

RESEARCH ARTICLE | MAY 23 2024

Inferring bifurcation diagrams with transformers

Lyra Zhornjak   ; M. Ani Hsieh  ; Eric Forgoston 

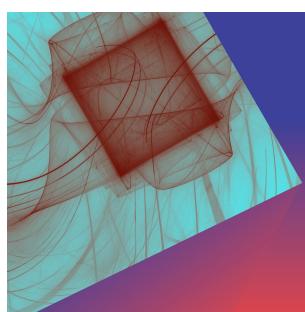


Chaos 34, 051102 (2024)

<https://doi.org/10.1063/5.0204714>


View
Online


Export
Citation



Chaos: An Interdisciplinary Journal of Nonlinear Science

Focus Issue:

From Sand to Shrimps: In honor of Jason A.C. Gallas

Guest Editors: Marcus W. Beims, Thorsten Pöschel, Pedro G. Lind

Submit Today!

 **AIP
Publishing**

Inferring bifurcation diagrams with transformers

Cite as: Chaos 34, 051102 (2024); doi: [10.1063/5.0204714](https://doi.org/10.1063/5.0204714)

Submitted: 22 February 2024 · Accepted: 25 April 2024 ·

Published Online: 23 May 2024



View Online



Export Citation



CrossMark

Lyra Zhorniyak,^{1,a)} M. Ani Hsieh,^{1,2} and Eric Forgoston^{2,3}

AFFILIATIONS

¹ Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Pennsylvania 19104, USA

² Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, Pennsylvania 19104, USA

³ School of Computing, Montclair State University, Montclair, New Jersey 07043, USA

^{a)}Author to whom correspondence should be addressed: zhorniyak@seas.upenn.edu

ABSTRACT

The construction of bifurcation diagrams is an essential component of understanding nonlinear dynamical systems. The task can be challenging when one knows the equations of the dynamical system and becomes much more difficult if only the underlying data associated with the system are available. In this work, we present a transformer-based method to directly estimate the bifurcation diagram using only noisy data associated with an arbitrary dynamical system. By splitting a bifurcation diagram into segments at bifurcation points, the transformer is trained to simultaneously predict how many segments are present and to minimize the loss with respect to the predicted position, shape, and asymptotic stability of each predicted segment. The trained model is shown, both quantitatively and qualitatively, to reliably estimate the structure of the bifurcation diagram for arbitrarily generated one- and two-dimensional systems experiencing a codimension-one bifurcation with as few as 30 trajectories. We show that the method is robust to noise in both the state variable and the system parameter.

Published under an exclusive license by AIP Publishing. <https://doi.org/10.1063/5.0204714>

We consider the problem of generating bifurcation diagrams from data. Traditionally, one must first use data-driven modeling techniques, such as system identification, to identify the governing equations of the dynamical system that is associated with the data. With the governing equations in hand, one can then employ standard analytical and numerical methods to find the bifurcation diagram. However, this approach is data-hungry and sensitive to noise, making it unsuitable for many scenarios. In this article, we provide a detailed description of how the bifurcation diagram prediction task can be instead reformulated as a task suitable for deep learning. Using a slightly modified DEtection TRansformer (DETR)¹ architecture, we demonstrate the approach on a variety of one-dimensional local bifurcations and show how the method can be extended for a type of two-dimensional local bifurcation. In each scenario, accurate bifurcation diagrams can be generated from a small number of very noisy trajectories by leveraging the inherent pattern-matching capabilities of transformers.

I. INTRODUCTION

Dynamic system behavior can often be impacted by changes in system parameters. In general, small continuous changes in

system parameters yield similarly small continuous changes in system behavior. Nevertheless, for a certain class of systems, small continuous changes in these parameters around certain values can lead to large changes in the overall dynamics of the system. At these values, fixed points, periodic orbits, and other phenomena representing the topological structure of the system may suddenly appear or disappear. This is known as *bifurcation* and each value where this occurs is known as a *bifurcation point* of the system. Understanding how and when bifurcations occur in dynamic systems is crucial to understanding a wide variety of phenomena across physics, biology, chemistry, and engineering.² For example, a saddle-node bifurcation can lead to an outbreak of spruce budworm,³ a transcritical bifurcation can cause a switch from spontaneous to stimulated laser emission for solid-state lasers,⁴ a pitchfork bifurcation is associated with the transition from conduction to convection in Rayleigh–Bénard convection,⁵ while a Hopf bifurcation can induce a transition of a swarm from a stationary to an oscillating state.⁶

Bifurcation diagrams provide a valuable tool for visualizing and understanding the topological structure of a system vs a parameter that contains a bifurcation point, a *bifurcation parameter*. Suppose we have a dynamical system $x'(t) = f(x(t), r)$ that has some bifurcation parameters $r \in \mathbb{R}^m$, where $x \in \mathbb{R}^n$ is the state of the system and $f: \mathbb{R}^{n+m} \rightarrow \mathbb{R}^n$ is the governing equation of the system. When f is

known, bifurcation diagrams are typically generated through continuation software such as AUTO.⁷ This approach requires familiarity with both the system and the software. Unfortunately, this can be problematic as AUTO is known to be very tricky to use and may also fail.⁸ In addition, when the governing equation f of the system is not known and only time-series data of the system are available, these tools cannot be used at all. One approach in this scenario involves two steps: first, identify a model to the underlying system at some values of the bifurcation parameter using system identification techniques, and second, interpolate between these to produce the bifurcation diagram. To produce accurate results with this method, a large amount of data are needed as typically a separate model is fitted at every examined value of the bifurcation parameter. Even when this data is available, the resulting prediction is coarse in the bifurcation parameter and sensitive to noise in the estimated value of the bifurcation parameter.

