scientific reports

Check for updates

OPEN

Tracking an untracked space debris after an inelastic collision using physics informed neural network

Harsha M[1]✉, Gurpreet Singh[2], Vinod Kumar[3], Arun Balaji Buduru[1] & Sanat K. Biswas[1]

With the sustained rise in satellite deployment in Low Earth Orbits, the collision risk from untracked space debris is also increasing. Often small-sized space debris (below 10 cm) are hard to track using the existing state-of-the-art methods. However, knowing such space debris' trajectory is crucial to avoid future collisions. We present a Physics Informed Neural Network (PINN)—based approach for estimation of the trajectory of space debris after a collision event between active satellite and space debris. In this work, we have simulated 8565 inelastic collision events between active satellites and space debris. To obtain the states of the active satellite, we use the TLE data of 1647 Starlink and 66 LEMUR satellites obtained from space-track.org. The velocity of space debris is initialized using our proposed velocity sampling method, and the coefficient of restitution is sampled from our proposed Gaussian mixture-based probability density function. Using the velocities of the colliding objects before the collision, we calculate the post-collision velocities and record the observations. The state (position and velocity), coefficient of restitution, and mass estimation of un-tracked space debris after an inelastic collision event along with the tracked active satellite can be posed as an optimization problem by observing the deviation of the active satellite from the trajectory. We have applied the classical optimization method, the Lagrange multiplier approach, for solving the above optimization problem and observed that its state estimation is not satisfactory as the system is under-determined. Subsequently, we have designed Deep Neural network-based methods and Physics Informed Neural Network (PINN) based methods for solving the above optimization problem. We have compared the performance of the models using root mean square error (RMSE) and interquartile range of the predictions. It has been observed that the PINN-based methods provide a better estimation performance for position, velocity, mass and coefficient of restitution of the space debris compared to other methods.

Since the launch of Sputnik in 1957, approximately 13,000 satellites have been deployed in the Low Earth orbit[1]. With collisions, explosions and fragmentations, the number of space debris sized less than 1 cm is now estimated to be more than 1,000,000[1]. The US Space Surveillance Network has a catalogue of about 30,000 resident space objects only[1]. Since 97% of space debris is not tracked, they pose a greater threat to active satellites as well as future space missions.

With this large number of space debris, the risk of in-orbit collision has increased, and the number of collision events has seen a rise in the past decade. Collision in space can be destructive, for example, Iridium-Cosmos collision[2] of 2009, or non-destructive, for example, Canadarm2 of the International Space Station hit by a small piece of space debris in 2021[3]. Another incident which was catalogued as a non-destructive collision was in 2013 when Blits satellite was possibly hit by un-tracked space debris[4]. In non-destructive collision events, fragmentation does not happen. While there has been exhaustive research on modelling destructive collisions[5,6], on risk assessment due to small space debris[7], and reconnecting fragments in space to their parent satellite body[8], the extraction of trajectory information of untracked space debris observing a non-destructive collision has not been studied extensively.

Recently, Harsha et al.[9] formulated a space debris position, velocity and mass estimation problem considering the position and velocity deviation of an active satellite as observation with non-destructive and elastic collision assumption. The performance of the classical estimation methods as well as deep learning (DL) based approaches, were examined for this problem. It was observed that the position, velocity, and mass estimation performance

[1]Indraprastha Institute of Information Technology Delhi, New Delhi 110020, India. [2]U R Rao Satellite Centre, ISRO, Bengaluru 560071, India. [3]Indian National Space Promotion and Authorization Center, Ahmedabad 380058, India. ✉email: harsham@iiitd.ac.in

of the Ensemble Neural Network-based technique are similar to those of classical methods. It should be noted that the elastic collision assumption in the above preliminary study was idealistic. The above estimation problem becomes complex when a more rigorous inelastic collision model is considered where the momentum is conserved, but the kinetic energy is not conserved. Hence, it is of interest to examine the trajectory estimation performance of both classical and ML-based methods under the inelastic collision assumption.

Deep Neural Network (DNN)-based techniques have already been proposed for asteroid exploration, spacecraft rendezvous and terrain navigation[10]. The convolutional neural network (CNN) has been proposed for pose estimation of uncooperative spacecraft[11] and for object detection and tracking in space[12]. ESA's Hera mission is planning to carry out image processing of double asteroid system Dimorphos-Didymos using CNN[13] and demonstrate optical navigation. This mission will also investigate the impact of kinetic impactor released by NASA's Double asteroid redirection test (DART) mission using deep learning techniques[14].

Deep learning techniques have also been demonstrated to carry out initial screening of conjunction assessment among 170 million space-object pairs[15], which otherwise is a computation intensive process in the domain of Space Situational Awareness. Automatic collision avoidance maneuver have been proposed using ML algorithms[16] combining with Dempster-Shafer theory of evidence. Deep Neural network achieve satisfactory performance when trained with large amounts of data. However, these standard neural network-based methods inevitably face the challenge of drawing conclusions and making decisions under partial information, where the data acquisition is costly and scarce while dealing with complex and non-linear physical systems[17]. In addition, the solution of the traditional neural network does not guarantee adherence to the underlying physical properties in problems involving physical systems. To address these issues, the Physics informed Neural network (PINN) was proposed[17]. PINN[18,19] uses the laws governing the system dynamics in the loss function as prior information. Note that the training of a neural network essentially implies finding the set of weights and biases for all the nodes in the network, which minimises the loss function. Hence, the inclusion of the system dynamics based on physical laws in the loss function act as a constraint for the output while training the neural network using the training data set. The PINNs can be used as surrogates in learning models used for autonomous navigation, uncertainty quantification and any real-time application that need inference[20].

In this article, we have studied the problem of tracking unknown space debris, observing an inelastic and non-destructive collision event. Additionally, we assumed the space debris was in a stable orbit before the collision. Under the inelastic collision assumption, the coefficient of restitution[21] is an unknown parameter in addition to the variables that need to be estimated under the elastic collision assumption. We have examined the performance of the classical estimation technique and various DNN and PINN-based methods for estimating the position, velocity, and mass of the space debris and the coefficient of restitution for the inelastic collision for the above-mentioned problem using 8235 collision simulations for training and 330 collision simulations for testing. The inelastic collisions were simulated using the model described by Schwager et al.[22] The performance of the DNN and PINN-based methods are compared with the classical method for the inelastic and non-destructive collision. Towards tracking unknown space debris, we have made the following key contributions:

1. Formulation of space debris tracking problem observing a non-destructive and inelastic collision: We have posed the unknown space debris tracking problem after an inelastic event as a position, velocity, mass, and the coefficient of restitution estimation problem considering the active satellite position and velocity deviation as observations.
2. Formulation of an appropriate physics loss function and application of PINN: We have formulated a physics loss function for the above problem and trained a PINN using the developed loss function to solve the above estimation problem.
3. A velocity sampling method for simulating random in-orbit collision events: For the training of DNN and PINN-based models as well for testing the performance of the ML-based approaches and the classical approach, we simulated a total of 8565 inelastic collision events. Note that, for a collision event, the active satellite position and the space debris position will be approximately the same at the time of the collision, while the velocities will be different. However, any random velocity vector at the given position does not necessarily result in a stable elliptical orbit. We have proposed a velocity sampling method to generate the velocity vector of space debris for a given active satellite position at the time of the collision, which guarantees an elliptical orbit with a periapsis above a given minimum allowable altitude from the mean sea level.

The rest of the article is organised as follows: We have formally defined the problem in the next section, and then discussed the classical estimation method and various DNN-based approaches and provided the physics loss function for the PINN for solving the problem. Then we have discussed the collision data generation method, including the formulation of the velocity sampling algorithm and simulation of collision in space. Next, we have compared the performance of the classical estimation technique as well as the gradient boosting, DNN and PINN-based methods for solving the estimation problem. We conclude the article by consolidating our observations and delineating the future research direction.

Problem definition

Consider a satellite and space debris undergoing an inelastic collision in space as depicted in Fig. 1. Our objective is to find the trajectory of the space debris by observing the position and velocity deviation of the active satellite after the collision.

Let $\boldsymbol{r}_{sat}$ and $\boldsymbol{v}_{sat}$ be the position and velocity of the satellite, and $\boldsymbol{r}_d$ and $\boldsymbol{v}_d$ be the position and velocity of the space debris. We assume satellite and space debris are spherical for ease of the analysis. Let t_c^- and t_c^+ denote the

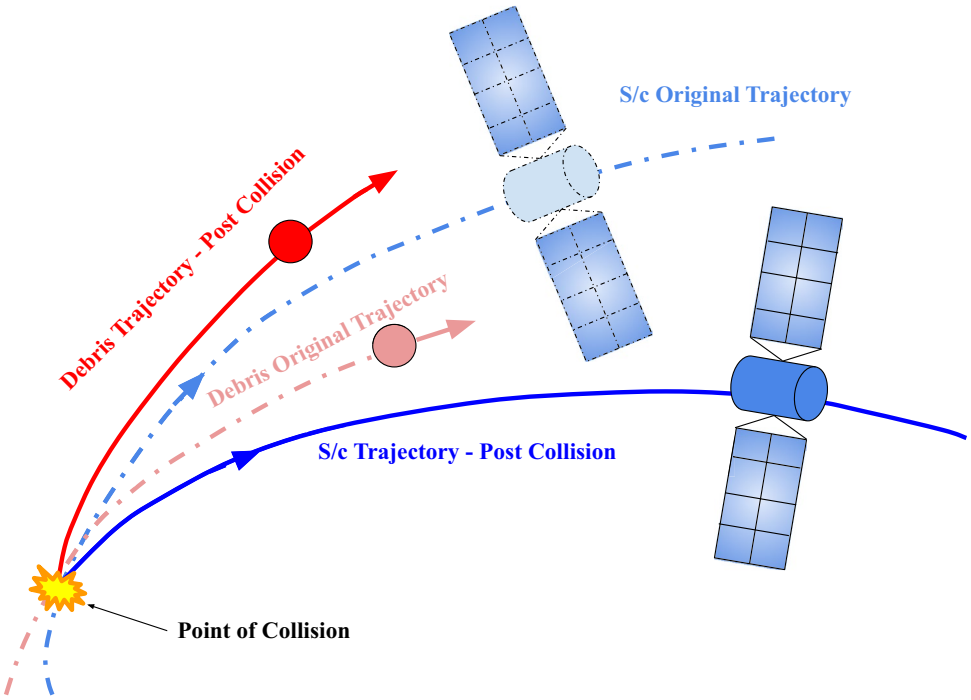


Figure 1. Illustration of a collision in space.

