1. Introduction

The modelling of temporal networks is an important task in many real world applications including symptom interactions for mental health [13, 11], epidemiology [9], and protein interactions [15, 8]. Temporal networks can be seen as dynamical systems: that is a system in which we have points, in our case nodes in a network, whose states, the edges connecting them, that vary dependent in time. Discovering the underlying equations governing these dynamical systems proves challenging, because changes in network structure are typically observed in the form of discrete jumps from one state to another, for example an edge between two nodes not being observed at the first time step then appearing at the next. Here, we propose a hybrid statistical and deep learning framework that allows us to model temporal networks as continuous-time dynamical systems, discover a fitting set of differential equations describing it, and, exploiting that discovery, predict the time evolution of a network.

Differential equations are useful for modelling systems where the state of one variable can effect the trajectories of other variables. We observe this behavior in temporal networks; nodes' connections within the network can influence the formation and decay of edges between other nodes, for example the phenomenon of preferential attachment observed in [5, 7], where a node is more lively to gain the connections the more connections it has. With this in mind we might wish to draw on the rich mathematical literature of differential equation modelling.

In the common representation of networks as binary-valued adjacency matrices, the events recorded in a temporal sequence of networks correspond to topological events, such as the appearance or the disappearance of link. Because of the discrete nature of events, directly modelling the temporal networks as dynamical systems would require us to handle discrete jumps. The topological nature of temporal networks,

and the discontinuous character of their temporal evolution, make it challenging to use differential equations techniques.

Here, we overcome the discreteness problem by interpreting networks as Random Dot Product Graphs; a well established statistical model for complex networks, that embeds nodes in a continuous, low-dimensional metric space by using a truncated singular value decomposition[10]. In this way we translate the hard problem of modelling discrete events in the space of networks to the easier problem of modelling continuous change in the embedding space. We then define and use systems of Neural Network Differential Equations (NNDE)[14] to approximate the time evolution of the embedding space, and symbolic regression techniques to discover the functional form of the fitted NNDEs. These functional forms are interpretable (as they read as classic differential equations) and allow us to predict forward in time the evolution of the temporal networks.

In this manuscript, we prove that the temporal network prediction problem can be successfully re-interpreted as a dynamical system modelling problem by taking the singular value decomposition of a sequence of adjacency matrices and training a NNDE to model an approximation to the underlying differential equation. We then go on to create a symbolic equation of this approximation.

We apply our proposed framework to three small example temporal networks with the hope of exploring the limitations and strengths of the proposed framework. The framework we are introducing is extremely flexible, and our research regarding the optimal structure of the Neural Networks used for the NNDEs is just started. We are confident that future research can identify more fitting Neural Network structures than the simple one adopted here. For this reason, we did not yet attempt to benchmark our model against other classic temporal network prediction methods. As it is completely general, we believe that the framework we are introducing can be

usefully applied to areas of medicine, especially protein interaction networks; population dynamics for network ecology; and social network modelling. In particular, we discuss how specific domain knowledge relative to the prediction scenario can be taken into account, moving from NNDEs to Universal Differential Equations.

2. Methodology

We are given a sequence of graphs, and our goal is to predict the next graph in the sequence. To achieve this, we reinterpret each graph as a sequence of points in a latent space; that is, a space in which similar vertices (ones with many of the same neighbours) are mapped to a similar point in space. We then Train a Neural Network Differential Equation (NNDE) to approximate the rate of change of the points in this latent space. With this numerical equation we can then use symbolic regression techniques to find a functional equation that matches the numerical equation. Using this process we discover an interpretable function that governs the evolution of a differential equation.

2.1. Singular Value Decomposition and Random Dot Product Graphs.

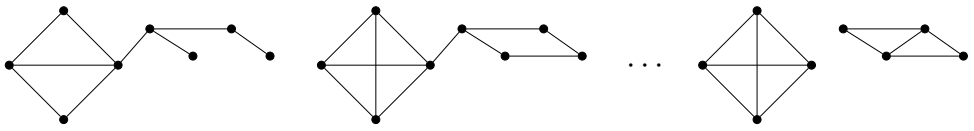
In this section we discuss the Singular Value Decomposition (SVD) of a matrix, and its usefulness in creating Random Dot Product Graphs (RDPG).

We define the matrix A to be a real $m \times n$ with $n \leq m$. A can be expressed as [1]:

$$A = L\Sigma R' \tag{1}$$

Where L, R are real valued, orthonormal matrices. The columns of L consist of the n largest eigenvectors of AA' and the columns R consist of the eigenvectors of $A'A$. Σ is a diagonal matrix whose entries are the square root of the positive eigenvalues of $A'A$ in decreasing order.

In order to reduce the problem size, we can truncate the L, R, Σ to the first d columns to yield, $\hat{L}, \hat{R}, \hat{\Sigma}$ such that:



(A) Example sequence of networks.

$$\begin{bmatrix} \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & 1 & \cdot & 1 & 1 & \cdot & \cdot & \cdot \\ 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & 1 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \end{bmatrix} \begin{bmatrix} \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & 1 & \cdot & 1 & 1 & \cdot & \cdot & \cdot \\ 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & 1 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot \end{bmatrix} \cdots \begin{bmatrix} \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 \\ \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot \end{bmatrix}$$

(B) Sequence of sparse adjacency matrices associated with the networks in Figure 1a.

