Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **520,** 1224–1242 (2023) https://doi.org/10.1093/mnras/stac2933
Advance Access publication 2022 October 14

Residual neural networks for the prediction of planetary collision outcomes

Philip M. Winter,[1★] Christoph Burger,[1★†] Sebastian Lehner,[2†] Johannes Kofler,[2] Thomas I. Maindl[3,4] and Christoph M. Schäfer[1]

[1]*Institut für Astronomie und Astrophysik, Eberhard Karls Universität Tübingen, Auf der Morgenstelle 10, D-72076 Tübingen, Germany*
[2]*Johannes Kepler University Linz, Altenberger Straße 69, A-4040 Linz, Austria*
[3]*SDB Science-driven Business Ltd, 85 Faneromenis Avenue, Ria Court 46, Suite 301, 6025 Larnaca, Cyprus*
[4]*Department of Astrophysics, University of Vienna, Türkenschanzstraße 17, A-1180 Vienna, Austria*

Accepted 2022 September 29. Received 2022 September 23; in original form 2022 April 21

ABSTRACT

Fast and accurate treatment of collisions in the context of modern *N*-body planet formation simulations remains a challenging task due to inherently complex collision processes. We aim to tackle this problem with machine learning (ML), in particular via residual neural networks. Our model is motivated by the underlying physical processes of the data-generating process and allows for flexible prediction of post-collision states. We demonstrate that our model outperforms commonly used collision handling methods such as perfect inelastic merging and feed-forward neural networks in both prediction accuracy and out-of-distribution generalization. Our model outperforms the current state of the art in 20/24 experiments. We provide a data set that consists of 10164 Smooth Particle Hydrodynamics (SPH) simulations of pairwise planetary collisions. The data set is specifically suited for ML research to improve computational aspects for collision treatment and for studying planetary collisions in general. We formulate the ML task as a multi-task regression problem, allowing simple, yet efficient training of ML models for collision treatment in an end-to-end manner. Our models can be easily integrated into existing *N*-body frameworks and can be used within our chosen parameter space of initial conditions, i.e. where similar-sized collisions during late-stage terrestrial planet formation typically occur.

Key words: hydrodynamics – methods: numerical – astronomical data bases: miscellaneous – celestial mechanics – planets and satellites: composition – planets and satellites: formation.

1 INTRODUCTION

1.1 Planet formation background

Planet formation is inherently connected to collisions on all scales, from μm-sized dust grains up to planet-sized bodies. The precise mechanisms of early planetary growth generally depend on the current conditions in the protoplanetary disc and the amount and (dominant) size of available building blocks (e.g. Kokubo & Ida 2002; McNeil, Duncan & Levison 2005; Johansen & Lambrechts 2017). Particularly for terrestrial planets, our current understanding suggests that their final phase of accretion comprises growth via pairwise collisions of up to planet-sized bodies, lasting on the order of tens to hundreds of Myr (e.g. Chambers & Wetherill 1998; Kokubo & Ida 1998; Agnor, Canup & Levison 1999; Chambers 2001; Kokubo, Kominami & Ida 2006). This is supported by the long accretion times of terrestrial planets in the Solar System, as well as features like Mercury's high bulk density, Earth's large moon, or Mars' hemispheric dichotomy, all believed to be the consequences of large-scale collisions of roughly similar-sized bodies. Indirect evidence for such encounters has also been found in extrasolar systems (e.g. Wyatt & Jackson 2016) in the form of observed infrared excess caused by warm dust, interpreted as collision debris. These large collision events are of particular interest as they shape the final characteristics of terrestrial planets, and likely contribute to the broad compositional diversity of observed low-mass exoplanets (Marcus et al. 2009, 2010; Inamdar & Schlichting 2016; Bonomo et al. 2019). This phase of planet formation naturally also leads to radial mixing of material and allows for (dynamical and collisional) transport of volatiles, such as water to the inner parts of the system, and especially to potential planets forming in the habitable zone (Morbidelli et al. 2000; Izidoro et al. 2013; O'Brien et al. 2014, 2018; Haghighipour & Winter 2016; Burger, Bazsó & Schäfer 2020b).

Modelling of this final phase of planet formation is typically based on *N*-body simulations, where mainly the gravitational interaction of hundreds to thousands of bodies is followed for up to few hundred Myr (e.g. Chambers 2013; Fischer & Ciesla 2014; O'Brien et al. 2014; Quintana & Lissauer 2014; Quintana et al. 2016, as some of the more recent work). As planet formation models become more sophisticated and aim to study more than the most basic outcome quantities, collision modelling has to keep up in order to avoid systematic errors caused by too crude approximations of the underlying physics.

★ E-mail: winter@murena.io (PMW); christoph.burger@uni-tuebingen.de (CB)
† Equal contribution.

1.2 The collision treatment problem

Accurate modelling of major collisions among large, up to planet-sized bodies plays an important role in understanding the formation, evolution, and diversity of planetary systems. The prediction task for two-body collisions is well-defined: Given the initial conditions, such as collision geometry and object properties, we ask for the outcome state at a specific later point in time.

Up to relatively recently, collisions in planet formation scenarios were typically modeled by assuming complete accretion in all encounters (e.g. Raymond, Quinn & Lunine 2004, 2007; Haghighipour & Raymond 2007; Izidoro et al. 2013; Fischer & Ciesla 2014; O'Brien et al. 2014; Quintana & Lissauer 2014), often referred to as perfect inelastic merging (PIM). This approach is simple and fast, but gives reasonably accurate predictions only for the lower end of the spectrum of characteristic collision velocities, or for large target-to-impactor mass ratios. In general, collisions between large and roughly similar-sized bodies can result in a diverse range of outcomes (e.g. Leinhardt & Stewart 2012), and often include significant material losses (Haghighipour & Maindl 2022). This can affect bulk and chemical composition (e.g. Carter et al. 2015; Dwyer, Nimmo & Chambers 2015; Carter et al. 2018), and even more so for volatile constituents, especially at or close to the surface (Marcus et al. 2010; Maindl et al. 2014, 2017; Burger, Maindl & Schäfer 2018; Kegerreis et al. 2020; Burger et al. 2020b). In addition, collisions among similar-sized bodies frequently result in two large and gravitationally unbound survivors, instead of a single dominant one, as exemplified in Fig. 1. These so-called hit-and-run events constitute up to half of all collision outcomes (e.g. Chambers 2013; Clement et al. 2019; Burger et al. 2020b). This can prolong planetary accretion considerably, naturally leading to a higher overall number of collisions, and resulting in very different behaviour in terms of material loss and transfer between colliding objects (Burger et al. 2018; Burger, Maindl & Schäfer 2020a; Burger et al. 2020b).

Several approaches have been developed to account for this diverse range of possible collision outcomes. Genda et al. (2017) developed scaling laws for collisional erosion with a focus towards smaller projectile-to-target mass ratios down to 1:10 000, where outcomes are generally dominated by a single large survivor. Zhou, Dvorak & Zhou (2021) propose an approach that also exclusively assumes a single survivor, but includes randomly picked material losses, based on statistics of a large number of Smooth Particle Hydrodynamic (SPH) collision simulations. Crespi et al. (2021) suggest an approach based on a catalogue of SPH collision outcomes, focusing on the distribution of smaller-scale collision fragments. A recent framework based on semi-analytical scaling laws (Leinhardt & Stewart 2012; Stewart & Leinhardt 2012; Leinhardt et al. 2015) has been applied in various planet formation studies (e.g. Chambers 2013; Bonsor et al. 2015; Carter et al. 2015; Quintana et al. 2016; Carter et al. 2018; Clement et al. 2019). Albeit fast and relatively straightforward to implement, its prediction accuracy for more complex behaviour, like the fate of surface volatiles, or individual material losses and transfer in hit-and-run, is naturally limited (Burger et al. 2018). Genda, Kokubo & Ida (2011), Genda et al. (2017), and Burger et al. (2020b) resolve collisions in *N*-body planet formation simulations by running dedicated SPH simulations for each event on the fly, which is the most accurate approach, but computationally complex and expensive.

To summarize, depending on the problem at hand and the available computational resources, one has to make design choices which method to use. Both, simple problems and/or sufficient computational resources allow the use of sophisticated collision treatment methods, whereas complex problems and/or limited resources require certain trade-offs between prediction accuracy and computation time. For many applications, it would be desirable to choose and adjust this trade-off more flexibly. Although analytic and heuristic approaches are efficient, they are typically neither very accurate, nor allow adjusting the accuracy-speed trade-off. In contrast, full hydrodynamic simulations for individual collisions are much more costly, but yet very accurate. In this paper, we aim to combine all three properties, yielding an efficient, still accurate and flexible approach, where flexible means that it can be easily adapted to different accuracy-speed trade-offs.

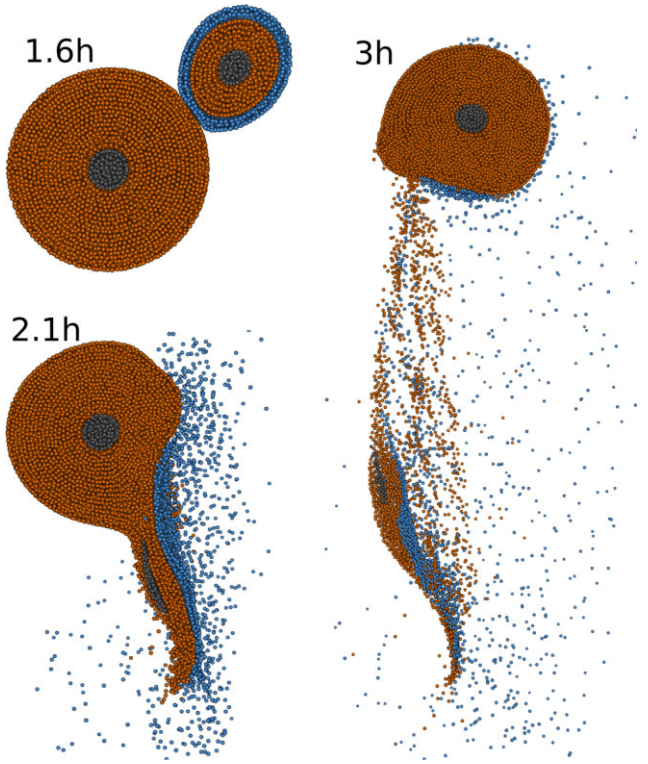

Figure 1. Exemplary snapshots at three different times of a SPH simulation of a planet–scale collision. The Mars-sized projectile hits the Earth-sized target at an impact angle of 43° and an impact velocity of 1.3 times the mutual escape velocity, resulting in a hit-and-run outcome. Colours indicate the different materials – an iron core, a silicate mantle, and a water/ice shell. Bodies are cut into halves for visualization. We perform 10 164 collision simulations, covering a large parameter space of initial conditions.

1.3 Machine learning for planetary collisions

The recent progress of cheap and efficient hardware caused a renaissance of machine learning (ML), enabling to solve complex tasks in different fields such as computer vision (Krizhevsky, Sutskever & Hinton 2017) and natural language processing (Brown et al. 2020) with unprecedented accuracy and speed. Recently, Tamayo et al. (2020) and Cranmer et al. (2021) applied ML for predicting long-term stability and dissolution of compact multiplanet systems, indicating that ML may serve as an efficient tool for fast and accurate approximation of astrodynamical processes.

Recurrent neural networks (RNNs; Jordan 1986; Pearlmutter 1989; Elman 1990) have been applied for approximating hydrodynamical simulations (Wiewel, Becher & Thürey 2019) and astrophysical simulations such as 2D mantle convection (Agarwal et al.

2021). Several works successfully demonstrated the applicability and usefulness of ML for planetary collision treatment, opening up a promising research direction for computational astrophysics: Valencia, Paracha & Jackson (2019) apply gradient boosting regression trees (Breiman et al. 1984; Friedman 2001), Gaussian processes (GPs; Rasmussen & Williams 2005), and a nested method for classifying collision scenarios and regressing the largest remnant mass. Cambioni et al. (2019) use a multiclass support-vector machine (Cortes & Vapnik 1995; Hearst et al. 1998) for the classification of different collision scenarios. They apply a small, three-layered feed-forward neural network (FFN; Rosenblatt 1961; Ivakhnenko & Lapa 1965) to regress accretion efficiencies, i.e. the mass of the largest remnant. Cambioni et al. (2021) extend this work and include surrogate models for predicting core mass-fractions of the largest and second-largest remnants. Emsenhuber et al. (2020) extend the work from Cambioni et al. (2019) to additionally predict orbital parameters of the two largest remnants with a separate regressor, resulting in a set of models that can be directly incorporated into *N*-body frameworks for collision treatment. However, this approach is limited to the main collision plane and does not allow prediction of orbital inclinations and longitudes of ascending nodes. The above-mentioned works use the SPH data from Reufer et al. (2012) that consists of collisions between non-rotating, differentiated iron-silicate bodies.

Timpe et al. (2020b) establish a high-quality data set that consists of 14 856 collisions between differentiated, rotating bodies (Timpe et al. 2020a). They apply a two-step classification-regression approach to predict post-collision properties. They study several different methods for collision treatment and find data-driven methods to outperform non-data driven methods. Gradient-boosted decision trees and FFNs are used for both classification and regression, whereas polynomial chaos expansion (Wiener 1938) and GPs are studied for regression only. They train different regressors for each individual post-impact property, and predict a variety of properties of the largest and second-largest remnant, and the remaining debris. FFNs and XGBoost (Chen & Guestrin 2016) perform best amongst data-driven methods. We regard that study as our closest related work.

The overall goal of our work is to improve the prediction of planetary collision outcomes via ML models. In particular, this includes minimizing systematic prediction errors as much as possible by outperforming the current state of the art. We improve upon the works above by providing a more general data set, reframing the ML task as a multi-task problem, and employing a simple, but problem-adapted ML model for the prediction of planetary collision outcomes. We train our model to predict masses, material fractions, positions, and velocities of the two largest post-collision remnants, and the remaining debris. Our contributions are summarized as follows:

(i) We perform extensive *N*-body simulations to determine realistic initial conditions for planetary collisions. We base the choice of the parameter space for our SPH data set on the outcome of the *N*-body simulations. To that end, we provide a comprehensive data set that consists of 10 164 SPH simulations of pairwise planetary collisions. We use between 20 and 50k SPH particles, which is relatively low resolution compared to state-of-the-art simulations in astrophysics with up to several million SPH particles. Our data set covers typical collision setups and is the first of its kind to combine all essential elements for a comprehensive treatment of collisions, including realistic object models (differentiated and rotating bodies), detailed pre- and post-collision geometries, and temporal information. The data set allows to study several generic topics, such as collision treatment in a broad range of scenarios, inverse problems (e.g. the Moon-forming impact), and collisional accretion during planet formation. While our data set is in general comparable to the one provided by Timpe et al. (2020b), it additionally includes volatile (water) layers, which opens up studies regarding collisional water/volatile transfer and loss, even though this is intended rather as a proof of concept in this work, mainly because of the difficulty to accurately resolve such surface layers.

(ii) In contrast to existing work, we follow a multi-task learning approach in the sense of multidimensional regression, in which a single ML model learns to predict the entire post-collision state rather than only specific, individual aspects of the state. Our ML task generalizes the collision treatment problem to 3D space, while at the same time avoiding the need for manual definition of class boundaries for different collision scenarios. Existing approaches often formulate the task as a classification problem, requiring somewhat arbitrary class definitions. We demonstrate that our multi-task learning approach leads to simple and computationally efficient models, while remaining relatively accurate compared to single-task learning.

