Predicting the Stability of Hierarchical Triple Systems with Convolutional Neural Networks

Florian Lalande\textsuperscript{1} and Alessandro Alberto Trani\textsuperscript{1,2}

\textsuperscript{1} Okinawa Institute of Science and Technology, 1919-1 Tancha, Onna, Kunigami, Okinawa 904-0495, Japan
\textsuperscript{2} Research Center for the Early Universe, School of Science, The University of Tokyo, Tokyo 113-0033, Japan

Received 2022 June 21; revised 2022 August 26; accepted 2022 August 31; published 2022 October 11

Abstract

The dynamical stability of hierarchical triple systems is a long-standing question in celestial mechanics and dynamical astronomy. Assessing the long-term stability of triples is challenging because it requires computationally expensive simulations. Here we propose a convolutional neural network model to predict the stability of equal-mass hierarchical triples by looking at their evolution during the first $5 \times 10^7$ inner binary orbits. We employ the regularized few-body code TSUNAMI to simulate $5 \times 10^9$ hierarchical triples, from which we generate a large training and test data set. We develop 12 different network configurations that use different combinations of the triples’ orbital elements and compare their performances. Our best model uses six time series, namely, the semimajor axes ratio, the inner and outer eccentricities, the mutual inclination, and the arguments of pericenter. This model achieves excellent performance, with an area under the ROC curve score of over 95% and informs of the relevant parameters to study triple systems stability. All trained models are made publicly available, which allows predicting the stability of hierarchical triple systems 200 times faster than pure $N$-body methods.

Unified Astronomy Thesaurus concepts: Three-body problem (1695); Dynamical evolution (421); Orbital elements (1177); Convolutional neural networks (1938); Time series analysis (1916)

1. Introduction

The gravitational three-body problem is one of the oldest issues in astronomy and classical mechanics. Despite being studied since the time of Newton himself, no closed-form solution of the three-body problem exists to date, and we are limited to understanding it using numerical or statistical approaches (Monaghan 1976a, 1976b; Nash & Monaghan 1978; Mikkola 1994; Valtonen & Karttunen 2006; Trani et al. 2019; Stone & Leigh 2019; Manwadkar et al. 2020, 2021; Kol 2021).

In addition to its importance in mathematical physics, the three-body problem has numerous applications in modern astrophysics. For example, the hierarchical three-body problem, in which a binary is orbited by a distant third companion, plays an important role in shaping the architectures of planetary systems (e.g., Muñoz et al. 2016) and in forming the gravitational wave sources that are observable by LIGO-Virgo-KAGRA (e.g., Hoang et al. 2018; Trani et al. 2022).

Modeling hierarchical triples is computationally challenging even when purely Newtonian gravitational forces are considered. The most straightforward approach is the direct numerical integration of the Newtonian equations of motion. However, long-term integration is computationally expensive because the integration time step needs to be short enough to resolve the motion of the inner binary. In addition, the inevitable accumulation of numerical errors makes individual simulations unreliable and requires running several realizations of the same system in order to faithfully predict its evolution (Portegies Zwart & Boekholt 2014). Fast approximate techniques such as the double-averaging approximation can alternatively be used to direct $N$-body integration, but they have limited validity regimes (Marchal 1990; Krymolowski & Mazeh 1999; Ford et al. 2000; Breiter & Ratajczak 2005; Tremaine et al. 2009; Naoz et al. 2013; Luo et al. 2016; Grishin et al. 2017). In particular, this approach is based on the assumption of hierarchical stability, i.e., that the system will remain in a hierarchical state forever and that no energy exchange occurs between the inner binary and the tertiary object.

In fact, the hierarchical state of some triples might only be temporary, with dynamical instability manifesting even after thousands of orbital periods of apparent stability. Once a triple becomes unstable, the hierarchy is disrupted and the three bodies undergo violent and chaotic interactions, until one body is ejected and only a binary is left. The mechanisms by which some triples become unstable are still not fully understood (Toonen et al. 2022; Hayashi et al. 2022), but many authors have tried to find analytical criteria to assess the stability of hierarchical triples (Mardling & Aarseth 2001; Mushkin & Katz 2020; Hayashi et al. 2022).

On a similar problem, Tamayo et al. (2020) have developed SPOCK, the Stability for Planetary Orbital Configurations Klassifier. This is a machine-learning model capable of predicting the stability of compact multiplanet systems. SPOCK uses a gradient-boosting classifier on 10 specific features computed onto the first $10^7$ inner orbits, and tries to predict the system stability after $10^9$ orbits.

In this work, we train an artificial neural network using the time series of the Keplerian elements in order to predict the stability of hierarchical triple systems. Convolutional neural networks (CNNs; LeCun et al. 1989) have notably shown great performances for time-series classifications (Fawaz et al. 2019), and we therefore develop a CNN tailored for a stability prediction of hierarchical triple systems. We make the trained models and the source code available as an open-source package.
Section 2 introduces the methods employed in this work, including the data-set generation, the training data selection, and the design of the CNNs. Section 3 presents the classification results of the networks and the diagnosis of the selected best model. Finally, Section 4 addresses possible limitations and summarizes our work.