FIGURE 1. Visual illustration of the proposed framework.

$$A \approx \hat{L}\hat{\Sigma}\hat{R}' \tag{2}$$

$$\hat{A} = \hat{L}\hat{\Sigma}\hat{R}' \tag{3}$$

The approximation of A is defined as $\hat{A}$ and is given by Equation 3 can be interpreted as a graph where the entry (i, j) of the matrix is the probability that an edge exists from $i \to j$. This approximation of A is known as an RDPG.

It is important to note that the SVD is not unique. The distances between points will be the same for every embedding, but the orientation may be different, and so, if the SVD of each time step is not aligned, learning will be effectively impossible as we will essentially be trying to model random noise. Here, we can use Procrustes

alignment XXX cite package XXX to align the matrices after they have been generated.

The SVD gives is an RDPG[10]. The SVD can be considered to give a latent space representation of the matrix[6]. That is, the rows of the matrices L, R can be viewed as vector points in a d-dimensional space, where d is the number of singular values we take for our SVD truncation. In this space similar nodes in the network are mapped to similar points. A visual representation of this process can be seen in Figure 1.

When obtaining the RDPGs for our temporal network, we take each network observation Figure 1a and represent them in the form of an adjacency matrix Figure 1b. We then use a truncated singular value decomposition to reduce the dimensionality of our adjacency matrices[2]. This truncation introduces some loss, but can dramatically reduce the size of our problem. Runghen, Stouffer, and Dalla Riva[16] effectively employ this method to reduce a problem involving over 130,000 visitors to a 6 dimensional latent space, whilst preserving 70% of the information.

To execute the SVD we use the ARPACK Julia package [4]. This package uses an iterative approach to approximate the eigenvectors and singular values of each of our adjacency matrices[3].

2.2. **Neural Network Differential Equation.** Once we have the SVD embedding of points, we look to model the trajectories of a target point. That is, with some input data, point location, distance to neighbours, etc, we want to model the differential equation that governs the movement of a point. Notably this method can be used to model multiple points by simply changing the node being predicted and trained on.

Our problem can be defined as:

$$A_{t+1} = f(A(t), t) \tag{4}$$

Where A_t is the adjacency matrix of the temporal network at time t. By using the SVD to obtain the approximation $\hat{A}_t$, which can evolve continuously, we can reinterpret our problem as:

$$\frac{d\hat{A}}{dt} = g(\hat{A}(t), t) \tag{5}$$

In this thesis, as a proof of concept, we focus on a modelling the change in structure of a single node, referred to as the target node u. In this case, the form of the problem becomes:

$$\frac{du}{dt} = g(u(t), t) \tag{6}$$

Because we are only modelling the movement of a single node, we assume that we have access to the rest of the network structure at every time step for the prediction. This would likely not be the case in a real world application, but will allow us to observe some of the strengths and weaknesses of this framework in a controlled setting. If we wanted to predict the whole network we would extrapolate out every point and use the dot product of each point to predict the edges.

Here we train a neural network to approximate the function $g(u(t), t) = NN(u(t), t, \theta)$. Where θ is the parameters of the neural network.

To obtain this differential equation we make use of the SciML ecosystem of packages in the Julia programming language [14].

As a way of limiting the complexity of our model, we limit the input of the neural network to the euclidean distances between the target node and its k nearest neighbours. The target node is the node that we are looking to model the movement of and stays the same throughout each of the temporal networks.

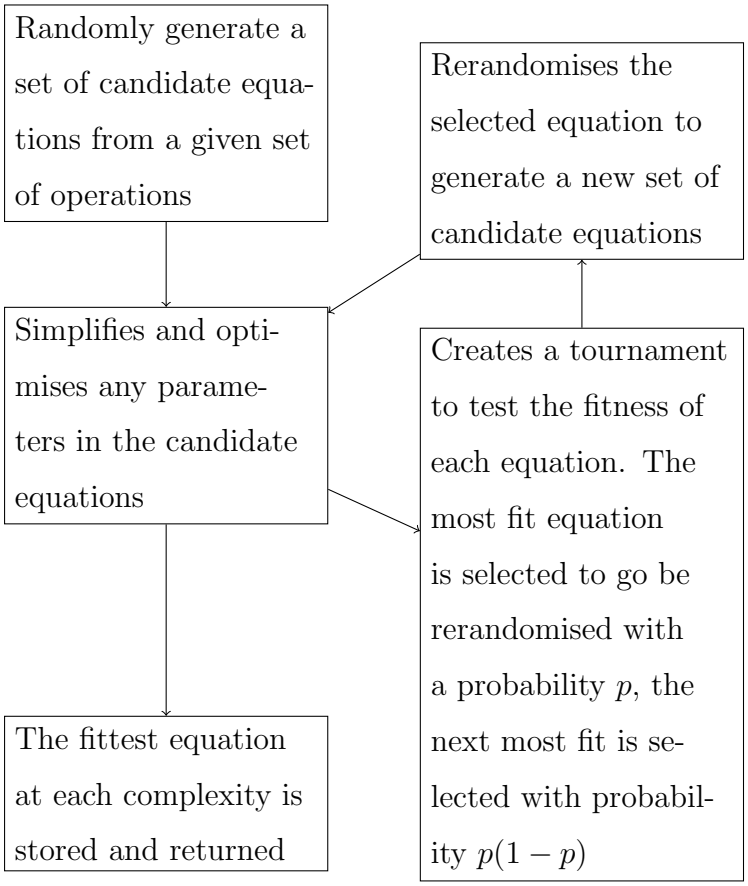


FIGURE 2. Illustration of the genetic programming symbolic regression process.