In this work, we instead present the first attempt at the novel problem of directly predicting the bifurcation diagram from time-series data. The model in this work is formulated to produce bifurcation diagrams of a single dimension against a single bifurcation parameter and, as such, this work is primarily focused on one-dimensional local bifurcations. In addition, however, we demonstrate the effectiveness on a slice of the Hopf bifurcation suggesting that this method can be easily extended to higher-dimensional systems in certain circumstances. Since systems that vary immensely in their overall topological structure can produce bifurcation diagrams that, at least locally, are very similar, directly predicting the bifurcation diagram allows knowledge on how the bifurcation parameter affects the topological structure of the system to be extracted when existing methods fail. In particular, by pooling information from the dynamics across multiple values of the bifurcation parameter, we can learn structural information about a system in low-data high-noise scenarios. Using a slightly modified version of the DETR (DEtection TRansformer) architecture,¹ we demonstrate the ability to accurately predict the bifurcation diagram within a given range with as few as 30 noisy impulse responses for an arbitrary continuous-time system. To avoid repetition, in the remainder of this work, we will refer to the simulated dynamics used as the input data for the model as the input trajectories and the predicted segments of the bifurcations diagram as the predicted branches. The code and trained model used in this paper is available at <https://github.com/lzhorniy/bifurcs>.

II. BACKGROUND

A. System identification

The task of identifying the underlying model for a dynamical system given numeric data is known as system identification. System identification generally makes the implicit assumption that the parameters of the system remain constant across all trajectories used or that these parameter values are known.

Sparse Identification of Nonlinear Dynamics (SINDy)⁹ provides a method to directly identify the underlying system given a set of trajectories. Bifurcating systems can be modeled by including the bifurcation parameter as an additional state variable and the method can be made to be somewhat resilient to noise by filtering the provided data. However, this is only possible due to strong assumptions

about the system and the data. SINDy assumes that the system can be expressed as the sum of a very sparse subset of pre-selected basis functions and, when examining a bifurcating system, the values of the parameter for each trajectory in the data are exactly known.

Opposed to explicit parameter identification provided by SINDy, a Proper Orthogonal Decomposition Neural Network (POD-NN)^{10,11} aims to build a single implicit model of the system for all its variables. POD-NN trains a neural network to map the variables (e.g., system parameter or position) onto their corresponding coefficients in the reduced basis space provided by a proper orthogonal decomposition, allowing a solution to be recalled. One can thus construct the bifurcation diagram of the system by mapping the asymptotic solution vs the system parameter provided as input. Since each parameter value can only correspond to one solution, this method can only generate one branch of the bifurcation diagram and cannot represent unstable branches. As a consequence, one must also be sure that all provided data corresponds to only one branch.

B. Sequence modeling

Transformers, first popularized by Vaswani *et al.*,¹² are a method to encode and relate a sequence of discrete tokens (vectors) in one domain and to use this information to recurrently decode a sequence of queries (vectors) in another domain. They differ from other sequence modeling approaches in their use of attention as the sole method of token comparison. In contrast to recurrent neural networks (RNNs) based methods such as Long Short-Term Memory networks (LSTMs)¹³ and Gated Recurrent Units (GRUs),¹⁴ attention compares all input tokens in parallel and thus does not need to balance retention of information and updating for new data. In contrast to convolutional methods,¹⁵ attention learns relations between all tokens and does not require a window with a uniform spacing of tokens. This is because attention learns structural information (e.g., ordering and spacing) from the token implicitly. These reasons make transformer architectures particularly well suited to predicting a bifurcation diagram from arbitrary impulse responses, where spacing in r of the trajectories used as tokens is random and even the ordering may be uncertain due to noise.

DETR¹ extends the basic transformer architecture by having the queries used for the decoder be a fixed set of learned queries, eliminating the need for recursion in the output. In particular, this allows an ordered set of input tokens to directly produce an unordered set of predictions. Each element in the output set is preferentially focused on some subset of the input set through the attention mechanism, the learned queries thus act as templates for different types of bifurcation branches. This effect is demonstrated in Fig. 1. Each branch selectively prioritizes the trajectories that converge to that branch but still allows some information from other trajectories to contribute, a feature of transformers that allow coordination between predictions.

