Predicting Orbital Elements of Planetary Systems in Star Clusters Using Deep Neural Networks

Author : Lisa Dombrovsky, Vaibhav Vaidya, Malavika Vasist, Brendon Walter, Zhongyue Zhang
Student ID : s1504819, s2048752, s2033046, s2078864, s2009315
Professors : Simon Portegies Zwart, Wojtek Kowalczyk
Supervisor : Maxwell Cai

Leiden, The Netherlands, August 6, 2018

Predicting Orbital Elements of Planetary Systems in Star Clusters Using Deep Neural Networks

Lisa Dombrovsky, Vaibhav Vaidya, Malavika Vasist, Brendon Walter, Zhongyue Zhang

Leiden Observatory, Leiden University
P.O. Box 9500, 2300 RA Leiden, The Netherlands

August 6, 2018

Abstract

Star clusters are considered to be a chaotic environment for planet formation, since they are very densely packed with stars. A close encounter with a perturber, such as a star passing by, can cause the orbital parameters of the planetary system to be excited to such an extent that a planet is ejected from the system. In order to simulate planet formation in star clusters, N-body simulations are used. These simulations are computationally expensive and take a long time to perform. As an alternative, we attempt to use a neural network to replace part of the simulation and predict parameter evolution. We find mixed results for our models, but expect that this method can be successful with more training data and further hyper parameter tuning.

Contents

Chapter 1

Introduction

A few centuries ago, once observational instruments became advanced enough, scientists started to observe star clusters. Thanks to the success of the Kepler mission, among others, we know that most stars have planets orbiting around them. Therefore, one would expect star clusters to be filled with planets, but only about $\sim$ 1% of the detected exoplanets have been detected in such star clusters. Part of the reason that only few exoplanets have been detected in star clusters is due to observational constraints and selection bias. However, these star clusters are also considered to be extreme environments for planet formation, because they are very densely packed with stars.

It is very interesting to know more about the planets in such chaotic environments, because the conditions of the star cluster might have great effects on the outcome of planet formation. Planetary systems that reside in the clusters could be greatly influenced by other stars in the cluster, which can have a significant impact on the planetary system as they pass by. For example, an initially circular orbit of a planet could become elliptical due to a perturbation. A close encounter with a perturber might cause the orbital parameters of a planetary system to be perturbed to such an extent that a planet is ejected from the system. The aftermath of these effects could be used to set constraints on the initial conditions of the star cluster from which a planetary system was born. It is also interesting to evaluate how the perturbation impact depends on the distance between the planetary system and the perturber, the mass of the perturber, and its velocity. Knowing this will provide one of the many puzzle pieces that we need to figure out how our own Solar System was formed.

An N-body system with $N \geq 3$, such as a multi- planetary system, is very difficult to analyse analytically. Scientists, therefore, carry out direct N-body simulations. Spurzem et al. (2009) use such an N-body simulation to simulate individual planetary systems and find that close encounters with perturbers can excite the eccentricities of the planets in the system. Also with an N-body simulation, Cai et al. (2017), find that perturbations induced by stellar encounters lead to distinct signatures in the orbital parameters of planetary systems, such as excited inclinations and eccentricities of the planets. They also find that the distance to the center of the cluster influences how prone the system is to perturbations. More specifically, planetary systems that form within the cluster's half-mass radius are found to be perturbed more.

Although direct N-body simulations have already shown promising results, they are computationally expensive and the simulation time is very long. Very recently, other methods have been presented that might be able to produce the same results in a shorter time period. For example, Tamayo et al. (2016), train an XGBoost machine-learning algorithm to predict the stability of tightly packed planetary systems, which is three orders of magnitude faster than direct N-body simulations. Lam and Kipping (2018) use a deep neural network (DNN) trained on N-body simulations to predict stability predictions for circumbinary planets. The accuracy of their DNN never drops below 86% and is much faster than the simulation time of the N-body systems.

Ideally, a neural network would be able to predict the evolution of the orbital elements of a planetary system, given the initial conditions of the system and the star cluster. Such a network could be used to generate a statistical overview of all of the systems in a star cluster. We explore the possibilities of a neural network applied in this context using time-integrated simulations of planets within a star cluster, provided by Maxwell Cai of the Leiden University Observatory. We have five simulations of star clusters with different amounts of stars to our disposal. In the star cluster with 32k stars, 200 systems with 5 planets each are simulated over 50 Myr with timesteps of 1000 years to provide the time evolution of the orbital parameters, as well as the position and velocity vectors of each planet and the nearest perturber. First, the star cluster is simulated to obtain the perturber