2.3. **Symbolic Regression.** With our trained NODE, we have a black box that approximates the system that governs the movement of the target node in the embedded space. To make this more interpretable we then look to approximate this black box into a combination of known functions using symbolic regression[12].

Symbolic regression is a type of regression analysis that searches a space of equations to find an expression that matches data. We use the genetic programming algorithm found in [12]. The process for which can be seen in Figure 2.

The fitness of a candidate function is determined by comparing the predicted value to the given data, as well as the derivatives of the candidate equations to numerically found partial derivatives of the data variable.

For this process we need to define a set of functions which the algorithm will use to generate a tree of functions to find the best approximation to our NNDE with. When using this method we are usually required to make two assumptions: that we have paired observations of $y(t), \frac{dy}{dt}$, and that the tree of expressions we will need is shallow. We will train the symbolic regression on the predictions of the neural network. By doing this, we can remove these assumptions[17].

2.4. **Reconstructing the Temporal Network.** To recover a final prediction, the point vector of the target node with the matrix R', this gives a vector of probabilities of connections between the target node and the rest of the network. As the results can be interpreted as a probability, the most likely graph is given by setting any entry with a value of 0.5 or higher to one (predict that the edge exists), and setting any entry with a value of less than 0.5 to zero (predict that an edge does not exist).

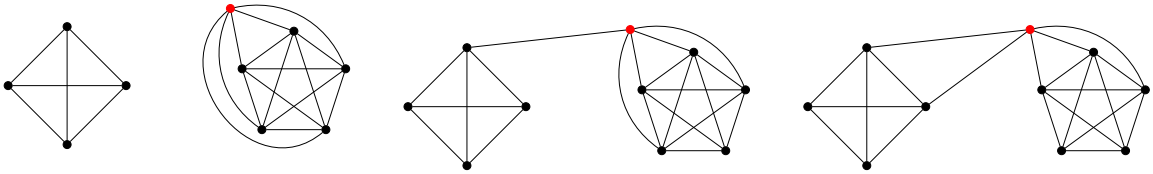
3. Data

In this paper we simulate three temporal sequences of undirected graphs. For each sequence we will be modelling the node in red, we will refer to this as the target node. For each sequence we train the model on the first 25 time steps and compare the predictions for the next 15 time steps. For clarity, each diagram Figure 3 Figure 4 Figure 5 has been simplified and illustrates the movement of the target node in each sequence.

3.1. **Two Community System.** The 2 community network has communities of size 40 and 50, each of which are fully connected. The target node is initially fully connected to the 50 node community. In our simplified diagram Figure 3a the 50

node community is represented by the 5 fully connected nodes, while the 40 node community is represented by the 4 node component. At each time step one edge is removed between the target node and the larger community and replaced with an edge between the target node and the smaller community Figure 3b. This process is then continued Figure 3c for thirty-five time steps to generate our training and test data. Once we have this data in the form of a sequence of adjacency matrices, we find the singular value decomposition using the framework described in subsection 2.1.

Figures obtained from [18].



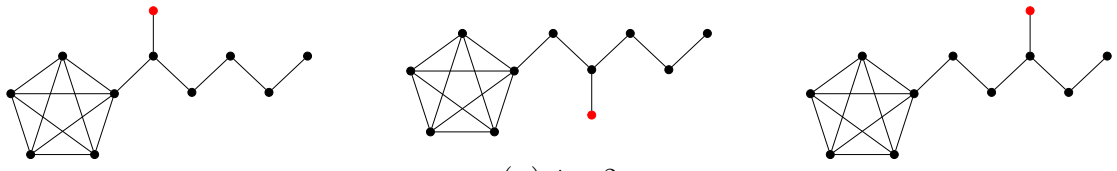
(A) $t = 1$ (B) $t = 2$ (C) $t = 3$

FIGURE 3. First 3 time steps of the simplified, synthetic 2 community network. The red node represents the target node that we are aiming to model. The 4 node and 5 node fully connected components represent the 40 and 50 node communities of the sequence respectively. At each time step, an edge from the target node to the larger community is replaced with and edge from the target node to the smaller community.

The 2 community temporal network was selected to test whether this framework can be used to model a node changing communities in systems where there are no other processes occurring. Notably, the movement in the two community system is difficult for the neural network to learn because opposite sets of inputs need to result in the same output. In a simplified version of this system with only the nearest node as input, the input for the model begins as the position of the larger community

at roughly $(0, -1)$ Figure 7a. This input needs to move the target towards the top left. However, as the sequence progresses, the nearest neighbours will be the small community at roughly $(-1, 0)$, which also needs to move the target node towards the top left.