III. METHODS

A. Architecture

We use the DETR architecture as the basis of our model. This consists of two parts: an encoder that takes a set of system

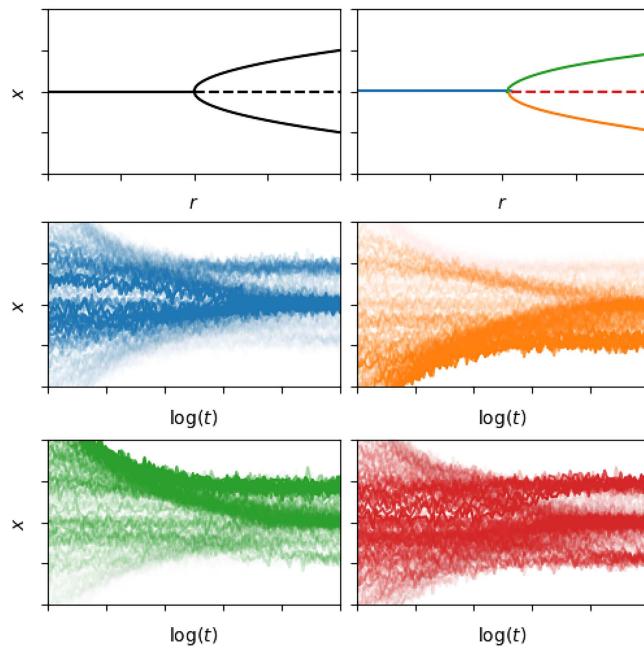


FIG. 1. Visualization of the attention in the last layer of the decoder from each valid prediction onto the noisy trajectories used as inputs. The top row shows the ground truth bifurcation diagram (left) and the prediction produced by the model (right), plotted as the state variable x vs the system parameter r . The bottom two rows show the system trajectories (see Fig. 2) shaded according to the attention from the corresponding predicted branch, indicated by color, plotted as state variable x vs time step t . A higher attention (darker color) indicates that trajectory is utilized more in producing the prediction. For example, for the blue stable branch at 0, those trajectories that converge to 0 generally have higher attention. All figures are plotted on the same scale for x .

trajectories and produces an embedding and a decoder that uses this embedding to transform a set of learned queries into a set of predictions that each represents a predicted bifurcation branch. For a brief overview of the function of this architecture and an explanation of the rationale behind its selection, see Sec. II B.

The encoder consists of $N = 4$ identically structured sequential blocks with two learned layers each, not shared between blocks. The encoder is provided with a set of trajectories, each composed of 128 points sampled as described in Sec. IV. These trajectories are each augmented with their associated value of the system parameter using sinusoidal position encoding.¹² The decoder likewise consists of $M = 4$ identically structured sequential blocks with two learned layers each with 50 learned queries provided as an input. The embeddings produced by the encoder are selected through the attention mechanism and once again are augmented with their associated value of the system parameter, before being combined with the learned queries. The output of the final layer of the decoder is passed through a feed-forward network (FFN) to four predictions representing a predicted segment: a set of parameters representing the shape of the curve, the start and end points of the curve on the bifurcation diagram, the stability of the segment, and the

model's confidence that its prediction is correct. Figure 2 summarizes how the system trajectories are transformed into the final prediction. For a more thorough explanation of attention and the DETR architecture, see Appendix A in [supplementary material](#).

B. Loss function

A four part loss function is used during training,

$$L = L_{\text{pos}} + L_{\text{shape}} + L_{\text{stab}} + 0.01L_{\text{conf}}.$$

Here, L_{pos} is the position loss, L_{shape} is the shape loss, L_{stab} is the stability loss, and L_{conf} is the confidence loss. We describe each term in Secs. III B 1–III B 5. The weights of each term are experimentally selected to ensure that the magnitude of the losses is approximately equal toward the end of the trajectory to prevent one loss term from dominating. For a detailed description of how the bifurcation diagram is created from the predicted curve segments, see Sec. IV.

1. Prediction assignment

In order to associate predicted curve segments with the ground truth, the Hungarian algorithm¹⁶ is used to assign one predicted segment to each ground truth to calculate the loss above. The assignment cost used is a weighted combination of the shape, position, and stability losses $C = L_{\text{pos}} + 0.1L_{\text{shape}} + L_{\text{stab}}$. This results in the creation of a mapping $H : b \leftrightarrow \hat{b}$ between the ground truth branches b and the assigned predictions \hat{b} , composed of n_H entries. In the equations below, let the index i correspond to indices of \hat{b} and index j correspond to indices of b .

2. Position loss

The position loss L_{pos} trains the start and end of each predicted branch. It is defined as the L_1 loss between the predicted start and end coordinates of the curve segment defining a branch and the ground truth as assigned above,

$$L_{\text{pos}} = \frac{1}{n_H} \sum_{i \in H} |\hat{r}_{0,i} - r_{0,j}| + |\hat{r}_{1,i} - r_{1,j}| + |\hat{x}_{0,i} - x_{0,j}| + |\hat{x}_{1,i} - x_{1,j}|,$$

where (r_0, x_0) is the start and (r_1, x_1) is the end of the i th segment in the bifurcation diagram space. Specifically, r and x are the value of the bifurcation parameter and fixed point, respectively.

3. Shape loss

The shape loss L_{shape} trains the path each branch takes between its start and end points. It is defined as the L_2 loss between the parameters of predicted approximation \hat{z} to the shape of a branch of the bifurcation diagram and the true shape z of the assigned branch,

$$L_{\text{shape}} = \frac{1}{n_H} \sum_{(i,j) \in H} \|\hat{z}_i - z_j\|_2.$$

4. Stability loss

The stability loss L_{stab} trains the predicted steady-state behavior of the branch, in this case either asymptotically stable or unstable.

