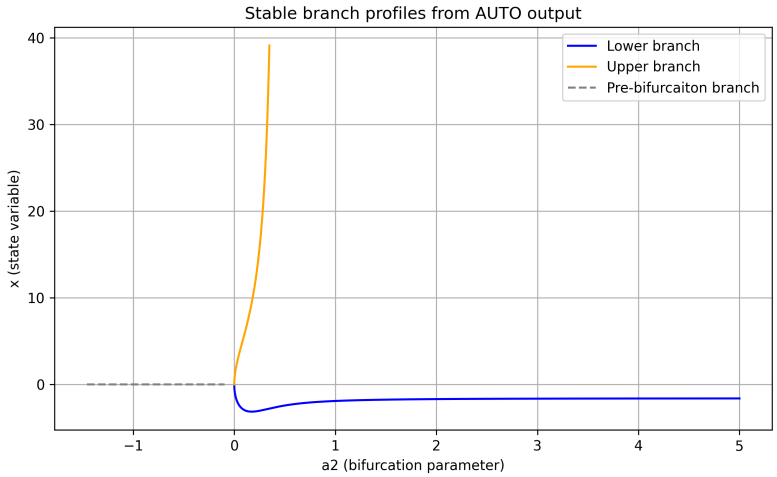

Figure 26: AUTO Branch continuation simulation from a random system, with the pre-bifurcation trajectory, complex case

7.3 Time constraint

One of the constraints encountered throughout this work,and that represents an obstacle in the presented alternative approach is the time required to generate the training dataset. The process involves creating random dynamical systems and systematically varying their parameters to identify potential bifurcations through AUTO. This procedure is computationally intensive, most of all because the bifurcation types discovered is not balanced among all types, largely dominated by fold bifurcations.

Since our goal is to construct a training set with a balanced representation of all bifurcation types, we are interested in those that occur less frequently, being the limiting factor, which happen to be the branch point bifurcations.

As an illustrative example, a 17-hour generation run yielded 10,123 fold bifurcations, 2,805 Hopf bifurcations, but only 891 branch point bifurcations. Considering a more or less even repartition between pitchfork and transcritical bifurcation, we obtain about 445 bifurcations of each type. This distribution is shown in Figure 27. As we have no way to emphasize the generation of a

certain type of bifurcation, without loosing generality (keeping total random systems), we have to wait for the least generated bifurcation type to reach the number of samples required.

To get some idea, Bury et al. generated a dataset with 500,000 time series, evenly distributed among four classes (Hopf, fold, branch point, and null), resulting in 125,000 time series per bifurcation type. To match this scale using our current setup, and considering the generation rate of the assumed transcritical bifurcations (approximately 445 in 17 hours), we would require:

$$\frac{17 \text{ hours}}{445} \times 125{,}000 \approx 4775 \text{ hours} \approx 199 \text{ days}.$$

Clearly, such a time frame is impractical for any realistic research timeline.

This time constraint justifies the need for parallelization. In Bury's work, a batch submission architecture was implemented, allowing for parallel generation of multiple datasets, which were later merged into one training set. This significantly reduced the overall generation time. More precisely, they submitted batches of 4,000 time series, taking about 17h to generate each, considering their specific hardware. In total, it gave 125 batches to submit, which would have resulted in 88 days to generate without parallelization [29].

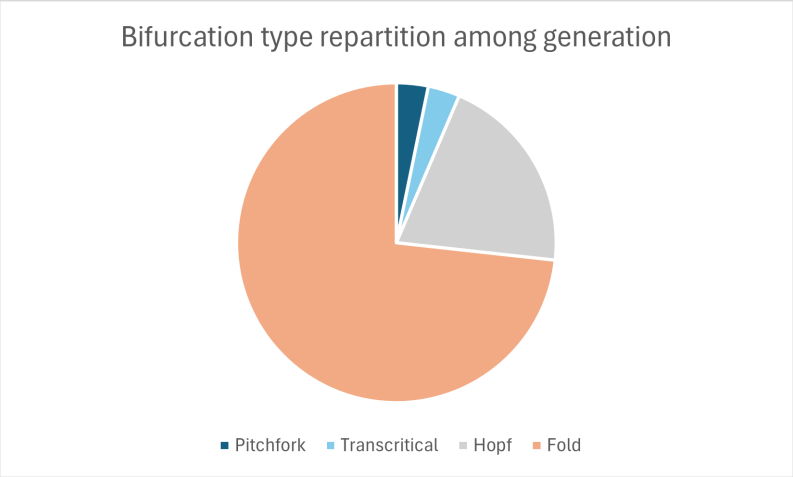

Figure 27: Distribution of bifurcation types generated over a 17-hour period. Pitchfork and transcritical bifurcations are significantly underrepresented.