3.2. **Long Tail System.** In the long tail network Figure 4, we have a long chain with 50 nodes and a fully connected component with forty nodes at one end of the chain. This is used to orient the embedding and model. The SVD does not take into account the index of the individual node in the adjacency matrix; it only distinguishes based on their distance to each other. Because of this a chain of nodes, even when aligned, could be flipped when it is embedded. That is, between one time step and the next the first node in the chain could be aligned with the last node. With a large, connected group at one end however, that group will always be aligned with itself and so the rest of the chain will also be aligned properly. The target node starts attached to the node that is attached to this large component Figure 4a. At each time step the target node move one node further down the tail Figure 4b. Again, this is repeated to generate the training and test data Figure 4c.

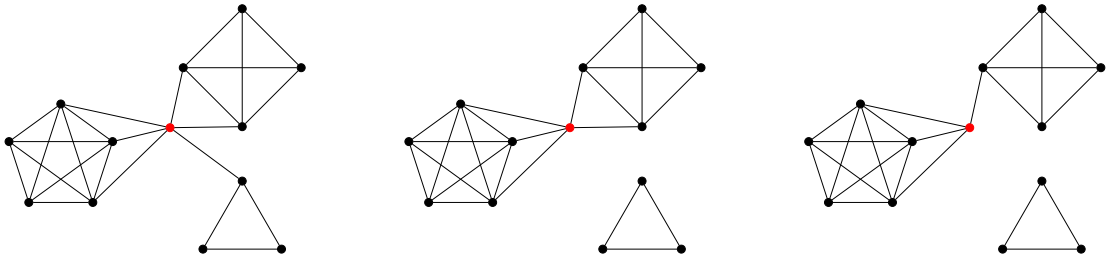


(A) $t = 1$ (B) $t = 2$ (C) $t = 3$

FIGURE 4. First 3 time steps of the simplified, synthetic long tail network. The red node represents the target node that we are aiming to model. The fully connected, 5 node component represents a 50 node community at the end of the long tail. At each time step the target node moves one step further along the chain, away from the large community.

The long tail network was selected because the SVD performs poorly on highly diagonal matrices; that is matrices with long chains. The long tail problem is difficult for the embedding because a large portion of the network is along the off diagonals; meaning that, to get an accurate embedding, the dimension of the embedding would need to be close to the length of the tail. As such we wish to see how the framework performs when it is presented with a system that the SVD cannot capture.

3.3. **Three community System.** We also simulate a sequence with three communities. The communities have sizes 40, 35, 30 and have 25, 24, and 23 edges to the target node respectively. In Figure 5 the communities have been simplified to have sizes of 5, 4, and 3. The target node starts connected to all of these communities with each community having a different number of edges Figure 5a. The community with the fewest edges between the target node will have one edge removed at each time step Figure 5b, and this process is repeated to create the required number of time steps Figure 5c.



(A) $t = 1$ (B) $t = 2$ (C) $t = 3$

FIGURE 5. First 3 time steps of the simplified, synthetic 3 community networks. The red node represents the target node which we aim to model. The 3, 4, and 5 node components represent the 30, 35, and 40 node communities respectively. At each time step the one edge is removed between the target node and the community it has the fewest connections with.

The three community was chosen as a best case scenario system to model. It has been constructed to avoid the challenges that are posed by the 2 community and long tail systems. That is, because in this system the target node is moving away from its least similar community, not moving from one community to another, it is not required to overcome the issue of symmetrical input. And because there are no long tails in this system, the SVD is able to capture a lot of the system with a low dimensional.

4. Results and Discussion

4.1. **Network Prediction.** To compare our models, we look at the loss of the predictions. This includes the loss from the SVD.

To find the approximation of the edges of node i, ${}_tA_{i,\cdot}$. We use:

$$ {}_tA_{i,\cdot} \approxeq (\bar{p})_t\hat{A} \tag{7}$$

Sequence	Embedding Loss	Neural Network Prediction Loss	Symbolic Regression Prediction Loss
2 Community	41.67	47.91	35.11
Long Tail	2.408	1.193	2.524
3 Community	35.43	35.50	35.57

FIGURE 6. A summary of the mean loss of each prediction at the fifth time step. The embedding loss comes from the true target node and loss is only introduced by the SVD. The neural network and symbolic prediction loss comes from the RDPG being reconstructed but with the position of the target node replaced with the location of the respective prediction. Loss is calculated as the total number of incorrectly predicted edges. That is an edge not being present when it should as well as the reverse.

Where $\bar{p}$ is the location of the node i in the embedding at time t.

To find calculate the loss of our prediction at time t, we use:

$$L = ||_t A_{i,\cdot} - (\bar{p})_t \hat{A}||_1 \tag{8}$$

When we do this for the SVD, NN, and symbolic regression, we get Figure 6.

The loss of one system should not be compared with another. That is the 2 community losses should not be compared to the long tail. This is because the long tail system will only ever have one edge to predict, whereas the 2 community will have 50 edges that need to be predicted.

Instead, we may compare predictions within the same system. However, we do not see much difference between these predictions, and so we plot our predictions at each time step to further understand the behavior of our framework.