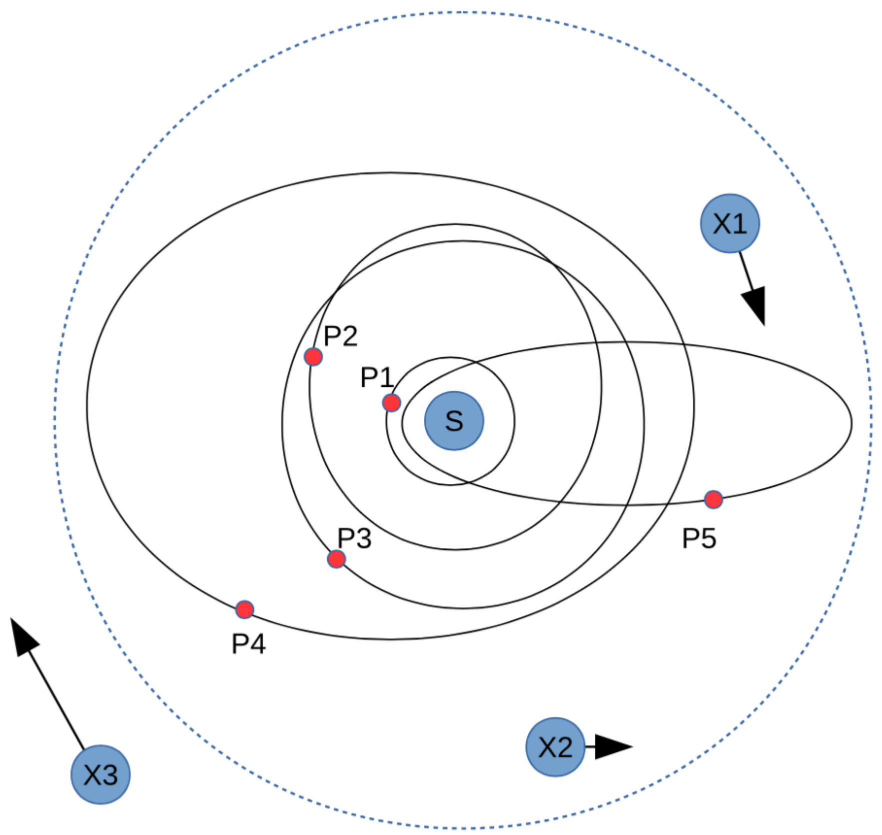

Figure 1.1: *Schematic illustration of a planetary system with multiple planets (P1-P5) being perturbed by multiple neighbor stars (X1, X2, X3). Stars X1 and X2 are sufficiently close to affect the planetary system, but star X3 is too far away to cause any harm to the system.*

information. This is then used to evolve the planetary system through the cluster. Because each system starts with five planets, there are a total of 60 parameters describing the evolution of the planetary system. A full list of parameters can be found in Table 1.1. The goal of this project is to replace the second simulation (the planetary system evolution) with a neural network in order to reduce computational time and cost. Another network is created to perform a binary prediction on planet ejections. The details of this network are described in Chapter 2. The results of the system evolution network can be found in Chapter 3.

Parameter	**Description**
Semi-major axis (a)	The farthest distance a planet gets from its star as it orbits
Eccentricity (e)	A parameter quantifying how circular or elliptical the orbit of a planet is.
Inclination (i)	The angle between the plane of the planets' orbit and the plane of the planetary system.
Time (t)	The time step.
Velocities (v_x, v_y, v_z)	The velocities of the star and each planet in the x, y, and z directions.
Positions (x, y, z)	The x, y and z positions of the star and each planet in its orbit.
Perturbation mass (m_{pert})	The mass of the perturber.
Perturber distance (d_{pert})	The distance of the perturber from the star.
Perturber velocity (v_{pert})	The velocity of the perturber in the x, y, and z direction.
Perturber position	The x, y, and z position of the perturber relative to the star.

Table 1.1: *Description of the parameters included in the time-integrated simulation output.*

Chapter 2

Binary Ejection

We design a simple neural network as a first step in the analysis to make predictions for the simulated data. We design two different neural network models. The first is to predict for each individual planet, whether it gets ejected or not. Second, we create a binary ejection model to predict if a planetary system has at least one ejection or not in the form of a 0 (No ejections) and 1 (at least one ejection). To design these models we need to select the features which would be fed into the model as an input. The selected features are as follows:

- **Number of close encounters:** This feature is extremely important to take it into the model since the ejections are directly dependent on the perturbations from the encounters from the perturbers (other systems). To calculate the number of close encounters, we need to define when would a perturbation from external influence be called a close encounter. We take two criteria into account. First, an encounter is considered to be close when the eccentricity of the planets change by a threshold value. We tried different values of threshold and finally settled to 0.01 since it gave the best results. The second criteria takes gravitational potential energy into account. An encounter is close when the M/r value (M being the mass of the perturber and r the distance) is greater than 10% of the maximum potential value. These two conditions taken together provide the number of close encounters.
- **Minimum distance of a perturber:** This feature is selected because the closer a perturber is the higher the change in the orbital parame-

ters of the planets would be.

- **Average perturber mass:** The perturber mass affects the potential energy value which is a criteria used in calculating the number of close encounters.
- **Average distance from the cluster center:** This feature is also important to consider since the density of systems towards the center of the cluster is high indicating that a system close to the center would suffer a more encounters as compared to a system far from the cluster.

In the following the sections we define the architecture, results and our interpretations of them.

2.1 Network Architecture

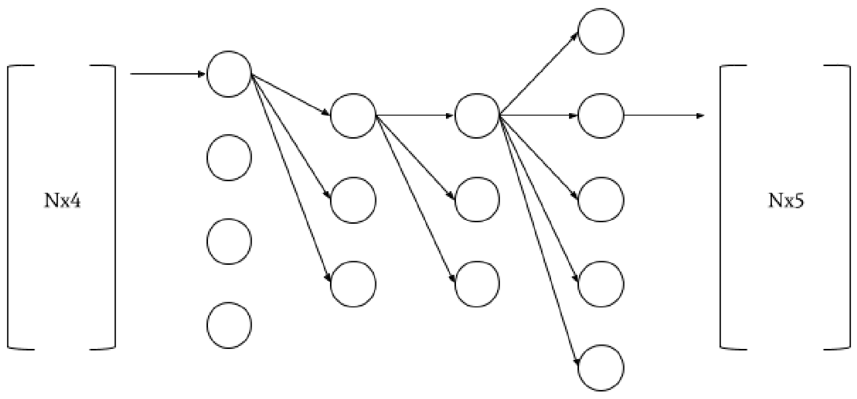

Figure 2.1: *One possible set up - 2 hidden layers with 3 nodes - of the MLP for predicting individual planetary ejections. Each node in the hidden layer uses the* `relu` *activation function while the output layer uses the* `sigmoid` *function. The loss function used is* `mse`*. This network is a fully connected network, however only some lines are shown for brevity. For each of the N systems, the network takes in four values: the number of close encounters, the minimum distance and average mass of the purtubers, and the mean distance of the system to the center of the cluster. As an output, the network predicts which of the five planets for each of the N systems has been ejected.*