Parallelization was not implemented in this study as it was not of use in our previous generation scheme. Indeed, it only involved the continuation of one parameter, the one expected to experiment the bifurcation (from the canonical form insight). However, if further research would be conducted on this data-generation scheme, parallelization of the data generation would be of high importance.

Remark on distribution In the systems derived from the canonical forms of the transcritical and pitchfork bifurcations, the bifurcation is supposed to occur along the parameter a_2 of the system described in 5. This imposes a limitation on the diversity of the generated systems in the training set, thus motivating the random systems generation scheme.

But to what extent do branch point bifurcations really occur along an other parameter different from a_2?

To answer this, we analyzed the same 17-hour generation experiment previously described. During this run, a total of 891 branch point bifurcations were identified. Among these, 337 occurred along the a_2 parameter. This corresponds to approximately:

$$\frac{337}{891} \approx 37\%$$

of the branch point bifurcations occurring on a_2.

If this proportion is significant, the majority (63%) of the bifurcations still occur along other parameters. Thus justifying the choice to move beyond canonical-form inspired systems to random system generation. Obtaining as such more diversity in the training set, which is, to my belief, crucial for training a classifier capable of generalizing to empirical and unseen systems.

Conclusion

Throughout this report, several limitations and open questions have been identified, which could serve as starting points for future research. The network architecture and the choice of training hyperparameters deserve further exploration and optimization. Additionally, generating bifurcation data from one-dimensional systems instead of two-dimensional ones could allow the inclusion of higher-order terms, potentially enriching the diversity and realism of the training data. The final approach presented ultimately shows promises and warrants further development. Lastly, one could consider narrowing the initial objective: instead of training a network to classify all four types of bifurcations (fold, Hopf, transcritical, and pitchfork), a more focused task, such as distinguishing between pitchfork and transcritical bifurcations only, assuming prior knowledge that the system undergoes a branch point bifurcation, might lead to improved classification performance.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to E. Massart for initiating and supervising this master's thesis and providing precious guidance throughout this study. I am also thankful to Pierre-Antoine Absil for his support during the semester.
I would also like to thank John Aoga for his assistance with the setup and usage of the Anatole server. And to Bastien Massion for his help in understanding the CISM interface.
Additionally, I would like to thank the École Polytechnique de Louvain (EPL) for their academic and institutional support, as well as the CÉCI and CISM for providing the computational resources that made the simulations possible.
Finally, I would like to thank my family and friends for their support.

References

[1] Thomas M. Bury. Deep learning for early warning signals of tipping points. *PNAS*, 118(39), 2021.

[2] Aditya Paranjape, Nandan Sinha, and N. Ananthkrishnan. Use of bifurcation and continuation methods for aircraft trim and stability analysis - a state-of-the-art. *Collection of Technical Papers - 45th AIAA Aerospace Sciences Meeting*, 18, 2007.

[3] Dylewsky D. Bauch C.T. et al Bury, T.M. Predicting discrete-time bifurcations with deep learning. *Nature Communications*, 14(6331), 2023.

[4] Steven H. Strogatz. *Nonlinear dynamics and chaos.* Perseus Books, New-York, United States, 1994.

[5] Jack Carr. *Applications of Centre Manifold Theory.* Springer Nature, 1982.

[6] Epureanu BI. Ghadami A. Deeplearning for centre manifold reduction and stability analysis in nonlinear systems. *Philosophical Transactions*, A380, 2022.

[7] H.B. Keller. Lectures on numerical methods in bifurcation problems. *Notes by A. K. Nandakumaran and Mythily Ramaswamy, Tata Institute Of Fundamental Research*, 1987.

[8] Timothy Lenton, Hermann Held, Elmar Kriegler, Jim Hall, Wolfgang Lucht, Stefan Rahmstorf, and Hans Schellnhuber. Tipping elements in the earth's climate system. *Proceedings of the National Academy of Sciences of the United States of America*, 105:1786–93, 2008.

[9] V. et al. Masson-Delmotte. Climate change 2021: The physical science basis. contribution of working group i to the sixth assessment report of the intergovernmental panel on climate change. *Cambridge University Press,*, IPCC, 2021.

[10] Ditlevsen S Ditlevsen, P. Warning of a forthcoming collapse of the atlantic meridional overturning circulation. *Nature Communications*, 14(4254), 2023.