(iii) We propose an ML model that helps modelling of temporal dynamics by evolving system states in an autoregressive manner. This closely resembles the data generation process, i.e. classical numerical solvers that iteratively solve the underlying hydrodynamic equations. This includes handling both, the properties of the colliding bodies and the spatio-temporal evolution of the system. Our model allows for flexible prediction of post-collision states at different times, and can be employed for collision treatment within existing *N*-body frameworks. We demonstrate superior prediction accuracy in comparison to commonly used baseline methods and the current state of the art. Moreover, our model requires little computational costs, reducing the prediction speed by approximately four orders of magnitude compared to the SPH simulations.

With our work, we aim to provide high-quality data and an ML model that is useful for various downstream applications. The paper is organized as follows. In Section 2, we describe our data generation pipeline, as well as the ML model used for collision treatment. In Section 3, we present our experiments and their results. Section 4 summarizes and concludes the paper.

2 METHODS

2.1 Data generation

2.1.1 N*-body simulations*

Burger et al. (2020b) developed a hybrid framework, based on extensive *N*-body simulations in combination with realistic collision treatment by direct SPH simulations. These results and collision statistics are also used to inform the choice of initial conditions for the SPH simulations performed in this study. In addition, we provide a cleaned and extended[1] version of their data set of approximately 10k collisions, which we refer to as '*N*-body data set'.[2]

[1]Based on new (yet unpublished) *N*-body + SPH simulations in a similar dynamical environment.

[2]The '*N*-body data set' based on the simulations by Burger et al. (2020b) provides data on collision parameters before contact, and basic data on the final state after the collision, like masses and composition of the two largest remnants, but no dynamical information (positions and velocities) and no data on intermediate states. Along with our other data and tools, it is available at https://github.com/littleblacksheep/csv/tree/main/misc.

The scenarios in Burger et al. (2020b) are based on an evolving disc of (∼Mars-mass) planetary embryos + smaller bodies (planetesimals). Their dynamical and collisional evolution is followed over several hundred Myr of terrestrial planet formation in an environment akin to the early Solar System. The embryos and planetesimals are modelled as differentiated, three-layered, self-gravitating bodies, similar to the SPH simulations in this work. The rotation state is not tracked across multiple collisions. The approach of on-the-fly SPH simulations allows not only for accurate treatment of each individual collision, but also a relatively straightforward re-integration of collision outcomes into the overall *N*-body dynamics (for our ML approaches this is discussed in Section B3). It also includes individual tracking of both large survivors in hit-and-run collisions, which comprise up to 50 per cent of outcomes between similar-sized bodies. Therefore, this data set also provides reliable collision (input parameter) statistics for the scenarios in this work.

2.1.2 SPH simulations

SPH is a numerical method for modelling visco-elastic fluid flows. The method was first proposed by Gingold & Monaghan (1977) and Lucy (1977) and has since been applied extensively to model various aspects of astrophysical collision processes, including planetary collisions. In this work, we use the SPH code `miluphcuda`[3] (Schäfer et al. 2016, 2020) to generate a planetary collision data set. An example is illustrated in Fig. 1. The SPH code solves the continuum mechanics equations for hydrodynamic flow, can handle 3D, multimaterial problems, and includes self-gravity. It also includes modules for the simulation of elasto-plastic solid-body physics based on several available material models and equations of state.

In this work, we perform pure hydro simulations, i.e. only solving the Euler equation with scalar pressure, instead of full tensorial treatment of material strength. Since we perform a large number of simulations, we trade some physical accuracy for numerical stability and more data (due to faster computation). However, this design choice is still a reasonably good proxy within the scope of our scenarios (Burger & Schäfer 2017; Burger et al. 2018). For actual collisions in an active planet formation environment, it can be assumed that the physical state of the colliding bodies – and hence their material (strength) response – varies over a broad range, even for otherwise identical scenarios in terms of masses, compositions, and collision parameters. This may be a function of their collision history, thermal state, and possibly various other factors. Considering those ambiguities, our rather simple material model allows the data set to remain as general as possible and at the same time consistent over our whole parameter space. We use the Tillotson equation of state (Tillotson 1962; Melosh 1989) for all simulations. Technical details are given in Section A1.

The SPH simulation pipeline is fully automated and includes all steps to initialize, run, and post-process individual simulations (see Fig. 2). For each run, a specific parameter set is sampled from the parameter space (Table 1). The chosen parameters cover a broad range of possible collision scenarios during terrestrial planet formation. The particular choices of parameter ranges are additionally informed by the robust statistics of our *N*-body data set (see Section 2.1.1). Note that we use the *N*-body data set exclusively for choosing meaningful parameter intervals representative of late-stage terrestrial planet formation. For creating the SPH data set, our parameter space of initial conditions is sampled randomly within the chosen intervals.

[3]The SPH code `miluphcuda` is in active development and publicly available at https://github.com/christophmschaefer/miluphcuda

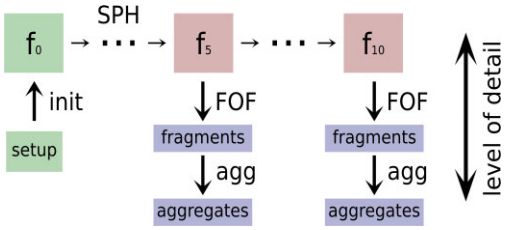

Figure 2. Data generation pipeline. For each individual simulation, a setup is sampled from the parameter space. The SPH particle distribution and its properties are set up in 'init', resulting in the input frame f_0. The SPH code then evolves the system, leading to a number of output frames. All output frames are post-processed with a friends-of-friends algorithm to compute all spatially connected material 'fragments'. Finally, 'aggregates' are identified, defined as gravitationally bound collections of fragments.

Table 1. Parameter space of initial conditions for our SPH simulations, covering a wide range of typical scenarios for rocky planet formation. See the text for detailed definitions. All parameters are randomly sampled.

Parameter	Min	Max	Description
$M_{\rm tot}(kg)$	$2 \times M_{\rm Ceres}$	$2 \times M_{\rm Earth}$	Total mass
$\gamma(1)$	0.05	1	Mass ratio m_p/m_t
$\zeta_{\rm iron}(1)$	0.01	0.25	Iron (core) fraction
$\zeta_{\rm water}(1)$	0; 0.1	0.25	Water (shell) fraction
$v_{\rm imp}(v_{\rm esc})$	1	8	Impact velocity
α(deg)	0	90	Impact angle
$P_{\rm rot}(P_{\rm rot,crit})$	0	0.2	Rotation period
$\theta_{\rm rot}$(deg)	0	180	Rotation axis polar
$\phi_{\rm rot}$(deg)	0	360	Rotation axis azimuthal
$f_i(1)$	3	7	Initial distance factor
$f_t(1)$	40	60	Simulation time factor
$N_{\rm tot}(1)$	20k	50k	Number of SPH particles

For initializing self-gravitating bodies in hydrostatic equilibrium, we adopt the approaches and tools from Burger et al. (2018), who calculate realistic density and pressure profiles for multilayered bodies. The colliding objects are referred to as *projectile* and *target*, the latter being the more massive body. They are initialized at a certain distance, on the order of several times the sum of their radii, to allow for pre-collision tidal deformation, relaxation of rotating configurations, and settling of residual numerical artefacts (e.g. at material boundaries). Based on the desired impact velocity and impact angle at 'touching-ball' distance (cf. Fig. 3), initial positions are calculated via backtracking the analytical two-body trajectories up to a distance of $d_{\rm initial} = f_i \times (R_{\rm t} + R_{\rm p})$. $R_{\rm t}$ and $R_{\rm p}$ are the target and projectile radii, and the initial distance factor f_i is a parameter. The total simulation time is calculated via $T_{\rm sim} = \tau_{\rm col} \times (f_i + f_t)$ and rounded up to the next full hour. $\tau_{\rm col}$ is the collision time-scale $\tau_{\rm col} = (R_{\rm t} + R_{\rm p})/v_{\rm imp}$. The impact velocity $v_{\rm imp}$ and the impact angle α are specified at touching-ball distance $R_{\rm t} + R_{\rm p}$, where $\alpha = 0^\circ$ corresponds to head-on collisions and $v_{\rm imp}$ is the absolute value of the relative velocity vector v at touching-ball distance (cf. Fig. 3). The minimum number of SPH particles is set such that the resulting water shell has a thickness of at least 2 SPH particles at $\zeta_{\rm water} = 0.1$ (Burger 2019). Note that this resolution may be too low to accurately simulate the water layers' response for a range of scenarios

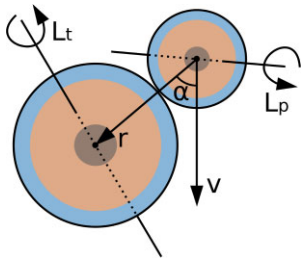

Figure 3. Collision geometry for planetary collisions. The impact angle α is measured between the relative position $\mathbf{r}$ and the relative velocity $\mathbf{v}$ between target and projectile at 'touching-ball' distance. Both objects comprise a core-mantle-shell structure and have random rotation axes $\mathbf{L}_t$ and $\mathbf{L}_p$, which can lie outside the plane spanned by $\mathbf{r}$ and $\mathbf{v}$.

(parameter combinations). This can be particularly problematic for the second-largest post-collision remnant, while reliable predictions are possible for the largest remnant already at resolutions similar to ours, as demonstrated by Burger et al. (2018). Nevertheless, results for water mass fractions on post-collision remnants should be taken with a grain of salt, and consequently we consider our ML model predictions for this aspect rather a proof of concept and not generally accurate at this point. For other basic outcome properties, on the other hand, like masses and kinematics of the two largest remnants, Burger et al. (2018) found resolution convergence for similar collision scenarios within 10 per cent for their 100k particles simulations. Our simulations contain either 2 or 3 materials, depending on ζ_{water}, where we remove the water shell if $\zeta_{\text{water}} < 0.1$ was sampled. The total colliding mass covers a range from $2 \times M_{\text{Ceres}}$ up to $2 \times M_{\text{Earth}}$. The mantle (basalt) mass-fraction is defined by $\zeta_{\text{basalt}} = 1 - \zeta_{\text{iron}} - \zeta_{\text{water}}$. Since the hydrostatic initialization routine is based on non-rotating objects, we set our maximum rotation period $P_{\text{rot,max}} = 0.2 \times P_{\text{rot,crit}}$ for both target and projectile in order to avoid excessive initial oscillations and instabilities, which typically occur once P_{rot} approaches $P_{\text{rot,crit}}$. The critical rotation period $P_{\text{rot,crit}}$ is defined such that material at the surface of the (idealized spherical) body is weightless according to Kepler's third law. Rotation axes are chosen randomly for both target and projectile. We refer to Section A2 for more details.

During simulation, the SPH code periodically produces output frames, which contain the state of all SPH particles at the respective time. We keep the first, the last, and intermediate frames for post-processing, where intermediate frames are saved at 5-h intervals (simulated time). All frames undergo the same post-processing procedure:

(i) Spatially connected collision fragments are calculated by the friends-of-friends algorithm (Geller & Huchra 1983).

(ii) Barycentres, orbital angular momentum, and spin angular momentum are calculated for each fragment, as well as for the entire system.

(iii) The two[4] largest aggregates of fragments are calculated. An aggregate is defined as a collection of gravitationally bound fragments, determined by an iterative procedure, which starts from the most-massive fragment as seed (see Burger et al. 2020b, for details). In the remainder of the paper, these aggregates are referred to as 'remnants' for clarity.

(iv) Basic visualization is done for the large fragments. A fragment is considered significant if it consists of at least 5 SPH particles.

(v) In this work, we focus on the prediction of macroscopic system states, requiring information on the level of remnants only. Moreover, we aim to keep memory requirements of the final data set low. Therefore, SPH output frames are subsampled, keeping 1 out of 10 SPH particles.

Keeping intermediate frames enables in-depth studies of temporal properties of the collision process. Moreover, they allow for the development of sophisticated ML models, i.e. models that not only predict the final state of the system, but the entire temporal evolution in detail. Note that since we sample our parameter space randomly, inputs to ML models do not require initial conditions that are similar to those in Burger et al. (2020b).

2.2 Machine Learning for collision treatment

From an ML perspective, the collision treatment task requires learning physical laws (e.g. conservation laws, material deformations, gravitational interactions, etc.) and handling the temporal evolution of the system (e.g. via time-series modelling). Various ML approaches can be applied in different contexts, mostly depending on which level of detail one is interested in. Therefore, we design our SPH data set such that it can be used at different levels of detail. For example, one can use remnant or fragment information ('macro states') rather than SPH particle representations ('micro states') for learning certain aspects (e.g. predicting certain quantities such as the mass of the largest remnant or the thermal energy of the system). Depending on which level of detail ML is applied to, different aspects may be able to be learned more or less efficiently. In this work, we focus on macro states because this setup is the most relevant one in order to incorporate ML models into *N*-body simulations for planet formation and evolution (see Table 2. and Section B3 for more details). In contrast, ML models operating on micro states may be a better choice if one is interested in studying details of the hydrodynamic flow and physical interactions in simulations such as SPH.

2.2.1 Collision treatment as a multi-task regression problem

Supervised learning is the task of selecting (learning) a specific model from a certain model class by using example input-target pairs. The difference between model outputs and desired target outputs results in an error, which is used to improve a model. We train our ML models in a supervised manner to predict several different quantities (mass, material fractions, position, and velocities) at once, which turns the problem into a multi-task problem. Our multi-task problem can be interpreted as a multidimensional regression problem of different physical quantities, since we use shared representations to predict different modalities. We motivate formulating and solving the problem as a multi-task problem due to inherent dependencies and correlations between the individual subtasks (e.g. trajectories of individual fragments highly depend on the overall mass distribution). Since all of our sub-tasks are highly correlated with each other, we hope that the multi-task setting supports generalization due to shared representations within ML models, acting as a form of regularization. Shared representations naturally allow for exploiting dependencies and correlations between different tasks, potentially improving the ML model's predictive performance. Note that in

[4]Gravity-dominated collisions of roughly similar-sized bodies generally result in either none (if highly destructive), one, or two (in hit-and-run scenarios) large surviving bodies, along with orders-of-magnitude smaller debris.

Table 2. Non-redundant ML features and normalization hyperparameters for feature normalization. Units indicate different physical quantities. All data x_{phys} is normalized during pre-processing. Note that since material fractions sum up to 1, only core (iron) and shell (water) fractions are required. Initial rotation speeds of the colliding bodies are encoded via the norms of their respective rotation axes.