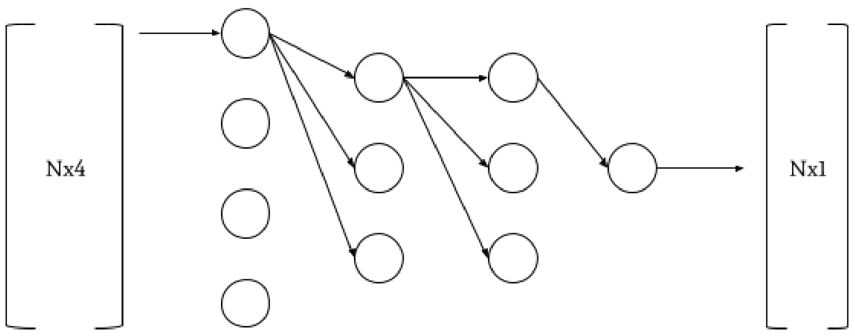

Figure 2.2: *A similar set up the network shown in Fig. 2.1, however with a binary planetary ejection prediction. In this case, the output layer only needs one node and the loss function is `binary crossentropy`. The output of this model indicates the likelihood of a system having at least one planet that is ejected.*

The planetary ejections are predicted using a simple multi-layer perceptron (MLP). We examine several different network architectures, built using the Python library Keras with a TensorFlow backend, by changing the number of hidden layers and the number of nodes in each hidden layer as well as different functions for evaluating the loss of the network.

Different network architectures result in varying levels of success depending on the simulation used (i.e., changing the number of stars in the cluster or initial size of the semi-major axis of the planet). For each simulation it is then important to tune the hyper-parameters individually. Through each simulation, the best loss and activation functions remained constant: For predicting individual planetary ejections, the best output activation function is `sigmoid` while the best loss function is `mse`; For the binary ejection model, `sigmoid` remains the best output activation function however, due to the nature of the problem, the loss function must be `binary crossentropy`. Two example models used for both problems can be seen in Figures 2.1 and 2.2.

2.2 Results

Figure 2.3 shows the predictions for each planet. The model takes all five simulations and thus predicts on all planets in all simulations taken together. The result predicts the trend that more outer planets are ejected correctly but clearly falls short in predicting the ejections of the inner planets and over-predicts the ejections of outermost planets. Overall, it over-predicts the total number of ejections. These results motivated us to try a different MLP model, the binary classification model.

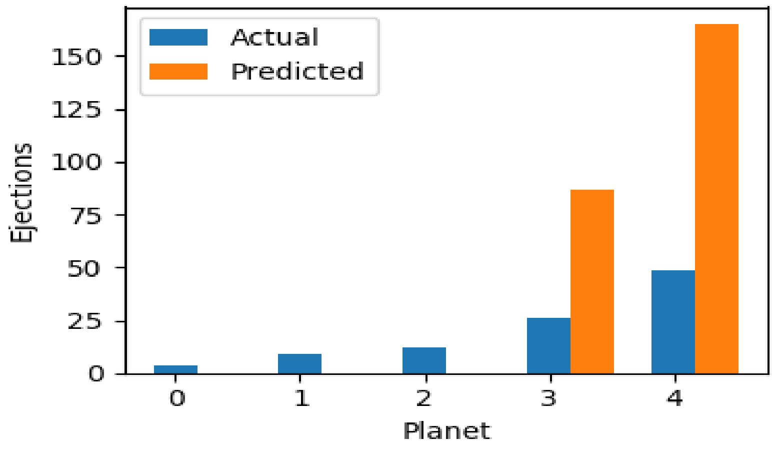

Figure 2.3: *Predictions made by the multi-layered perceptron model (orange) compared with the true ejections for all the 5 simulations available. Planet-0 is the innermost planet and planet-4 is the outermost.*

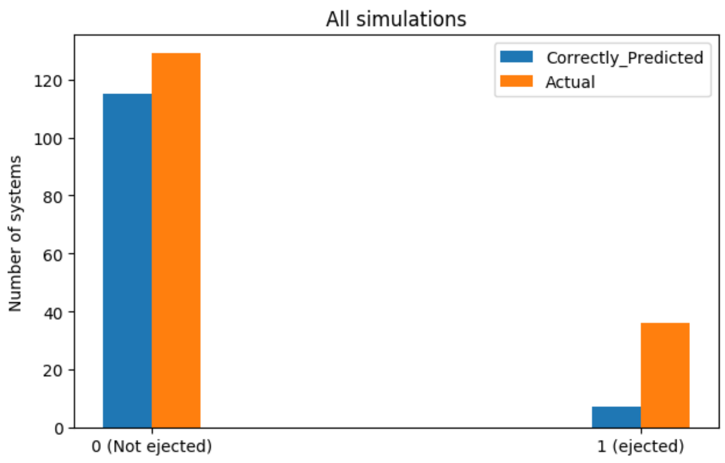

Figure 2.4: *Binary classification model predictions(blue) compared to the true values (orange)*

Figure 2.4 shows the results from the binary classification model. The model provides a binary output as the predicted value. 0 shows that no ejections were detected from the system. 1 shows that at least one ejection happened in the system. The result shows that the binary classification model does a better job than the previous model. However, it still falls short in predicting the ejections although it is quite successful in predicting the non-ejections. Availability of more simulation data may provide better results for ejection case too.

2.3 Discussion

We test the binary classification model by feeding different amount of simulation data in order to see and validate if get better results with feeding more data into the model. The results of this analysis (Figure 2.5) show that the percentage accuracy of non-ejected cases stays roughly the same (since the model is already predicting them well). The accuracy for the ejected case has a general up trend with more simulation data. There is a dip in the ejected accuracy when all the simulation data is used. We hope that this is just a local minima. Testing the model with more simulation data would provide a clarity in the trend even further.