2. Methods

In this section we first describe the generation of the data set of hierarchical triples (Section 2.1), next, we introduce the set of Keplerian elements and their transformations used as input parameters for training (Section 2.2), and finally provide the description of the various CNN architectures as well as the training strategy (Section 2.3).

2.1. Data Set of Hierarchical Triple Systems

We use TSUNAMI (Trani et al. 2019), a modern regularized code suited to evolve few-body self-gravitating systems. The algorithm can handle close encounters with extreme accuracy thanks to a combination of numerical techniques, namely algorithmic regularization (Mikkola & Tanikawa 1999), chain coordinate system (Mikkola & Aarseth 1993), and Bulirsch-Stoer extrapolation (Stoer & Bulirsch 2002).

In the algorithmic regularization scheme, we solve the equations of motion using a second-order leapfrog derived from a transformed Hamiltonian extended in phase space. In the resulting kick-drift-kick scheme, time becomes an integrated variable, just like positions and velocities, because the system is advanced along a new independent variable that can be considered as a fictional time. The advantage of this algorithm is that the integration time step can remain large even for collision orbits, where the acceleration going to infinity would cause numerical problems in traditional integration schemes. Because the leapfrog scheme is only second order, we increase the accuracy of the integration by employing Bulirsch-Stoer extrapolation with an adaptive time step. The chain coordinate system is instead a coordinate system alternative to the more common center-of-mass coordinates, which serves to reduce round-off errors when two close particles are very far from the center of mass. TSUNAMI comes with a Python interface, which makes it particularly convenient for the generation of our data set of hierarchical triple systems.

The hierarchical triple systems we consider comprise three bodies of $10M_\odot$, which we refer to as 1, 2, and 3. Bodies 1 and 2 form the inner binary of the hierarchical system. The outer orbit is composed of the inner binary center of mass and body 3. The initial parameters of the system are sampled such that we observe both long-lived systems and disruptions.

We sample the initial inner eccentricity with the square of a uniform distribution ($e_1 \sim U[0, 1]^2$) and uniformly sample the initial outer eccentricity ($e_2 \sim U[0, 1]$). The distribution for the inner binary eccentricity was inspired by the $e_1$ distribution of triples formed in stellar cluster simulations (Trani et al. 2022). We sample the initial inner semimajor axis from a log-uniform distribution between 1 and 100 au ($a_1 \sim U[log 1, log 100]$). We generate the initial mutual inclination by uniformly sampling the arc cosine of the angle ($\cos i_{\text{mut}} \sim U[-1, 1]$). Then, we uniformly and independently sample both arguments of periapsis ($\omega_1, \omega_2 \sim U[-\pi, \pi]$). Next, we sample the initial mean anomaly for the inner and outer binaries uniformly

\begin{align*}
\frac{m_1}{M_1} &= 10M_\odot \\
\frac{m_2}{M_2} &= 10M_\odot \\
\frac{m_3}{M_3} &= 10M_\odot \\
e_1 &\sim U[0, 1]^2 \\
e_2 &\sim U[0, 1] \\
a_1 &\sim \exp U[\ln 1; \ln 100] \\
a_2 &\sim U[0.85, 0.95] \times a_1^{\text{lim}} \\
i_{\text{mut}} &\sim \arccos(U[-1, 1]) \\
\omega_1, \omega_2 &\sim U[-\pi, \pi] \\
M_1, M_2 &\sim U[0, 2\pi]
\end{align*}

(1)

For our simulations, we uniformly sample the initial semimajor axis of the outer binary $a_2$ in a range slightly below $a_2^{\text{lim}}$: $a_2 \sim U[0.85, 0.95] a_2^{\text{lim}}$. This sampling strategy prevents us from simulating obviously stable or unstable triple systems. The uniform distribution range of [0.85, 0.95] has been chosen empirically, such that we obtain both long-lived and medium-lived systems in a reasonable amount of computational time. In the most unstable scenario, we have $a_2 = 0.85 a_2^{\text{lim}}$, and the system is likely to undergo fast disruption; while in the most stable scenario, we have $a_2 = 0.95 a_2^{\text{lim}}$, and the simulation is expected to last longer. Very short-lived simulations are not computationally expensive and can simply be discarded.

Finally, we make use of the numerical stability criterion of Mardling & Aarseth (2001) to compute a theoretical limit for the semimajor axis of the outer binary $a_2^{\text{lim}}$. Mardling & Aarseth (2001) proposed a simple expression to assess the stability of hierarchical triple systems, empirically derived from simulations. This criterion aims at providing a boundary between stable and unstable initial configurations for hierarchical triple systems. According to their criterion, the triple is stable if the outer semimajor axis $a_2$ is greater than

\begin{align*}
a_2^{\text{lim}} &= 2.8 \left( \frac{(m_1 + m_2 + m_3)(1 + e_2)}{(m_1 + m_2)(1 - e_2)^{3/5}} \right)^{2/5} \\
&\times \left(1 - \frac{3i_{\text{mut}}}{\pi} \right) a_1.
\end{align*}

Figure 1. Initial configuration and input parameters for the hierarchical triple systems simulated with TSUNAMI. The masses are fixed (in blue) for all simulations, while all other parameters are sampled (in red) to explore the initial parameter space. The outer semimajor axis $a_2$ is sampled such that we expect medium-lived and long-lived systems, while avoiding obviously stable configurations.