State	Feature	Dim	Description	σ (ours)	σ (Timpe)
Initial	$N_{\text{tot}}(1)$	1	Number of SPH particles	5e + 4	2.3e + 5
	$M_{\text{tot}}(kg)$	1	Total mass	1e + 25	100
	$\gamma(1.)$	1	Mass ratio m_p/m_t	1	1
	$\zeta_p(1.)$	2	Material fractions projectile	1	1
	$\zeta_t(1.)$	2	Material fractions target	1	1
	$rot_p(\text{rad s}^{-1})$	3	Rotation axis projectile	6.5e-05	100
	$rot_t(\text{rad s}^{-1})$	3	Rotation axis target	6.5e-05	100
	$x_p(\text{m})$	3	Barycentre position projectile	[5e+07, 2e+08, 2e + 07]	[6, 41, 3]
	$v_p(\text{m s}^{-1})$	3	Barycentre velocity projectile	[2e+03, 1e+04, 6e + 02]	[2, 16, 0.5]
	$x_t(\text{m})$	3	Barycentre position target	[5e+07, 2e+08, 2e + 07]	[6, 41, 3]
	$v_t(\text{m s}^{-1})$	3	Barycentre velocity target	[2e+03, 1e+04, 6e + 02]	[2, 16, 0.5]
final	$m_1(\text{kg})$	1	Mass largest remnant	1e + 25	100
	$m_2(\text{kg})$	1	Mass 2nd-largest remnant	1e + 25	100
	$m_r(\text{kg})$	1	Mass rest	1e + 25	100
	$\zeta_1(1.)$	2	Material fractions largest remnant	1	1
	$\zeta_2(1.)$	2	Material fractions 2nd-largest remnant	1	1
	$\zeta_r(1.)$	2	Material fractions rest	1	1
	$x_1(\text{m})$	3	Barycentre position largest remnant	[5e+07, 2e+08, 2e + 07]	[6, 41, 3]
	$v_1(\text{m s}^{-1})$	3	Barycentre velocity largest remnant	[2e+03, 1e+04, 6e + 02]	[2, 16, 0.5]
	$x_2(\text{m})$	3	Barycentre position 2nd-largest remnant	[5e+07, 2e+08, 2e + 07]	[6, 41, 3]
	$v_2(\text{m s}^{-1})$	3	Barycentre velocity 2nd-largest remnant	[2e+03, 1e+04, 6e + 02]	[2, 16, 0.5]
	$x_r(\text{m})$	3	Barycentre position rest	[5e+07, 2e+08, 2e + 07]	[6, 41, 3]
	$v_r(\text{m s}^{-1})$	3	Barycentre velocity rest	[2e+03, 1e+04, 6e + 02]	[2, 16, 0.5]

contrast to our multi-task setting, Cambioni et al. (2019, 2021), Emsenhuber et al. (2020), and Timpe et al. (2020b) use individual ML models for predicting either single or subsets of collision outcome quantities, i.e. following single-task approaches. We believe that single-task approaches introduce unnecessary restrictions to the generalization capabilities of ML models, because individual models are exclusively able to specialize in their respective regime. Also, using individual regression-models for individual outcome scenarios (i.e. erosion, accretion, and hit-and-run) may lead to various issues caused by data-scarcity due to class-imbalances, which are common in planetary collision data sets. Moreover, using a single ML model for solving several subtasks at once may allow much better accuracy-speed trade-offs, especially in the presence of many subtasks. In our work, the importance of the individual subtasks are implicitly given via data pre-processing, effectively weighting loss terms of the respective subtasks. However, the importance can be explicitly adjusted depending on specific use cases.

To our knowledge, this is the first work to fully formulate the ML task as a regression problem. Our formulation allows simple, yet efficient training in an end-to-end manner and facilitates easy integration of ML models into existing *N*-body frameworks. Our regression objective does not explicitly optimize for classification performance, but rather for regression of macroscopic properties of the system. At the same time, our objective avoids the need for complicated approaches, which require two separate ML models for regression and classification, respectively.

We believe that it is not beneficial to train classifiers that explicitly discriminate between different outcome scenarios such as accretion or hit-and-run, because such classification can be easily performed as a post-processing step on top of regression-model predictions. We favour performing classification via a post-processing step rather than training dedicated classifiers because the former can be used in combination with variable class definitions, whereas the latter is bound to fixed class definitions. When training dedicated classifiers, changing class definitions would require re-training the classifiers, which can be quite cumbersome in practice. Moreover, pure classifiers cannot be used as a full replacement for collision treatment in *N*-body simulations.

In general, we believe that defining and learning fixed classification schemes is not optimal due to continuous transitions between classes and the associated arbitrariness of class definitions. We believe that training dedicated classifiers is only reasonable if one is explicitly interested in accurate classification under the restriction of fixed class definitions.

The integration of ML models for collision treatment into *N*-body simulations might require additional post-processing steps on top of ML predictions. This includes restricting predictions to conserve certain quantities such as the total mass, e.g. by re-scaling predicted masses and/or distributing debris material across the two largest remnants. For the actual application in *N*-body simulations, it is especially important how (if at all) the remaining collision debris is treated, which typically consists mostly of physically non-connected and gravitationally unbound fragments. This naturally opens up many possibilities depending on the respective use case (i.e. the precise physical and numerical model). In our experiments, we do not apply any additional post-processing steps in order to remain as general as possible and to obtain conservative performance estimates.

Fragments that are formed from collisions between non-rotating objects mostly remain in the collision's main symmetry plane (the x–y plane in our case) with only marginal z-components. However, collisions between rotating objects generally break this symmetry, and may produce large fragments with significant z-components. This symmetry breaking is also confirmed by the data from Timpe et al. (2020b). We thus generalize the prediction task from Emsenhuber et al. (2020) to 3D space, treating all dimensions equally to consistently handle deviations from the main collision plane.

2.2.2 *Autoregressive ML models for temporal evolution*

The use of autoregressive ML models for predicting collision outcomes can be motivated by studying the data generation process, i.e. the SPH simulations. We know that the data generation process has the Markov property, i.e. states s_{t+h} at a time $t + h$ depend entirely on their previous states s_t at time t. We assume continuous transitions between states for infinitesimal stepsizes h. The transition from s_t to s_{t+h} is described by a transition function g.

$$s_{t+h} = g(s_t, h) \tag{1}$$

Historically, g refers to a set of hand-crafted equations that incorporate certain physical laws (e.g. gravity, friction, etc.), as well as a procedure to evolve the system in time (e.g. numerical integration) by means of differential equations. In practice, these approaches often suffer from limitations, such as the requirement to use small stepsizes when using classical solvers. Too large stepsizes typically introduce large systematic errors, often leading to diverging or unstable solutions.

In this work, we aim to approximate solutions obtained using hand-crafted transition functions via an ML model that is learned from data. We believe that ML models are – once trained – efficient, powerful, and flexible transition functions for modelling the underlying physical processes in planetary collisions over time. In contrast to FFNs, our proposed model class exploits the Markov property of the data generation process, i.e. taking multiple, autoregressive steps to predict system states at a desired time T. There are several arguments that support the use of autoregressive ML models:

(i) Neural networks are universal function approximators (Hornik, Stinchcombe & White 1989) that allow learning highly complex functions. This property allows direct prediction of system states at various times T, entirely circumventing the need for time-series modelling. ML models should thus be much more computationally efficient compared to numerical simulations. The most extreme case would be to predict the final state directly, as typically achieved in the literature. Depending on the choice for the stepsize, our model allows a flexible accuracy-speed trade-off. Small stepsizes can be expected to better model the physical processes and lead to more accurate predictions at the cost of computational resources, whereas large stepsizes lead to less accurate, but faster predictions.

(ii) Autoregressive models subdivide the prediction of system states by taking multiple iterative steps. Since the universal function approximation theorem also applies to autoregressive ML models, they can use magnitudes larger, more complex time-steps compared to classical transition functions (numerical solvers) before getting unstable. This property typically makes these ML models much more efficient in terms of computational costs compared to hand-crafted transition functions.

(iii) Learned transition functions allow context-dependent time-steps, i.e. adjusting the transition function automatically, based on data-specific information. This property avoids algorithmic design decisions, making ML-based transition functions more general and flexible compared to hand-crafted transition functions.

(iv) Using autoregressive ML models allows for improved interpretability by enabling analysis of intermediate states. Such an analysis is not possible for ML models like FFNs or regression trees, which typically try to predict final post-collision states directly.

(v) Due to their design, we believe that autoregressive ML models can achieve better generalization compared to methods that try to predict final states directly, allowing more accurate predictions and improved o.o.d. generalization capabilities. Autoregressive ML models that learn fixed time intervals (i.e. taking multiple steps with the same stepsize) are effectively time-invariant per design, in the sense that they have to learn physical processes at only a single timescale. Thereby, the models are not forced to spend their parameters[5] for learning to become time-invariant. This property can potentially also lead to improved parameter efficiency compared to non-time-invariant ML approaches.

(vi) Longer physical interactions typically lead to the emergence of more complex dynamical processes during planetary collision events. Using autoregressive models naturally accounts for these effects by allocating computational resources that linearly scale with time, which is consistent with fluid-flow approaches.

Gated architectures (Hochreiter & Schmidhuber 1997; Cho et al. 2014) and regularized RNNs (Schmidt et al. 2021) are able to produce chaotic dynamics, but often suffer from the exploding gradient problem (Metz et al. 2021), typically leading to diverging sequences (Monfared, Mikhaeil & Durstewitz 2021). On the other hand, non-chaotic sequences have bounded loss gradients and converge to fixed points. Thus, training autoregressive ML models is typically non-trivial and often very sensitive to hyperparameters, especially when the data-generating process is itself chaotic. Exploding gradients and diverging sequences in LSTMs[6] can be mitigated via the forget gate (Gers, Schmidhuber & Cummins 1999), thereby reintroducing the vanishing gradient problem (Hochreiter 1998). The vanishing gradient problem can prohibit efficient training of deep neural networks. For our autoregressive ML model, we find regularization of hidden states to be a robust strategy against diverging sequences. Moreover, we find that gradient descend with backpropagation through time (Robinson & Fallside 1987; Werbos 1988; Mozer 1995) works fine for training.

2.2.3 *Residual neural network for planetary collision handling*

Our proposed model for prediction of collision outcomes can be interpreted as a residual neural network (ResNet; He et al. 2016). ResNets were originally introduced to ease the training of deep neural networks. The 'residual' aspect refers to reformulating neural network layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. Comparable to LSTMs without forget gate, ResNets efficiently mitigate the vanishing gradient problem.

We refer to our architecture as RES herein. In contrast to a classical ResNets, learned parameters in our model are shared across different steps (see Fig. 4). Our architecture treats temporal dynamics consistently by evolving system states in an autoregressive manner. Individual steps of a trained model can be interpreted as evolving the system for a fixed, but learned time interval. This approach is comparable to explicit iterative methods such as the Euler method or the Runge–Kutta method (Runge 1895; Kutta 1901). Ideally, smaller stepsizes should allow for better modelling of physical processes and should thus lead to better performance at the cost of computational resources. Our architecture allows for flexible prediction of system states at different times T by taking the respective number of update steps. Our architecture is autoregressive, i.e. only requiring the initial state y^0 and the number of steps $n_{\text{steps}} = s_h \times T$ as input. The

[5]Note that herein term 'parameters' can either refer to individual parameters of initial conditions for our SPH simulations, or to learnable parameters of an ML model. Both cases should be apparent from the respective text passages.
[6]Long Short-Term Memory, a special kind of RNN that is widely used for processing sequential data such as written texts, time series, or DNA sequences.

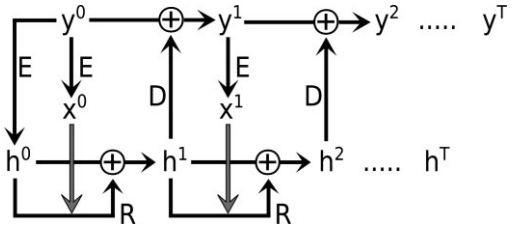

Figure 4. Operations of our weight-tied residual neural network architecture. The neural network modules E, R, and D are learned from data and shared across individual steps. The initial state y^0 is encoded into the initial hidden state h^0 and the input x^0 of R at time $t = 0$. R then predicts additive updates to h^t by using h^t and x^t. At each step, D outputs relative updates, which are used to evolve system states y via Euler integration. Sequences y^t and h^t are calculated iteratively, where the number of steps T correlate linearly to simulation time.

hyperparameter s_h allows to take different accuracy-speed trade-offs by adjusting the temporal resolution (i.e. the stepsize).

We want to stress that predicted sequences y^t and h^t [with $t \in (0, n_{\text{steps}})$] should not be considered as time sequences per se, but may nevertheless be closely related/correlated to time sequences, especially when considering our task of predicting system states at different points in time. We incorporate this close temporal correlation by choosing the number of steps to correlate linearly to the simulated time of the respective SPH simulations. This strong assumption may require additional and more detailed consideration in future work.

Our entire model architecture can be formalized as follows:

$$(h^t, x^t) = E(y^t, \phi_E) \tag{2}$$

$$h^t = h^{t-1} + R(h^{t-1}, x^{t-1}, \phi_R) \tag{3}$$

$$y^t = y^{t-1} + D(h^t, \phi_D) \tag{4}$$

E, R, and D correspond to the encoder module, residual module, and decoder module, respectively. Hidden states that are predicted by E are only used for the initial hidden state h^0. Our architecture avoids the vanishing gradient problem via additive updates to hidden states h and physical states y. Additive updates to the physical states y can be interpreted as Euler discretization of a continuous transformation and is closely related to the works of Chen et al. (2018), He et al. (2016), and Srivastava, Greff & Schmidhuber (2015). E, R, and D are FFN networks[7] (Rosenblatt 1961; Ivakhnenko & Lapa 1965) with learnable parameters ϕ_E, ϕ_R, and ϕ_D.

2.2.4 *Baseline models*

We choose three baseline methods for comparison with our newly proposed RES model class. Existing work (Cambioni et al. 2019, 2021; Emsenhuber et al. 2020; Timpe et al. 2020b) use FFNs as regressors for collision outcomes. We thus choose an FFN as our first baseline. We choose a linear regression model (LIN) as our second baseline to study the benefit of deep learning models compared to a simple, data-driven model. The third baseline is PIM, which is still widely used in astrophysical problems involving collisions because the method is purely analytic and fast. PIM assumes a perfect inelastic collision of target and projectile, always leading to a single surviving body, and conserving mass and momentum of the system by design.

[7]Historically also referred to as multilayer perceptrons

To enable learning non-linear mappings, artificial neural networks require so-called activation functions, which are applied element-wise to individual neurons usually after calculating the matrix-vector product for the respective layers. Due to its sound theoretical advantage compared to other activation functions, we use the scaled exponential linear unit (SELU) activation function (Klambauer et al. 2017) for hidden layers and linear activation functions for output layers of our deep learning models. SELU activations have self-normalizing properties, where neuron activations automatically converge towards zero mean and unit variance in the case of many hidden layers, leading to substantial advantages for training, regularization, and robustness when compared to other approaches. An optional rotation module can be incorporated into the ML models as additional pre- and post-processing steps, rendering the models rotation-equivariant (see Section B1 for details).

Although our SPH results naturally contain approximations and assumptions about the simulated physical processes, and are also subject to typical numerical inaccuracies, we define the SPH data as our ground truth. This definition is generally motivated by the fact that hydrodynamical simulations are currently considered the most accurate method for planetary collision treatment.

2.2.5 *ML experiment setup*

We split our data into a development set and a test set. The development set includes approximately 88 per cent of the data (8927 data points) and consists of training and validation splits, whereas the test set covers the remaining 12 per cent (1237 data points). The entire data set contains 10 164 data points. Using the development set, we perform fivefold crossvalidation[8] (Hastie, Tibshirani & Friedman 2017) for all experiments, allowing to calculate confidence intervals for our results. All training and validation splits share the same data distribution. Note that validation data are inappropriate for estimating performance on future data because validation data are used for hyperparameter optimization, which can be a source of information leakage. A holdout test set is required to estimate performance on completely new, unseen data.