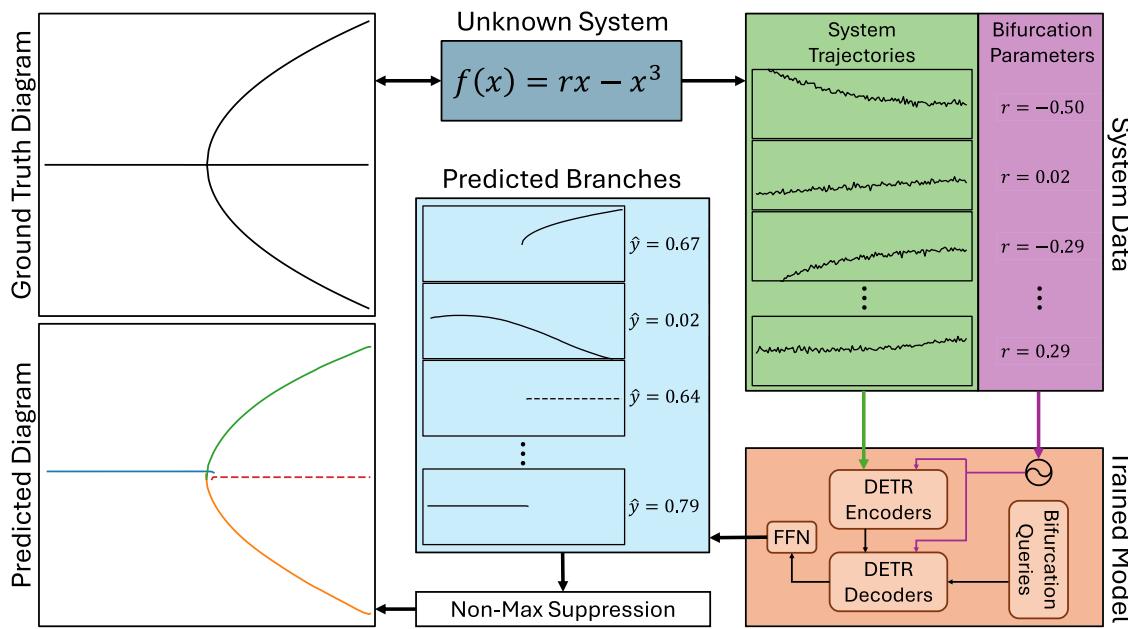


FIG. 2. Given a set of data from an unknown system, comprised of system trajectories and the estimated bifurcation parameter in that configuration, our approach can reliably reconstruct the predicted bifurcation diagram. The predicted confidence \hat{y} is used as the basis for the non-max suppression. Figure 6 in [supplementary material](#) provides a more detailed view of the internal architecture of the trained model.

It is defined as the binary cross-entropy loss between the predicted stability \hat{s} and an indicator function I_s that is 1 if ground truth curve segment j represents a branch of the bifurcation diagram that is asymptotically stable and 0 otherwise,

$$L_{\text{stab}} = \frac{1}{n_H} \sum_{(i,j) \in H} - (I_s[j] \log(\hat{s}) + (1 - I_s[j]) \log(1 - \hat{s})).$$

5. Confidence loss

Finally, the confidence loss L_{conf} trains the model's confidence in its own predictions, \hat{y} . It is described by the balanced sum of the binary cross-entropy loss of the correct and incorrect predictions,

$$L_{\text{conf}} = -\frac{1}{n_H} \sum_{i \in H} \log(\hat{y}_i) - \frac{1}{n - n_H} \sum_{i \notin H} \log(1 - \hat{y}_i).$$

C. Non-max suppression

We have so far defined a model that predicts a set of segments defined by their start and end positions, shape, stability, and the confidence the model has in the prediction. Unfortunately, the model tends to duplicate its prediction, producing many predictions for each leg of the bifurcation diagram. While DETR-based systems can typically forgo non-maximum (non-max) suppression (NMS) systems in vision contexts,¹ we find that some such system is needed in this context. Since curve segments can be continuously deformed to any other segment, this may be the result of the model "hedging" between nearby possibilities.¹⁷

To remove duplicate prediction, we need to define the *distance* between curve segments. There are a variety of techniques used to find this distance. From basic methods such as mean absolute error (MAE) and the mean squared error (MSE) to more specialized techniques such as dynamic time warping¹⁸ and curve length,¹⁹ most methods will produce adequate results for the context of NMS.

Predicting bifurcation diagrams has unique requirements that help in selecting an appropriate measure. As the start and end points of each segment define critical points of the model, the metric should *not* reject outliers as these are often what differentiates nearby branches. Additionally, the approach must be computationally efficient to allow for the computation of set statistics on the test sets, ruling out most specialized comparison algorithms. However, since predicted segments are directly compared against each other with no external source of error, noise is not a concern. For these reasons, we select the MSE as the basis of our distance function,²⁰

$$d(a, b) = \min(1, \alpha \text{MSE}(a, b)). \quad (1)$$

Here, MSE is the MSE loss between the two segments and α is a fixed scaling factor.