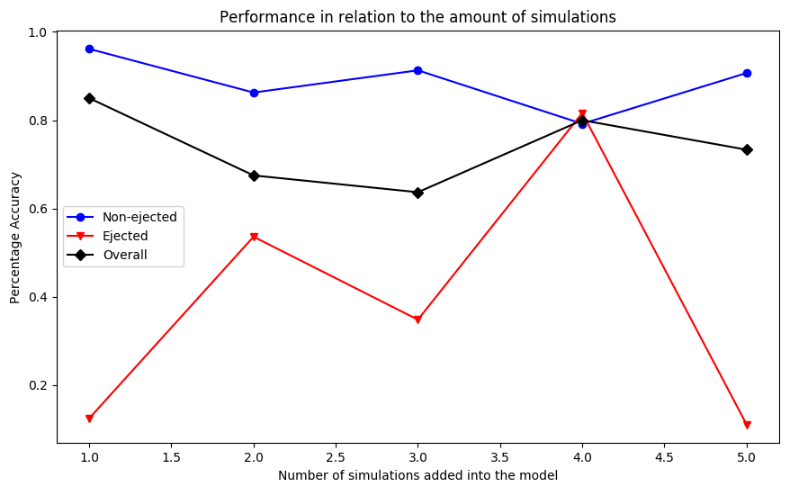

Figure 2.5: *Binary classification model predictions (blue) compared to the true values (orange).*

In the binary classification model, there may be some potential biases that may be the reason behind the limitations of the model. First, the definition of a close encounter can be improved since the threshold value was just taken by a hit-and-trial methodology. There can be more criteria involved that may have been overlooked. Second, the average distance taken from the center of the cluster may not be a good definition for the feature which represents the proximity to the cluster center. The reason behind this is that every system over its time evolution seems to be passing very close to the cluster center. Thus there is a requirement of another definition other than the average distance.

Chapter 3

Time Series Prediction

The next step beyond predicting if a planet has been ejected at the end of 50 Myr is using neural networks to predict how the orbital elements each planet change in time. This is a task prime for use with recurrent neural networks in which the output of a node is passed back into itself as an additional input allowing for forecasting of time-series events. In particular, we choose to use Long-Short Term Memory (LSTM) networks due to their ability to determine which information should be retained over long periods of time and forgetting unnecessary information.

If a network is successful in tracing how the orbital parameters change in time for individual planets, it means that neural networks are capable of replacing simulations of planetary systems for some use cases. In particular, we are less interested in how individual systems perform, preferring instead in a statistical overview of how every planetary element changes in the star cluster over time. However the performance of individual systems is still important to measure as it can give insight into the success of the statistical overview of the star cluster.

3.1 Data Structure

The input data comprises of 200 systems with 5 planets each. Specifically, we use information about the planets' time evolution in terms of changing orbital parameters over 50 Myrs with a time step of 1000 years.

The data is essentially 4-Dimensional. The dimensions include the windows of initial time steps to be predicted on, the number of such windows, the orbital parameters of the planets in each system, and the number of systems. However, since the LSTM layers in Keras take a 3D input, this data needs to be reduced by one dimension. Two approaches have been taken to do so.

The first approach considers 200 systems with 5 planets one at a time. After each system, the weights are retained and used for the next iteration. The second approach considers all the planets together, essentially viewing it as one system with 995 planets in it. A time series prediction is then performed.

3.2 Method 1 - One at a time

In this first approach, we build an LSTM model which trains on one system at a time. A single system is passed into the model and the model is trained for a certain number of epochs. Then, the next system is passed in and the model parameters are updated to reflect the new system. The weights for the next system that are initialized are the same as the weights from the old system. This way the network retains the weights and keeps refining them with each system.

There are some inherent biases with this method. Namely, the model would favor the last-seen systems as the model state was most recently updated to reflect these final systems. To counteract this, we ensure that the model sees each system multiple times. All the systems are shuffled in random order and once each system is fed through the model, the systems are reshuffled and fed through again. We dub the number of times this process happens as the number of *iterations* of the training process. It should be noted that this process does not completely solve the problem of later systems being more favored by the model, however the impact of this bias will decrease.

The data is structured into a 3-dimensional data-cube with dimensions of 200 systems by 7 features (eccentricities for the 5 planets, as well as the mass and distance of the closest perturber) by 50,000 time steps. This data-cube is then split 70-to-30 percent between the training and test data and the training data is normalized. Additionally, in the simulation data, planets which have been ejected have an eccentricity denoted as `Nan`. These

Epochs	Iterations	LSTM layers	Conv. layers
1	1	1	0
1	1	2	0
1	1	2	1
5	1	1	0
5	1	2	0
5	**1**	**2**	**1**
1	3	1	0
1	3	2	0
1	**3**	**2**	**1**
5	**3**	**1**	**0**
5	3	2	0
5	**3**	**2**	**1**

Table 3.1: *The variation in the number of epochs, iterations, and architecture of the LSTM network. For most models, no change in eccentricity was predicted. The models which preformed 'well' (i.e., the predicted eccentricity was non-zero in more than several of the test systems) are indicated in bold red.*

values are replaced by -0.01 to ensure the model is able to train properly.

Next, the training and testing data are divided into windows where the model will train on ten time steps and predict on the eleventh time step. This windowed data then has 4 dimensions: 5000 windows, by the number of systems, by 10 time steps per window, by 7 features.

Finally, the systems are fed through the network one system at a time. To figure out the best parameters for the model, we train twelve different networks with varying complexity (see Table 3.1). We try variations of only 1 iteration and 1 epoch as well as combinations of 5 epochs and 3 iterations. For each of these, we choose 3 different network architectures: the most simple one with only a single LSTM layer, a slightly more complex network with two LSTM layers, and finally the most complex network (albeit still arguably simple for the problem at hand) with one convolutional layer and two LSTM layers with dropout layers (each with a value of 0.3) after each of the three layers. Due to time constraints, we were unable to explore other combinations in more depth.