Let us recap that our data set covers the parameter space as defined in Table 1. Although the parameter space is carefully chosen, parameters of real collisions are naturally not strictly limited to our defined parameter ranges (Quionero-Candela et al. 2009), i.e. so-called o.o.d. data points. In practice, ML models often fail to generalize to such o.o.d. data points. In order to study o.o.d. generalization of our ML models, we establish an o.o.d. test set. We expect that problem-specific models have better o.o.d. generalization capabilities compared to general-purpose models (Mitchell 1980). It is widely known that the impact velocity and the impact angle are two of the most important parameters in the context of planetary collisions. Thus, we manually select four regions in the impact angle – impact velocity space that compose our o.o.d. test set (see Table 3). We use this o.o.d. test set as our default test set in experiments unless stated otherwise.

Our data set D consists of $N = 10\,164$ tuples (y_i^0, z_i^T), $i \in [1, N]$, representing initial system states (at $t = 0$) and final system states (at $t = T$). For our supervised learning task, y_i^0 and T are used as model inputs, whereas z_i^T are used as ground truth labels. For intermediate

[8]Folds are non-intersecting, same-sized subsets of a data set. For five-fold crossvalidation, a total of five models are trained independently. Each training session consists of training on four-folds, whereas the remaining fifth fold is used for validation.

Table 3. Selected regions for the out-of-distribution (o.o.d.) test set. We select regions that may result in qualitatively different outcomes compared to the development set. The test set contains about 12 per cent of all data points.

Parameter	Region 1	Region 2	Region 3	Region 4
α_{min}(deg)	10	65	80	0
α_{max}(deg)	30	75	90	20
$v_{\text{imp,min}}(v_{\text{esc}})$	1.5	2	1	6
$v_{\text{imp,max}}(v_{\text{esc}})$	2.5	4	2	8

states $0 < t < T$ holds. D is split into a development set D_{dev} and a test set D_{test}. Our training and validation splits are derived from D_{dev} depending on the respective crossvalidation fold.

$$D = \left\{ \left(y_1^0, z_1^T\right), \left(y_2^0, z_2^T\right), ..., \left(y_N^0, z_N^T\right) \right\} \tag{5}$$

$$D_{\text{dev}} \subset D, \quad D_{\text{test}} \subset D, \quad D_{\text{dev}} \cap D_{\text{test}} = \{\} \tag{6}$$

We perform data pre-processing to transform features into appropriate value ranges for ML. We apply feature-wise normalization $x_{\text{ML}} = \frac{(x_{\text{phys}} - \mu)}{\sigma}$ to transform data x_{phys} given in SI units into data x_{ML}, whose value ranges are better suited for ML. We set $\mu = 0$ for all features. Note that the barycentre of the system still remains at the origin of the coordinate system for all data points even after normalization. Table 2. summarizes our ML features, along with normalization hyperparameters σ. Note that μ and σ implicitly define the importance between different subtasks during ML model training. Importance of subtasks can be further adjusted via introducing dedicated weights for the corresponding loss terms. The detailed pre-processing pipeline can be found in the provided source code (see Section 1.1).

Note that although accurate tracking of the rotation state is important for many aspects of planet formation and evolution modelling, we do not include rotation in our model predictions. This is because it is non-trivial to derive physically reliable (and unique) post-collision rotation states within our post-processing chain for SPH collision simulations. A major point to consider is that our definition of a remnant is not restricted to a single physically connected fragment, but also includes all gravitationally bound fragments in addition. In reality, these fragments may or may not be actually accreted at some later point in time, or interact otherwise with each other. Nevertheless, we consider including these fragments into the definition of *remnant* as the best possible option, considering the alternative of simply ignoring them. This comes on top of the general issue that approximate rotational equilibrium has to be achieved after the collision in order to extract a reliable rotation state, which is highly scenario-dependent in terms of the relevant dynamics and time-scales. Considering those difficulties, we decided not to include the rotation state in our ML model predictions. Therefore, while pre-collision rotation is fully accounted for, we do not attempt to predict post-collision rotation states in this work.

Our optimization objective is to minimize the mean absolute error (MAE) between model predictions y_i^T and ground truth labels z_i^T over our training data:

$$y_i^T = f\left(y_i^0, T, \phi\right) \tag{7}$$

$$\mathcal{L} = \frac{1}{M} \sum_{i=1}^{M} \xi \cdot ||y_i^T - z_i^T|| \tag{8}$$

$f : \mathbb{R}^d \rightarrow \mathbb{R}^k$ refers to an ML model that regresses final states when given initial states. In this work, we focus on handling macroscopic system states for both model inputs and outputs, resulting in $d = 25$ and $k = 27$. f has learnable parameters ϕ that we aim to optimize. M refers to the size of the training split for individual crossvalidation folds.

We treat every unit of mass (i.e. every kilogram) as equally important in our ML task. We account for this treatment by calculating mass-dependent weights ξ in order to re-weight errors for output features that correspond to the largest remnant, second-largest remnant, and the rest (of material), respectively.

$$\xi = [\xi_{\text{lr}}, \ \xi_{\text{2lr}}, \ \xi_{\text{rest}}] \tag{9}$$

$$\xi = \left[\frac{m_{\text{lr}}}{m_{\text{tot}}}, \frac{m_{\text{2lr}}}{m_{\text{tot}}}, \frac{m_{\text{rest}}}{m_{\text{tot}}} \right] \tag{10}$$

$$m_{\text{tot}} = m_{\text{lr}} + m_{\text{2lr}} + m_{\text{rest}} \tag{11}$$

We consider this re-weighting to be essential to accurately reflect the prediction problem, especially if $m_{\text{2lr}} << m_{\text{lr}}$ or $m_{\text{rest}} << m_{\text{lr}}$. Early experiments without re-weighting lead to poor prediction performance originating from small objects (remnants and fragments). In general, small objects are more difficult to predict compared to large objects. Moreover, the assignment of the second-largest remnant tends to jump in the presence of many small objects, making it almost impossible to predict robustly. This labelling noise then leads to large error gradients, hampering learning significantly. This problem is ameliorated with our re-weighting approach.

We use identical training hyperparameters for our deep learning models (FFN and RES). Training is performed via stochastic gradient descend (Robbins & Monro 1951), utilizing the backpropagation algorithm (Kelley 1960; Rumelhart, Hinton & Williams 1986). We use the adamax optimizer (Kingma & Ba 2014) with default hyperparameters, a minibatch size of $bs = 128$, a constant learning rate of $\eta = 0.0005$, and a weight decay of $wd = 0.0001$. We apply gradient-norm clipping (Pascanu, Mikolov & Bengio 2013), allowing for maximum gradient norms of $n_{\text{grad}} = 50$. Moreover, we use exponential moving average models (Ruppert 1988; Polyak 1990; Tarvainen & Valpola 2017) with a rate of $r_{\text{ema}} = 0.999$ for validation and testing. We find that the mean-squared error leads to worse validation performance than the MAE, which is more robust to outliers. To alleviate the exploding gradient problem as described by Metz et al. (2021), we additionally penalize too large activations of hidden states h_t in our RES model. We train each of our models for 5000 epochs, which is sufficient to reach convergence.

Since different ML architectures are inherently difficult to compare, we try to find the best architectures and respective models in terms of validation performance for each model class (i.e. FFN and RES) separately. We optimize hyperparameters manually in an iterative manner (i.e. always optimizing one hyperparameter at a time while keeping others fixed, and repeating the procedure until convergence in validation performance) and dedicate approximately the same amount of time and computational resources to optimize each set of hyperparameters for FFN and RES. Table B1 summarizes all optimized hyperparameters for FFN and RES, while LIN has no model-class hyperparameters. In order to prevent information leakage and misleading test performance, we solely perform hyperparameter finetuning based on validation performance. Test performance is measured after model development was completed. In principle, RES allow using intermediate states as additional learning signals. Unless stated otherwise, we only use final states for training to ensure a fair comparison with our baselines.

We use the root mean squared error (RMSE) as our validation metric. For certain applications, prediction speed may play a significant role. We note that the RMSE metric does not account for this aspect and thus purely focuses on prediction accuracy. All performance results reported below are obtained by first taking the best RMSE

(minimum) over all epochs for every fold individually, then averaging them over the folds. Errors indicate the minimum and maximum (over folds) of best RMSE values. The same procedure holds for classification accuracies, except for first taking the maximum instead of the minimum. Measuring and interpreting RMSE in the data space is unintuitive in our multi-task setting due to vastly different value ranges of individual quantities. Thus, RMSE is measured in ML feature-space. Moreover, since RMSE values have to be interpreted in consideration of the overall ML task, we recommend comparing reported results relative to each other.

We use the balanced accuracy score for validating classification performances of our models. Balanced accuracy in the multiclass classification setting is defined by taking the average of true-positive rates for individual classes. The true-positive rate is also referred to as sensitivity or recall. Considering the strong class-imbalances that are typically present in planetary collision data, we consider the balanced accuracy score to be much more problem-focused and more applicable compared to the unbalanced accuracy score.

2.2.6 *Efficiency considerations of ML*

In practice, researchers are interested at which point using ML starts to pay off compared to classical approaches for a fixed computational budget. Consider the goal of predicting m collision outcomes. For classical approaches such as PIM or direct SPH simulations, we only need to consider inference times, which scale linearly with m. On the other hand, ML requires consideration of data generation, training, and inference. In general, the required computation times for these three components are mostly independent from each other. In our case, data generation requires by far the most time, followed by training. Finally, ML inference requires only a tiny fraction of the overall computation budget. Thus, we recommend using ML approaches in case of extensive inference, i.e. large m. Let us define the three computation times τ_d for generating one data point, τ_t for ML model training, and τ_i for inference of one data point. In our case, we consider τ_t as the total wall-clock training time for fivefolds, each having 5000 epochs. N refers to the training data set size. We can calculate for which m the use of ML pays off (i.e. $T_\mathrm{ML} < T_\mathrm{CL}$) when comparing the overall computation times T_CL for classical approaches with computation times T_ML for ML approaches:

$$T_\mathrm{CL} = m \times \tau_\mathrm{i,CL} \tag{12}$$

$$T_\mathrm{ML} = N \times \tau_\mathrm{d} + \tau_\mathrm{t} + m \times \tau_\mathrm{i,ML} \tag{13}$$

$$m > \frac{N \times \tau_\mathrm{d} + \tau_\mathrm{t}}{\tau_\mathrm{i,CL} - \tau_\mathrm{i,ML}} \tag{14}$$

3 RESULTS

3.1 SPH collision data

The provided SPH data serves as the basis for ML models in order to solve the collision treatment problem accurately and fast. To our knowledge, this data set is the first of its kind to combine different aspects such as object rotation, realistic object models including water layers, and providing time-series data. All data can be freely accessed (see Section 1.1).

Our results are consistent with Leinhardt & Stewart (2012) and Stewart & Leinhardt (2012) in identifying three major outcome regimes, erosion, accretion, and hit-and-run. We define these regimes as

(i) erosion: $m_{lr} < m_t$
(ii) accretion: $m_{lr} > m_t \wedge m_{2lr} \leq 0.1 m_p$
(iii) hit-and-run: $m_{lr} > m_t \wedge m_{2lr} > 0.1 m_p$

where subscripts indicate the largest remnant, second-largest remnant, target, and projectile, respectively. Each of these regimes can be further divided into subclasses, based on thresholds for remnant masses, as defined in Table 4, and plotted in Fig. 5. Erosion typically results from high impact velocities and/or low impact angles, whereas accretion mostly emerges for lower impact velocities. Hit-and-run either results from high impact angles, or from a combination of lower impact angles, large-enough projectile-to-target mass ratios, and impact velocities that are low enough to avoid global disruption but high enough to avoid an accretion-type outcome.

Table 4. Class counts for different outcome regimes, and for random and realistic collision parameters. The three major outcome regimes are each further divided into subclasses, based on remnant masses. Random parameters are obtained from uniform, random sampling, whereas realistic conditions are obtained from dynamically consistent N-body simulations.

Class	Subclass	Random	Realistic
Erosion	$m_{lr} < m_t/2$	18.3 % (1856)	3.0 % (151)
Erosion	$m_{lr} > m_t/2$	55.1 % (5600)	58.5 % (2968)
Accretion	$m_{lr} < m_t + m_p/2$	1.3 % (135)	2.0 % (102)
Accretion	$m_{lr} > m_t + m_p/2$	5.2 % (526)	5.5 % (281)
Hit-and-run	$m_{2lr} < m_p/2$	0.4 % (40)	4.4 % (221)
Hit-and-run	$m_{2lr} > m_p/2$	19.7 % (2007)	26.7 % (1353)

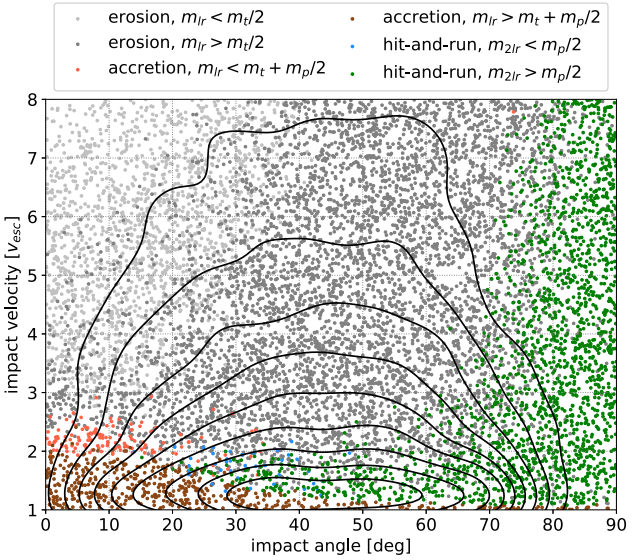

Figure 5. Overview of collision outcomes in impact angle – impact velocity space. Each data point represents a simulation in our SPH data set. Colours indicate major outcome regimes: erosion (grayish), accretion (reddish), and hit-and-run (greenish). Each regime is further divided into subcategories, depending on remnant masses. The contour overlay indicates collision statistics in a realistic dynamical environment, obtained from N-body simulations by Burger et al. (2020b), see Section 2.1.1. Contour levels correspond to iso-proportions of the density (in 10 per cent steps).

Table 5. Test performance (RMSE and balanced accuracy, both measured on the final state) of different approaches for planetary collision treatment. Classification is performed as a post-processing step on top of predicted masses. We assume the SPH simulation data to be the ground truth. For single-task learning (last six rows), each entry corresponds to the performance of individually trained ML models and column headers indicate optimized tasks, respectively. We use data from Timpe et al. (2020b) to obtain the results in the last four rows. Best results are indicated in bold, whereas $*$ indicate statistically significant results according to a Wilcoxon test (comparing FFN and RES). Our proposed RES model outperforms the other baseline methods in most experiments.