The distance function is restricted to the range (0, 1) to allow it to be used as a discounting function for NMS. Inspired by Soft-NMS,²¹ the NMS procedure is composed of the following steps:

1. Remove all predictions whose confidence is less than some threshold β_1 .
2. Select the highest confidence prediction and add it to the list of predicted segments.

3. Discount the confidence of all remaining predictions by multiplying their confidence by their distance to this segment.
4. If the highest confidence prediction remaining has a confidence greater than $\beta_2 > 0$, return to step 2.

D. Evaluation

To evaluate the predicted set of curve segments against the true bifurcation diagram, we need to extend the notion of distance between two segments to the distance between two sets of segments. While this appears similar to the problem of matching two shapes, traditional shape matching metrics strive to be invariant to some set of transformations, noise, and outliers.²² This makes most traditional shape and skeleton similarity metrics unsuitable for measuring the similarity between two bifurcation diagrams. Instead, we create a simple extension to the distance measure in Eq. (1).

To extend a pairwise distance measure to a bifurcation similarity metric, pairwise correspondences between the set of predicted curve segments and the ground truth must first be created; as in Sec. III B, the Hungarian algorithm is used to produce this set of correspondences \hat{H} . Note that, instead of using losses to define the assignment cost, here the distance function is used directly. We thus define the bifurcation distance metric between the set of predicted curve segments A and true segments B as

$$BDM(A, B)$$

$$= \frac{1}{n_B} \left[\sum_{(i,j) \in \hat{H}} d(a_i, b_j) + \sum_{i \notin \hat{H}} \left(\min_{b \in B} d(a_i, b) \right) + (n_B - n_{\hat{H}}) \right]. \quad (2)$$

Here, the sum of the distances between the assigned pairs of predictions and ground truths is the basis of the metric. For unassigned predictions, the closest distance to the ground truth is added, while for each unassigned ground truth, the maximum distance (1) is added. This is then normalized by the number of ground truth segments. The bifurcation similarity score (BSS) is thus defined as

$$BSS(A, B) = \min(0, 1 - BDM(A, B)). \quad (3)$$

IV. IMPLEMENTATION

A. System definition

We consider all one-dimensional (1D) polynomial differential equations with a single variable of the form

$$\frac{dx}{dt} = f(x, r) = \sum_{i=0}^3 [(a_i + b_i(r + c/2))(x + d/2)^i]. \quad (4)$$

This formulation is selected to compactly represent all prototypical local bifurcations of a single variable phase space and their combinations. These can be grouped into three types identified by their prototypical form,²

- saddle-node bifurcation: $\dot{x} = r + x^2$,
- transcritical bifurcation: $\dot{x} = rx - x^2$,
- pitchfork bifurcation: $\dot{x} = rx \pm x^3$.

The terms a_i , b_i , c , and d define the system and, without loss of generality, are sampled from a uniform distribution with a range of

-1 to $+1$. To ensure that the space of possible bifurcations is adequately covered, a random subset of these parameters is set to zero for each equation in the training data.

B. Input data

To generate the simulated dynamics of each system that serves as the input to the model, we randomly select 200 configurations of the system. Each configuration consists of an initial value x_0 randomly selected from -4 to $+4$ and the value of the bifurcation parameter, randomly set to a value from -1 to $+1$. This system is then simulated from $t = 0$ to $t = 10$ and 128 points, geometrically spaced, are sampled from the resulting trajectory. The geometric spacing of the sampled points ensures that the model is provided with sufficient fidelity in the small timescale transient dynamics of the system while still providing the long-term behavior of the system. Additionally, it conveniently allows for representing configurations with different time constant with the same time-series.

C. Bifurcation diagram

As we are considering only local bifurcations of a single variable, the bifurcation diagram is equivalent to the zero level set of f , namely, $f(x, r) = 0$. To convert this into a set of curve segments that can be predicted by the transformer model, a consistent criterion must be selected. This criterion must split the zero level set into curve segments such that (a) curve segments meet only at their end points and (b) curve segments end whenever their stability changes (since, by our definition, each curve segment is assumed to have the same stability across it).

This leads naturally to the selection of the critical points of the bifurcation diagram, those points where $f'(x) = 0$, as the points where the curve segments are split. Specifically, curve segments are split when $|f'(x)| < \epsilon$ along the curve, for some epsilon. The results of this method can be seen in Fig. 3(a).

For the model to predict these curve segments, they now must be converted into a vector. These vectors will contain the start and end points of the curve, the stability of the curve (0 if stable and 1 otherwise), and a parametric representation of the shape of the curve between the end points. For simplicity, 100 points, evenly spaced along the curve in the axis of the bifurcation parameter, are used to represent the curve; these are normalized using the start and end points to values from 0 to 1, decoupling the shape and the positional prediction of the curve. Other parametric representations, such as Bezier curves, are possible. However, our representation carries the additional benefit of not making implicit assumptions on the form of branches, allowing more complicated branches to be represented without modification.