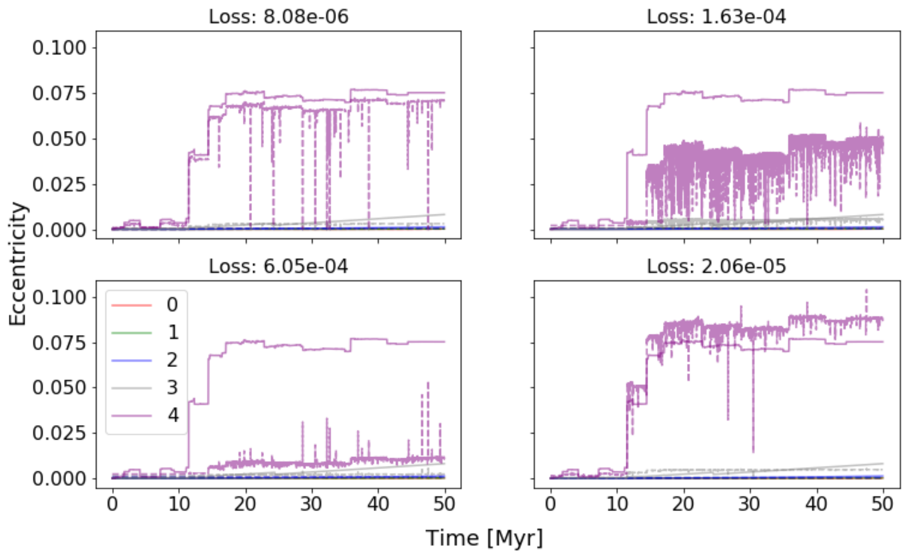

Figure 3.1: *Eccentricity predictions for each of the five planets (shown in different colors) for a single system. The solid lines depict the real values and the dashed values show the predictions. The loss function used is the* `MSE` *function.*

- ***Upper left**: 1 LSTM layer with 3 iterations and 5 epochs*
- ***Upper right**: 2 LSTM layers, 1 Conv. layer with 1 iteration and 5 epochs*
- ***Lower left**: 2 LSTM layers, 1 Conv. layer with 3 iterations and 1 epoch*
- ***Lower right**: 2 LSTM layers, 1 Conv. layer with 3 iterations and 5 epochs*

Comparisons of of additional systems can be found at the end of this section.

3.2.1 Discussion

In general we find that the more complex the network is the better it performs. The most simple networks fail to predict any change in eccentricity for the test systems while the more successful models are able to predict the eccentricity changes to varying degrees. In general, the networks are correctly able to trace the largest jumps in eccentricity however they tend to either underestimate or overestimate the predicted changes. Additionally, a lot of 'noise' seems to be present in the predictions as there is a lot of variation in the eccentricity over very short periods of time.

Figure 3.1 shows the predictions of one system using the four most successfully models highlighted in red in Table 3.1. A comparison of three additional systems for each of these four networks can be found in Figures 3.2, 3.3, 3.5, and 3.5 at the end of this section.

Some models performed better than others with the more complicated ones being able to trace the eccentricity trends with more accuracy. Others, however, under-performed for some systems while working well for others. It appears that the number of times each system is seen by the network (larger number of epochs and iterations) is more important than the architecture of the network.

Case in point, for the system shown in Figure 3.1, the model shown in the upper left has the most simple architecture yet gives one of the better predictions, most likely due to the fact that it trained with 3 iterations and 5 epochs per system.

The figures in the upper right and bottom left have the most complex network architecture, however only have 1 iteration with 5 epochs and 3 iterations with 1 epoch per system respectively. These models consistently perform worse, sometimes predicting zero change in eccentricity.

Finally, the most complex model, shown in the bottom right, has the most number of epochs, iterations, and layers and gives predictions that are comparable to that of the model shown in the upper left, however with less noise.

The results of this network are promising as it suggests that neural networks are capable of predicting how planetary orbital elements change in time. However, much work is still needed in order to fully explore these implications. We have shown that is is possible to predict how the eccentricity changes, however we have not attempted to predict how the other orbital elements, such as the inclination and semi-major axis, change in time. Further, the systematic analysis of the simulation as a whole has not been examined.