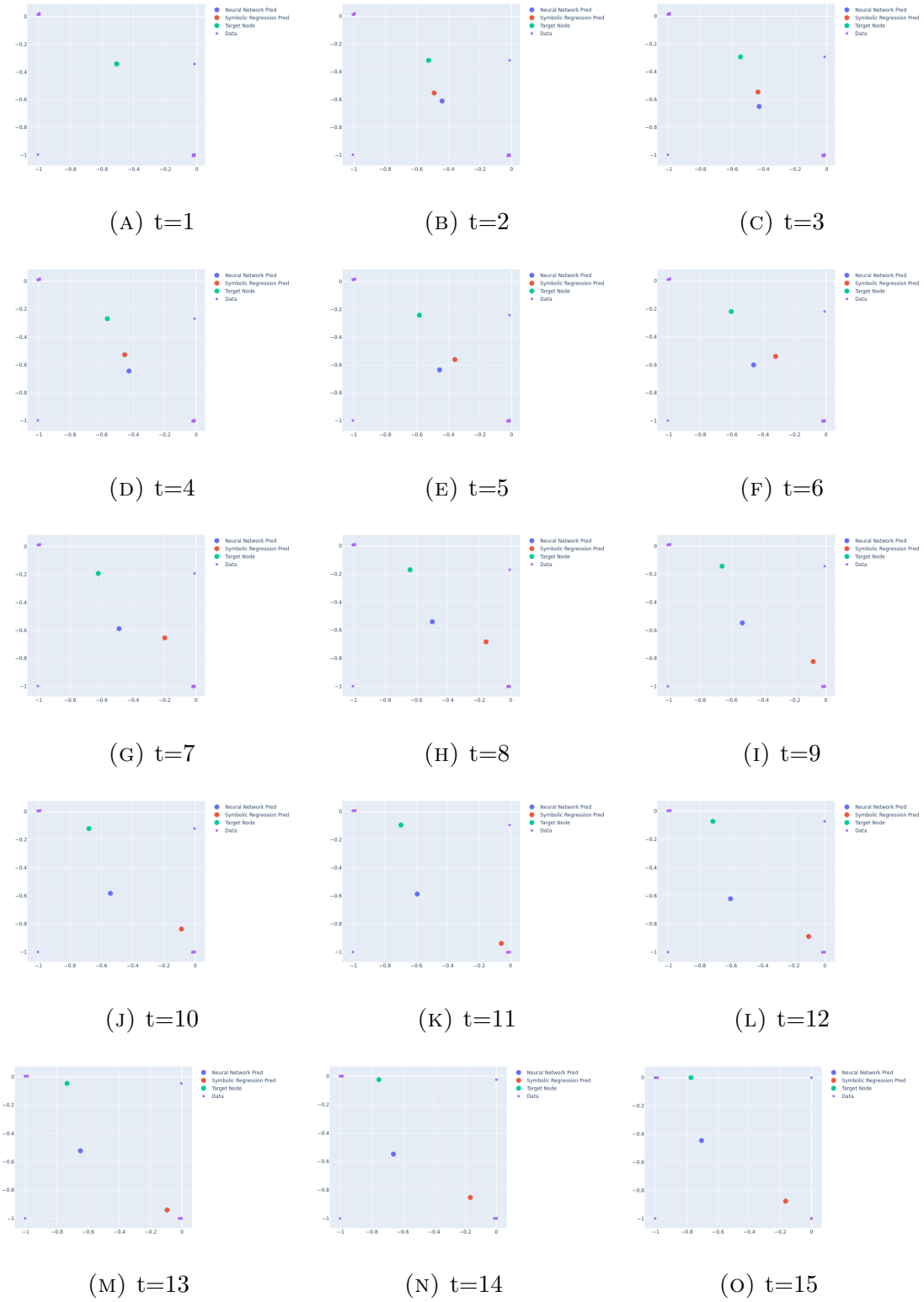
For each of these systems, we took embedded the temporal network in two dimensions at each time step. The temporal embeddings were then divided into a training set of 20 and a testing set of 15. The output of the trained neural networks was then used to train a symbolic regression model that could use a simple set of addition, subtraction, division, and multiplication.

In fig Figure 7, Figure 8,Figure 9 the green points are the embedded coordinates of the node in each of the communities. Each cluster is one separate community. The orange point is the true coordinate of the embedded target node at each time step. The blue point is our model prediction from the true target node at the first time step. As the time progresses, we see the true node move from one community cluster to the other. This is expected from the embedding; the target node starts as very similar to the first community (as it has many connections with nodes in that community) and as time progresses, it gradually becomes less and less similar to the first community and more similar to the second (as the edges between the target node and the first community are replaced with edges to the second community). As such, we see the target node move towards the second community.

4.2. **2 Community.** We see the neural network model quickly jumps to around $(-0.5, -0.5)$ where it sits for the entire simulation. This happened across many training periods; either a few time steps after the training period or right at the end of it, the neural network model moved towards a stable point where it remained for the test period.

In contrast, the symbolic regression predicted the location of the target node very well at the beginning of the test period, but after about 5 steps starts to drift away.