V. RESULTS

A. Performance

As shown in Fig. 3 and in Appendix B in supplementary material and summarized in Table II the trained model is able to accurately and reliably reconstruct the structure of all basic bifurcations, as described in Table I. In contrast, the model is only able to locate the bifurcation points (indicated by the ends of the predicted branches) to within about 0.1 units of the true bifurcation point

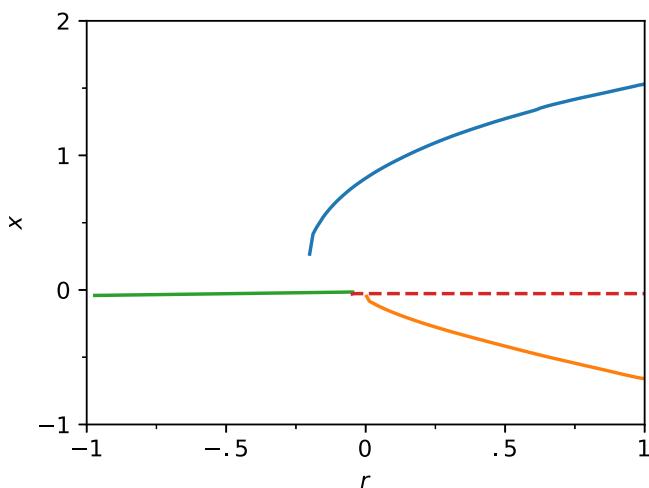
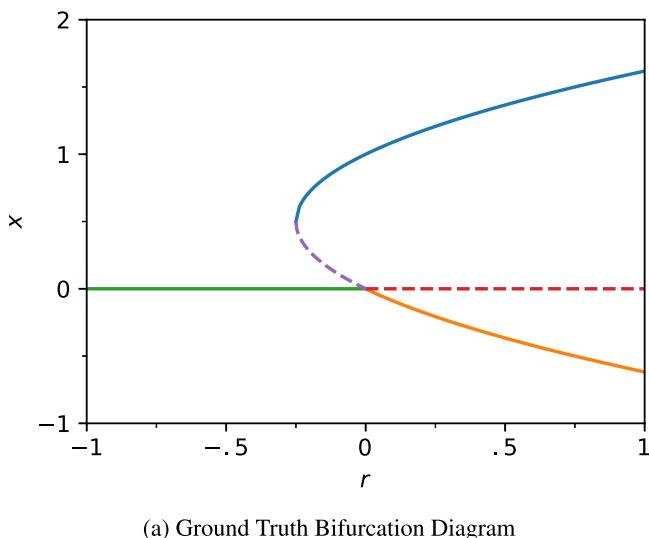


FIG. 3. Ground truth and predicted bifurcation diagram for the system $f(x) = rx + x^2 - x^3$ with each segment shown in a different color; unstable branches of the diagram are shown with a dashed line. The bifurcation diagram in (b) is predicted using 30 trajectories with additive Gaussian noise of standard deviation 0.1. (a) Ground truth bifurcation diagram, (b) predicted diagram with 30 noisy trajectories; $BSS = 0.7991$.

when a branch is predicted. This may simply be the natural result of bifurcation points and how trajectories are sampled in the data generation scheme provided in Sec. IV; trajectories converge or diverge increasingly slowly as one approaches the bifurcation point while the geometry of the system can change rapidly. Given that there may be only a handful of trajectories in proximity to the bifurcation point, the model must rely on interpolation between trajectories to determine when a branch ends. The interpretation of the model as an interpolator between trajectories is also supported by the structure

TABLE I. Randomized forms of several different types of bifurcation. Here, a_i , b_i , c , d , and e_i are sampled from a uniform distribution with a range of -1 to $+1$.

Bifurcation	Randomized form
Saddle-node	$f(x, r) = b_0(r + c/2) - a_2(x + d/2)^2$
Transcritical	$f(x, r) = b_1(r + c/2)(x + d/2) - a_2(x + d/2)^2$
Pitchfork	$f(x, r) = b_1(r + c/2)(x + d/2) - a_3(x + d/2)^3$
Arbitrary	See Eq. (4)
Hopf	$f(x, y, r) = (b_0(r + c/2) - a_0(x + d/2)^2 - e_0 y^2) \times (x + d/2) - 10 e_1 y$ $g(x, y, r) = (b_0(r + c/2) - a_0(x + d/2)^2 - e_0 y^2))y - 10 e_1 (x + d/2)$

of the DETR model itself: the learned queries act as templates that selectively match to the input trajectories that inform the branch that the query is responsible for, producing the selection behavior observed in Fig. 1.

B. Robustness

Another consequence of using the complete set of trajectories to generate entire branches rather than a more local approach may be the robustness to both large additive noise and a reduction in the number of supplied trajectories summarized in Table II. Figure 4 shows that small noise produces little to no impact on the resulting predictions, with larger values resulting in a gradual reduction in score. Similarly, Fig. 5 shows that, for a test set of arbitrary functions generated as in Sec. IV, there is minimal reduction in the resulting score as the number of trajectories decreases until there is insufficient information and a sharp drop in score occurs. These results show how performance decreases with randomly sampled trajectories; appropriate coverage of the initial positions and bifurcation parameter when selecting trajectories can maintain performance with fewer trajectories by ensuring that trajectories are sampled from all regions of the bifurcation diagram. This robustness is present despite the model being trained without data augmentation.