time before collision and time after collision respectively. At the time of the collision, the position of the satellite and the position of debris can be written as

$$\boldsymbol{r}_{sat}(t_c^+) = \boldsymbol{r}_{sat}(t_c^-) \tag{1}$$

$$\boldsymbol{r}_d(t_c^+) = \boldsymbol{r}_d(t_c^-) = \boldsymbol{r}_{sat}(t_c^-) - (\rho_{sat} + \rho_d)\hat{\boldsymbol{v}}_{rel} \tag{2}$$

where ρ_{sat} and ρ_d are radii of the satellite and debris, respectively and $\hat{\boldsymbol{v}}_{rel}$ is the unit vector along the relative velocity of debris with respect to the satellite and is given by

$$\hat{\boldsymbol{v}}_{rel} = \frac{\boldsymbol{v}_{sat}(t_c^-) - \boldsymbol{v}_d(t_c^-)}{||\boldsymbol{v}_{sat}(t_c^-) - \boldsymbol{v}_d(t_c^-)||} \tag{3}$$

The velocity of the satellite and debris after an inelastic collision is[21]

$$\boldsymbol{v}_{sat}(t_c^+) = \frac{m_{sat}\boldsymbol{v}_{sat}(t_c^-) + m_d\boldsymbol{v}_d(t_c^-) + \epsilon m_d(\boldsymbol{v}_d(t_c^-) - \boldsymbol{v}_{sat}(t_c^-))}{m_{sat} + m_d} \tag{4}$$

$$\boldsymbol{v}_d(t_c^+) = \frac{m_d\boldsymbol{v}_d(t_c^-) + m_{sat}\boldsymbol{v}_{sat}(t_c^-) + \epsilon m_{sat}(\boldsymbol{v}_{sat}(t_c^-) - \boldsymbol{v}_d(t_c^-))}{m_{sat} + m_d} \tag{5}$$

where $\boldsymbol{v}_{sat}(t_c^-)$ indicates velocity of satellite just before the collision and $\boldsymbol{v}_{sat}(t_c^+)$ indicates velocity of satellite just after the collision and ϵ is the coefficient of restitution[22] of the inelastic collision. Then change in velocity of the active satellite can be written as

$$\Delta\boldsymbol{v}_{sat}(t_c^+) = \boldsymbol{v}_{sat}(t_c^+) - \boldsymbol{v}_{sat}(t_c^-) \tag{6}$$

Define the measurement $\boldsymbol{Z}$ as

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{r}_{sat}(t_c^+) \\ \Delta\boldsymbol{v}_{sat}(t_c^+) \end{bmatrix} + \omega(t_c^+) = \begin{bmatrix} \boldsymbol{r}_{sat}(t_c^+) \\ \boldsymbol{v}_{sat}(t_c^+) - \boldsymbol{v}_{sat}(t_c^-) \end{bmatrix} + \omega(t_c^+) \tag{7}$$

where $\omega(t_c^+)$ is noise in measurement. As the active satellite is being tracked, $\boldsymbol{v}_{sat}(t_c^+)$ and $\boldsymbol{v}_{sat}(t_c^-)$ can be measured, and therefore measurement $\boldsymbol{Z}$ is available. Using Eqs. (2), (4) and (5), $\boldsymbol{Z}$ can be written as

$$\boldsymbol{Z} = f(\boldsymbol{r}_d(t_c^-), \boldsymbol{v}_d(t_c^-), m_d, \epsilon) + \omega(t_c^+) \tag{8}$$

Obtaining $\boldsymbol{r}_d(t_c^-)$, $\boldsymbol{v}_d(t_c^-)$, m_d and ϵ from $\boldsymbol{Z}$ is essentially an estimation problem. Using the estimated parameters, $\boldsymbol{v}_d(t_c^+)$ can be obtained and thus the trajectory of the space debris after the collision can be computed.

Classical approach

One can estimate $\boldsymbol{r}_d(t_c^-), \boldsymbol{v}_d(t_c^-), m_d$ and ϵ from observation $\boldsymbol{Z}$ using the Least Squared Estimation (LSE):

$$\widehat{\boldsymbol{r}}_d(t_c^-), \widehat{\boldsymbol{v}}_d(t_c^-), \widehat{m}_d, \widehat{\epsilon} = \underset{\boldsymbol{r}_d(t_c^-), \boldsymbol{v}_d(t_c^-), m_d, \epsilon}{\arg\min} (\boldsymbol{\Delta Z}^T \boldsymbol{\Delta Z}) \tag{9}$$

where $\widehat{\boldsymbol{r}}_d(t_c^-), \widehat{\boldsymbol{v}}_d(t_c^-), \widehat{m}_d, \widehat{\epsilon}$ are the estimate of the desired quantities and

$$\boldsymbol{\Delta Z} = \boldsymbol{Z} - f(\widehat{\boldsymbol{r}}_d(t_c^-), \widehat{\boldsymbol{v}}_d(t_c^-), \widehat{m}_d, \widehat{\epsilon}) \tag{10}$$

Note that the above estimation problem formulation does not consider any mass constraints for the debris. However, we are particularly interested in the untracked debris, which is of small size and essentially has a very small mass. The inclusion of the mass constraint transforms the original problem into a constrained optimization problem, which can be solved using the Lagrange Multiplier approach[9]. The Lagrangian L for this constrained optimization problem can be defined as

$$L = \boldsymbol{\Delta Z}^T \boldsymbol{\Delta Z} + \lambda_1(1 - m_d) + \lambda_2(m_d - \delta m) \tag{11}$$

which includes the mass constraint $\delta m < m_d < 1\ kg$. Here, λ_1 and λ_2 are Lagrange multipliers. The position, velocity, mass and coefficient of restitution of the debris can be estimated by solving

$$\left[\frac{\partial L}{\partial \boldsymbol{r}_d} \quad \frac{\partial L}{\partial \boldsymbol{v}_d} \quad \frac{\partial L}{\partial m_d} \quad \frac{\partial L}{\partial \epsilon}\right] = \boldsymbol{0} \tag{12}$$

$$\left[\frac{\partial L}{\partial \lambda_1} \quad \frac{\partial L}{\partial \lambda_2}\right] = \boldsymbol{0} \tag{13}$$

There are various techniques available to solve this minimisation problem. We have used the gradient descent approach to solve the problem numerically.

Deep Learning-based approaches

An alternate approach for estimating the position, velocity, mass and coefficient of restitution of the debris after a collision can be based on a DNN model. Using (8) one can write

$$E[\boldsymbol{r}_d(t_c^-), \boldsymbol{v}_d(t_c^-), m_d, \epsilon] = f^{-1}(\boldsymbol{Z}) \tag{14}$$

considering $\boldsymbol{\omega}(t_c^+)$ as a zero mean noise vector. Essentially the trained DNN model approximates the inverse function $f^{-1}(\cdot)$. The DNN model can be trained using simulated collision data. We consider $\boldsymbol{Z}$ as the input to the DNN and $\boldsymbol{\Delta r}, \boldsymbol{\Delta v}, m_d, \epsilon$ as the output of the DNN, where

$$\boldsymbol{\Delta r} = \begin{bmatrix} \Delta x_d \\ \Delta y_d \\ \Delta z_d \end{bmatrix} = \boldsymbol{r}_d(t_c^-) - \boldsymbol{r}_{sat}(t_c^-) \tag{15}$$

and

$$\boldsymbol{\Delta v} = \begin{bmatrix} \Delta v_{xd} \\ \Delta v_{yd} \\ \Delta v_{zd} \end{bmatrix} = \boldsymbol{v}_d(t_c^-) - \boldsymbol{v}_{sat}(t_c^-) \tag{16}$$

We consider two distinct DNN architectures for this problem. The first architecture is the usual DNN with an input layer with 6 inputs, multiple hidden layers and an output layer with 8 outputs as shown in Fig. 2. The second architecture comprises of 8 parallel DNNs with 6 inputs, multiple hidden layers and 1 output. Essentially, each of these parallel DNNs learns one of the 8 outputs. The outputs of these 8 parallel DNNs are connected to another DNN with 8 inputs and 8 outputs. The last DNN block ensures learning the dependency of each output variable with other output variables. The architecture is shown in Fig. 3.

We refer to this architecture as the stacked Deep Neural Network (StackDNN). For both architectures, Mean Squared Error (MSE) loss function is used for training the models. We define the MSE loss for this problem as

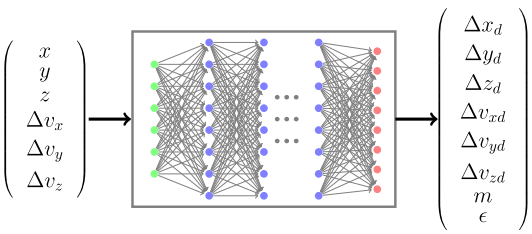


Figure 2. Deep Neural Network (DNN) architecture.