Figure 1 shows the initial configuration and input parameters for the hierarchical triple systems simulated with TSUNAMI. The masses are fixed (in blue) for all simulations, while all other parameters are sampled (in red) to explore the initial parameter space. The outer semimajor axis $a_2$ is sampled such that we expect medium-lived and long-lived systems, while avoiding obviously stable configurations.
broken and dynamically unstable, but the system might still take a long time to finally decay into an unbound-binary single (e.g., Manwadkar et al. 2020). For this reason, we adopt the above condition on the semimajor axes ratio as our definition of instability. While some oscillations of the semimajor axis ratio can occur over short timescales, we note that after $a_1/a_2$ changes significantly (about 15%), the energy transfer between inner and outer orbits can no longer stop, and the triple will eventually break.

It is worth noting that our criterion for instability ignores the effects of the secular evolution of triples, such as the von Zeipel-Kozai–Lidov mechanism (von Zeipel 1910; Kozai 1962; Lidov 1962), which causes long-term oscillations in mutual inclination and inner binary eccentricity of highly inclined triples. We do not regard the secular evolution as an instability itself, but as a process that can trigger the instability in terms of an energy exchange between inner and outer orbit. For example, the inner binary eccentricity may increase due to the von Zeipel-Kozai–Lidov mechanism, which then causes the inner binary to become more perturbed (see Section 6 of Hayashi et al. 2022). Eventually, this will manifest in a change of the semimajor axis ratio. The reverse is not always verified: secular evolution does not necessarily trigger the dynamical instability, and the triple may proceed to evolve in a stable configuration.

During simulations, we periodically sample the following eight parameters: current time in years $t$, inner orbit eccentricity $e_1$, outer orbit eccentricity $e_2$, inner orbit major-axis $a_1$, outer orbit major-axis $a_2$, mutual inclination between inner and outer orbits $i_{\text{mut}}$, inner orbit argument of periapsis $\omega_1$, and outer orbit argument of periapsis $\omega_2$. These parameters are recorded every $\Delta t = 5P_1$, from the beginning of the simulation until $5 \times 10^5 P_1$, that is, 0.5% of the total simulation duration. This arbitrary cutoff at 0.5% is a tradeoff between accuracy, which increases by proving longer time series, and computational efficiency, which naturally decreases for higher cutoffs. The relevant data correspond to seven time series of the following orbital elements: $e_1$, $e_2$, $a_1$, $a_2$, $i_{\text{mut}}$, $\omega_1$, and $\omega_2$. Note that these seven attributes completely describe the secular evolution of the system. Therefore, other quantities such as angular momentum or orbital energies can be derived from these seven orbital elements or their combination. If any of these derived quantities is fundamentally important for our classification problem, we expect the network to find them automatically.

We simulate the evolution of 5,000,000 triple hierarchical systems with TSUNAMI. Since we are interested in the analysis of the first 0.5% of the maximum simulation lifetime, we discard all simulations that underwent disruption before $5 \times 10^5 P_1$. This corresponds to about 90% of our simulations. Note that this process may introduce a bias in our data set because the neural network will not be trained on very short-lived ($< 5 \times 10^5 P_1$) simulations. Including all 5,000,000 simulations would have led to artificially high classification performances as the very short-lived simulations break up too quickly. Since very short-lived systems can be simulated until disruption at a negligible computational cost, we decided to focus on the critical simulations that develop the instability much later in time, and are more complicated to classify. Training an artificial neural network on imbalanced data sets remains a complex open problem (Fernández et al. 2018).

About 5% of our simulations were found to be medium-lived and underwent disruption between $5 \times 10^5 P_1$ and $10^6 P_1$. These 238,289 simulations are included in the data set and are labeled “unstable”. Finally, 241,456 simulations (another 5% of the total) reached the final time $t_f$ without disruption. We include these simulations in the data set as “stable”. We purposefully discarded the simulations for which $t_f < 100 P_2$, because the third body does not have enough time to interact with the inner binary, making the simulations appear artificially stable. As a result, our hierarchical triple systems data set is comprised of 479,745 simulations, with approximately half being stable and the other half unstable. Figure 2 shows a summary of the simulation results and the hierarchical triple systems data-set generation.

### 2.1.1. Comparison with the Mushkin & Katz Estimate

It is worth noting that the unstable simulations of our data set are strongly biased toward short-lived systems. As can be seen in Figure 3, the disruption time distribution of our TSUNAMI unstable simulations decreases exponentially.