Method	Notes	Mass	Material	Position	Velocity	Total	Accuracy
PIM	o.o.d. test	$0.1501^{+0.0000}_{+0.0000}$	$0.0605^{+0.0000}_{+0.0000}$	$0.3046^{+0.0000}_{+0.0000}$	$0.2229^{+0.0000}_{+0.0000}$	$0.2450^{+0.0000}_{+0.0000}$	$0.1667^{+0.0000}_{+0.0000}$
LIN	o.o.d. test	$0.0460^{+0.0009}_{-0.0012}$	$0.0250^{+0.0002}_{-0.0002}$	$0.1880^{+0.0007}_{-0.0007}$	$0.1509^{+0.0015}_{-0.0014}$	$0.1479^{+0.0009}_{-0.0010}$	$0.2340^{+0.0043}_{-0.0057}$
FFN	o.o.d. test	$0.0121^{+0.0008}_{-0.0008}$	$0.0129^{+0.0006}_{-0.0003}$	$0.0487^{+0.0014}_{-0.0012}$	$0.0433^{+0.0010}_{-0.0010}$	$0.0408^{+0.0008}_{-0.0009}$	$\mathbf{0.4964^{+0.0480}_{-0.0538}}$
RES	o.o.d. test	$\mathbf{*0.0108^{+0.0003}_{-0.0005}}$	$\mathbf{0.0127^{+0.0002}_{-0.0002}}$	$\mathbf{*0.0386^{+0.0015}_{-0.0018}}$	$\mathbf{0.0428^{+0.0004}_{-0.0003}}$	$\mathbf{*0.0364^{+0.0009}_{-0.0010}}$	$0.4887^{+0.0160}_{-0.0139}$
FFN	o.o.d. test, + labels	$0.0122^{+0.0009}_{-0.0010}$	$\mathbf{0.0136^{+0.0002}_{-0.0004}}$	$0.0500^{+0.0015}_{-0.0020}$	$0.0433^{+0.0005}_{-0.0004}$	$0.0416^{+0.0009}_{-0.0010}$	$0.5309^{+0.0673}_{-0.1678}$
RES	o.o.d. test, + labels	$\mathbf{*0.0107^{+0.0008}_{-0.0009}}$	$0.0136^{+0.0002}_{-0.0002}$	$\mathbf{*0.0372^{+0.0012}_{-0.0014}}$	$\mathbf{0.0426^{+0.0011}_{-0.0006}}$	$\mathbf{*0.0358^{+0.0007}_{-0.0009}}$	$\mathbf{0.5311^{+0.0548}_{-0.0311}}$
FFN	o.o.d. test, single	$\mathbf{0.0083^{+0.0003}_{-0.0004}}$	$0.0098^{+0.0002}_{-0.0002}$	$0.0422^{+0.0009}_{-0.0010}$	$\mathbf{0.0409^{+0.0007}_{-0.0008}}$	–	$0.4720^{+0.0417}_{-0.0470}$
RES	o.o.d. test, single	$0.0083^{+0.0008}_{-0.0004}$	$\mathbf{*0.0090^{+0.0003}_{-0.0003}}$	$\mathbf{*0.0364^{+0.0024}_{-0.0017}}$	$0.0422^{+0.0004}_{-0.0010}$	–	$\mathbf{*0.5165^{+0.0349}_{-0.0228}}$
PIM	i.i.d. test, Timpe	$0.2088^{+0.0000}_{+0.0000}$	$0.1925^{+0.0000}_{+0.0000}$	$0.3691^{+0.0000}_{+0.0000}$	$0.2880^{+0.0000}_{+0.0000}$	–	$0.1667^{+0.0000}_{+0.0000}$
LIN	i.i.d. test, Timpe	$0.0518^{+0.0003}_{-0.0002}$	$0.0473^{+0.0003}_{-0.0005}$	$0.2105^{+0.0005}_{-0.0002}$	$0.1693^{+0.0002}_{-0.0005}$	–	$0.3160^{+0.0025}_{-0.0015}$
FFN	i.i.d. test, Timpe	$0.0132^{+0.0002}_{-0.0001}$	$0.0144^{+0.0002}_{-0.0001}$	$0.0877^{+0.0017}_{-0.0012}$	$0.0631^{+0.0006}_{-0.0004}$	–	$0.4675^{+0.0039}_{-0.0021}$
RES	i.i.d. test, Timpe	$\mathbf{*0.0126^{+0.0003}_{-0.0004}}$	$\mathbf{0.0141^{+0.0006}_{-0.0006}}$	$\mathbf{*0.0840^{+0.0037}_{-0.0036}}$	$\mathbf{0.0628^{+0.0007}_{-0.0009}}$	–	$\mathbf{0.4690^{+0.0026}_{-0.0023}}$

3.2 ML experiments

Below we present our results for the experiments described in Section 2.2.5. Results indicated by * are statistically significant ($p < 0.05$) according to a Wilcoxon test when comparing FFN with RES for the respective experiments.

3.2.1 Performance

We compare commonly used methods for planetary collision treatment with our proposed RES model and summarize the results in Tables 5 and B2. Our o.o.d. test set consists of data points within manually selected regions in the impact angle – impact velocity space (dashed regions in Fig. 7).

All deep learning models outperform the PIM and LIN baselines by a large margin. Improved performance over PIM was expected, since it is an analytic model that applies very simplistic assumptions on collision dynamics. However, we still regard PIM as a useful method in case of limited computational resources. Improved results over the LIN baseline were also expected since it assumes that the data are linearly dependent. RES perform best amongst the deep learning approaches that we studied, significantly outperforming the FFN baseline, also illustrated in Fig. 6. In general, our deep learning models generalize well to the o.o.d. test set, indicating that they might even generalize beyond our covered parameter space (Table 1). RES consistently outperforms the FFN baseline in terms of RMSE on the o.o.d. test set.

We do not observe performance gains when using intermediate states as additional labels during training. We observe a shrinkage effect (regression to the data set mean) in model predictions when using intermediate states, which may harm performance measured on the final state. Moreover, non-converged intermediate states may have a relatively high labelling noise due to the discretization into remnants, potentially making intermediate states more difficult to predict compared to final states. We also believe that intermediate macro states are somewhat redundant, unless operating on a microscopic scale, i.e. directly learning from SPH particle representations or similar.

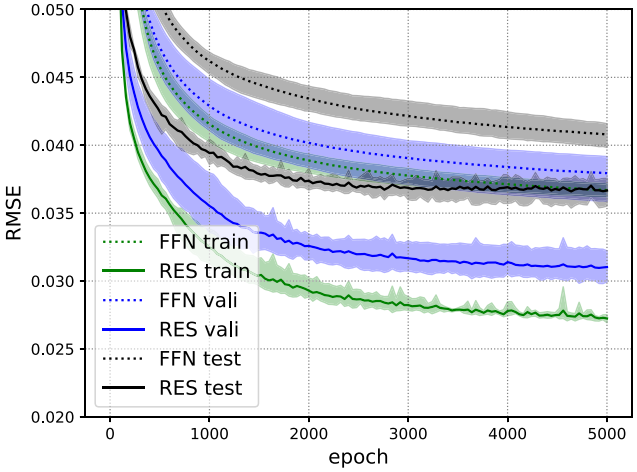

Figure 6. Learning curves depicting the training, validation, and test performance for our multi-task objective. RMSE is measured on the final state ($t = T$). Shaded regions indicate the minimum and maximum performance over the five crossvalidation folds. Performance increases significantly during the first 1000 epochs before converging. RES significantly outperforms the FFN baseline.

Direct prediction of final states with FFNs only allows analysis of hidden representations, which are typically too abstract to interpret due to their high-dimensional, learned nature. In addition to the analysis of hidden representations, our RES architecture allows analysing predicted intermediate states, opening up an entry point for interpreting model predictions in more detail. This is also illustrated in Fig. B1. Although our learning objective contains no incentive for predicted remnant trajectories to align with ground truth trajectories, we observe spatio-temporally continuous transitions from initial to final states. This indicates that steps in RES may correlate to the temporal evolution of the physical system to a certain extent, even though our results do not allow for strong conclusions.

We also train our ML models on the planetary collision data set provided by Timpe et al. (2020a, b; single-task setting) and report

Table 6. Number of learnable parameters and number of optimized hyperparameters of our methods, as well as typical computation times for data generation, training, and inference on the same hardware (GPU: Nvidia GTX 1080Ti). τ_d and τ_i are the average computation times for generating one data point and model inference for one data point, whereas τ_t is the total wall-clock training time for fivefolds, each having 5000 epochs. The required time for data generation is $\tau_D = N \times \tau_d$ and takes the largest part of the overall computational budget. s_h is the number of RES model steps taken per simulated hour.

Method	#Param	#Hyper	τ_d	τ_t	τ_i
PIM	0	0	–	–	<1 s
LIN	729	0	0.75 h	<1 s	<1 s
FFN	22203	2	0.75 h	11.95 h	<1 s
RES, $s_h = 1$	64417	3	0.75 h	33.51 h	<1 s
RES, $s_h = 2$	64417	3	0.75 h	44.78 h	<1 s
RES, $s_h = 3$	64417	3	0.75 h	56.45 h	<1 s
RES, $s_h = 4$	64417	3	0.75 h	66.74 h	<1 s
SPH	0	∼5	–	–	0.75 h

Table 7. RMSE of deep learning models on the fixed-size o.o.d. test set when using 100 per cent, 50 per cent, and 25 per cent of training data. RES requires 50 per cent less data to achieve similar performance compared to FFN.

Method	100 per cent data	50 per cent data	25 per cent data
FFN	$0.0408^{+0.0008}_{-0.0009}$	$0.0442^{+0.0004}_{-0.0005}$	$0.0516^{+0.0012}_{-0.0012}$
RES	$\mathbf{*0.0364^{+0.0009}_{-0.0010}}$	$\mathbf{*0.0398^{+0.0009}_{-0.0013}}$	$\mathbf{*0.0469^{+0.0013}_{-0.0010}}$

performance results on the independent and identically distributed (i.i.d.) test set in Table 5. Without performing any additional hyperarameter finetuning, we observe that RES outperforms the other baseline methods in all sub-tasks, verifying that our RES model is data set-agnostic. However, our results are not statistically significant for 2/4 subtasks.

3.2.2 *Efficiency*

We report the required computational resources of our methods in Table 6. As expected, PIM marks one extremum of the accuracy-speed trade-off, requiring the least amount of computational resources. On the other hand, SPH marks the other extremum, scaling badly with m (see Section 2.2.6). ML models cover intermediate trade-off regions, depending on the model class and hyperparameter choices. ML models are approximately 4 magnitudes faster in inference compared to SPH (both running on a single GPU), allowing ML models to be efficiently used in large-scale planetary evolution simulations. Considering the results from Table 6, for our deep learning models it holds that $N \times \tau_d \gg \tau_t \gg N \times \tau_i$ for $N = 10164$. This indicates that the most effective way to save computation when using ML is by requiring less training data, i.e. small N. However, using less data will inevitably lead to worse performance in terms of RMSE and to degrading generalization.

Therefore, we perform ablation studies using different training data set sizes to investigate data-efficiency of our deep learning models and report the corresponding o.o.d. test performances in Table 7. The results indicate that RES requires 50 per cent less data to achieve comparable performance to the FFN baseline. In other words, using RES is much more efficient than the FFN baseline in the case of comparable performance. Once trained, ML methods are practically as fast as PIM, while maintaining high prediction accuracy.

Table 6 also summarizes the number of learnable parameters and the number of model class hyperparameters of all collision treatment methods we studied in this work. ML training requires additional hyperparameters such as bs, η, wd, and n_grad. The number of learnable parameters for LIN is fully determined by the ML task, i.e. the dimensions of input and label vectors. For deep learning methods, the number of learnable parameters depends on the hyperparameter choices that ultimately define model architectures. We optimize hyperparameters w.r.t. validation performance and perform extensive ablation studies for both FFN and RES to verify their optimal hyperparameters. We consider the reported model capacities (i.e. the number of learnable parameters) to be optimal in terms of validation performance for our data. We find that the optimal FFN architecture requires less learnable parameters compared to the optimal RES architecture. In particular, increasing FFN's size does not improve its prediction performance anymore. We refer to Section B2 for more details about hyperparameters. The number of hyperparameters for the direct-SPH method accounts for the most important method-specific aspects such as the smoothing length and settings related to the equation of state of the simulated material.

We study the effects of multi-task learning and single-task learning on model performance and report ablation studies in Table 5. Multi-task learning is computationally more efficient by design (i.e. prediction speed and required parameters), requiring only a single model for predicting several different modalities. However, we observe a performance decrease when comparing multi-task learning (one model with four tasks) to single-task learning (four models, each with one task) in favour of single-task learning for both FFN and RES. We perform statistical significance tests (Wilcoxon) between single-task and multi-task experiments. We find single-task significantly ($p < 0.05$) outperforms multi-task for almost all subtasks for both FFN and RES. Two exceptions are performances for the position and velocity tasks on the o.o.d. test set for the RES model. We conclude that our hypothesis of improved generalization due to mutual benefit via exploiting shared representations does not hold in our experiments. Since we optimized model hyperparameters for the multi-task setting, single-task models might be even better with further finetuning. RES outperforms FFN in 4/4 subtasks for multi-task learning, but only in 2/4 for single-task learning, indicating that RES benefits a bit more from multi-task learning.

We verify that our regression-approach is suited to perform classification of different collision scenarios as a post-processing step, avoiding the need for two-step classification-regression approaches. Classification accuracies are calculated on top of regression results w.r.t. the six different classes as defined in Table 4. Balanced accuracies are reported in Table 5. Fig. 7 visualizes predicted collision outcomes based on predicted masses of the largest and second-largest remnants.

We observe that our ML models typically tend to mispredict actual accretion scenarios as hit-and-run, ultimately resulting in misclassifications in post-processing. This is pronounced for low-velocity (close to v_esc) collisions, and particularly for lower impact angles ($\lesssim 30^\circ$), and directly visible when comparing Figs 7 and 5. We assume that this mistreatment stems from the relative under-representation of accretion scenarios in our data (see Table 4), which results in models having poor classification performance for the respective scenarios. This is a common problem in imbalanced classification tasks and can be tackled with different approaches such as generating more data for under-represented classes, oversampling of under-represented classes, regularizing the model during training, or introducing problem-specific model architectures. Note that the balanced accuracy score succeeds in reflecting the reduced classi-

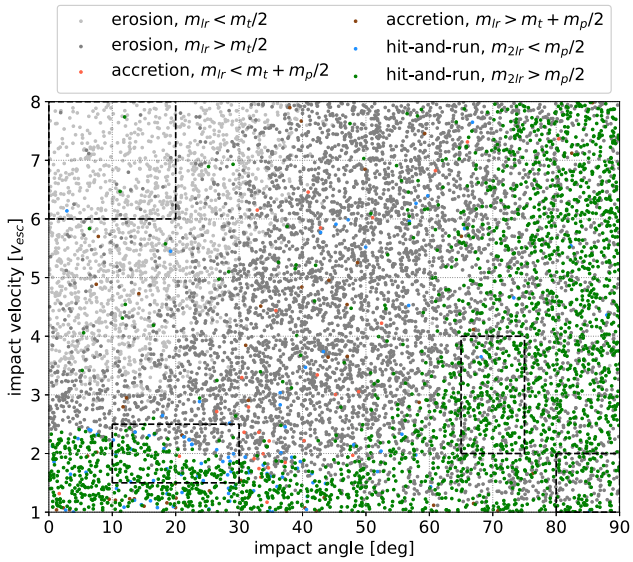

Figure 7. Classification results on validation and o.o.d. test data using predicted remnant masses of the RES model. Data points for validation are obtained by combining the respective validation splits that originate from different crossvalidation folds, whereas o.o.d. test set regions are marked with dashed boxes. The model learned to differentiate between typical collision scenarios/outcomes and generalizes to the o.o.d. test set.

fication performance on under-represented classes. In contrast, the unbalanced accuracy score typically tends to be much higher, since under-represented classes are not accounted for properly.

4 CONCLUSION

We perform *N*-body and SPH simulations to tackle the problem of accurate and fast treatment of planetary collisions with ML methods. We use the SPH data to employ a simple but problem-adapted ML model for predicting masses, material fractions, positions, and velocities of the two largest post-collision remnants and the remaining debris. Our model helps the modelling of temporal dynamics by evolving system states in an autoregressive manner, which closely resembles the data-generating process. The model allows for flexible prediction of post-collision states at different times and can be employed for collision treatment within existing *N*-body simulation frameworks.