[11] Bascompte J. Brock W. et al. Scheffer, M. Early-warning signals for critical transitions. *Nature*, 461:53–59, 2009.

[12] Peter et al. Ashwin. Tipping points in open systems: bifurcation, noise-induced and rate-dependent examples in the climate system. *Phil. Trans. R. Soc. A*, 370:1166–1184, 2011.

[13] Early warning toolbox. https://github.com/earlywarningtoolbox.

[14] Ewstools repository. https://github.com/ThomasMBury/ewstools.

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

[16] Boulton C. A. Buxton J. E. Abrams J. F. Arellano-Nava B. Armstrong McKay D. I. Bathiany S. Blaschke L. Boers N. Dylewsky D. López-Martínez C. Parry I. Ritchie P. van der Bolt B. van der Laan L. Weinans E. Dakos, V. and S. Kéfi. Tipping point detection and early warnings in climate, ecological, and human systems. *Earth Syst. Dynam.*, 15:1117–1135, 2024.

[17] T. Kleinen H. Held. Detection of climate system bifurcations by degenerate fingerprinting. *GEOPHYSICAL RESEARCH LETTERS*, 31(L23207), 2004.

[18] Krkošek Martin and Drake John M. On signals of phase transitions in salmon population dynamics. *Proceedings of the royal society B*, 281, 2014.

[19] Bury T. M. Sadria, M. Fatenet: an integration of dynamical systems and deep learning for cell fate prediction. *Bioinformatics*, 4O:9, 2024.

[20] Hommes C. Wang J. Diks, C. Critical slowing down as an early warning signal for financial crises? *Empirical Economics*, 57:1201–1228, 2019.

[21] Sharma Y. John T. et al. Gopalakrishnan, E. Early warning signals for critical transitions in a thermoacoustic system. *Scientific Reports*, 6(35310), 2016.

[22] Jeff Gore Lei Dai, Kirill S. Korolev. Relation between stability and resilience determines the performance of early warning signals under different environmental drivers. *PNAS*, 112(32), 2015.

[23] T. Smith, R.-M. Zotta, C. A. Boulton, T. M. Lenton, W. Dorigo, and N. Boers. Reliability of resilience estimation based on multi-instrument time series. *Earth System Dynamics*, 14(1):173–183, 2023.

[24] Yuan N. Yang Q. Ma Z. Kurths J. Lu, Z. Early warning ofthe pacific decadal oscillation phase-transition using complex networkanalysis. *Geophysical Research Letters*, 48(e2020GL09167), 2021.

[25] Blaine D Griffen Drake, John M. Early warning signals of extinction in deteriorating environments. *Nature*, 467(7314), 2010.

[26] Ricarda Winkelmann Jonathan F Donges Ann Kristin Klose, Nico Wunderling. What do we mean, 'tipping cascade'? *Environmental Research Letters*, 16(12), 2021.

[27] Held H. Petschel-Held G. Kleinen, T. The potential role of spectral properties in detecting thresholds in the earth system: application to the thermohaline circulation. *Ocean Dynamics*, 53:53–63, 2003.

[28] Clements CF-Krishnan NC Dutta PS Deb S, Sidheekh S. Machine learning methods trained on simple models can predict critical transitions in complex natural systems. *Royal Society Open Science*, 9:161519–161541, 2022.

[29] Deep learning for early warning signals of tipping points. https://github.com/ThomasMBury/deep-early-warnings-pnas.

[30] R. Hennekam et al. Early-warning signals for marine anoxic events. *Geophysical research letters*, 47, 2020.

[31] L. Zhang H. Xie and C. P. Lim. Evolving cnn-lstm models for time series prediction using enhanced grey wolf optimizer. *IEEE Access*, 8:161519–161541, 2020.

[32] Mosquera C. Nápoles G Van Houdt, G. A review on the long short-term memory model. *Artificial Intelligence Review*, 53:5929–5955, 2020.

[33] Hinton GE Krizhevsky A, Sutskever I. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[34] 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*, 33:917–963, 2019.

[35] Investigation on ews classification through deep neural networks git repository. https://github.com/adrienguidat/ews$_c lassification_i nvestigation.git$.

[36] R. H. MacArthur M. L. Rosenzweig. Graphical representation and stability conditions of predator-prey interactions. *The American Naturalist*, 97(895):209–223, 1963.

[37] Cism documentation. https://www.cism.ucl.ac.be/doc/index.html.

[38] Maurice G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[39] Younger JG. Grzybowski M. Statistical methodology: Iii. receiver operating characteristic (roc) curves. *Academic Emergency Medicine*, 4(8):818–826, 1997.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**