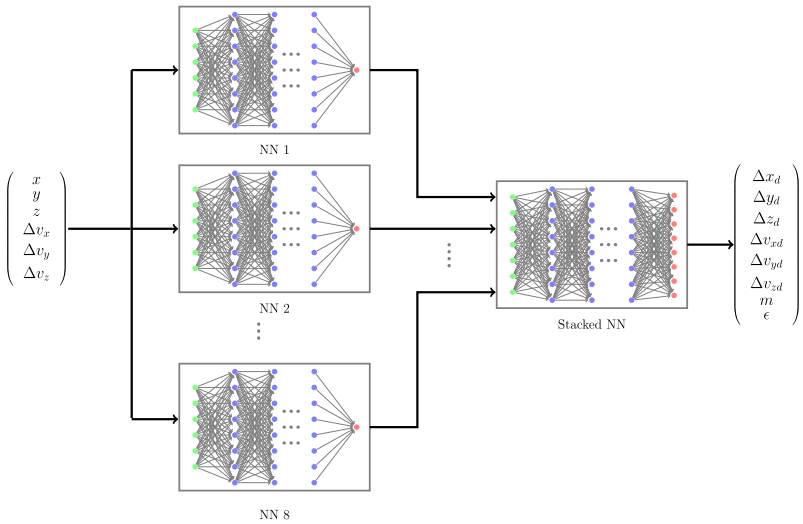


Figure 3. Stacked Deep Neural Network (StackDNN) architecture.

$$L_{MSE}(\widehat{\boldsymbol{r}}_d, \widehat{\boldsymbol{v}}_d, \hat{m}_d, \hat{\epsilon}) = \frac{1}{N}\sum_{i=1}^{N}\left[(\boldsymbol{r}_{d_i} - \widehat{\boldsymbol{r}}_{d_i})^T(\boldsymbol{r}_{d_i} - \widehat{\boldsymbol{r}}_{d_i}) + (\boldsymbol{v}_{d_i} - \widehat{\boldsymbol{v}}_{d_i})^T(\boldsymbol{v}_{d_i} - \widehat{\boldsymbol{v}}_{d_i}) + (m_{d_i} - \hat{m}_{d_i})^2 + (\epsilon_i - \hat{\epsilon}_i)^2\right] \tag{17}$$

where $(\cdot)_{d_i}$ and $\widehat{(\cdot)}_{d_i}$ denote the output data for the ith collision event and the corresponding predicted output data using the DNN. N is the number of collision events used to generate the training data.

Physics informed neural network

Having discussed the DNN-based method for estimating the position, velocity, mass and coefficient of restitution of debris after the collision, it is essential to note that the estimated position and velocity must be the solution to the differential equation that governs the motion of the debris. Based on Newton's Law of Gravitation, the debris dynamics can be written as

$$\begin{bmatrix}\dot{\boldsymbol{r}}_d \\ \dot{\boldsymbol{v}}_d\end{bmatrix} = \begin{bmatrix}\boldsymbol{v}_d \\ -\frac{\mu}{r_d^3}\boldsymbol{r}_d\end{bmatrix} \tag{18}$$

However, DNN-based solutions do not guarantee the above constraint. The Physics Informed Neural Network[18] preserves the laws of physics in the solution by incorporating a physics loss in the loss function used to train the DNN. Using (18), we defined the physics loss for the PINN as

$$L_{phy}(\widehat{\boldsymbol{r}}_d, \widehat{\boldsymbol{v}}_d) = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\frac{\mu}{r_{d_i}^3}\boldsymbol{r}_{d_i} - \frac{\mu}{\hat{r}_{d_i}^3}\widehat{\boldsymbol{r}}_{d_i}\right)^T\left(\frac{\mu}{r_{d_i}^3}\boldsymbol{r}_{d_i} - \frac{\mu}{\hat{r}_{d_i}^3}\widehat{\boldsymbol{r}}_{d_i}\right) + \left(\boldsymbol{v}_{d_i} - \widehat{\boldsymbol{v}}_{d_i}\right)^T\left(\boldsymbol{v}_{d_i} - \widehat{\boldsymbol{v}}_{d_i}\right)\right] \tag{19}$$

for the problem under consideration. The significance of this loss function is that the minimisation of $\mathcal{L}_{phy}(\widehat{\boldsymbol{r}}_d, \widehat{\boldsymbol{v}}_d)$ minimises the difference between the acceleration computed using the output of the training data and the PINN output, and the difference between the velocity computed using the output of the training data and the PINN output. As a result, this loss function minimises the distance between the derivative of the vector $[\boldsymbol{r}_d \;\; \boldsymbol{v}_d]^T$ computed using the training data, and the derivative of the same computed using the PINN predicted output. Since the velocity loss is already included in $L_{phy}(\widehat{\boldsymbol{r}}_d, \widehat{\boldsymbol{v}}_d)$ we define the MSE loss for the PINN as

$$L_{MSEP}(\widehat{\boldsymbol{r}}_d, \hat{m}_d, \hat{\epsilon}) = \frac{1}{N}\sum_{i=1}^{N}\left[(\boldsymbol{r}_{d_i} - \widehat{\boldsymbol{r}}_{d_i})^T(\boldsymbol{r}_{d_i} - \widehat{\boldsymbol{r}}_{d_i}) + (m_{d_i} - \hat{m}_{d_i})^2 + (\epsilon_i - \hat{\epsilon}_i)^2\right] \tag{20}$$

The complete loss function for the PINN is

$$L_{PINN} = L_{phy}(\widehat{\boldsymbol{r}}_d, \widehat{\boldsymbol{v}}_d) + L_{MSEP}(\widehat{\boldsymbol{r}}_d, \hat{m}_d, \hat{\epsilon}) \tag{21}$$

We use this loss function in both DNN and StackDNN for performance comparison.

Collision data generation

Having discussed the DNN, StackDNN, PINN, and StackPINN approaches for tracking the debris after an inelastic collision event, let us now turn to the preparation of the training data. Publicly available data on inelastic and non-destructive collision events are very few and not sufficient to train the neural networks. Hence simulated collision data are required for training the deep learning models.

The position and velocity information of active satellites can be obtained from publicly available catalogues. However, the position and velocity of the debris which collides with an active satellite at a given epoch must be simulated. The simulated debris must be in a complete orbit, i.e. the orbit eccentricity must be less than 1. Additionally, to form a stable orbit, the periapsis of the debris cannot be at a very low altitude.

The debris position is given by (2) at the time of the collision. From (2), we can approximate that the debris position as nearly equal to the active satellite position because $r_{sat} \gg \rho_{sat}, \rho_d$. Corresponding to this position, infinitely many velocity vectors are possible for which the debris will be on a complete orbit. However, the magnitude and the direction of the velocity vector can not be any arbitrary magnitude and direction. In the next subsection, we describe a method of sampling the velocity of debris for a given position vector in space and a given minimum periapsis altitude.

Velocity sampling

Consider r is the norm of the active satellite as well as the debris position vector $\boldsymbol{r}$ at the time of the collision, h is the norm of the angular momentum vector $\boldsymbol{h}$ of the colliding debris, e is the eccentricity of the debris orbit, ϕ is the true anomaly of the debris during the collision, R_e is the radius of the Earth and a_{min} minimum allowable periapsis altitude. If the semi-major axis for the debris orbit is a, then the periapsis $r_p = a(1-e)$. Also

$$\frac{h^2}{\mu} = r_p(1+e) \tag{22}$$

and

$$r = \frac{h^2}{\mu}\frac{1}{1+e\cos\phi} \tag{23}$$

Considering $R_{min} = R_e + a_{min}$ as the minimum allowable distance from the earth centre

$$r_p = \frac{r(1+e\cos\phi)}{1+e} > R_{min} \tag{24}$$

then

$$0 < e < \frac{r - R_{min}}{R_{min} - r\cos\phi} < 1 \tag{25}$$

and

$$\cos^{-1}\frac{2R_{min}-r}{r} < \phi < 2\pi - \cos^{-1}\frac{2R_{min}-r}{r} \tag{26}$$

Fig. 4 shows the variation of maximum possible eccentricity with the true anomaly.

Now, the position of the debris at the time of collision is

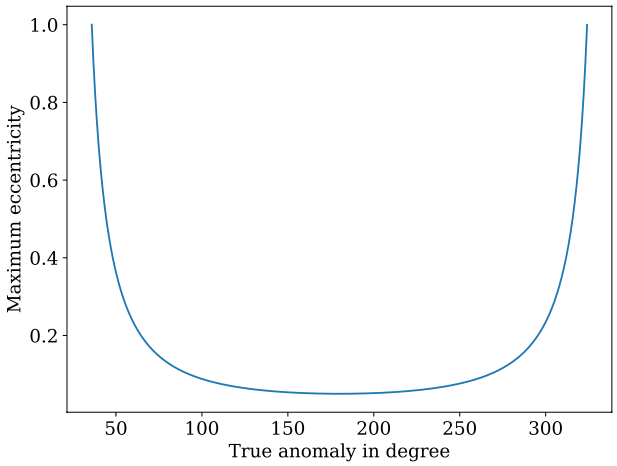


Figure 4. Maximum eccentricity vs. True anomaly.