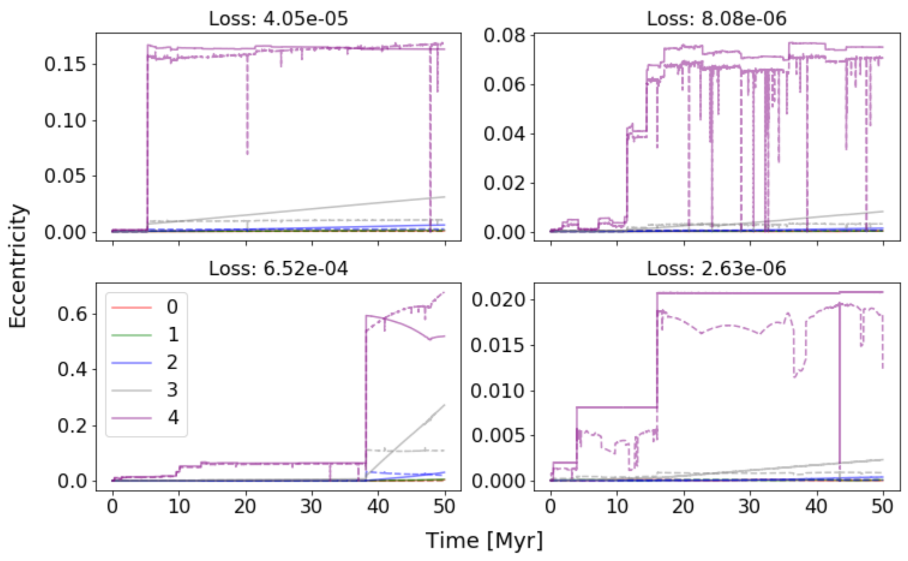

Figure 3.2: *1 LSTM layer with 3 iterations and 5 epochs.*

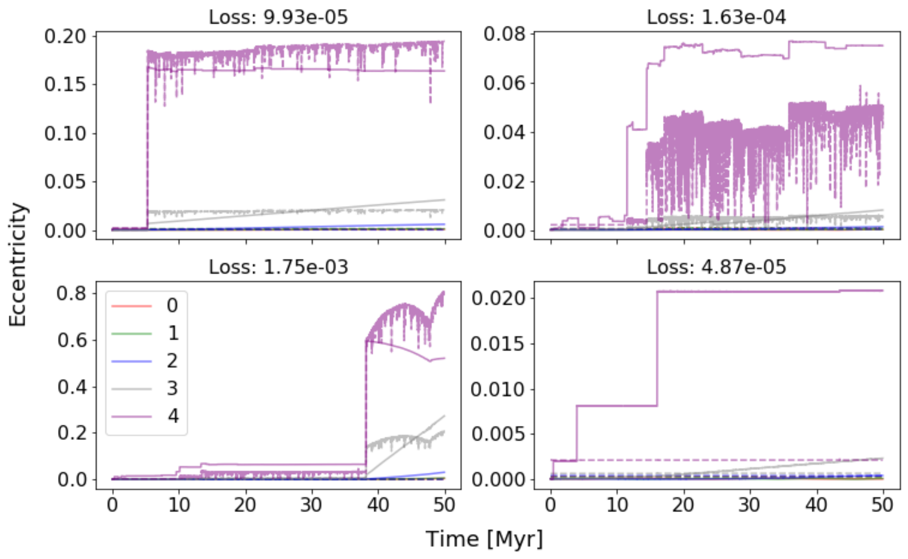

Figure 3.3: *2 LSTM layers, 1 Convolutional layer with 1 iteration and 5 epochs.*

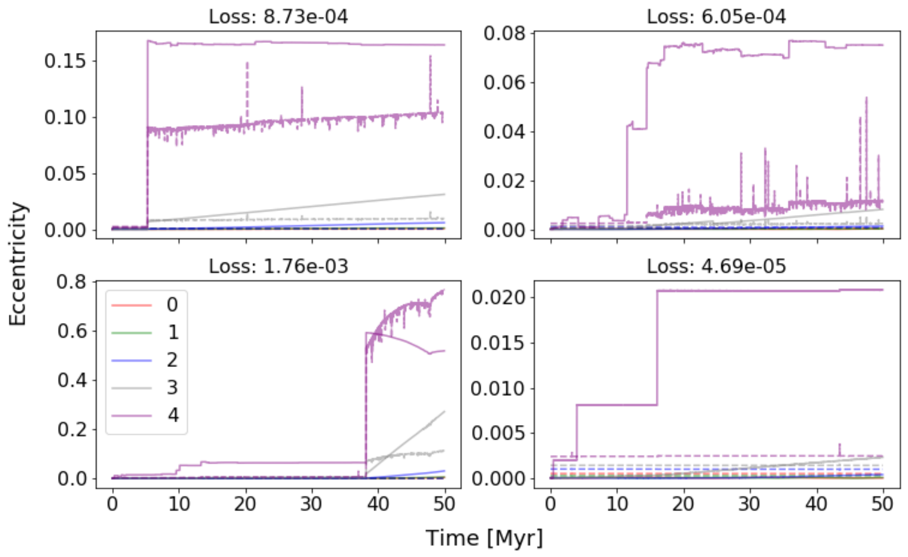

Figure 3.4: *2 LSTM layers, 1 Convolutional layer with 3 iterations and 1 epoch.*

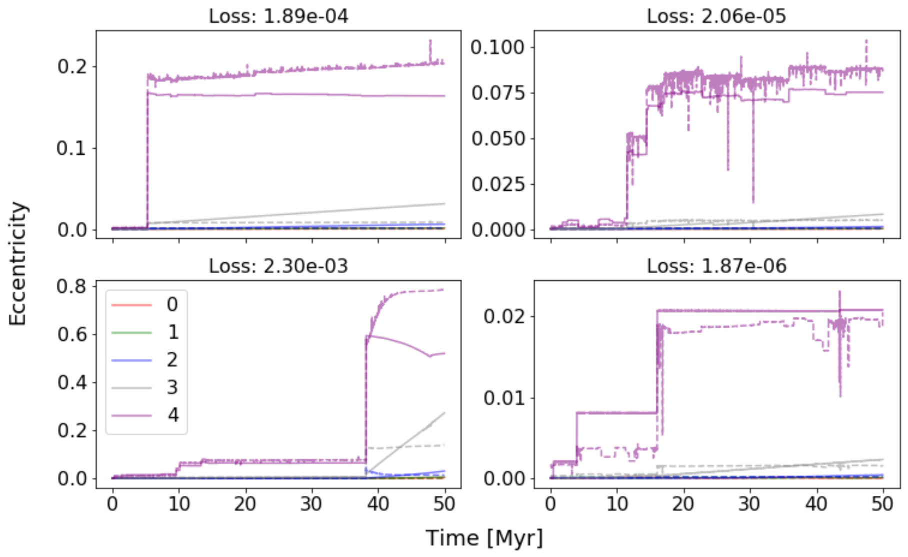

Figure 3.5: *2 LSTM layers, 1 Convolutional layer with 3 iterations and 5 epochs.*

3.3 Method 2 - All together

In the second approach, the network is trained by considering all the systems together and treating it as one big system with $200 * 5$ planets in it. This assumes no distinction between planetary systems. This approach is majorly aimed at assessing the accuracy of the network to recreate the statistics of the evolution of ejections with time. This work is based off Cai et al. (2017). The 200 systems are from the 32k model, a simulation of 32000 stars each with their own planetary system containing 5 planets each.

For each system, the training input has 17 features that include the 15 orbital parameters of the planets - semi-major axis, eccentricity, and inclination for each of the 5 planets - and the 2 perturber features - mass and distance of the closest perturber to the planetary system at each time step. This makes 3800 features for 200 systems. The training input data is thus a 3D cube with 5000 windows with a window size of 10 each with 3800 features ($5000 * 10 * 3800$).