The poor predictive behaviour of the neural network especially may be attributed to this being a symmetrical problem. That is the same set of distances as input into the neural network, the target node needs to move away from its nearest neighbours in



(A) t=1 (B) t=2 (C) t=3

(D) t=4 (E) t=5 (F) t=6

(G) t=7 (H) t=8 (I) t=9

(J) t=10 (K) t=11 (L) t=12

(M) t=13 (N) t=14 (O) t=15

FIGURE 7. 2 Community test series. This series shows the comparison of the neural network model, the symbolic regression model trained on the neural network model, and the true solution of the two community system.

the first half, then move towards its nearest neighbours in the second half. Potential ways of breaking this symmetry may be to include information of the previous time step, or to include the absolute position of the target node as input into the neural network.

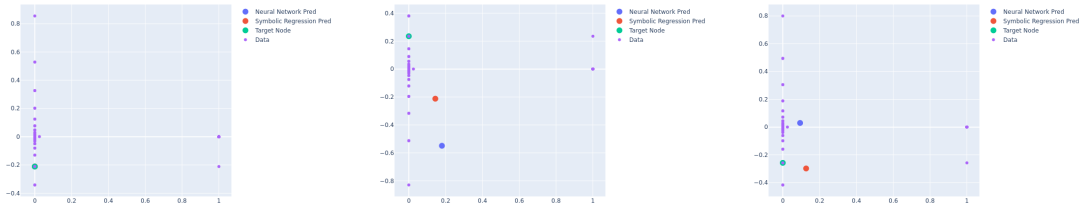
Notably, using the small neural network model to train the symbolic regression gave much better predictions than the large neural network**??**.

4.3. **Long Tail.** In fig Figure 8 we see the true embedding of the target node jump from one arm to the other at each time step. It is clear that neither the symbolic regression model nor the neural network model capture this movement. We also see that the symbolic regression model maintains a relatively stable position when compared to the neural network model.

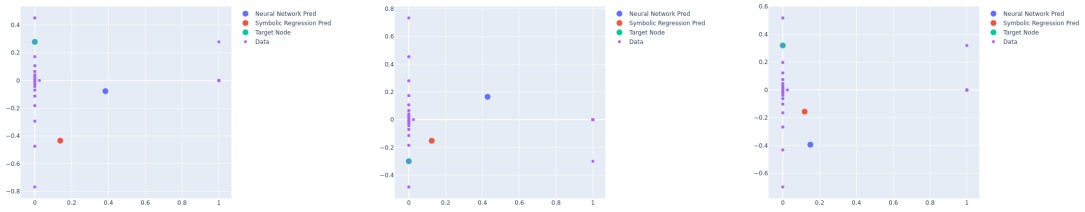
We see that neither the neural network model capture the movement of the target node to any real extent. One reason for this may again be an issue of symmetrical input. The distances from the k nearest neighbours to the target node do not seem to change much as the target node jumps from one arm to the other, but the neural network somehow needs to learn to jump left then right at alternating time steps. The solutions to this would be the same as for the two community system.

4.4. **Three Community.** In fig Figure 9 we again, as the edges between the community with the fewest edges are removed, we see the target node move away from it in the embedding and towards the other two clusters of points (communities).

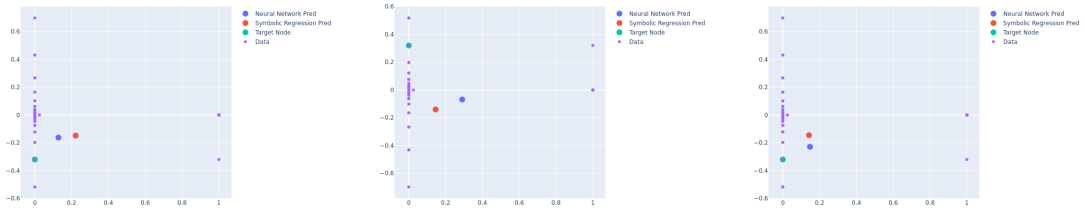
For the first ten time steps, we see similar behaviour between the neural network model and the neural network model. However, around the eleventh time step, the symbolic regression model begins to diverge from the neural network. The symbolic regression moves back towards the target node towards the end. This could indicate that the symbolic regression is a somewhat more robust model when extrapolating, but more testing should be done to corroborate this.