$$\boldsymbol{r}_d = \boldsymbol{r} = r \begin{bmatrix} r_{ux} \\ r_{uy} \\ r_{uz} \end{bmatrix} \tag{27}$$

and velocity of the debris is

$$\boldsymbol{v}_d = v \begin{bmatrix} v_{ux} \\ v_{uy} \\ v_{uz} \end{bmatrix} \tag{28}$$

where $\begin{bmatrix} r_{ux} & r_{uy} & r_{uz} \end{bmatrix}^T$ and $\begin{bmatrix} v_{ux} & v_{uy} & v_{uz} \end{bmatrix}^T$ are the position and velocity unit vectors respectively, and

$$v = \sqrt{\frac{\mu}{r}\left(2 - \frac{1-e^2}{1+e\cos\phi}\right)} \tag{29}$$

The angle between $\boldsymbol{r}$ and $\boldsymbol{v}$ is θ[23]

$$\theta = \sin^{-1}\frac{1+e\cos\phi}{\sqrt{1+2e\cos\phi+e^2}} \tag{30}$$

The angular momentum vector of the debris is

$$\begin{aligned} \boldsymbol{h} &= \boldsymbol{r} \times \boldsymbol{v} \\ &= rv \begin{bmatrix} r_{uy}v_{uz} - r_{uz}v_{uy} \\ r_{uz}v_{ux} - r_{ux}v_{uz} \\ r_{ux}v_{uy} - r_{uy}v_{ux} \end{bmatrix} \end{aligned} \tag{31}$$

and

$$h = rv\sin\theta \tag{32}$$

then the cosine of the inclination i of the orbit

$$\cos i = \frac{rv(r_{ux}v_{uy} - r_{uy}v_{ux})}{rv\sin\theta} \tag{33}$$

Consequently, the velocity unit vector needs to satisfy the following:

$$r_{ux}v_{ux} + r_{uy}v_{uy} + r_{uz}v_{uz} = \cos\theta \tag{34}$$

$$-r_{uy}v_{ux} + r_{ux}v_{uy} + 0v_{uz} = \sin\theta\cos i \tag{35}$$

$$\sqrt{v_{ux}^2 + v_{uy}^2 + v_{uz}^2} = 1 \tag{36}$$

Geometrically, the condition for forming a complete orbit is - the line of intersection of planes (34) and (35) must lie within the unit sphere defined by (36). We can write (35) in the normal form:

$$-\frac{r_{uy}}{\sqrt{r_{ux}^2 + r_{uy}^2}}v_{ux} + \frac{r_{ux}}{\sqrt{r_{ux}^2 + r_{uy}^2}}v_{uy} + 0 \cdot v_{uz} = \frac{\sin\theta\cos i}{\sqrt{r_{ux}^2 + r_{uy}^2}} \tag{37}$$

Since $\sqrt{r_{ux}^2 + r_{uy}^2 + r_{uz}^2} = 1$, (34) is already in the normal form. Observing (34) and (37), one can deduce that the angle between the two planes is 90°. Then from the cross-sectional geometry shown in Fig. 5, the criteria for the intersection of the two planes within the unit sphere is

$$\cos^2\theta + \frac{\sin^2\theta\cos^2 i}{r_{ux}^2 + r_{uy}^2} \leq 1 \tag{38}$$

Then

$$-\sqrt{\frac{r_{ux}^2 + r_{uy}^2}{\sin^2\theta(1-\cos^2\theta)}} \leq \cos i \leq \sqrt{\frac{r_{ux}^2 + r_{uy}^2}{\sin^2\theta(1-\cos^2\theta)}} \tag{39}$$

Using the ranges of ϕ, e and $\cos i$ given by (26), (25) and (39) one can independently sample these quantities and subsequently solve for v_{ux}, v_{uy} and v_{uz} from (34), (35) and (36). Note that the solution will result in two unit vectors for velocity. Finally, the velocity magnitude can be calculated using (29). Algorithm 1 describes the velocity sampling steps.

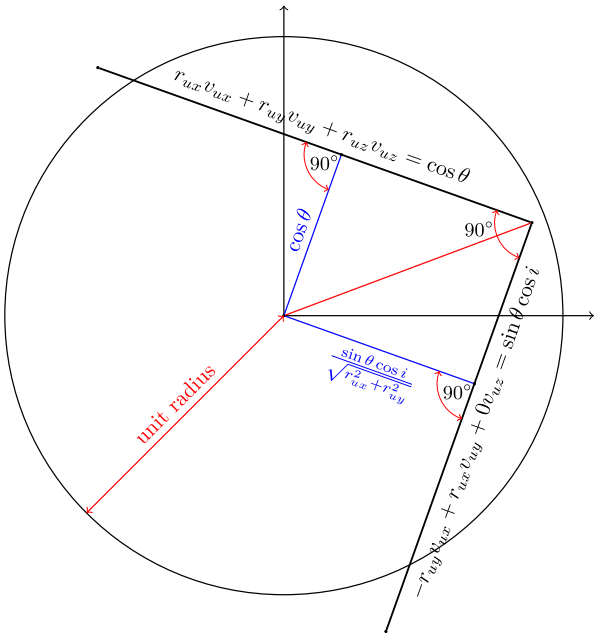


Figure 5. Cross-section of the intersection of the unit sphere and planes (34) and (35).

Input : $\boldsymbol{r}$, minimum altitude, Number of samples
Output : Velocity samples
for *i=1:Number of samples* **do**
  Given the allowed minimum altitude, find the minimum and maximum true anomaly ϕ_{min} and ϕ_{max} for the given position vector using (26);
  Sample $\phi \sim U(\phi_{min}, \phi_{max})$;
  For a sampled ϕ calculate maximum eccentricity e_{max} using (25);
  Sample $e \sim U(0, e_{max})$;
  Calculate θ;
  Compute the lower bound $\cos i_{min}$ and upper bound $\cos i_{max}$ for $\cos i$ using (39);
  Sample $\cos i \sim U(\cos i_{min}, \cos i_{max})$;
  Solve for v_{ux}, v_{uy}, v_{uz} using (34), (35) and (36);
  Calculate v using (29)
end

Algorithm 1. Velocity sampling

Using this algorithm, we have sampled 2000 velocity vectors for an active satellite position at

$$\boldsymbol{r} = \begin{bmatrix} 3664.75536654 & 3893.60049258 & 5062.87536598 \end{bmatrix}^T km \tag{40}$$

and a minimum perigee altitude of 300 km. Figure 6 shows that the generated samples have perigees at altitudes higher than 300 km. Fig. 7 shows the relative velocities for the samples with respect to the active satellite. It can be deduced from Fig. 7 that collision cannot occur from any arbitrary direction for a given position vector in space.

Simulation of the collision

We have generated inelastic collision observations considering the collision model described in the "Problem Definition" section. We have used Python 3.7 with astropy[24] and poliastro[25] libraries for orbit simulations of the active satellites as well as the debris.

We have considered 5 collision epochs t_c each at 1200 UTC, starting from April 29, 2023, to May 3, 2023. We consider 1647 satellite orbits from the Starlink constellation and 66 satellite orbits from the LEMUR constellation for generating collision simulations. The Two Line Element (TLE) data for each constellation were obtained from www.spacetrack.org on April 27, 2023. Using TLE data, we have calculated the time difference Δt between the collision time t_c and the the TLE time, and propagated the active satellites. The orbits are propagated considering J2 and J3 zonal harmonics and the exponential atmospheric drag model.

For the debris simulations, we sampled the mass of each untracked debris from a normal distribution with 0.5 kg mean and varying standard deviations of (0 kg, 0.001 kg, 0.002 kg, 0.003 kg, 0.004 kg, 0.005 kg, 0.006 kg).

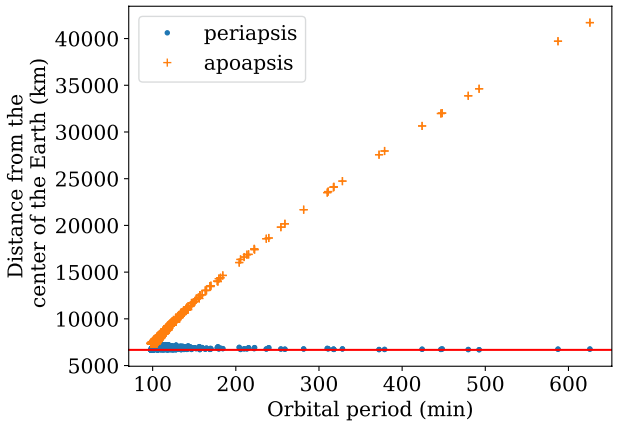


Figure 6. Gabbard chart for the generated velocity samples.

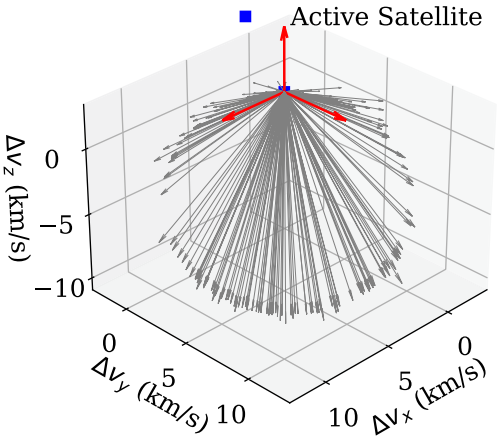
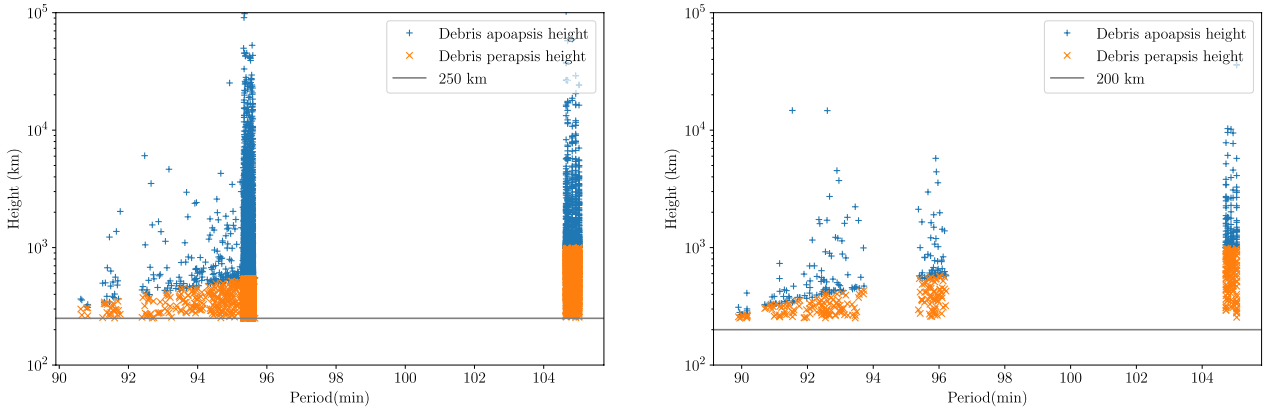


Figure 7. Relative velocities for the sampled space debris velocity with respect to the active satellite.