For comparison, we computed the estimated disruption time of hierarchical triple systems following Mushkin & Katz (2020). Their estimate was derived after modeling the disruption of hierarchical three-body systems by a random-walk process. Mushkin & Katz (2020) estimate the number of inner orbital periods $N$ before a triple disrupts as

\[
N = (1 - e_2^2) \left( \frac{a_2(1 - e_2)}{a_1} \right)^{-7/2} \left( \frac{(m_1 + m_2)^2}{m_1 m_2} \right)^2 \times \left( \frac{m_1 + m_2 + m_3}{m_1 + m_2} \right)^{5/2} \times \exp \left( \frac{4\sqrt{2}}{3} \sqrt[3]{m_1 + m_2 + m_3} \left( \frac{a_2(1 - e_2)}{a_1} \right)^{3/2} \right),
\]

which for our equal-mass systems with $m_1 = m_2 = m_3 = 10 M_\odot$ reduces to

\[
N = \frac{36\sqrt{3}}{\sqrt{2}} \left( \frac{a_2}{a_1} \right)^{-7/2} (1 - e_2)^{-3/2} \times \exp \left( \frac{8}{3\sqrt{3}} \left( \frac{a_2}{a_1} \right)^{3/2} (1 - e_2)^{3/2} \right).
\]
The Astrophysical Journal, 938:18 (9pp), 2022 October 10

Lalande & Trani

Figure 3. Disruption time distribution for TSUNAMI unstable simulations and comparison with the Mushkin & Katz (2020) disruption time estimates. The y-axis is logarithmic: our unstable data set (in orange) is very biased toward short-lived simulations. The blue histogram displays the distribution of disruption time estimates following Mushkin & Katz (2020), where the predictions below $5 \times 10^5 P_1$ have been removed (382,183 – 295,746 = 86,437). The contingency table shows the comparison over all simulations: the approximation of Mushkin & Katz (2020) overestimates instability.

| Channel number | Value | Range |
|----------------|-------|-------|
| 0              | $e_1$  | [0, 1]|     |
| 1              | $e_2$  | [0, 1]|     |
| 2              | $a_1$  | $\mathbb{R}^+$|     |
| 3              | $a_2$  | $\mathbb{R}^+$|     |
| 4              | $\frac{\omega_1 + \pi}{2}$ | [0, 1]|     |
| 5              | $\frac{\omega_2 + \pi}{2}$ | [0, 1]|     |
| 6              | $\frac{\omega_1}{a_1}$ | $\mathbb{R}^+$|     |
| 7              | $\frac{\omega_2}{a_2}$ | $\mathbb{R}^+$|     |
| 8              | $\frac{\omega_1 + \omega_2}{a_1 + a_2}$ | $\mathbb{R}^+$|     |

Note. The raw time series are standardized to facilitate the information processing of the CNNs. Various combinations of the presented input channels are fed into the CNNs for classification.

First, the mutual inclination $i_{\text{mut}}$ and the arguments at periapsis $\omega_1$ and $\omega_2$ are scaled by $\pi$ and $2\pi$, respectively, to lie in the interval $[0, 1]$. Second, we compute derived quantities from the semimajor axes $a_1$ and $a_2$. The ratio of semimajor axes $a_1/a_2$ and the ratio

$$a_1(1 + e_1) 
\frac{a_2(1 - e_2)}{e_2}.$$

This second quantity, which we term apsidal separation ratio, is the ratio of the inner binary apocenter distance to the outer binary pericenter distance, and estimates how well the two orbits are separated. Both quantities are dimensionless and normalized to 1 at $t = 0$. Figure 4 displays all available input channels for a random simulation in each class.

Because some channels are redundant, we select only a subset of input channels for the CNN. Six sets of input channels are tested:

1. Channels $\{0, 1, 4, 7\}$ as benchmark.
2. Channels $\{0, 1, 4, 5, 6, 7\}$ to assess the importance of the arguments of periapsis.
3. Channels $\{0, 1, 4, 8\}$ to compare the ratio of the semimajor axes to the apsidal separation ratio.
4. Channels $\{0, 1, 8\}$ to assess the importance of the mutual inclination.
5. Channels $\{4, 7\}$ to evaluate the network performances in the absence of the eccentricities.
6. Channels $\{4, 8\}$ to check whether the information on the eccentricities encoded in the apsidal separation ratio is sufficient.

The hierarchical triple-system data set is comprised of 241,456 stable simulations and 238,289 unstable simulations. We create a training data set of 150,000 stable and 150,000 unstable simulations. We use all remaining simulations—that is, 91,456 stable and 88,289 unstable simulations—for testing. The lower part of Figure 2 shows the split of the original data set between training and test data sets. Having slightly morestable than unstable simulations in the test data set is not an issue when testing.

2.3. CNN Designs

Convolutional neural networks (CNNs) are a type of artificial neural networks that is tailored for image processing. They are...
translation invariant thanks to the use of convolutions with trainable filters sliding along the input data (LeCun et al. 1989).