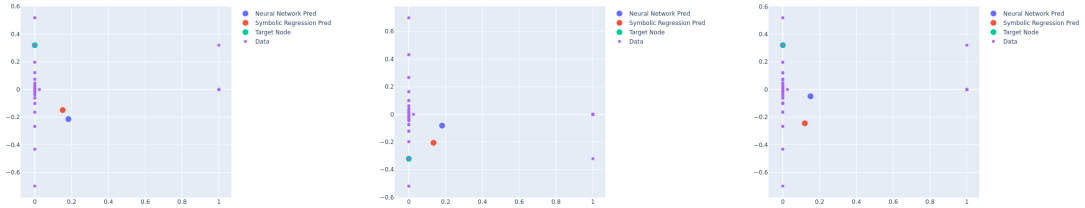
(A) t=1 (B) t=2 (C) t=3



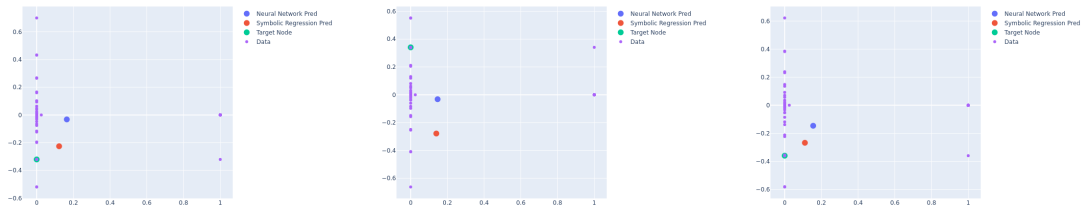
(D) t=4 (E) t=5 (F) t=6



(G) t=7 (H) t=8 (I) t=9

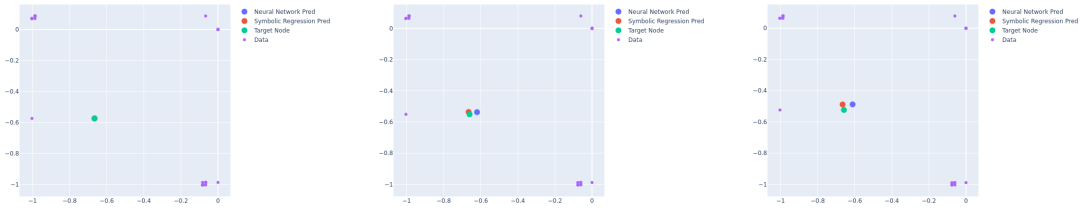


(J) t=10 (K) t=11 (L) t=12

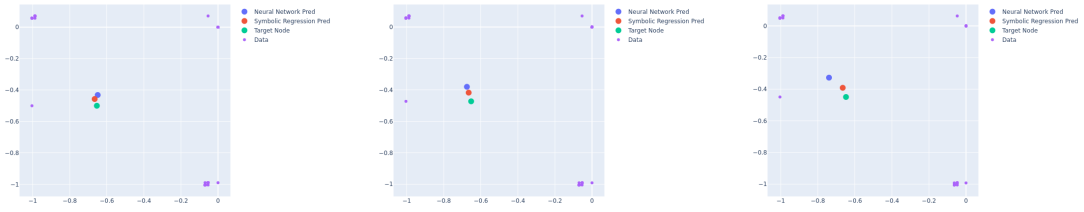


(M) t=13 (N) t=14 (O) t=15

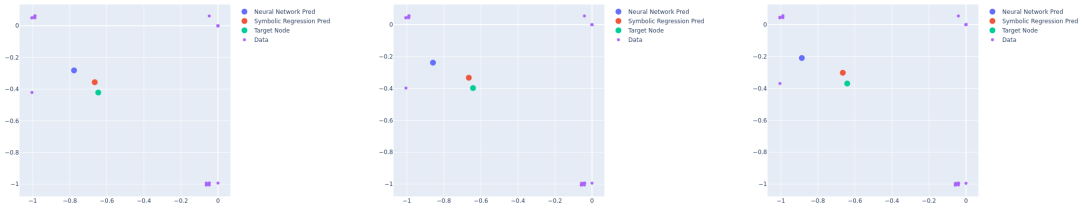
FIGURE 8. Long Tail test series. This series shows the comparison of the neural network model, the symbolic regression model trained on the neural network model, and the true solution of the long tail system.



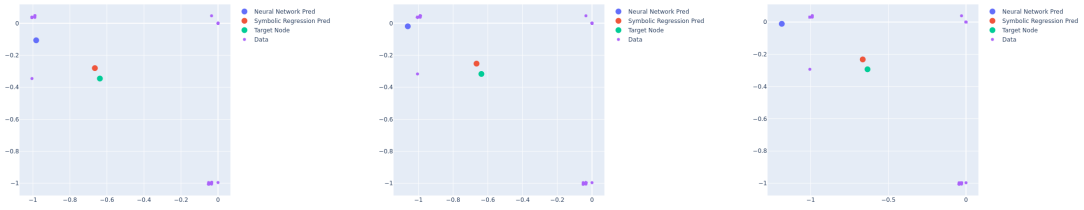
(A) t=1 (B) t=2 (C) t=3



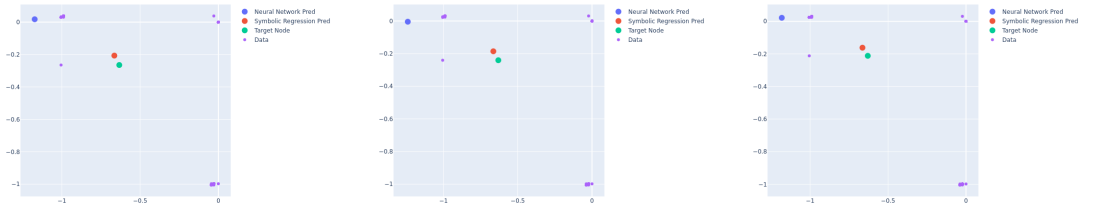
(D) t=4 (E) t=5 (F) t=6



(G) t=7 (H) t=8 (I) t=9



(J) t=10 (K) t=11 (L) t=12



(M) t=13 (N) t=14 (O) t=15

FIGURE 9. 3 community test series. This series shows the comparison of the neural network model, the symbolic regression model trained on the neural network model, and the true solution of the 3 community system.