For simplicity, satellites and debris are considered spherical objects of radius 0.5 m and 0.1 m, respectively, and the mass of the satellite is 100 times that of sampled mass of debris. The position of the debris $\boldsymbol{r_d}$ is considered to be the same as $\boldsymbol{r_{sat}}$, whereas the velocity of the debris $\boldsymbol{v_d}$ is obtained using algorithm 1. The minimum perigee altitude for the debris corresponding to the Starlink satellites was set at 250 km, and for LEMUR, it was 200 km. The Gabbard plots for the debris generated for the Starlink and LEMUR constellations are shown in Fig. 8.



(**a**) Gabbard Plot of Space Debris for Starlink Satellites

(**b**) Gabbard Plot of Space Debris for LEMUR Satellites

Figure 8. Gabbard plot of space debris.

Coefficient of restitution (ϵ) values for each of the collisions have been sampled from a Gaussian mixture distribution, as shown in Fig. 9. It consists of 4 Gaussian distributions with 0.015 standard deviations sampled around uniformly distributed means ranging from 0.5 to 1 where the mean has been assumed based on the ϵ values of the satellite materials (6061-T6 Aluminium, Stainless Steel 304, Nitronic 60A and Titanium)[21].

Inelastic collision simulations are generated using the position, velocity, mass and coefficient of restitution values for the respective satellite and debris pairs in Eqs. (4) and (5). Post-collision at time t_c^+, the observations are generated using Eq. (7). Here, the noise $\omega(t_c^+)$ signifies the tracking uncertainty of the active satellite. We consider varying standard deviation (0m, 100 m, 200 m, 300 m, 400 m, 500 m) for position and (0 m/s, 50 m/s, 100 m/s, 150 m/s, 200 m/s, 250 m/s) for velocity in each axis.

A total of 8235 collision events were simulated for the orbits of the Starlink constellation using Param Pravega Super Computer. 66 collision events with 6 varying noise factors were simulated for the LEMUR constellation using a local workstation. For each collision, we have recorded $\boldsymbol{Z}$ and $\boldsymbol{r}_d, \boldsymbol{v}_d, m_d$ and ϵ. We consider the collision data corresponding to the orbits of the Starlink constellation as the training data and the same corresponding to the orbits of the LEMUR constellation as the test data.

Results and discussion

We used the training data described in the previous section for training the DNN, PINN, StackDNN, and StackPINN models. We trained four different models with 2, 4, 6, and 8 hidden layers for each of the ML models, with 5 nodes in each hidden layer. Rectified Linear Unit (ReLU) is selected as the activation function for each node. We have used adaptive moment (adam) optimizer with the learning rate equal to 1e−3 for all the models. All the models are trained for 50 epochs with a batch size equal to 32. We have also trained the Extreme Gradient Boosting (XGBoost)[26] regressor model with the same training data.

The training is performed in a workstation with 8 Intel Xeon processor cores and 62GB of RAM. The training of all the ML models is done using 5-fold cross-validation for different sizes of training data. All the ML models are trained with 8235 samples and 4000 samples (All results analysed during the current study are available in the **collisionAI** repository, https://github.com/HarshaSSL/collisionAI.). Figure 10 shows the change in training loss with respect to training epochs for the DNN and the PINN models trained using 8235 samples. Similarly, Fig. 11 shows the change in training loss with respect to training epochs for the StackDNN and the StackPINN models trained using 8235 samples. It is observed in most of the folds, for models with an increased number of hidden layers, the training epochs decreases for the training loss to converge. In StackDNN and StackPINN architectures (Fig. 3), the input to stacked NN is obtained from already trained parallel models. This reduces the initial loss value to the order of 10^1 when compared to DNN and PINN models' initial loss value which is of the order 10^7.The results for models trained using 4000 samples can be found in the supplementary information.

After analysing the performance of the trained models, we pick the best-performing model based on the lowest RMSE for the validation data during the 5-fold cross-validation. Inference is done using the selected model for the prediction of position, velocity, mass, and coefficient of restitution of inelastic collisions of the test dataset generated using LEMUR satellites.

Using the observation $\boldsymbol{Z}$ for all 66 test collision events with 6 different noise standard deviations as mentioned in section "Simulation of the collision" in the selected ML models, $\boldsymbol{r}_d(t_c^-), \boldsymbol{v}_d(t_c^-), m_d$ and ϵ were estimated. Additionally, the same variables were estimated using the Lagrange multiplier method for the same set of observations. We recorded the error in estimation for performance comparison.

Figure 12 shows the box plots of the position, velocity, mass and coefficient of restitution estimation errors recorded using selected DNN and PINN models. The position and velocity estimation errors for all the methods are represented by the Euclidian norm of the x, y, and z axes in the ECI frame. It can be observed that for position estimation error, the median and the interquartile range increase for the test cases with increasing noise standard deviation. A similar trend in the interquartile range can be found in the mass estimation error plots . In position, velocity and coefficient of restitution error estimation plots, it is observed that the performance of DNN and PINN models with 4, 6, and 8 hidden layers does not improve compared to 2 layers. However, in mass

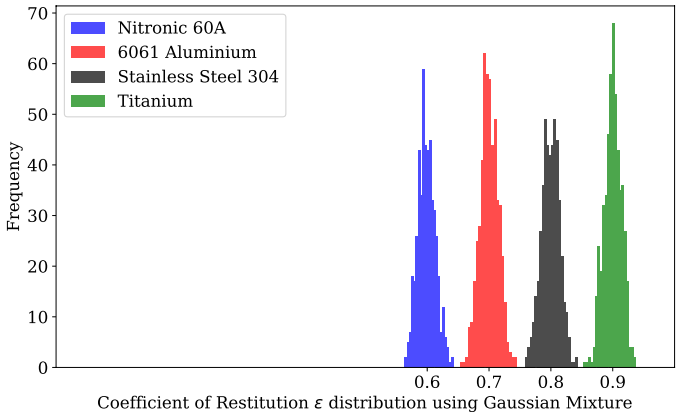


Figure 9. Coefficient of restitution (ϵ) distribution using Gaussian mixture.

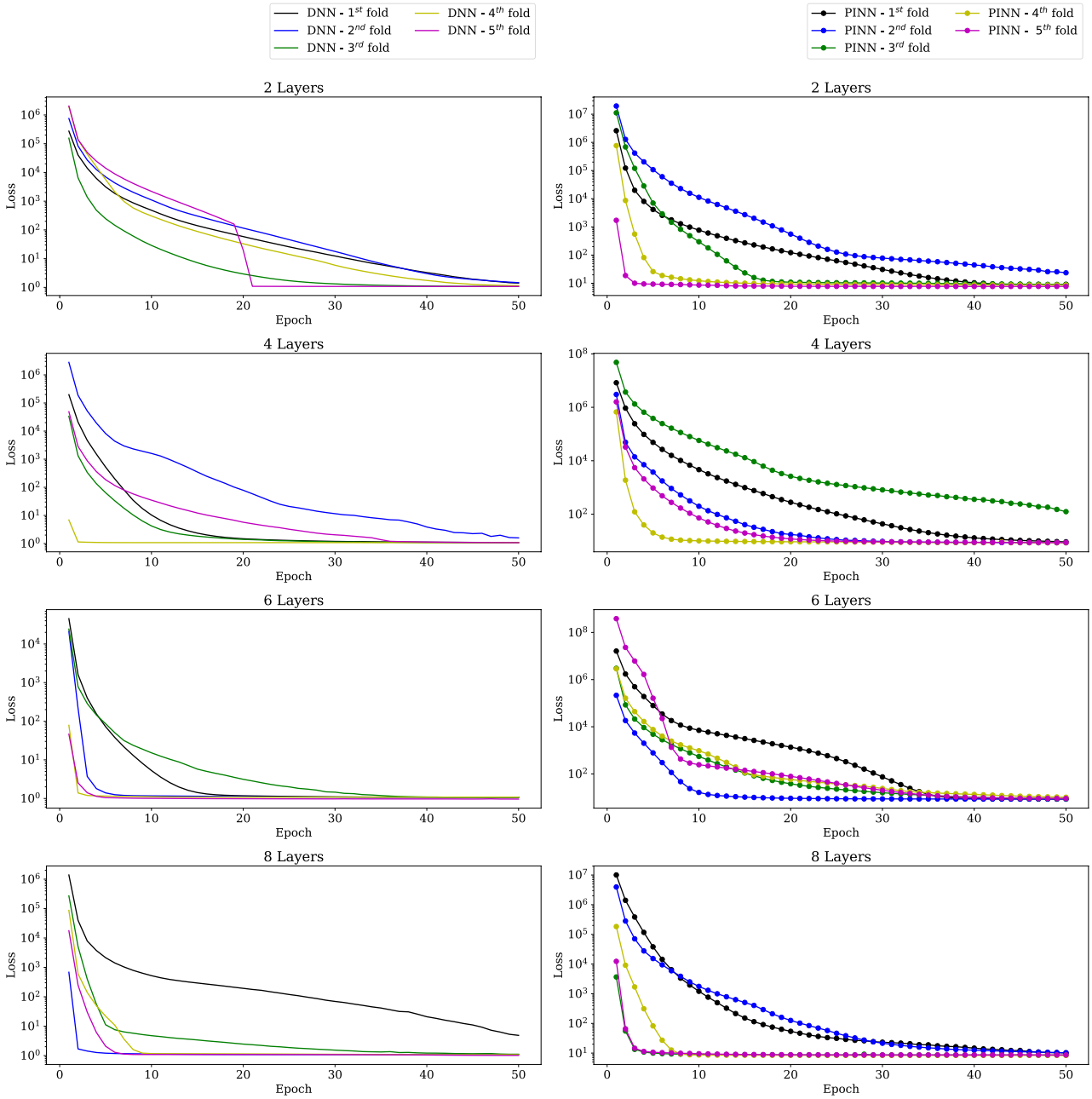


Figure 10. Training loss for DNN and PINN models in 5-Fold Cross-validation using 8235 samples and 50 epochs.

estimation error plots, DNN and PINN with 8 layers provide marginally better results as these models with more hidden layers are better function approximators[27].