The network is trained with the above mentioned training data with reference to the labeled data. Since this model is aimed at recreating the statistics only, the prediction is done for all the features for 4 time steps namely- 1 Myr, 5 Myr, 10 Myr and 50 Myr for all the planets in 200 systems. The statistics is then compared. The network built has 2 LSTM layers and 2 Dropout layers alternating in succession. The model is trained for 100 epochs.

The comparison between the predicted statistics and the actual distribution is plotted for several bin sizes. The time evolution is also plotted for system 0.

3.3.1 Discussion

It is known that in the beginning of the simulation most of the planets have very stable orbits and hence very low eccentricities. As time progresses it is expected for the eccentricities to become larger due to perturbations, thus changing their distribution. Therefore there are two expectations of the network. One is to accurately predict the statistics of the distribution in eccentricities at a given time instant and the second is to catch the trend of time evolution of this distribution.

The network trained isn't competent enough to predict the statistics accurately as seen from the figures. Also, the network doesn't seem to catch the trend in the evolution of statistics with time as effectively as one would hope, however this might be because of the simplistic model. With a more complex network, the quality of the predictions are expected to improve.

The network described in Method 2 was also used to make a time evolution prediction but it failed to make an accurate prediction. The network missed the sudden changes in eccentricity (Fig. 3.7) which might be because the number of features that the network is trained is simply too heavy and it is confusing the network. This is in comparison with Method 1 where the orbital parameters in the feature list included only the eccentricity while Method 2 considered inclination and semi-major axis as well. It may be worth noting that earlier experiments with Method 1 included these features as well, however with very poor results. Further analysis of including these features or not is surely needed.

The network could be trained only on eccentricity of the planets, the mean distance and mass of the perturber. Considering fewer features might improve the network. Time evolution might be attempted for this approach once the statistics prove to be accurate.

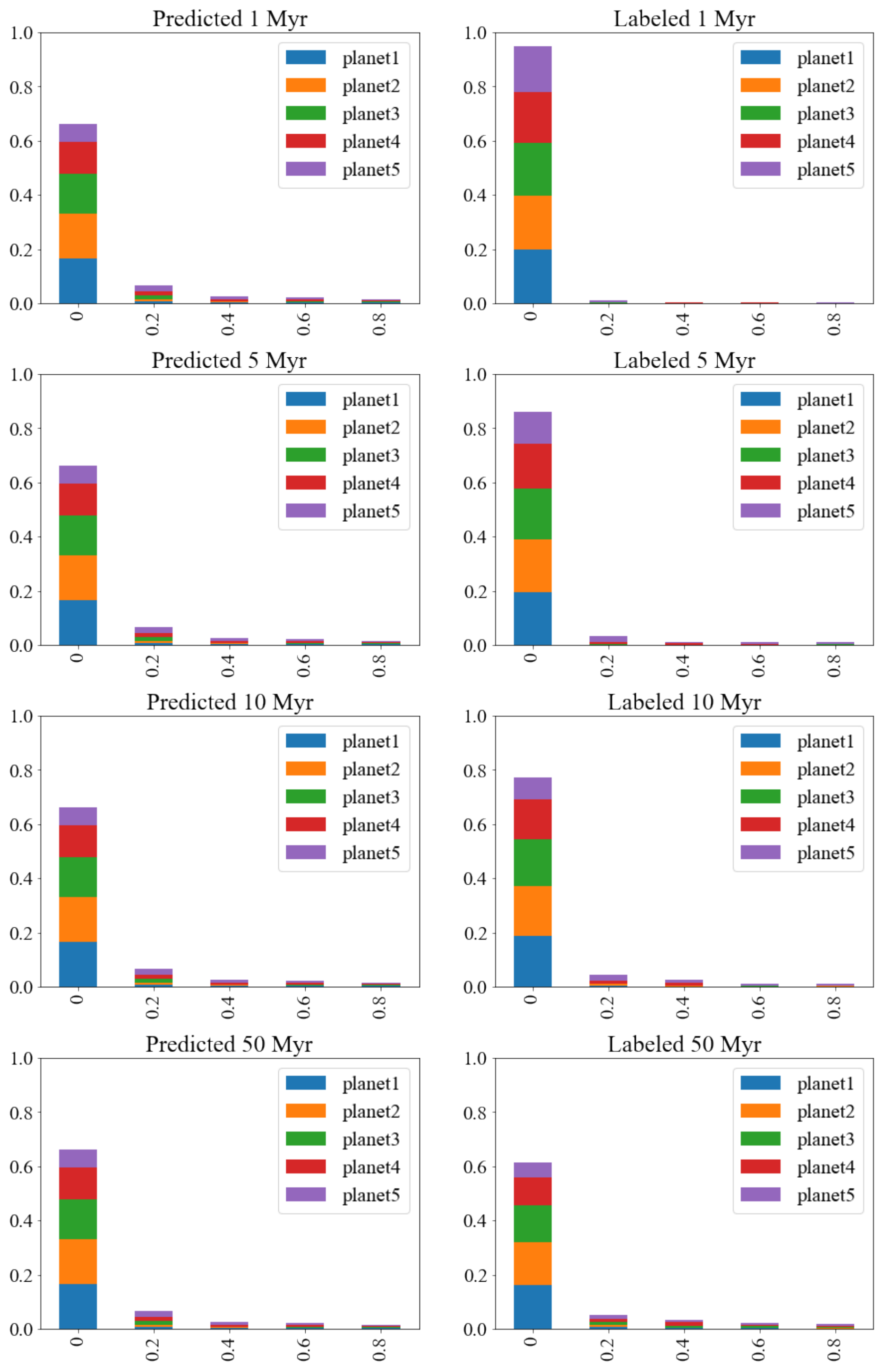

Figure 3.6: *A time evolving comparison is made between the labeled and predicted distribution of eccentricity. It is observed that the model picks up neither the distribution nor the trend in the change in the eccentricity with time.*