We summarize our experiment results by comparing the performance of our two best methods, the FFN baseline and our proposed RES model, for all experiments on the respective test sets. We count a total of 24 comparisons. RES outperforms FFN in 20/24 cases. In 13/20 cases for RES and 0/4 cases for FFN, improvements are statistically significant ($p < 0.05$). Moreover, RES is also more data-efficient, achieving similar performance while requiring approximately half as much data compared to FFN. Although multi-task learning is more computationally efficient than single-task learning, we do not observe improved generalization induced by shared representations.

We demonstrate that the FFN baseline is outperformed by our RES model due to its problem-adapted algorithmic bias (i.e. its autoregressive structure), which is better suited for modelling the underlying physical processes in planetary collisions over time (see also Section 2.2.2). The optimal RES architecture requires more learnable parameters than the optimal FFN architecture because RES consists of three neural network modules, whereas FFN consists of only one module. However, we find that increasing FFN's size does not increase its performance anymore. This finding indicates that the performance improvement from RES originates from its better algorithmic bias rather than from its relatively larger number of learnable parameters. Moreover, the superiority of RES is apparent in both our own data and data from Timpe et al. (2020b), indicating a general trend rather than a data-specific effect.

Measuring the actual effects of systematic errors induced by ML model predictions still remains an open topic in the context of planetary formation and evolution modelling. Thus, a natural follow-up work could be to test various ML methods for collision treatment in such simulations.

Beyond studying the basic task of collision outcome prediction, our data and methods also open up further interesting lines of research related to planet formation in general. This includes studying inverse problems or focusing on specific collision scenarios (Canup, Barr & Crawford 2012; Chau et al. 2018). Other possible directions are the extension of our methods to different regimes such as small bodies or objects including (proto-)atmospheres, probably requiring to extend the underlying physics model. The latter can include more sophisticated equations of state for more realistic thermodynamics and advanced models for material strength to accurately simulate solid-body behaviour.

ML models might benefit from learning directly based on microscopic representations, i.e. fragments or even down to the SPH particle level, and thereby improve aspects regarding generalization and interpretability. Incorporating certain aspects like symmetries, conserved quantities, and sophisticated numerical approaches that have been developed in recent years (Chen et al. 2020; Alet et al. 2021; Hoedt et al. 2021; Satorras, Hoogeboom & Welling 2021; Brandstetter et al. 2022) could be promising directions for further improvements of ML architectures. For example, graph neural networks (GNNs; Scarselli et al. 2009; Defferrard, Bresson & Vandergheynst 2016; Kipf & Welling 2017) and regression forests (Ho 1995) have been successfully applied to the approximation of numerical simulations (Ladický et al. 2015; Martinkus, Lucchi & Perraudin 2020; Pfaff et al. 2020; Sanchez-Gonzalez et al. 2020; Mayr et al. 2021). Efficient ML approaches begin to replace traditional PDE solvers in the context of hydrodynamic simulations (Li et al. 2020a,b). Residual neural networks (He et al. 2016) showed promising results in modelling complex dynamical processes by formulating the neural network layer-structure as a continuous-depth model (Queiruga et al. 2020) in the context of neural ODEs (Chen et al. 2018; Kidger 2022). ML methods allow accurate collision modelling at scale, while at the same time being orders of magnitude faster compared to classical, non data-driven approaches.

ACKNOWLEDGEMENTS

Christoph Burger (CB) and Christoph M. Schäfer (CMS) appreciate support by the German Research Foundation (DFG) project 398488521. CB acknowledges support by the Austrian Science Fund (FWF) project S11603-N16. Thomas I. Maindl (TIM) acknowledges support from the Austrian Science Fund (FWF): P33351-N. The authors acknowledge support by the High Performance and Cloud Computing Group at the Zentrum für Datenverarbeitung of the University of Tübingen, the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant no. INST 37/935-1 FUGG.

We thank Sepp Hochreiter, Günter Klambauer, Johannes Brandstetter, Pieter-Jan Hoedt, Max Zimmermann, Rudolf Dvorak, Kajetan Schweighofer, Andreas Mayr, Miles Timpe, and Christian Reinhardt

for contributing valuable feedback and comments. Simulations for our data sets made use of the REBOUND code (Rein & Liu 2012). We thank the developers of PYTHON (Van Rossum & Drake 2009), PYTORCH (Paszke et al. 2019), MATPLOTLIB (Hunter 2007), and NUMPY (Harris et al. 2020), whose contributions facilitate the advancement of our work.

DATA AVAILABILITY

As extensive simulation data sets for ML begin to emerge (Villaescusa-Navarro et al. 2022), we advocate making these data sets publicly available in order to optimize the benefit from network effects among the research community. Data sharing helps to reduce spending computational resources for data generation, making ML approaches more efficient and sustainable in the long term.

We provide our data, source code, and pre-trained ML models to encourage independent researchers to reproduce our results and to incorporate our methods into their own work. The data underlying this article are available in the Phaidra repository at https://phaidra.univie.ac.at/o:1206181, and can be accessed with the handle 11353/10.1206181. Our source code and pre-trained ML models can be accessed at https://github.com/littleblacksheep/csv/tree/main. We kindly ask users to report possible bugs to winter@murena.io.

REFERENCES

Agarwal S., Tosi N., Kessel P., Breuer D., Montavon G., 2021, Phys. Rev. Fluids, 6, 113801
Agnor C. B., Canup R. M., Levison H. F., 1999, Icarus, 142, 219
Alet F., Doblar D., Zhou A., Tenenbaum J. B., Kawaguchi K., Finn C., 2021, preprint (arXiv:2112.03321)
Bonomo A. S. et al., 2019, Nat. Astron., 3, 416
Bonsor A., Leinhardt Z. M., Carter P. J., Elliott T., Walter M. J., Stewart S. T., 2015, Icarus, 247, 291
Brandstetter J., Hesselink R., van der Pol E., Bekkers E. J., Welling M., 2022, preprint (arXiv:2110.02905)
Breiman L., Friedman J. H., Olshen R. A., Stone C. J., 1984, Classification and Regression Trees. Wadsworth and Brooks, Monterey, CA
Brown T. B. et al., 2020, in Larochelle H., Ranzato M., Hadsell R., Balcan M. F., Lin H., eds, Language Models are Few-Shot Learners, vol. 33, Curran Associates, Inc., p. 1877, https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf
Burger C., 2019, Dissertation. University of Vienna, Austria
Burger C., Schäfer C. M., 2017, The First Greek-Austrian Workshop on Extrasolar Planetary Systems, Proceedings of the First Greek-Austrian Workshop on Extrasolar Planetary Systems. p. 63
Burger C., Maindl T. I., Schäfer C. M., 2018, Celest. Mech. Dyn. Astron., 130, 2
Burger C., Maindl T. I., Schäfer C., 2020a, in Elmegreen B. G., Tóth L. V., Güdel M., eds, IAU Symposium, Vol. 345, IAU Symposium. p. 287 preprint (arXiv:2002.00231), doi:10.1017/S1743921318008621
Burger C., Bazsó Á., Schäfer C. M., 2020b, A&A, 634, A76
Cambioni S., Asphaug E., Emsenhuber A., Gabriel T. S. J., Furfaro R., Schwartz S. R., 2019, ApJ, 875, 40
Cambioni S., Jacobson S. A., Emsenhuber A., Asphaug E., Rubie D. C., Gabriel T. S. J., Schwartz S. R., Furfaro R., 2021, Planet. Sci. J., 2, 93
Canup R. M., Barr A. C., Crawford D. A., 2012, Icarus, 222, 200
Carter P. J., Leinhardt Z. M., Elliott T., Walter M. J., Stewart S. T., 2015, ApJ, 813, 72
Carter P. J., Leinhardt Z. M., Elliott T., Stewart S. T., Walter M. J., 2018, Earth Planet. Sci. Lett., 484, 276
Chambers J. E., 2001, Icarus, 152, 205
Chambers J., 2013, Icarus, 224, 43
Chambers J. E., Wetherill G. W., 1998, Icarus, 136, 304
Chau A., Reinhardt C., Helled R., Stadel J., 2018, ApJ, 865, 35
Chen T., Guestrin C., 2016, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16. ACM, New York, NY, USA, p. 785
Chen R. T. Q., Rubanova Y., Bettencourt J., Duvenaud D., 2018, Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18. Curran Associates Inc., Red Hook, NY, USA, p. 6572, http://doi.acm.org/10.1145/2939672.2939785
Chen Z., Zhang J., Arjovsky M., Bottou L., 2020, in 8th International Conference on Learning Representations, ICLR 2020, April 26-30, OpenReview.net, 2020,. Addis Ababa, Ethiopia, https://openreview.net/forum?id = BkgYPREtPr
Cho K., van Merriënboer B., Bahdanau D., Bengio Y., 2014, in Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation. Association for Computational Linguistics, Doha, Qatar, p. 103, https://aclanthology.org/W14-4012
Clement M. S., Kaib N. A., Raymond S. N., Chambers J. E., Walsh K. J., 2019, Icarus, 321, 778
Cortes C., Vapnik V., 1995, Mach. Learn., 20, 273
Cranmer M., Tamayo D., Rein H., Battaglia P., Hadden S., Armitage P. J., Ho S., Spergel D. N., 2021, Proc. Natl. Acad. Sci., 118, e2026053118
Crespi S., Dobbs-Dixon I., Georgakarakos N., Haghighipour N., Maindl T. I., Schäfer C. M., Winter P. M., 2021, MNRAS, 508, 6013
Defferrard M., Bresson X., Vandergheynst P., 2016, in Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16. Curran Associates Inc., Red Hook, NY, USA, p. 3844
Dong X., Thanou D., Rabbat M., Frossard P., 2019, IEEE Signal Process. Mag., 36, 44
Dwyer C. A., Nimmo F., Chambers J. E., 2015, Icarus, 245, 145
Elman J. L., 1990, Cogn. Sci., 14, 179
Emsenhuber A., Cambioni S., Asphaug E., Gabriel T. S. J., Schwartz S. R., Furfaro R., 2020, ApJ, 891, 6
Fischer R. A., Ciesla F. J., 2014, Earth Planet. Sci. Lett., 392, 28
Friedman J. H., 2001, Ann. Stat., 29, 1189
Geller M. J., Huchra J. P., 1983, ApJS, 52, 61
Genda H., Kokubo E., Ida S., 2011, 42nd Lunar and Planetary Science Conference. p. 2090
Genda H., Fujita T., Kobayashi H., Tanaka H., Suetsugu R., Abe Y., 2017, Icarus, 294, 234
Genda H., Iizuka T., Sasaki T., Ueno Y., Ikoma M., 2017, Earth Planet. Sci. Lett., 470, 87
Gers F. A., Schmidhuber J., Cummins F., 1999, Learning to forget: Continual Prediction with LSTM, Vol. 2, . p. 850
Gingold R. A., Monaghan J. J., 1977, MNRAS, 181, 375
Haghighipour N., Maindl T. I., 2022, ApJ, 926, 197
Haghighipour N., Raymond S. N., 2007, ApJ, 666, 436
Haghighipour N., Winter O. C., 2016, Celest. Mech. Dyn. Astron., 124, 235
Harris C. R. et al., 2020, Nature, 585, 357
Hastie T., Tibshirani R., Friedman J., 2017, The Elements of Statistical Learning. Springer, New York, NY
He K., Zhang X., Ren S., Sun J., 2016, Deep Residual Learning for Image Recognition, IEEE, Las Vegas, NV, USA, p. 770
Hearst M., Dumais S., Osuna E., Platt J., Scholkopf B., 1998, IEEE Intell. Syst. Appl., 13, 18
Ho T. K., 1995, Proceedings of 3rd international conference on document analysis and recognition, IEEE, Montreal, QC, Canada, p. 278
Hochreiter S., 1998, Int. J. Uncertain. Fuzziness Knowl.-Based Syst., 6, 107
Hochreiter S., Schmidhuber J., 1997, Neural Comput., 9, 1735
Hoedt P.-J., Kratzert F., Klotz D., Halmich C., Holzleitner M., Nearing G. S., Hochreiter S., Klambauer G., 2021, Proceedings of the 38 th International Conference on Machine Learning, PMLR, p. 139
Hornik K., Stinchcombe M., White H., 1989, Neural Netw., 2, 359
Hunter J. D., 2007, Comput. Sci. Eng., 9, 90
Inamdar N. K., Schlichting H. E., 2016, ApJ, 817, L13
Ivakhnenko A. G., Lapa V. G., 1965, Cybernetic Predicting Devices. CCM Information Corporation.U.S. department of commerce. Clearinghouse for federal scientific and technical information, Washington, D.C.
Izidoro A., de Souza Torres K., Winter O. C., Haghighipour N., 2013, ApJ, 767, 54