In spite of their traditional use for two-dimensional image data, CNNs can well be applied for one-dimensional time-series data processing. CNNs have been shown to provide state-of-the-art performances for time series classification (Fawaz et al. 2019). For this reason, we decide to develop two CNN models for the binary classification of time series.

We introduce two CNN model architectures to account for different dependences between input channels. Architecture A and architecture B are presented in Appendix A. The activation functions between the network layers are presented in Appendix B. Both network architectures were inspired from well-established CNNs for classification tasks (e.g., Alexnet or VGG16; Russakovsky et al. 2015; Krizhevsky et al. 2017). We adapted their two-dimensional architectures to match our time-series input data.

All CNNs are trained with the same strategy, regardless of the set of input channels used. We use the Keras library (Chollet 2015) for TensorFlow (Abadi et al. 2016) in Python. We use the Adam optimizer (Kingma & Ba 2015) to minimize the binary cross-entropy with respect to the trainable parameters during training. Details of the training strategy are provided in Appendix C.

3. CNN Predictions

As we use two CNN architectures with six sets of input channels for the training data set, we are left with 12 configurations to analyze. We label them in the same fashion as $A_{014567}$, where the first letter (either A or B) stands for the CNN architecture type, and the trailing series of digits corresponds to the input channels used during training. This section first presents the overall CNN performances (Section 3.1), and then focuses on the diagnosis of a chosen model (Section 3.2).

3.1. Overall CNN Performances

Figure 5 shows the receiver operating characteristic curve (ROC curve) for the 12 configurations under study. The area under the curve (AUC) is given in parentheses next to each configuration name. The AUC is to be interpreted as the probability that given a random observation from class 0 (stable simulation) and a random observation from class 1 (unstable simulation), the CNN outputs a higher value for the latter. In other words, it is the probability that the CNN determines which one is which. The right panel of Figure 5 is a closer look at the upper left corner of the left panel.

First, we see that both CNN architectures are capable of predicting whether a hierarchical triple system is stable, provided the beginning of the evolution of its Kepler elements.

This result holds for most input channels: only sets $\{4, 7\}$ and $\{4, 8\}$ provide poor classification results. This means that the raw time series of the inner and outer eccentricities are necessary to accurately predict the stability of the hierarchical triple-system evolution.

Interestingly, architecture B performs about as well as architecture A, even though it has far fewer trainable parameters (see Tables 2 and 3). This indicates that the coupling between input channels is important for predicting the fate of hierarchical triple systems. This also means that the Keplerian elements should not be treated as independent in the analysis of the system evolution.

We now focus on the best models, whose ROC curves are better shown in the right panel of Figure 5. The information provided by $T_{\text{mut}}$ (channel 4) seems to provide relevant information as models $A_{0148}$ and $B_{018}$ perform worse than models $A_{0147}$, $A_{0148}$, $B_{0147}$, and $B_{0148}$. The redundancy of channels 0, 1, and 8 does not seem to worsen the performances of the networks, but these channels are unnecessary.

Finally, the best two models both make use of all Keplerian elements, indicating that they are all meaningful to the CNN. It turns out that the best configurations are determined by the selected input channels, rather than by the CNN architecture type.

3.2. Best-model Diagnosis

Although configuration $A_{014567}$ has the highest AUC, we prefer to select the configuration $B_{014567}$ for parsimony reasons. Given these training input channels, architecture B has a slightly lower AUC, but far fewer trainable parameters than architecture A (59,585 versus 96,129). Also, architecture B allows us to take into account stronger dependences between the input channels from the beginning, which is not permitted by architecture A.

The left panel of Figure 6 shows the CNN prediction outputs of the test data set for the stable simulations (class 0). We see that the CNN predicts values close to 0 when it is provided stable simulations. Few stable simulations happen to be more confusing for the network, which outputs values greater than 0.2. However, studying the stability of hierarchical triple system is not a binary question (stable or unstable?), but rather a quantitative one: how long will the system be stable? We interpret the output values between 0 and 1 as the probability that the simulation will not disrupt before $10^8 P_1$. 

![Figure 4: Instance of all available input channels. The left panel corresponds to a stable simulation in our data set, and the right panel shows an unstable simulation that undergoes disruption after $5.1 \times 10^7 P_1$. Note the scale on the y-axis for channels 2 and 3 (semimajor axes): there, the magnitudes greatly vary in the data set as they are not normalized. Channels 2 and 3 are not presented to the network; we use channels 7 and 8 instead.](image-url)
The mutual eccentricity performances. The information. (MaxPooling (MaxPooling (MaxPooling Input Layer Output Shape Nb Parameters

| Layer         | Output Shape | Nb Parameters |
|---------------|--------------|---------------|
| Input         | (c, 31830, 1)| 0             |
| Conv2D        | (c, 31826, 16)| 96            |
| MaxPooling    | (c, 7956, 16)| 0             |
| Conv2D        | (c, 7952, 32)| 2592          |
| MaxPooling    | (c, 1988, 32)| 0             |
| Dropout       | (c, 1988, 32)| 0             |
| Conv2D        | (c, 1984, 32)| 5152          |
| MaxPooling    | (c, 496, 32)| 0             |
| Conv2D        | (c, 492, 64)| 10304         |
| MaxPooling    | (c, 123, 64)| 0             |
| Dropout       | (c, 123, 64)| 0             |
| Conv2D        | (c, 119, 64)| 20544         |
| MaxPooling    | (c, 29, 64)| 0             |
| Conv2D        | (c, 25, 64)| 20544         |
| MaxPooling    | (c, 6, 64)| 0             |
| Dropout       | (c, 6, 64)| 0             |
| Flatten       | 384          | 0             |
| FullyConnected| 16           | 6144c + 16    |
| FullyConnected| 1            | 17            |