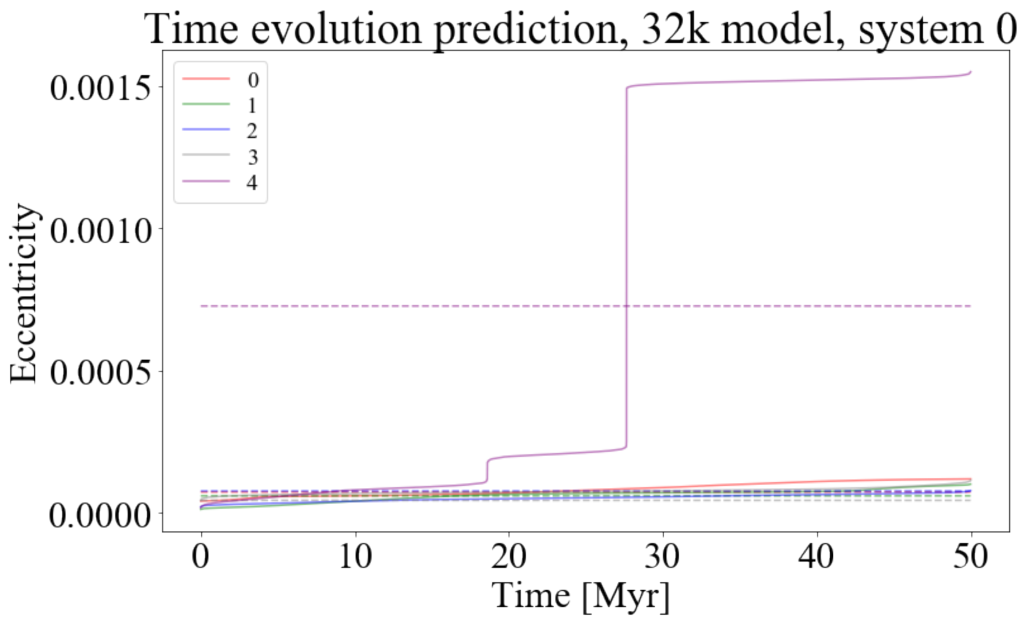

Figure 3.7: *The time evolution of eccentricity is plotted for system 0 in model 32k. The continuous lines are labeled values and the dotted lines indicate predicted values. It is evident that the model is not sensitive enough to pick up the sudden jumps in eccentricity over time.*

Chapter 4

Conclusion & Discussion

Overall, we find mixed success with using neural networks to predict how planetary systems evolve within the chaotic environment of a star cluster.

To predict planetary ejection events, we use multi-layer perceptrons to both predict which planets tend to be ejected as well as how many systems experience one or more ejections. We found that while these networks perform poorly on individual systems, the networks managed to capture a somewhat accurate although under-predicted statistical view of the cluster. Planets starting in the outer regions of the system tend to experience larger perturbations in their orbital elements due to close encounters with other objects and so are more likely to be ejected from the system. Indeed, our networks predict that the outermost planets are ejected more often. The binary classification model, which examines if any planets have been ejected from a system, performed better however the number of ejections are still under-predicted. We believe this is in part due to the lack of training data, due to the physics of each simulation being different, we cannot easily add more data by combining simulations. Thus, we are stuck with using only 200 total systems to both train and test on which is an insufficient number to ensure proper results.

Next, we build LSTM networks to take this problem a step further and predict how the systems evolve in time. Once again, we find mixed results. Method 1, which trained the network one system at a time with input features of only the eccentricity of the planets as well as information about the the closest perturbers, worked moderately well at predicting the evolution of eccentricities. Not all models created with method 1 worked so well as many of them failed to predict any change in eccentricity. Even amongst the top performing models, models in which were either more complex or saw each system numerous times, some systems proved to be more difficult to predict than others. Moreover, the models predicted large variations in the eccentricity from time step to time step. This errors can likely be reduced by building larger, more complex models as well as putting each system through the model in the training process many more times.

The second method for predicting the evolution of the eccentricity looked at a the statistical overview of the cluster. The model was aimed at predicting the distribution of eccentricity at a time instant and its evolution of trend over time. It was trained considering all the 200 star systems as a single system. The large number of features deemed the model ineffective as it confused the system. This approach failed to perform as anticipated. The model would perform better with fewer features.

Despite the mixed results, we are hopeful about the implications of our findings. Many of the errors, we believe, can be corrected by either adding more training data or taking more time to tune the hyper parameters of the networks as well as increasing the training time. A well-performing neural network could replace the need for running a full simulation of planetary evolution in star clusters for some situations, which would greatly reduce the amount of time and computational power required to gain insights into the evolution of planetary systems.

Bibliography

Cai, M. X., M. B. N. Kouwenhoven, S. F. Portegies Zwart, and R. Spurzem
2017. Stability of multiplanetary systems in star clusters. *Monthly Notices of the Royal Astronomical Society*, 470(4):4337–4353.

Cai, M. X., S. Portegies Zwart, and A. van Elteren
2018. The signatures of the parental cluster on field planetary systems. *Monthly Notices of the Royal Astronomical Society*, 474(4):5114–5121.

Lai, G., W.-C. Chang, Y. Yang, and H. Liu
2017. Modeling long- and short-term temporal patterns with deep neural networks. *arXiv:1703.07015 [cs]*. arXiv: 1703.07015.

Lam, C. and D. Kipping
2018. A machine learns to predict the stability of circumbinary planets. *Monthly Notices of the Royal Astronomical Society*, 476(4):5692–5697. arXiv: 1801.03955.

Spurzem, R., M. Giersz, D. Heggie, and D. Lin
2009. Dynamics of planetary systems in star clusters. *The Astrophysical Journal*, 697(2).

Tamayo, D., A. Silburt, D. Valencia, K. Menou, M. Ali-Dib, C. Petrovich, C. X. Huang, H. Rein, C. v. Laerhoven, A. Paradise, and et al.
2016. A machine learns to predict the stability of tightly packed planetary systems. *The Astrophysical Journal*, 832(2):L22.