Johansen A., Lambrechts M., 2017, Ann. Rev. Earth Planet. Sci., 45, 359
Jordan M. I., 1986, Technical report AD-A-173989/5/XAB; ICS-8604, Serial order: A Parallel Distributed Processing Approach.US, CA
Kegerreis J. A., Eke V. R., Massey R. J., Teodoro L. F. A., 2020, ApJ, 897, 161
Kelley H. J., 1960, Ars J., 30, 947
Kidger P., 2022, On neural differential equations, University of Oxford
Kingma D. P., Ba J., 2014, in Bengio B., LeCun Y., eds, 3rd International Conference on Learning Representations, San Diego, CA, preprint (arXiv:1412.6980)
Kipf T., Welling M., 2017, preprint (arXiv:1609.02907)
Klambauer G., Unterthiner T., Mayr A., Hochreiter S., 2017, preprint (arXiv:1706.02515)
Kokubo E., Ida S., 1998, Icarus, 131, 171
Kokubo E., Ida S., 2002, ApJ, 581, 666
Kokubo E., Kominami J., Ida S., 2006, ApJ, 642, 1131
Krizhevsky A., Sutskever I., Hinton G. E., 2017, Commun. ACM, 60, 84
Kutta W., 1901, Beitrag zur näherungsweisen Integration totaler Differentialgleichungen, B.G Teubner, Leipzig, https://www.bibsonomy.org/bibtex/24ee695c671c31151cc9816e314dfa6c2/brouder
Ladický L., Jeong S., Solenthaler B., Pollefeys M., Gross M., 2015, ACM Trans. Graph., 34, 199
Leinhardt Z. M., Stewart S. T., 2012, ApJ, 745, 79
Leinhardt Z. M., Dobinson J., Carter P. J., Lines S., 2015, ApJ, 806, 23
Li Z., Kovachki N., Azizzadenesheli K., Liu B., Bhattacharya K., Stuart A., Anandkumar A., 2020a, preprint (arXiv:2010.08895)
Li Z., Kovachki N., Azizzadenesheli K., Liu B., Bhattacharya K., Stuart A., Anandkumar A., 2020b, Workshop on Integration of Deep Neural Models and Differential Equations, ICLR, Virtual Event, preprint (arXiv:2003.03485)
Lucy L. B., 1977, AJ, 82, 1013
Maindl T. I., Dvorak R., Schäfer C., Speith R., 2014, IAU Symp. 310, Complex Planetary Systems, Cambridge University Press, p. 138
Maindl T. I., Schäfer C. M., Haghighipour N., Burger C., Dvorak R., 2017, Proceedings of the First Greek-Austrian Workshop on Extrasolar Planetary Systems, CreateSpace Independent Publishing Platform, South Carolina, United States, p. 137
Marcus R. A., Stewart S. T., Sasselov D., Hernquist L., 2009, ApJ, 700, L118
Marcus R. A., Sasselov D., Stewart S. T., Hernquist L., 2010, ApJ, 719, L45
Martinkus K., Lucchi A., Perraudin N., 2020, preprint (arXiv:2010.06948)
Mayr A., Lehner S., Mayrhofer A., Kloss C., Hochreiter S., Brandstetter J., 2021, preprint (arXiv:2105.01636)
McNeil D., Duncan M., Levison H. F., 2005, AJ, 130, 2884
Melosh H. J., 1989, Impact Cratering: A Geologic Process, Oxford University Press ; Clarendon Press New York, Oxford
Metz L., Freeman C. D., Schoenholz S. S., Kachman T., 2021, preprint (arXiv:2111.05803)
Mitchell T. M., 1980, Technical report, The Need for Biases in Learning Generalizations, Department of Computer Science, Laboratory for Computer Science Research, Rutgers Univ., New Jersey, p. 184
Monfared Z., Mikhaeil J. M., Durstewitz D., 2021, preprint (arXiv:abs/2110.07238)
Morbidelli A., Chambers J., Lunine J. I., Petit J. M., Robert F., Valsecchi G. B., Cyr K. E., 2000, Meteorit. Planet Sci., 35, 1309
Mozer M. C., 1995, Complex Syst., 3, 349
O'Brien D. P., Walsh K. J., Morbidelli A., Raymond S. N., Mandell A. M., 2014, Icarus, 239, 74
O'Brien D. P., Izidoro A., Jacobson S. A., Raymond S. N., Rubie D. C., 2018, Space Sci. Rev., 214, 47
Pascanu R., Mikolov T., Bengio Y., 2013, Proceedings of the 30th International Conference on Machine Learning, Vol. 28, PMLR, Atlanta, USA, p. 1310
Paszke A. et al., 2019, in Advances in Neural Information Processing Systems 32. Curran Associates, Inc., p. 8024
Pearlmutter B. A., 1989, Neural Comput., 1, 263
Pfaff T., Fortunato M., Sanchez-Gonzalez A., Battaglia P. W., 2020, preprint (arXiv:2010.03409)
Polyak B., 1990, Avtom. Telemekh., 7, 98
Press, William H. F. B. P., Teukolsky S. A., Vetterling W. T., 1992, Numerical Recipes in Fortran 77: The Art of Scientific Computing. Cambridge Univ. Press, Cambridge
Queiruga A. F., Erichson N. B., Taylor D., Mahoney M. W., 2020, preprint (arxiv:2008.02389)
Quintana E. V., Lissauer J. J., 2014, ApJ, 786, 33
Quintana E. V., Barclay T., Borucki W. J., Rowe J. F., Chambers J. E., 2016, ApJ, 821, 126
Quionero-Candela J., Sugiyama M., Schwaighofer A., Lawrence N. D., 2009, Dataset Shift in Machine Learning. The MIT Press, Cambridge, Massachusetts, USA
Rampásek L., Wolf G., 2021, preprint (arXiv:2107.07432)
Rasmussen C. E., Williams C. K. I., 2005, Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press, Cambridge, Massachusetts, USA
Raymond S. N., Quinn T., Lunine J. I., 2004, Icarus, 168, 1
Raymond S. N., Quinn T., Lunine J. I., 2007, Astrobiology, 7, 66
Rein H., Liu S. F., 2012, A&A, 537, A128
Reufer A., Meier M. M., Benz W., Wieler R., 2012, Icarus, 221, 296–299
Robbins H., Monro S., 1951, Annals Math. Stat., 22, 400
Robinson A. J., Fallside F., 1987, Technical Report CUED/F-INFENG/TR.1, The Utility Driven Dynamic Error Propagation Network. Engineering Department, Cambridge Univ., Cambridge, UK
Rosenblatt F., 1961, Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms, Spartan Books, Washington DC
Rumelhart D., Hinton G. E., Williams R., 1986, Nature, 323, 533
Runge C., 1895, Math. Ann., 46, 167
Ruppert D., 1988, Technical Report, Efficient Estimations from a Slowly Convergent Robbins-Monro Process, Cornell University Operations Research and Industrial Engineering
Sanchez-Gonzalez A., Godwin J., Pfaff T., Ying R., Leskovec J., Battaglia P. W., 2020, preprint (arXiv:2002.09405)
Satorras V. G., Hoogeboom E., Welling M., 2021, preprint (arXiv:2102.09844)
Scarselli F., Gori M., Tsoi A. C., Hagenbuchner M., Monfardini G., 2009, IEEE Trans. Neural Netw., 20, 61
Schäfer C., Riecker S., Maindl T. I., Speith R., Scherrer S., Kley W., 2016, A&A, 590, A19
Schäfer C. et al., 2020, Astron. Comput., 33, 100410
Schmidt D., Koppe G., Monfared Z., Beutelspacher M., Durstewitz D., 2021, Identifying nonlinear dynamical systems with multiple time scales and long-range dependencies, ICLR, Virtual Event, Austria
Srivastava R. K., Greff K., Schmidhuber J., 2015, Highway Networks, 32nd International Conference on Machine Learning, Lille, France
Stewart S. T., Leinhardt Z. M., 2012, ApJ, 751, 32
Stoer J., Bulirsch R., 1980, Introduction to Numerical Analysis, Vol. §7.2.14. Springer-Verlag, New York
Tamayo D. et al., 2020, Proc. Natl. Acad. Sci., 117, 18194
Tarvainen A., Valpola H., 2017, in Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17. Curran Associates Inc., Red Hook, NY, USA, p. 1195
Tillotson J. H., 1962, Metallic Equations of State For Hypervelocity Impact, General Atomic Report GA-3216. 1962. Technical Report.Air Force Weapons Lab, San Diego, California
Timpe M. L., Han Veiga M., Knabenhans M., Stadel J., Marelli S., 2020a, Machine learning applied to simulations of collisions between rotating, differentiated planets, Springer International Publ.
Timpe M. L., Han Veiga M., Knabenhans M., Stadel J., Marelli S., 2020b, Comput. Astrophys. Cosmol., 7, 2
Valencia D., Paracha E., Jackson A. P., 2019, ApJ, 882, 35
Van Rossum G., Drake F. L., 2009, Python 3 Reference Manual. CreateSpace, Scotts Valley, CA
Villaescusa-Navarro F., et al., 2022, ApJ, 929, 132
Werbos P. J., 1988, Neural Networks, 1, 339
Wiener N., 1938, Am. J. Math., 60, 897
Wiewel S., Becher M., Thürey N., 2019, Comput. Graph. Forum, 38, 71
Wyatt M. C., Jackson A. P., 2016, Space Sci. Rev., 205, 231
Zhou L., Dvorak R., Zhou L.-Y., 2021, MNRAS, 505, 4571

APPENDIX A: SPH DATA SET DETAILS

A1 Technical details

We use three Nvidia GeForce GTX1080 Ti (11 GB) GPUs for the generation of SPH collision data. The total computation time can be split up into pre-processing, SPH simulations, and post-processing. Pre-processing and post-processing are performed on the CPU, whereas SPH simulations are performed on the GPU. The average computation times per simulation (average over 10 164 simulations) in these categories are 169 s, 42 min, and 4.8 s using a single GPU, which resulted in approximately 317 GPU days for the entire SPH data set. Similar computation times hold for the *N*-body data set (see Section 2.1.1) from Burger et al. (2020b).

The relatively large number of data points (one data point corresponds to one simulation) does not allow for high-resolution runs due to hardware limitations. We perform data reduction on the fly to significantly reduce the memory footprint from approximately 200 TB of raw data. After post-processing, individual simulation runs require less than 2 MB of storage space. The complete data-set requires about 12.2 GB of storage space.

We performed a total of 10 794 SPH simulations. 630 of those runs failed, mostly because they ran into numerical issues either in the setup script or during the actual simulation with `miluphcuda`. The density of failed runs is higher in the disruption regime compared to other regimes due to excessively small time-steps (via the adaptive time integration). This tends to be more likely in the disruption regime, where very high pressures are more common. We end up with 10 164 valid simulations for our data set. Note that due to the inhomogeneous distribution of invalid simulations, the distribution of valid simulations is also somewhat inhomogeneous, leading to a slightly worse coverage of high-velocity, low-impact-angle scenarios (see Fig. A1). We keep configuration files of invalid simulations for possible future data analysis.

A2 Pre-collision spin

Let us consider a single rotating body, consisting of n SPH particles. The rotation axis can either be defined in spherical coordinates (radius r, azimuth ϕ, polar angle θ) or in Cartesian coordinates (x, y, z):

$$x = r\ \cos(\phi)\ \sin(\theta) \tag{A1}$$

$$y = r\ \sin(\phi)\ \sin(\theta) \tag{A2}$$

$$z = r\ \cos(\theta) \tag{A3}$$

In our case, r is either the length of the angular momentum vector $\mathbf{L}$, or the rotation period $P_{\rm rot}$.

$$\vec{L} = \sum_{i=1}^{n} \vec{r}_i \times (m_i\, \vec{v}_i) \tag{A4}$$

Here, m_i, $\vec{r}_i$, and $\vec{v}_i$ refer to the masses, positions, and velocities of individual SPH particles w.r.t. the barycentre of the object. The critical rotation period $P_{\rm rot,crit}$ is defined such that a test mass at the surface of the (idealized spherical) body is weightless according to Kepler's third law:

$$P_{\rm rot,crit} = \sqrt{\frac{4\pi^2 r^3}{G\, m}} = \frac{2\pi}{\omega_{\rm crit}} \tag{A5}$$

Here, r, m, and G refer to the object radius, its mass, and the gravitational constant. For data set generation, the rotation speed (angular velocity) ω is randomly sampled between $\omega = 0$ and $\omega = 0.2 \times \omega_{\rm crit}$, whereas the rotation axis is randomly sampled in Cartesian coordinates.

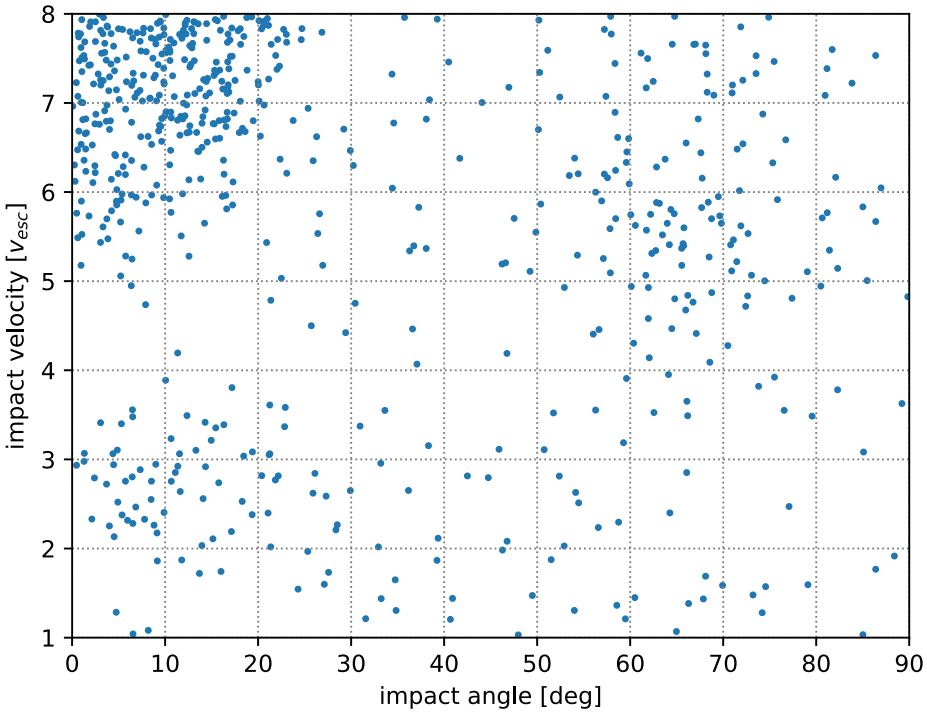

Figure A1. Distribution of invalid simulation runs. The chance that `miluphcuda` suffers from numerical issues is increased for collision scenarios with high velocities and low impact angles.

APPENDIX B: MACHINE LEARNING

B1 Rotation equivariance

An optional rotation module can be used as a pre- and post-processing step. The module rotates the system in the 3D domain, making the ML models equivariant to rotations. Thus, the models are geometrically consistent and do not require to spend their capacity on learning to be rotation equivariant. We propose to rotate and de-rotate the entire system before and after applying learnable modules.

$$y_i^0 = rot\left(\hat{y}_i^0, R_i\right) \tag{B1}$$

$$y_i^T = f\left(y_i^0, T, \Phi\right) \tag{B2}$$

$$\hat{y}_i^T = rot\left(y_i^T, R_i^{-1}\right) \tag{B3}$$

rot refers to the rotation module, f is an ML model with its respective learnable parameters Φ, and $\hat{y}$ refers to systems with any orientation, whereas y have a fixed orientation. We calculate a rotation matrix R_i for each data point such that the system is rotated into a fixed basis. The pre-rotation basis is given via the impact geometry and is calculated as follows (see Fig. 3):

$$\vec{e}_0 = \frac{\vec{v}}{|\vec{v}|} \qquad \vec{e}_1 = \frac{\vec{e}_0 \times \vec{r}}{|\vec{e}_0 \times \vec{r}|} \qquad \vec{e}_2 = \frac{\vec{e}_0 \times \vec{e}_1}{|\vec{e}_0 \times \vec{e}_1|} \tag{B4}$$

v and r are the relative velocity and relative distance vectors between the projectile and the target. The rotation matrix R_i is chosen as the inverse of the pre-rotation basis:

$$R_i = \begin{pmatrix} e_{00} & e_{01} & e_{02} \\ e_{10} & e_{11} & e_{12} \\ e_{20} & e_{21} & e_{22} \end{pmatrix}^{-1} \tag{B5}$$

The rotation module can be easily embedded into ML frameworks for collision treatment.

B2 Miscellaneous

Table B1 summarizes optimized hyperparameters for our deep learning models. Formally, model architectures can also be interpreted as hyperparameters. We tested a handful of different possible architectures and performed ablation studies before settling on the best-performing architecture, which is presented in the main text. For FFN, we tested a residual neural network approach and different activation functions.

We studied LSTMs (Hochreiter & Schmidhuber 1997; Hochreiter 1998; Gers et al. 1999) in-depth and investigated the following setups:

(i) Direct prediction of final states without Euler updates for studying performance of direct predictions versus relative predictions.
(ii) LSTM without forget gate for mitigating the vanishing gradient problem.

We also investigated several different GNN architectures (Scarselli et al. 2009; Defferrard et al. 2016; Kipf & Welling 2017). Our best architecture was a latent GNN (Dong et al. 2019), which slightly outperformed the FFN baseline. For the GNN, we investigated several architecture variations that use the initial SPH representation in combination with clustering approaches. These variations were mostly inspired by the architecture of Sanchez-Gonzalez et al. (2020). Increasing the number of graph nodes may require hierarchical graph structures (Martinkus et al. 2020; Rampásek & Wolf 2021) to account for learning long–range interactions such as gravity.

For the weight-tied residual network RES, we tested state updates via the Bulirsch–Stoer method (Stoer & Bulirsch 1980; Press, Teukolsky & Vetterling 1992) instead of the classical Euler update, but we did not observe any improvements.