Note. This architecture treats each time series separately until the last three layers, where the channels are flattened and the information is blended. The convolution masks are shared across all channels. The total number of parameters is $59,265 + 6,144c$, where $c$ is the number of channels in the input time series.

In the unstable case, the network gives more robust predictions. The right panel of Figure 6 shows a scatter plot of the selected model predictions as a function of the true breakup time for the $N=88,289$ unstable simulations of the test data set. The CNN mainly predicts values close to 1 for unstable simulations. The scatter plot can be misleading, as we can see many points close to the lower end of the $x$-axis, even in the bottom left corner where we would like to see none. This is because the unstable data set is biased toward short-lived simulations (see Figure 3), and so is the test subset.

To better understand the prediction performances of the network, we averaged the CNN predictions over 100 equally spaced bins, starting from $5 \times 10^3 P_1$ up to $10^6 P_1$. They correspond to the larger dark blue dots in the right panel of Figure 6. The average is performed over more observations on the left side of the plot than on the right side, due to the bias toward short-lived simulations in our unstable data set. This explains the higher scatter of the averaged predictions at longer disruption time. Nevertheless, we can see that the network predicts values very close to 1 for short-lived systems. The
averaged predictions become smaller as the system lives longer, however. In other words, the network gives very robust predictions for short-lived systems, and less robust predictions when the system lasts longer.

4. Discussion

4.1. Generalization to Arbitrary Hierarchical Triple Systems of Equal Masses

Our results have been derived from a data set comprised of hierarchical triple systems with fixed equal masses \( (m_1 = m_2 = m_3 = 10 M_\odot) \). Because of the scale-free nature of Newtonian gravity, any global rescaling of the masses (e.g., \( m_1 = m_2 = m_3 = 30 M_\odot \)) can be absorbed by rescaling the time unit. Numerical experiments also confirmed that the disruption time of hierarchical triples can be expressed in units of the inner orbital period, without loss of generality, even for systems with different mass or length scales (e.g., Hayashi et al. 2022). This consideration is no longer valid when one includes non-Newtonian forces, such as tides and general relativity corrections, which can affect the evolution of triple systems.

The code and the CNN-trained models presented in this work are provided as an open-source package. Currently, the code is optimized to work with TSUNAMI simulation outputs (Trani et al. 2019), but it can easily be extended to other codes. Because of the above considerations, the body masses have to be the same (but not necessarily \( 10 M_\odot \)) as long as the time series is generated in units of the inner orbital period \( P_1 \).

4.2. Limitations

The CNNs we have presented are set up to receive orbital inputs on a fixed time step in units of initial inner orbital periods. Using the initial inner orbital period as time unit may not necessarily be an optimal choice. Indeed, the long-term evolution of the system may lead to small changes in the inner orbital period. As long as the changes even out (see, e.g., channel 2 in Figure 4), the unit system remains valid. When these changes happen to become large and consistent, however, the time-unit system will no longer be meaningful as the mean of \( P_1 \) will be different. In principle, this could degrade the performances of the artificial neural network.

That said, large changes in the evolution of the inner orbital period also mean that the system becomes very unstable. Because we are interested in the long-term stability of the system, we would not need the help of a network in these cases, given that the system is obviously unstable. Our simulations have been selected to lie in the gray area between stability and instability. We therefore expect that the chosen time-unit system does not significantly deteriorate the CNN predictions.

During the review process of this manuscript, Vynatheya et al. (2022) presented a similar work on estimating the stability of triple systems using machine learning. The main difference to our work is that they estimate the stability of triples only using a limited set of initial conditions (masses, semimajor axes, eccentricities, and mutual inclinations), whereas we take into consideration the full set of initial parameters. By obtaining the early evolution of the triples, we can predict the stability for any individual realization of a hierarchical triple system. On the other hand, any stability criterion based only on partial initial conditions, such as the analytic Mardling & Aarseth (2001) criterion or the neural network by Vynatheya et al. (2022), neglect parameters such as true anomalies and the apsidal orientations. N-body simulations have shown that these parameters also determine the long-term stability of a triple and can turn stable triples into unstable ones and vice versa (e.g., Hayashi et al. 2022). This is because triple systems are inherently chaotic, and the boundary between stable and unstable orbits is fractal in the initial phase space. Consequently, any criterion based only on partial initial conditions is destined to be approximate at best.