While the model can overcome noise and a reduction in trajectories by interpolating between the remaining trajectories, the model is unable to reliably extrapolate outside of the range of provided trajectories. Specifically, the model is unable to predict the behavior of the system on one side of a bifurcation only given

TABLE II. Average BSS across 1000 random systems as described in Table I. With the exception of the Hopf bifurcation (see Sec. V C), all predictions are produced from the same model. Scores are shown for 200 provided trajectories, 200 trajectories distorted with Gaussian noise with standard deviation 0.2, 30 trajectories, and 30 trajectories with the same noise.

	200 Traj.	$+0.2\sigma$	30 Traj.	$+0.2\sigma$
Saddle-Node	0.9799	0.9686	0.9604	0.9523
Transcritical	0.8956	0.8931	0.8478	0.8288
Pitchfork	0.9223	0.9386	0.8839	0.8759
Arbitrary	0.7793	0.7703	0.7574	0.7470
Hopf	0.7916	0.7786	0.7625	0.7603

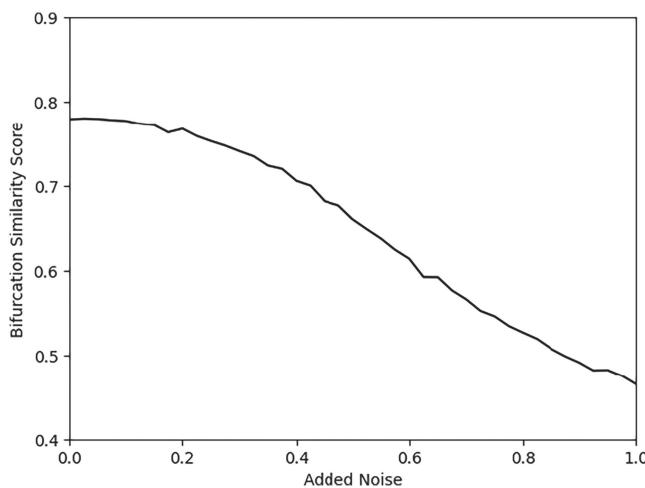


FIG. 4. Performance of the model with 200 trajectories as measured by the BSS [Eq. (3)] vs the standard deviation of the normal noise added to both the state and parameter values of the supplied trajectories. Results shown are for systems as defined in Eq. (4).

trajectories on the other side of the bifurcation. Given how a bifurcation point represents a complete change in the geometry of a system, the task of predicting the diagram beyond the bifurcation fundamentally requires identifying the underlying system using the dynamics exhibited by the provided trajectories. Further exploration is required to determine if this is feasible.

C. Higher dimensions

While the results discussed so far have all been bifurcations in 1D systems, this model can easily be extended to systems of higher

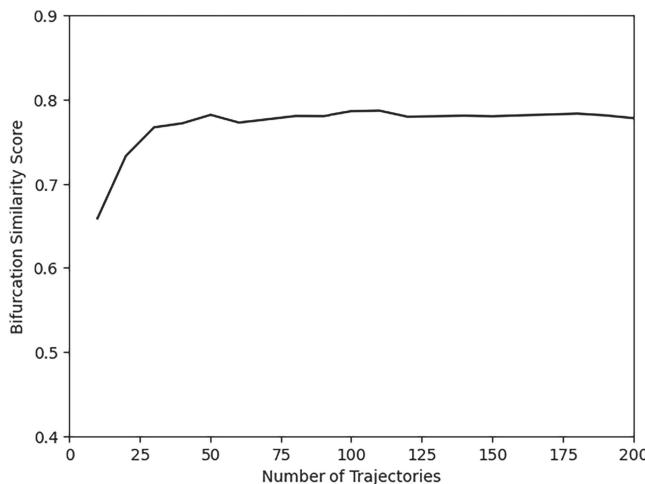


FIG. 5. Performance of the model as measured by the BSS vs the number of supplied trajectories, with no added noise. Results shown are for systems as defined in Eq. (4).

dimensions if some restrictions are placed on the system. As the model predicts a bifurcation diagram of one variable against the bifurcation parameter, higher-dimensional systems should have all examined fixed points on a single line through the state space.

Consider an arbitrary system that experiences a Hopf bifurcation, as defined in Table I, where y is the second dimension of the system. If we assume (as defined) that all fixed points in this system occur when $y = 0$, we can define a bifurcation diagram as the locations where $f(x, 0, r) = 0$ on the (x, r) plane. This captures the limits of the oscillations induced by the Hopf bifurcation and produces a bifurcation diagram that appears visually similar to a pitchfork bifurcation; see Appendix B in [supplementary material](#), for examples.

If we fine-tune the model used previously on this Hopf bifurcation, providing only the trajectories in x and not y , we achieve the results described in Table II. This suggests the following: if we can generate simulated data such that our desired bifurcation diagram lies in the domain of the bifurcation diagrams of the training systems, this approach should be able to reliably predict the major features of the topological structure of the system.

VI. CONCLUSION

We introduce a novel approach to directly estimate the bifurcation diagram of a dynamical system from noisy data. By dividing a bifurcation diagram at the bifurcation points into its component branches, we are able to structure the problem to leverage the DETR architecture. The transformer is trained to predict the number of branches and minimize the loss with respect to the position, shape, and stability of each predicted branch. The input trajectories are augmented with their associated system parameter using a sinusoidal embedding scheme, allowing the uncertainty in the parameter for a single trajectory to be mitigated through relational information to other input trajectories. The resulting model, despite no augmentation with regard to noise or trajectory count, is able to reliably predict the structure of the bifurcation diagram with as few as 30 trajectories even when there is significant noise present in both the state variables and system parameters.