Figure 13 shows the box plots of the position, velocity, mass, and coefficient of restitution estimation errors recorded using selected StackDNN and StackPINN models. The results follow a similar trend as the DNN and PINN models. However, StackDNN and StackPINN provides marginally better mass estimation using 2 layers. Figure 14 shows the box plots of estimation errors recorded using Lagrange and XGBoost models. The results of the Lagrange method and XGBoost models follow a similar trend to that of DNN and PINN models for different noise standard deviations. However, the median and interquartile range for the estimation errors of XGBoost models is the least among all the models. The median and interquartile range of the estimation errors of the Lagrange method are the highest among all the methods.We have found through experiments that the output is sensitive to position noise standard deviation compared to the standard deviation for velocity and mass noises[1].

The summary of 5-Fold cross-validation of the models with 2 hidden layers is shown in Table 1. The table shows the mean ± standard deviation of the Root Mean Squared Error (RMSE) for the validation data of the 5-fold cross-validation analysis. We have observed that the mean RMSE across all the training is higher when

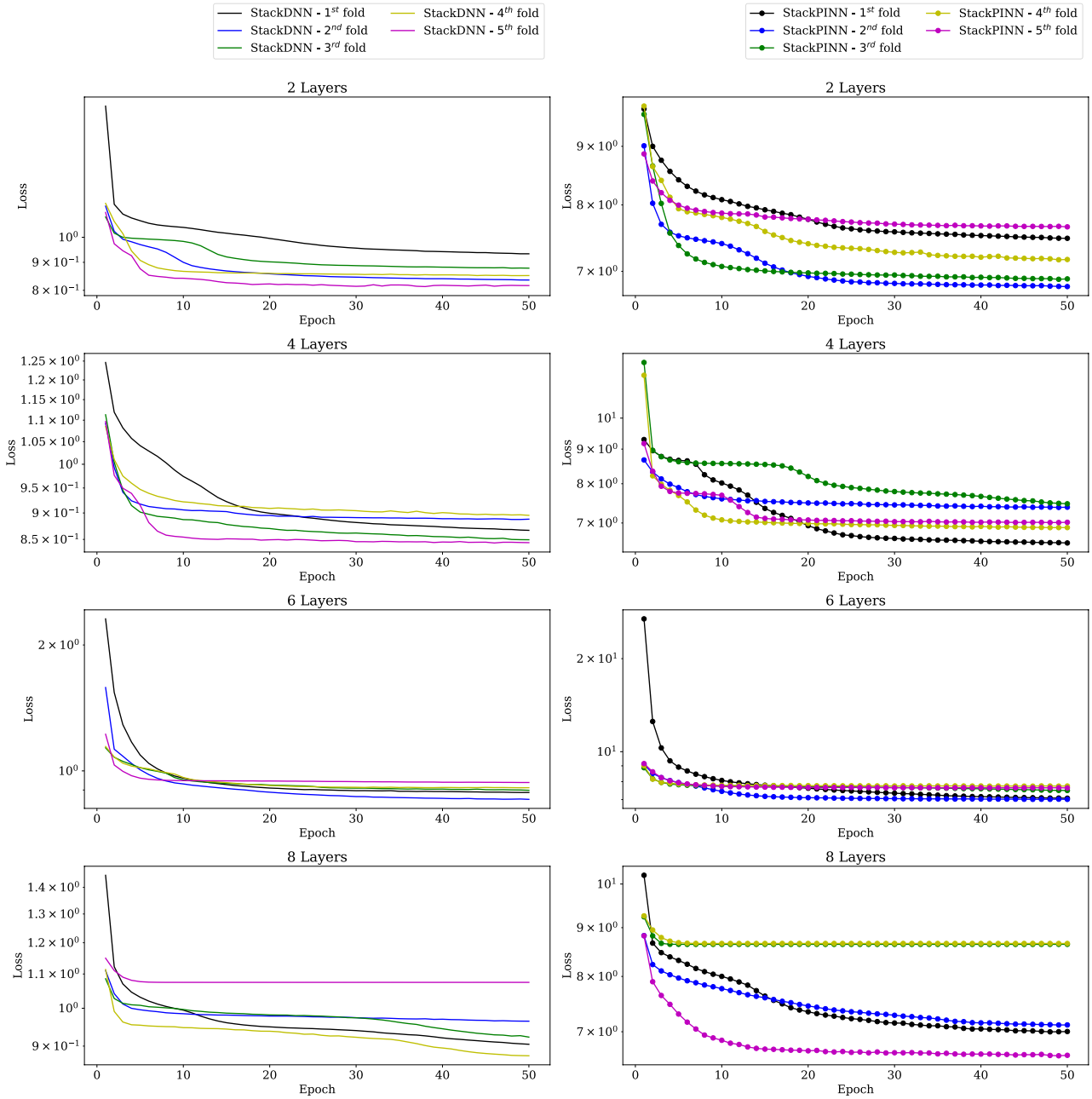


Figure 11. Training loss for StackDNN and StackPINN models in 5-Fold Cross-validation using 8235 samples and 50 epochs.

trained with smaller datasets. It is also observed that the StackPINN produces lesser variation in the RMSE when the smaller dataset is used.

From the above discussion, it can be discerned that, for 2 hidden layers, the PINN improves the space debris position, mass, and coefficient of restitution estimation performance significantly compared to the Lagrange multiplier-based and DNN-based methods. The performance of the DNN becomes similar to the PINN for higher numbers of hidden layers. However, the results show no significant advantage to increasing the number of layers for the problem under consideration. In addition, the StackDNN and the StackPINN performances are similar for the different numbers of hidden layers and comparable to the PINN model's performance with 2 hidden layers. However, the stacked architecture is 9 different Neural Networks, increasing the complexity of the implementation.

The training time required for different folds during 5-fold cross-validation, inference time for the selected models, and stored memory for reusability for each selected model with 2 hidden layers is shown in Table 2. XGBoost model requires 4000 KB of memory, which is 100 times more than the memory required to store PINN models.

Tables 3 and 4 summarize the inference of the selected trained models along with the Lagrange method for the LEMUR collision dataset. From these tables, we can conclude that the PINN with 2 hidden layers provides