Notably the symbolic regression trained on the large neural network performed extremely well, while the predictions of the large neural network model itself did not seem to make any significant improvement**??**. This may indicate that...XXXXX

In the neural network model, we see the prediction remain at close to its initial location for the duration of the test. This seems to indicate an attractor in the model.

In contrast, we see that the symbolic regression model quickly moves ahead of the true location where it remains for the duration of the test.

4.5. **Summary.** We see a large difference in the training predictions between the 2 and 3 community systems. In the 2 community system, even in the training period, the predictions tended to wander and be less accurate. Whereas the 3 community training predictions remained very close to the true target node. The movement of the target node in the embedding was very similar between the two systems, and so the difference seems to be that the 2 community has symmetrical inputs.

These were trained on a relatively small neural network (one hidden layer of 64 nodes), when training this small network one of the issues we encountered was that the weights would become stuck in local minima. To manage this we found that simulate annealing was helpful. When training on a larger network (4 hidden layers (64,8,8,8)) this does not seem to be as much of a problem.

5. Conclusion

In this paper, we proposed a novel framework for modelling temporal networks. We then tested this framework on three types of small, synthetic network sequences. Throughout these tests the neural network performed generally more poorly than

the symbolic regression model, with the large three community sequence being especially notable due to th symbolic regression remaining so close to the target node throughout the test period.

References

[1] George E Forsythe and Cleve B Moler. "Computer solution of linear algebraic systems". In: (1967).

[2] Gene H Golub and Christian Reinsch. "Singular value decomposition and least squares solutions". In: *Linear algebra*. Springer, 1971, pp. 134–151.

[3] Richard B Lehoucq and Danny C Sorensen. "Deflation techniques for an implicitly restarted Arnoldi iteration". In: *SIAM Journal on Matrix Analysis and Applications* 17.4 (1996), pp. 789–821.

[4] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.

[5] Mark EJ Newman. "Clustering and preferential attachment in growing networks". In: *Physical review E* 64.2 (2001), p. 025102.

[6] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. "Latent space approaches to social network analysis". In: *Journal of the american Statistical association* 97.460 (2002), pp. 1090–1098.

[7] Andrea Capocci, Vito DP Servedio, Francesca Colaiori, Luciana S Buriol, Debora Donato, Stefano Leonardi, and Guido Caldarelli. "Preferential attachment in the growth of social networks: The internet encyclopedia Wikipedia". In: *Physical review E* 74.3 (2006), p. 036116.

[8] Ruoming Jin, Scott McCallen, Chun-Chi Liu, Yang Xiang, Eivind Almaas, and Xianghong Jasmine Zhou. "Identifying dynamic network modules with

temporal and spatial constraints". In: *Biocomputing 2009*. World Scientific, 2009, pp. 203–214.

[9] Naoki Masuda and Petter Holme. "Predicting and controlling infectious disease epidemics using temporal networks". In: *F1000prime reports* 5 (2013).

[10] Avanti Athreya, Donniell E Fishkind, Minh Tang, Carey E Priebe, Youngser Park, Joshua T Vogelstein, Keith Levin, Vince Lyzinski, and Yichen Qin. "Statistical inference on random dot product graphs: a survey". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 8393–8484.

[11] Alba Contreras, Carmen Valiente, Alexandre Heeren, and Richard Bentall. "A temporal network approach to paranoia: A pilot study". In: *Frontiers in psychology* 11 (2020), p. 544565.

[12] Miles Cranmer. *PySR: Fast & Parallelized Symbolic Regression in Python/Julia*. Sept. 2020. DOI: 10.5281/zenodo.4041459. URL: http://doi.org/10.5281/zenodo.4041459.

[13] D Gage Jordan, E Samuel Winer, and Taban Salem. "The current status of temporal network analysis for clinical science: Considerations as the paradigm shifts?" In: *Journal of clinical psychology* 76.9 (2020), pp. 1591–1612.

[14] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Jasim Ramadhan. "Universal Differential Equations for Scientific Machine Learning". In: *CoRR* abs/2001.04385 (2020). arXiv: 2001.04385. URL: https://arxiv.org/abs/2001.04385.

[15] Maxime Lucas, Arthur Morris, Alex Townsend-Teague, Laurent Tichit, Bianca Habermann, and Alain Barrat. "Inferring cell cycle phases from a partially temporal network of protein interactions". In: *Cell Reports Methods* (2021).

[16] Rogini Runghen, Daniel B Stouffer, and Giulio V Dalla Riva. "Exploiting node metadata to predict interactions in large networks using graph embedding and neural networks". In: *bioRxiv* (2021). DOI: 10.1101/2021.06.10.447991.

eprint: https://www.biorxiv.org/content/early/2021/06/11/2021.06.10.447991.full.pdf. URL: https://www.biorxiv.org/content/early/2021/06/11/2021.06.10.447991.

[17] Patrick Kidger. "On neural differential equations". In: *arXiv preprint arXiv:2202.02435* (2022).

[18] E. Hogan. *Tikz Diagrams*. https://github.com/roonil-wazlib/tikz_diagrams. 2023.