The other differences with the work of Vynatheya et al. (2022) include that their criterion is based on the short-term evolution of the triples because they stop the simulations at \( t_f = 100 P_2 \), which is the minimum allowed duration of the simulations we consider. Additionally, we fully explore the parameter space in initial conditions, while they explore only slices of the initial parameter space, which could result in biased training data.
Hierarchical triple systems are of major importance in astrophysics from the study of planetary system evolution to the formation of gravitational waves. These systems are mainly studied through complete $N$-body integration, which can be computationally expensive. Instead, we propose to use a CNN to predict the stability of a hierarchical triple system using only 0.5% of the total integration time (therefore it is 200 times faster).

We have shown that the time-series evolution of the Keplerian elements allows us to accurately predict the long-term survival of hierarchical triple systems. In particular, the eccentricities provide relevant information for the stability prediction. Our selected CNN model has an AUC of 0.956, making it robust for the classification of new unseen data.

Our model is computational efficient and can be incorporated into more complex models of a triple-system evolution (e.g., Hamers et al. 2021; Toonen et al. 2022), or it can be used to place stability constraints on observed systems. We provide all trained models, their diagnosis, and the code in an open-source GitLab repository.7

Possible extensions of our work include the generation of a larger training data set involving very short-lived simulations and non-equal-mass systems, larger and deeper convolutional architectures, and an optimized training scheme with a validation test data splits. We leave these developments to future works.

A. A. T. acknowledges support from JSPS KAKENHI grant numbers 17H06360, 19K03907, and 21K13914. The simulations were run on the Deigo high-performance computing cluster at OIST. We are grateful for the help and support provided by the Scientific Computing section at OIST. F. L. wishes to thank Kenji Doya and Jun Makino for stimulating discussions. We finally thank the anonymous referees for their careful reading of our manuscript, and their constructive feedback, suggestions, and questions.

### Appendix A

#### Two CNN Architectures

We introduce two CNN architectures to account for different dependences between input channels and assess their significance regarding binary classification performances.

Architecture A exclusively uses convolution filters of shape (1, 5). These filters slide along the time axis of each input channel independently. They are shared by all input channels, which means that the CNN starts looking for patterns in each input channel independently. Only the last two layers are fully connected and allow for potential dependences between input channels. We alternate convolutional layers with max-pooling layers of size 4, and we repeat this pattern six times. The six convolutional layers have 16, 32, 32, 64, 64, and 64 filters, respectively, to allow for a reasonable number of parameters (see Table 2).

Architecture B, on the other hand, starts with a convolution filter of size (c, 5) across all input channels. This large filter transforms the multiple input time series into one single output time series (a sort of mixture of all input channels). This allows us to account for stronger dependences between input channels from the beginning. The large convolution has 64 output filters, which is flexible and allows considering several kinds of dependences from the beginning. The rest of architecture B is similar to architecture A. Due to its design, architecture B has far fewer trainable parameters than architecture A (see Table 3).

The input layer of both CNN architectures has a length of 31,830. This value corresponds to $t_f/\Delta t$ and is the same for all simulations. We also note that we employ Conv2D filters (and not Conv1D), even though we do not use the first dimension of the filter in practice. Indeed, most filters have the size $(1, 5)$, where the first dimension allows us to input different channels. The first layer of architecture B employs a convolutional filter of size $(c, 5)$, which uses the first dimension to blend all time series together.

### Appendix B

#### Activation Functions

For both network architectures, all activation functions are rectified linear units (ReLU), except for the activation function between the last two fully connected layers, which is a sigmoid function. The ReLU activation function, defined as

$$f(x) = \max(0, x),$$

is standard in artificial neural networks and is used to break the linearity between subsequent layers. The sigmoid activation function is defined as

$$\sigma(x) = \frac{e^x}{1 + e^x},$$

and outputs a value between 0 and 1, which can be interpreted as the probability for the input data of being from class 1 (unstable simulation) rather than class 0 (stable simulation).

### Appendix C

#### Training Strategy

Both CNN architectures are trained with the same strategy. As is standard for binary classification tasks, we use the binary cross-entropy loss function to assess the network training performances,

$$\mathcal{L} = \frac{1}{N_b} \sum_{i=1}^{N_b} y_i \log(p_i) + (1 - y_i) \log(1 - p_i),$$

where $N_b$ is the batch size, $y_i \in \{0, 1\}$ is the true label for observation $i$ in the data set (1 for unstable simulation, 0 for stable simulation), and $p_i \in [0, 1]$ is the output of the softmax activation at the last layer of the CNN.

The loss function is optimized with respect to the network parameters. The network parameters are the convolutional filter coefficients. The optimization is performed with the Adam optimizer with default values: Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07).