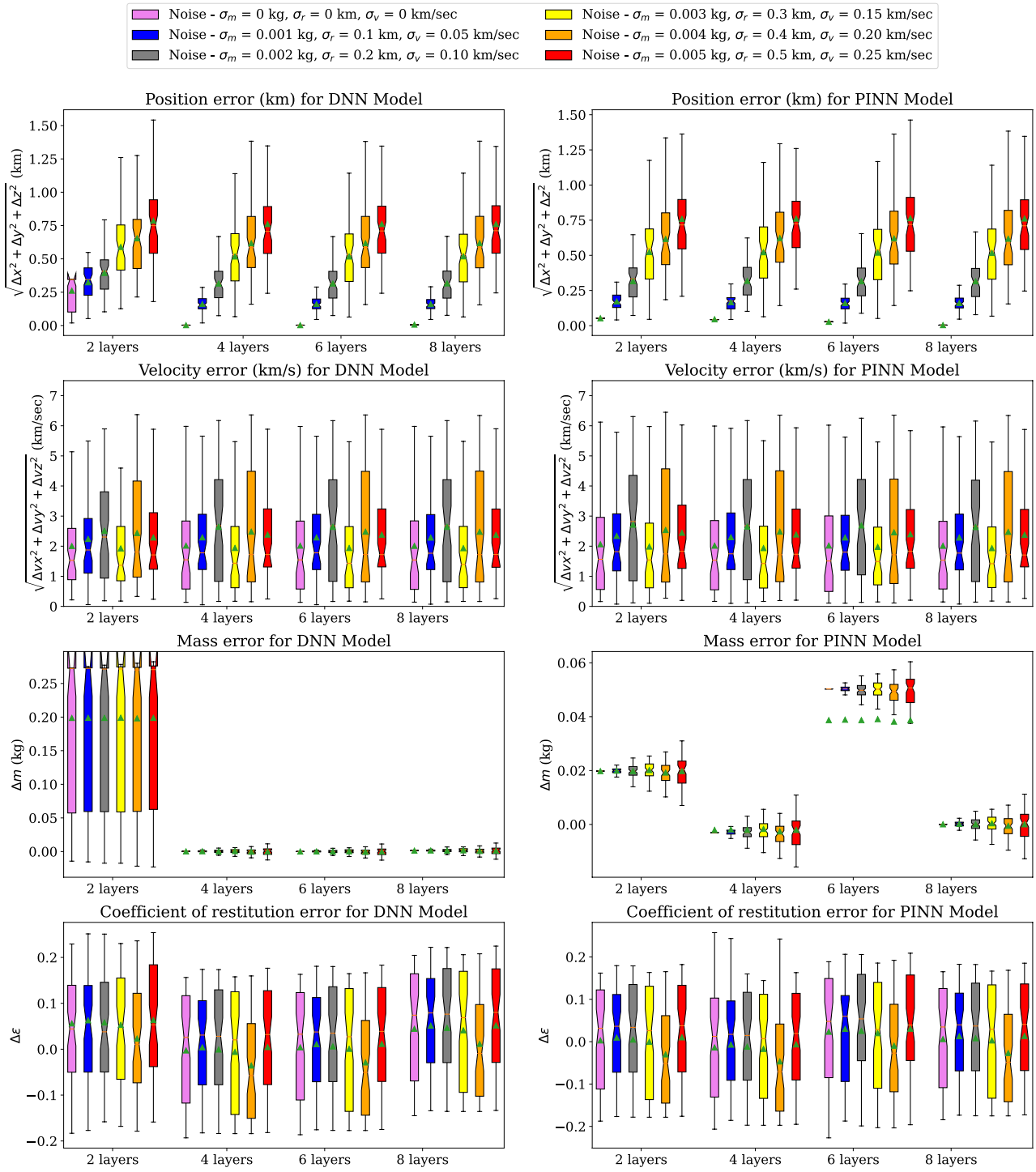


Figure 12. Error comparison of DNN and PINN models trained using 8235 samples and 50 epochs for multiple noise variations in LEMUR Satellites collision data.

better performance to resource requirement trade-off for tracking untracked space debris from a non-destructive and inelastic collision with an active satellite than the DNN, Stacked neural networks, and classical methods. We have found that the XGBoost method provides better results than the DNN and the PINN. However, this method requires approximately 100 times more memory than the DNN and PINN-based techniques. As a result, the PINN-based model would be more suitable for on-board deployment on the active spacecraft. Additionally, the XGBoost is a black box model, whereas the PINN is intrinsically explainable due to the physics-based loss function.

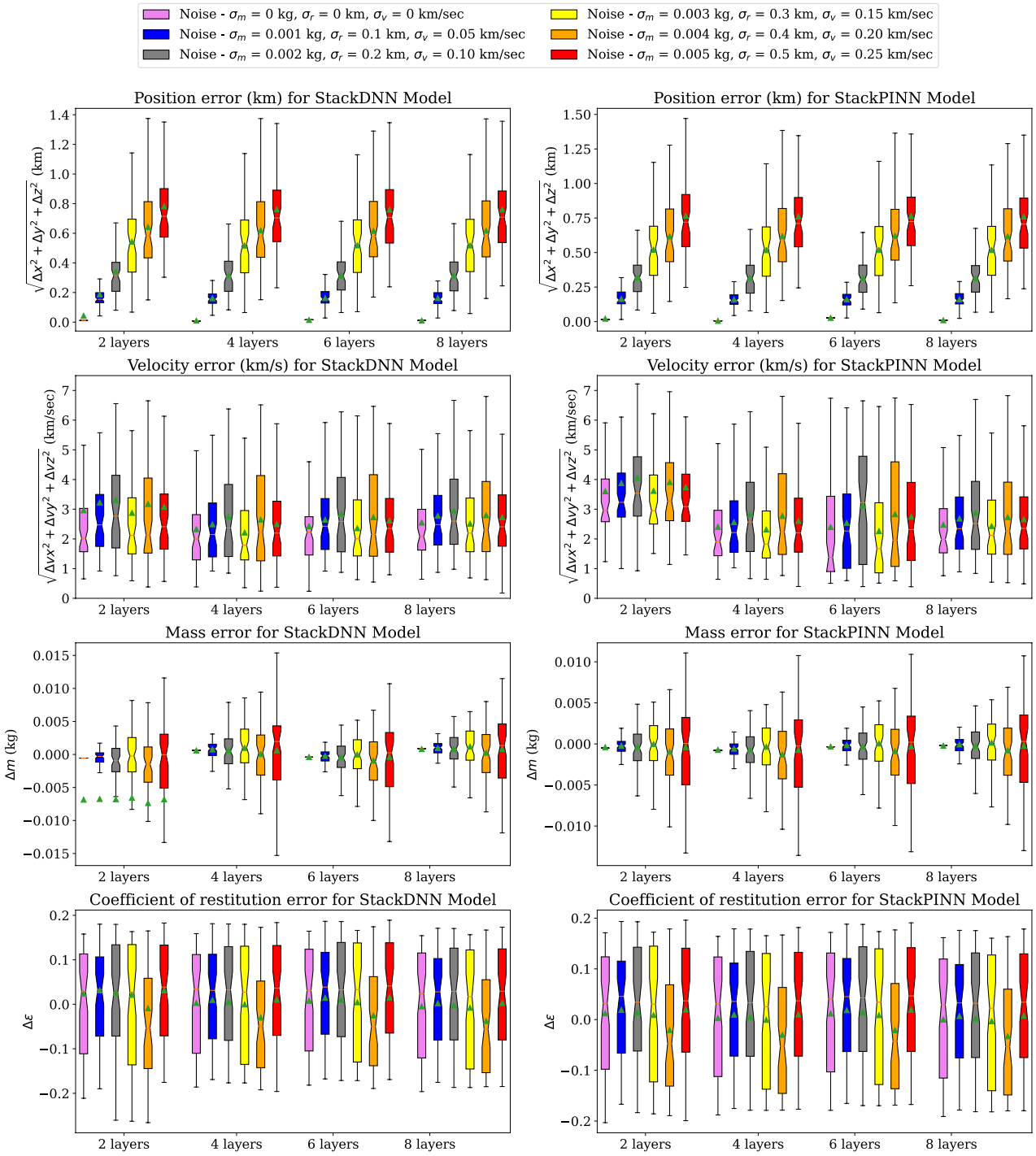


Figure 13. Error comparison of StackDNN and StackPINN models trained using 8235 samples and 50 epochs for multiple noise variations in LEMUR satellites inelastic collision data.

Conclusion

We have formulated a problem of space debris position, velocity, mass, and coefficient of restitution estimation from a non-destructive inelastic collision event under the assumption that the occurrence of the collision event is known. We have shown that this problem can be posed as a function approximation problem; hence, an ML model can approximate the function. We proposed a loss function based on Newton's law of gravitation to design a PINN to solve this problem. We have also presented an algorithm for selecting the velocity of space debris for a collision simulation. Using simulated collision data, we have trained the DNN, PINN, StackDNN, and StackPINN with varied numbers of hidden layers and the XGBoost model. We have also compared the estimation performance of these ML-based methods with the Lagrange multiplier-based optimization technique.

The results demonstrate that the PINN with 2 hidden layers performs better estimation than the DNN with 2 hidden layers and the classical method. Additionally, the DNN performance becomes similar to the PINN

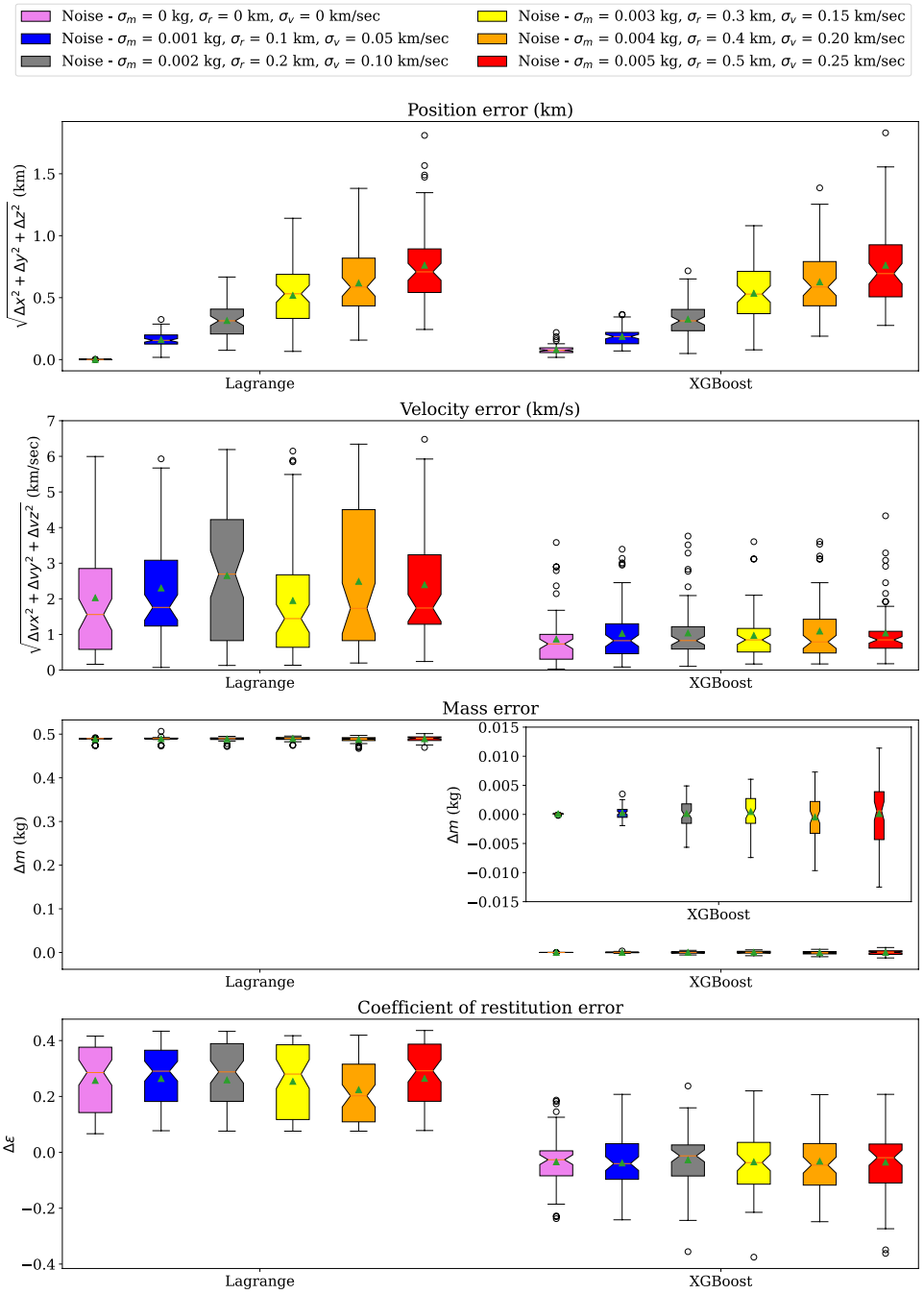


Figure 14. Error comparison of Lagrange method and XGBoost models trained using 8235 samples for multiple noise variations in LEMUR Satellites inelastic collision data.