SUPPLEMENTARY MATERIAL

The [supplementary material](#) is composed of two appendixes. Appendix A provides a brief overview of the mechanism of attention and the components of DETR. Appendix B includes additional results on randomly selected systems.

ACKNOWLEDGMENTS

We thank Tom Zhang Jiahao, Jasleen Dhanoa, and Ben Shaffer for their many helpful discussions. This work was funded by the National Science Foundation (Award Nos. CMMI 2121887 and CMMI 2121919).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Lyra Zhorniyak: Conceptualization (equal); Data curation (lead); Formal analysis (lead); Investigation (lead); Methodology (lead); Project administration (equal); Resources (lead); Software (lead); Validation (lead); Visualization (lead); Writing – original draft (lead); Writing – review & editing (equal). **M. Ani Hsieh:** Conceptualization (equal); Funding acquisition (lead); Investigation (supporting); Methodology (supporting); Project administration (equal); Supervision (lead); Writing – original draft (supporting); Writing – review & editing (equal). **Eric Forgoston:** Investigation (supporting); Supervision (supporting); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding authors upon reasonable request.

REFERENCES

- ¹N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *European Conference on Computer Vision* (Springer, 2020), pp. 213–229.
- ²S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering* (Westview Press, 1994).
- ³D. Ludwig, D. D. Jones, and C. S. Holling, “Qualitative analysis of insect outbreak systems: The spruce budworm and forest,” *J. Anim. Ecol.* **47**, 315 (1978).
- ⁴H. Haken, “Laser theory,” in Atoms, Molecules and Lasers, *Trieste, 17 January–10 April 1973* (International Atomic Energy Agency, Vienna), pp. 283–314, <https://www.iaea.org/publications/3120/atoms-molecules-and-lasers-trieste-17-jan-10-apr-1973>.
- ⁵P. Bergé and M. Dubois, “Rayleigh-Bénard convection,” *Contemp. Phys.* **25**, 535 (1984).
- ⁶E. Forgoston and I. B. Schwartz, “Delay-induced instabilities in self-propelling swarms,” *Phys. Rev. E* **77**, 035203 (2008).
- ⁷E. J. Doedel and B. Oldeman, *AUTO-07p: Continuation and Bifurcation Software* (Concordia University, Montreal, QC, 1998).
- ⁸B. Ermentrout and A. Mahajan, “Simulating, analyzing, and animating dynamical systems: A guide to XPPAUT for researchers and students,” *Appl. Mech. Rev.* **56**, B53 (2003).
- ⁹S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” *Proc. Natl. Acad. Sci. U.S.A.* **113**, 3932 (2016).
- ¹⁰J. S. Hesthaven and S. Ubbiali, “Non-intrusive reduced order modeling of nonlinear problems using neural networks,” *J. Comput. Phys.* **363**, 55 (2018).
- ¹¹F. Pichi, F. Ballarin, G. Rozza, and J. S. Hesthaven, “An artificial neural network approach to bifurcating phenomena in computational fluid dynamics,” *Comput. Fluids* **254**, 105813 (2023).
- ¹²A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Adv. Neural Inf. Process. Syst.* **30**, 5998–6008 (2017).
- ¹³S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.* **9**, 1735 (1997).
- ¹⁴R. Dey and F. M. Salem, “Gate-variants of gated recurrent unit (GRU) neural networks,” in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)* (IEEE, 2017), pp. 1597–1600.
- ¹⁵J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, “Convolutional sequence to sequence learning,” in *International Conference on Machine Learning* (PMLR, 2017), pp. 1243–1252.
- ¹⁶H. W. Kuhn, “The Hungarian method for the assignment problem,” *Naval Res. Logist.* **52**, 7 (2005).
- ¹⁷R. Jena, L. Zhorniyak, N. Doiphode, P. Chaudhari, V. Buch, J. Gee, and J. Shi, “Beyond map: Towards better evaluation of instance segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Computer Vision Foundation, 2023), pp. 11309–11318.
- ¹⁸D. J. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series,” in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (Association for the Advancement of Artificial Intelligence, 1994), pp. 359–370.
- ¹⁹A. Andrade-Campos, R. De-Carvalho, and R. Valente, “Novel criteria for determination of material model parameters,” *Int. J. Mech. Sci.* **54**, 294 (2012).
- ²⁰C. F. Jekel, G. Venter, M. P. Venter, N. Stander, and R. T. Haftka, “Similarity measures for identifying material parameters from hysteresis loops using inverse analysis,” *Int. J. Mater. Form.* **12**, 355 (2019).
- ²¹N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, “Soft-NMS—Improving object detection with one line of code,” in *Proceedings of the IEEE International Conference on Computer Vision* (Institute of Electrical and Electronics Engineers, 2017), pp. 5561–5569.
- ²²R. C. Veltkamp and M. Hagedoorn, “Shape similarity measures, properties and constructions,” in *International Conference on Advances in Visual Information Systems* (Springer, 2000), pp. 467–476.