During training, we use a generator yielding batches of size $N_b = 64$ observations from the training data set. The training generator successively draws individual observation, with an equal probability of coming from the stable or the unstable training data set. One training step corresponds to one update of the network parameters using the Adam optimizer over the batch of size $N_b$. We perform 50 training steps per epoch, which amounts to 3,200 observations seen by the network at every epoch. We recall that the training data set has $2 \times 150,000 = 300,000$ observations. All CNN models have been trained for 250 epochs.
ORCID iDs
Florian Lalande @ https://orcid.org/0000-0001-6676-1484
Alessandro Alberto Trani @ https://orcid.org/0000-0001-5371-3432

References
Abadi, M., Barham, P., Chen, J., et al. 2016, arXiv:1605.08695
Breiter, S., & Ratajczak, R. 2005, MNRAS, 364, 1222
Chollet, F. 2015, Keras.io, https://github.com/fchollet/keras
Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P. A. 2019, Data Min. Knowl. Discov., 33, 917
Fernández, A., García, S., Galar, M., et al. 2018, Learning from Imbalanced Data Sets (Berlin: Springer), doi:10.1007/978-3-319-98074-4
Ford, E. B., Kozinsky, B., & Rasio, F. A. 2000, ApJ, 535, 385
Grishin, E., Perets, H. B., Zenati, Y., & Michaely, E. 2017, MNRAS, 466, 276
Hamers, A. S., Rantala, A., Neunteufel, P., Preece, H., & Vynatheya, P. 2021, MNRAS, 502, 4479
Hayashi, T., Trani, A. A., & Suto, Y. 2022, arXiv:2207.12672
Hoang, B.-M., Naoz, S., Kocsis, B., Rasio, F. A., & Dosopoulou, F. 2018, ApJ, 856, 140
Hut, P., & Bahcall, J. N. 1983, ApJ, 268, 319
Kingma, D. P., & Ba, J. L. 2015, 3rd Int. Conf. on Learning Representations, ICLR 205 (San Diego, CA: ICLR)
Kol, B. 2021, CeMDA, 133, 17
Kozai, Y. 1962, AJ, 67, 591
Krishchuk, A., Sutskever, I., & Hinton, G. E. 2017, Communications of the ACM (New-York: ACM), 60
Krymolowski, Y., & Mazeh, T. 1999, MNRAS, 304, 720
LeCun, Y., Boser, B., Denker, J. S., et al. 1989, Neural Comput., 1, 541
Lidov, M. L. 1962, P&SS, 9, 719
Luo, L., Katz, B., & Dong, S. 2016, MNRAS, 458, 3060
Manwadkar, V., Kol, B., Trani, A. A., & Leigh, N. W. C. 2021, MNRAS, 506, 692
Manwadkar, V., Trani, A. A., & Leigh, N. W. C. 2020, MNRAS, 497, 3694
Marchal, C. 1990, The Three-body Problem (USA: Elsevier Science)
Mardling, R. A., & Aarseth, S. J. 2001, MNRAS, 321, 398
Mikkola, S. 1994, MNRAS, 269, 127
Mikkola, S., & Aarseth, S. J. 1993, CeMDA, 57, 439
Mikkola, S., & Tanikawa, K. 1999, MNRAS, 310, 745
Monaghan, J. J. 1997a, MNRAS, 176, 63
Monaghan, J. J. 1976b, MNRAS, 177, 583
Muñoz, D. J., Lai, D., & Liu, B. 2016, MNRAS, 460, 1086
Mushkin, J., & Katz, B. 2020, MNRAS, 498, 665
Naoz, S., Farr, W. M., Lithwick, Y., Rasio, F. A., & Teyssandier, J. 2013, MNRAS, 431, 2155
Nash, P. E., & Monaghan, J. J. 1978, MNRAS, 184, 119
Portegies Zwart, S., & Boekholt, T. 2014, ApJL, 785, L3
Russakovsky, O., Deng, J., Su, H., et al. 2015, Int. J. Comp. Vision, 115, 211
Stoer, J., & Bulirsch, R. 2002, Texts in Applied Mathematics, Vol. 12, Introduction to Numerical Analysis, Vol. 12 (3rd edn.; New York: Springer)
Stone, N. C., & Leigh, N. W. C. 2019, Natur, 576, 406
Tamayo, D., Cranmer, M., Hadden, S., et al. 2020, AAS/Division of Dynamical Astronomy Meeting, 51, 300.01
Toonen, S., Boekholt, T. C. N., & Portegies Zwart, S. 2022, A&A, 661, A61
Trani, A. A., Fuji, M. S., & Spera, M. 2019, ApJ, 875, 42
Trani, A. A., Rastello, S., Di Carlo, U. N., et al. 2022, MNRAS, 511, 1362
Trani, A. A., Spera, M., Leigh, N. W. C., & Fuji, M. S. 2019, ApJ, 885, 135
Tremaine, S., Touma, J., & Namouni, F. 2009, AJ, 137, 3706
Valtonen, M., & Karttunen, H. 2006, The Three-Body Problem (Cambridge: Cambridge Univ. Press)
von Zeipel, H. 1910, AN, 183, 345
Vynatheya, P., Hamers, A. S., Mardling, R. A., & Bellinger, E. P. 2022, MNRAS, 516, 4146