Preliminary investigation of predicted water mass fractions did not allow for in-depth analysis of generalization to the low-water-content regime (i.e. $\zeta_{\text{water}} < 0.1$), because model predictions were too inaccurate.

We summarize the training and validation performance of our experiments in Table B2 . Results are consistent with the corresponding test results, which can be found in Table 5.

We provide pre-trained FFN and RES models for direct integration into existing N-body frameworks. We train these models using the entire development set (i.e. training + validation data) and report their performance in Table B3.

Table B1. Model-class hyperparameters for deep learning collision treatment approaches. Hyperparameters are optimized w.r.t. validation performance.

Method	Description	Value
FFN	Number of layers	8
FFN	Hidden state size	56
RES	Number of steps per simulated hour	4
RES	Number of layers for each module E, R, and D	3
RES	Hidden state size	64

B3 Incorporation of ML models into N-body frameworks

All our models can in principle be employed for collision treatment within existing N-body simulation frameworks. In the following, we list some important aspects related to implementation and technical details thereof:

(i) As our data are highly imbalanced (see Fig. 7 and Table 4), different use cases might require re-training the ML model using different data subsets. This benefit might be especially true for classification accuracy, which is sensitive to class-imbalances. Note that using the entire data set might lead to miss-classifications of collision outcome types (e.g. compare Fig. 7 versus Fig. 6). Also, depending on the use case, one might want to incorporate additional restrictions (such as imposing conservation laws) as post-processing step on top of model predictions. Further details are elaborated in Section 2.2.1.

(ii) Our models require consistent input features, produced by collisions that lie within our parameter space as defined in Table 1. Having consistent input features also includes applying the identical pre-processing pipeline (e.g. the numerical scaling of features) in analogy to the pre-processing pipeline that is used for model training. We believe that producing consistent input features is the most error-prone process when implementing ML methods for collision treatment. For sophisticated frameworks or if o.o.d. data points are expected to be common, one may even consider applying anomaly detection methods to check for potential invalid inputs.

(iii) Note that we define the onset of the collision process before objects come in direct contact in order to include tidal effects. This should be considered for triggering collision events in N-body frameworks. We propose taking two measurements, one when tidal interaction begins, and another one once the objects come into direct contact (cf. Fig. 3). We initialize the colliding bodies at a distance of $d_{\text{initial}} = f_i \times (R_t + R_p)$. R_{t} and R_{p} are the target and projectile radii, while the initial distance factor f_i is a hyperparameter (between $f_i = 3$ and $f_i = 7$ for our data). Therefore, reasonable estimates of R_{t} and R_{p} might be required to define the first measurement in N-body simulations.

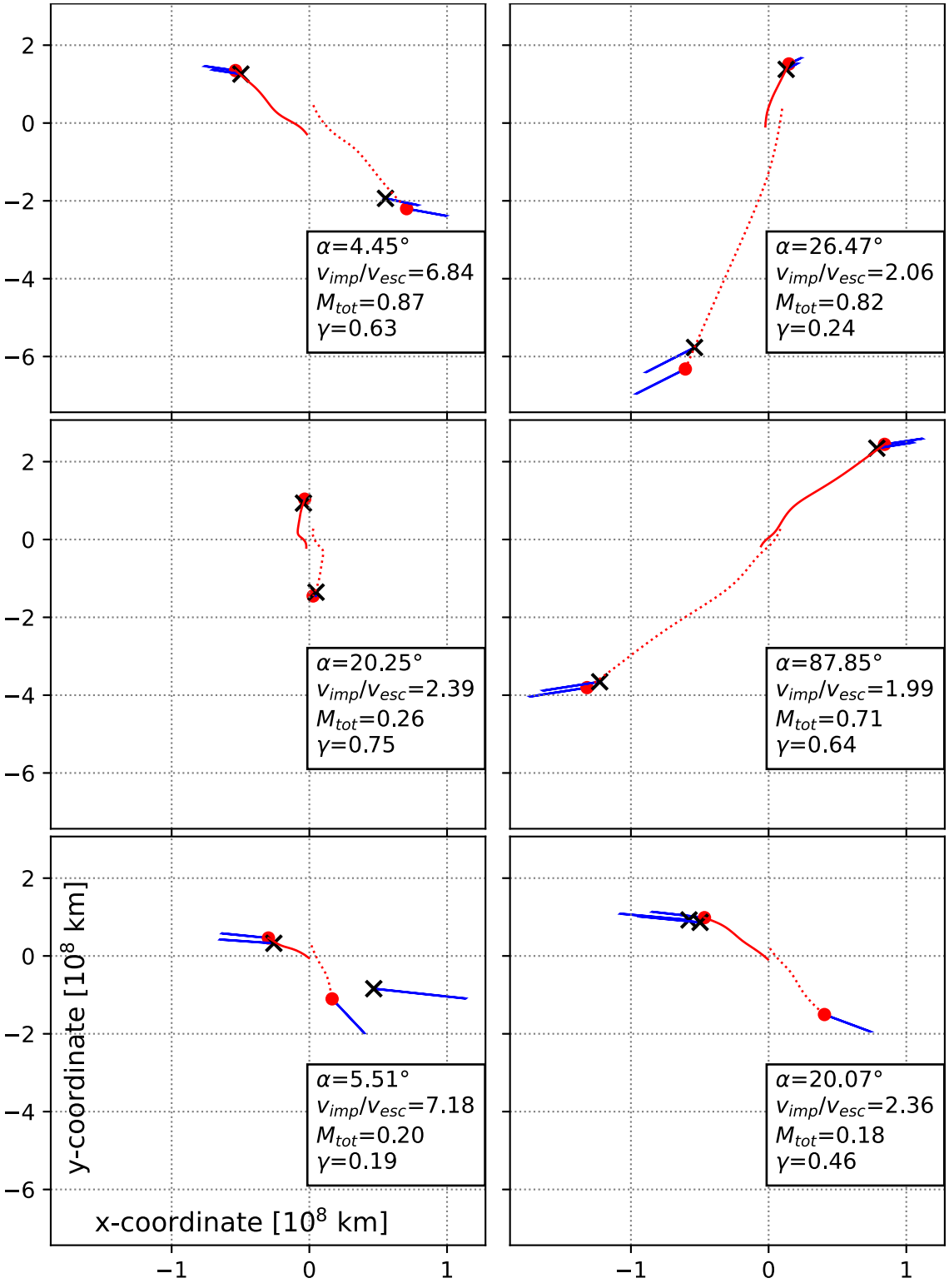

Figure B1. Visualization of predicted intermediate states (in red) of our RES model for four different data points of the o.o.d. test set. Each plot depicts two predicted trajectories, where one corresponds to the largest remnant (solid) and the other one to the second-largest remnant (dotted). Final positions are indicated by red dots (predictions) and black crosses (ground truth). Ground truth initial positions are indicated by the starting point of the respective predicted trajectories. Final velocity vectors (for both prediction and ground truth) are indicated by blue arrows for all objects. Note that projectiles are flying into the negative y-direction initially and the x–y plane is the main collision plane. The labels indicate the initial conditions (mass in units of 10^{25} kg, cf. Table 1). We find that z-components (which are caused by rotating bodies) were quite significant in many simulations. Prediction performance in the z-component is comparable to the x- and y-components. We observe spatio-temporally continuous transitions from initial to final states, possibly indicating a correlation to the temporal evolution of the physical system to a certain extent. The first four panels were cherry-picked for predicted positions in the x–y plane. The last two panels visualize typical failure cases of model predictions, where positions and velocities of the second-largest remnants are poorly approximated.

Table B2. Training and validation performance (RMSE and balanced accuracy, both measured on the final state) of different approaches for planetary collision treatment. Classification is performed as a post-processing step on top of predicted masses. We assume the SPH simulation data to be the ground truth. For single-task learning (lower half of the table), each entry corresponds to the performance of individually trained ML models and column headers indicate optimized tasks, respectively. We use data from Timpe et al. (2020b) to obtain results in the lower half. Best results are indicated in bold, whereas $*$ indicate statistically significant results according to a Wilcoxon test (comparing FFN and RES). Our proposed RES model outperforms the other baseline methods in most experiments.

Method	Split	Mass	Material	Position	Velocity	Total	Accuracy
PIM	Training	$0.1388^{+0.0008}_{-0.0012}$	$0.0619^{+0.0002}_{-0.0001}$	$0.4055^{+0.0010}_{-0.0008}$	$0.4411^{+0.0022}_{-0.0018}$	$0.3619^{+0.0010}_{-0.0007}$	$0.1667^{+0.0000}_{+0.0000}$
LIN	Training	$0.0441^{+0.0003}_{-0.0003}$	$0.0280^{+0.0001}_{-0.0001}$	$0.1617^{+0.0005}_{-0.0006}$	$0.1579^{+0.0011}_{-0.0015}$	$0.1375^{+0.0006}_{-0.0007}$	$0.2296^{+0.0068}_{-0.0088}$
FFN	Training	$0.0060^{+0.0003}_{-0.0004}$	$0.0115^{+0.0002}_{-0.0001}$	$0.0403^{+0.0007}_{-0.0007}$	$0.0446^{+0.0006}_{-0.0007}$	$0.0367^{+0.0005}_{-0.0005}$	$0.4422^{+0.0444}_{-0.0416}$
RES	Training	$\mathbf{*0.0047^{+0.0001}_{-0.0001}}$	$\mathbf{*0.0108^{+0.0002}_{-0.0003}}$	$\mathbf{*0.0272^{+0.0002}_{-0.0003}}$	$\mathbf{*0.0339^{+0.0003}_{-0.0006}}$	$\mathbf{*0.0272^{+0.0002}_{-0.0002}}$	$\mathbf{*0.5234^{+0.0089}_{-0.0182}}$
PIM	Validation	$0.1388^{+0.0048}_{-0.0032}$	$0.0619^{+0.0004}_{-0.0008}$	$0.4056^{+0.0031}_{-0.0041}$	$0.4411^{+0.0071}_{-0.0087}$	$0.3619^{+0.0030}_{-0.0041}$	$0.1667^{+0.0000}_{+0.0000}$
LIN	Validation	$0.0443^{+0.0012}_{-0.0009}$	$0.0280^{+0.0005}_{-0.0008}$	$0.1619^{+0.0024}_{-0.0018}$	$0.1581^{+0.0066}_{-0.0042}$	$0.1377^{+0.0033}_{-0.0021}$	$0.2380^{+0.0485}_{-0.0220}$
FFN	Validation	$0.0066^{+0.0003}_{-0.0004}$	$0.0117^{+0.0002}_{-0.0001}$	$0.0420^{+0.0016}_{-0.0017}$	$0.0458^{+0.0023}_{-0.0032}$	$0.0379^{+0.0012}_{-0.0021}$	$0.4558^{+0.0328}_{-0.0334}$
RES	Validation	$\mathbf{*0.0052^{+0.0003}_{-0.0004}}$	$\mathbf{*0.0111^{+0.0003}_{-0.0004}}$	$\mathbf{*0.0320^{+0.0019}_{-0.0012}}$	$\mathbf{*0.0382^{+0.0015}_{-0.0022}}$	$\mathbf{*0.0309^{+0.0014}_{-0.0010}}$	$\mathbf{*0.5381^{+0.0438}_{-0.0631}}$
PIM	Training, Timpe	$0.2080^{+0.0003}_{-0.0003}$	$0.1933^{+0.0014}_{-0.0009}$	$0.3598^{+0.0019}_{-0.0014}$	$0.2897^{+0.0017}_{-0.0017}$	–	$0.1667^{+0.0000}_{+0.0000}$
LIN	Training, Timpe	$0.0528^{+0.0005}_{-0.0004}$	$0.0493^{+0.0004}_{-0.0003}$	$0.2040^{+0.0013}_{-0.0014}$	$0.1691^{+0.0009}_{-0.0007}$	–	$0.3019^{+0.0063}_{-0.0037}$
FFN	Training, Timpe	$0.0124^{+0.0003}_{-0.0002}$	$0.0145^{+0.0002}_{-0.0002}$	$0.0757^{+0.0017}_{-0.0009}$	$0.0589^{+0.0002}_{-0.0003}$	–	$0.4172^{+0.0040}_{-0.0040}$
RES	Training, Timpe	$\mathbf{*0.0112^{+0.0004}_{-0.0004}}$	$\mathbf{*0.0138^{+0.0005}_{-0.0006}}$	$\mathbf{*0.0658^{+0.0010}_{-0.0008}}$	$\mathbf{*0.0548^{+0.0007}_{-0.0007}}$	–	$\mathbf{*0.4252^{+0.0063}_{-0.0059}}$
PIM	Validation, Timpe	$0.2080^{+0.0013}_{-0.0011}$	$0.1933^{+0.0035}_{-0.0058}$	$0.3598^{+0.0056}_{-0.0077}$	$0.2897^{+0.0066}_{-0.0070}$	–	$0.1667^{+0.0000}_{+0.0000}$
LIN	Validation, Timpe	$0.0529^{+0.0015}_{-0.0009}$	$0.0495^{+0.0014}_{-0.0005}$	$0.2044^{+0.0046}_{-0.0064}$	$0.1695^{+0.0043}_{-0.0046}$	–	$0.3010^{+0.0181}_{-0.0105}$
FFN	Validation, Timpe	$0.0137^{+0.0010}_{-0.0008}$	$0.0155^{+0.0009}_{-0.0006}$	$0.0834^{+0.0056}_{-0.0025}$	$\mathbf{0.0642^{+0.0024}_{-0.0019}}$	–	$0.4196^{+0.0163}_{-0.0116}$
RES	Validation, Timpe	$\mathbf{*0.0130^{+0.0013}_{-0.0011}}$	$\mathbf{0.0152^{+0.0005}_{-0.0003}}$	$\mathbf{0.0826^{+0.0054}_{-0.0035}}$	$0.0649^{+0.0028}_{-0.0017}$	–	$\mathbf{0.4265^{+0.0059}_{-0.0049}}$

Table B3. Performance of provided pre-trained models.

Method	Split	Mass	Material	Position	Velocity	Total	Accuracy
FFN	Development	0.0055	0.0111	0.0399	0.0442	0.0364	0.4535
RES	Development	0.0046	0.0108	0.0273	0.0341	0.0273	0.5035
FFN	o.o.d. test	0.0119	0.0120	0.0483	0.0432	0.0405	0.5593
RES	o.o.d. test	0.0104	0.0125	0.0375	0.0425	0.0357	0.5091

(iv) Collisions can happen at arbitrary orientations. We provide a rotation module to account for rotation-equivariance of our ML models. We recommend re-training ML models using data augmentation (i.e. randomly rotating the systems) in combination with the rotation module as a sanity check for guaranteeing rotation-equivariance. Note that all ML models in this paper were trained without using the rotation module, i.e. projectiles flying into the negative y-direction initially, where the main collision plane is the x–y plane. Our data pre-processing is also specifically adjusted for this fixed-orientation setup, i.e. different axes getting scaled differently in order to account for the variability in the respectiv e directions.

(v) Our ML models can be ran on CPUs if using GPUs is unfeasible or none are available. We estimate that CPU inference times $\tau_{\mathrm{i,ML}}$ are in the order of 1–10 s.

(vi) As discussed in Section 2.2.5, extracting reliable post-collision rotation states from our data remains an open issue, potentially restricting the full usability of our ML models in *N*-body simulations, specifically for full tracking of rotation states over several collisions. However, once post-collision rotation states can be extracted reliably from collision simulations, they can be easily incorporated into our multi-task regression objective.

This paper has been typeset from a TEX/LATEX file prepared by the author.