when the number of hidden layers increases and when Stacked neural network architecture is used. However, this increase in the model complexity does not improve the estimation performance compared to the PINN with 2 hidden layers. It is also observed that the XGBoost models provide better estimation performance than the neural network-based counterparts. It is noteworthy to mention that the XGBoost model is a black-box model, and as a consequence, the output of the model is not explainable. On the other hand, the PINN, being trained using a loss function derived from the laws of physics, provides explainability of the outputs. In addition, the XGBoost model requires significantly higher memory than the proposed PINN. However, it is anticipated that model-agnostic explanation techniques, for example, SHapley Additive exPlanations (SHAP)[28] can be utilised to explain the XGBoost model in this context and can be deployed on the ground or onboard spacecraft leveraging the state-of-the-art embedded systems. A physics-informed approach for the explainability of the gradient boosting technique[29] can also be explored for tracking space debris by observing a collision.

Number of train samples	Model	Δr (km)	Δv (km/s)	Δm (kg)	$\Delta \epsilon$
4000	DNN	5.2704 ± 7.0132	6.821 ± 6.0505	1.544 ± 1.8549	2.1735 ± 1.9109
	PINN	9.2218 ± 11.667	19.410 ± 21.356	3.8594 ± 4.8815	3.7805 ± 2.0651
	StackDNN	5.2704 ± 7.0132	6.821 ± 6.0505	1.5440 ± 1.8549	2.1735 ± 1.9109
	StackPINN	0.3264 ± 0.3149	3.0294 ± 0.1066	0.0909 ± 0.1328	0.1467 ± 0.0386
	XGBoost	0.1832 ± 0.0020	1.0837 ± 0.0325	0.001 ± 9.47e-06	0.0917±0.0013
8235	DNN	1.2813 ± 1.2237	3.3195 ± 0.6529	0.421 ± 0.4746	0.8052 ± 0.6608
	PINN	0.9443 ± 0.7666	3.0712 ± 0.2684	0.3405 ± 0.2371	0.4976 ± 0.4599
	StackDNN	1.2813 ± 1.2237	3.3195 ± 0.6529	0.421 ± 0.4746	0.8052 ± 0.6608
	StackPINN	0.1752 ± 0.0098	4.3062 ± 1.8048	0.0252 ± 0.0538	0.1177 ± 0.0062
	XGBoost	0.1771 ± 0.0010	0.9979 ± 0.0187	0.001 ± 1.863e-05	0.0854 ± 0.0010

Table 1. RMSE tabular summary for 5-fold cross validation of training the models with different number of samples.

Number of samples	Model	Time required (s) for training the models					Inference time (s)	Memory required
		1*st*Fold	2*nd*Fold	3*rd*Fold	4*th*Fold	5*th*Fold		
4000	DNN	11.215	10.994	13.418	21.693	12.223	0.26417	40 KB
	PINN	12.970	22.370	17.891	18.063	16.883	0.33131	40 KB
	StackDNN	118.229	97.498	98.634	101.169	100.175	0.26417	360 KB
	StackPINN	100.464	100.062	101.656	101.386	102.123	0.33131	360 KB
	XGBoost	11.250	11.791	41.222	31.340	61.009	0.27845	3900 KB
8235	DNN	22.230	21.886	22.479	22.337	21.955	0.26594	40 KB
	PINN	25.407	25.273	25.082	25.214	25.093	0.32936	40 KB
	StackDNN	200.484	200.760	201.941	201.911	200.889	0.26594	360 KB
	StackPINN	204.296	203.725	203.401	204.594	205.888	0.32936	360 KB
	XGBoost	7.963	8.183	8.119	8.323	8.295	0.28983	4232 KB

Table 2. Tabular results of models consisting of time taken for training during 5-Fold cross validation, inference time for predicting and evaluating the LEMUR collision data (Nominal Noise) and their respective stored memory for reusability.

	Model	Δr (km)	Δv (km/s)	Δm (kg)	$\Delta \epsilon$
4000	DNN	0.674506	2.027031	0.394279	1.076511
	PINN	0.473144	2.010821	0.178281	0.767108
	StackDNN	0.152939	1.764392	-0.017718	0.034151
	StackPINN	0.158544	2.013816	0.002398	0.030450
	XGBoost	0.189282	0.617731	0.000088	-0.035985
8235	DNN	0.344972	1.873850	0.272428	0.059095
	PINN	0.165065	1.816507	0.019991	0.036363
	StackDNN	0.154050	2.475177	-0.000288	0.033286
	StackPINN	0.156674	3.232480	-0.000230	0.045742
	XGBoost	0.185646	0.825226	0.000249	-0.040400
	Lagrange	0.155496	1.558974	0.490096	0.290463

Table 3. Median error for the test dataset.

The problem presented in the article can be further extended to the estimation of space debris position, velocity, mass, and coefficient of restitution using a sequence of tracking data of the active satellite. The Recurrent Neural Network (RNN) or transformer may be suitable for this case. However, it will require the generation of a large number of sequences of tracking data for each collision to train such models. RNNs, Transformers, and other such models tend to have significantly more inference times, model sizes, and complexity. Further research is necessary to study the performance of these models in the problem under consideration. It is anticipated that the application of the PINN can be further extended to trajectory extraction of the debris after a break-up event. In addition, the velocity sampling formulation can be used to derive a velocity probability density function for a

	Model	Δr (km)	Δv (km/s)	Δm (kg)	$\Delta\epsilon$
4000	DNN	0.186429	2.010423	0.002568	0.231941
	PINN	0.163924	2.238280	0.001413	0.201985
	StackDNN	0.073994	2.002598	0.001530	0.185718
	StackPINN	0.069802	2.313196	0.002019	0.184570
	XGBoost	0.126994	0.562895	0.001296	0.103200
8235	DNN	0.205209	1.814919	0.213681	0.188687
	PINN	0.088477	1.894849	0.001394	0.183433
	StackDNN	0.067543	1.750979	0.001412	0.178207
	StackPINN	0.085123	1.482897	0.001205	0.181471
	XGBoost	0.091459	0.840224	0.001372	0.127520
	Lagrange	0.073416	2.193088	0.001430	0.183433

Table 4. Interquartile range of the errors for the test dataset.

given position in space, which can be used to design new filters for selecting candidate space debris for collision assessment as well as enhancing the collision probability computation.

Data availability

The datasets generated and analysed during the current study are available in the **collisionAI** repository, https://github.com/HarshaSSL/collisionAI.

Received: 24 July 2023; Accepted: 9 January 2024
Published online: 09 February 2024

References

1. Montaruli, M. F. *et al.* Adaptive track estimation on a radar array system for space surveillance. *Acta Astronaut.* **198**, 111–123. https://doi.org/10.1016/j.actaastro.2022.05.051 (2022).
2. Tan, A., Zhang, T. X. & Dokhanian, M. Analysis of the iridium 33 and cosmos 2251 collision using velocity perturbations of the fragments. *Adv. Aerospace Sci. Appl.* **3**, 145 (2013).
3. Datta, A. Op-ed|Damage to Canadarm2 on ISS once again highlights space debris problem (2021).
4. Kelso, T. S. *et al.* What Happened to BLITS? An Analysis of the 2013 Jan 22 Event. In *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference* 4 (2013).
5. Braun, V., Funke, Q., Lemmens, S. & Sanvido, S. Drama 3.0-upgrade of esas debris risk assessment and mitigation analysis tool suite. *J. Space Saf. Eng.* **7**, 206–212 (2020).
6. Braun, V., Horstmann, A., Lemmens, S., Wiedemann, C. & Böttcher, L. Recent developments in space debris environment modelling, verification and validation with master. In *8th European Conference on Space Debris* (organizationESA Space Debris Office Darmstadt, 2021).
7. Lopez-Calle, I. & Franco, A. I. Comparison of cubesat and microsat catastrophic failures in function of radiation and debris impact risk. *Sci. Rep.* **13**, 385. https://doi.org/10.1038/s41598-022-27327-z (2023).
8. Celletti, A., Pucacco, G. & Vartolomei, T. Reconnecting groups of space debris to their parent body through proper elements. *Sci. Rep.* **11**, 22676. https://doi.org/10.1038/s41598-021-02010-x (2021).
9. Dave, A. A., Singh, G., Buduru, A. B. & Biswas, S. K. A Deep Neural Network-based Space debris trajectory prediction after an elastic collision event. In *SMOPS Conference 2023* (Bangalore, India, 2023).
10. Song, J., Rondao, D. & Aouf, N. Deep learning-based spacecraft relative navigation methods: A survey. *Acta Astronaut.* **191**, 22–40. https://doi.org/10.1016/j.actaastro.2021.10.025 (2022).
11. Becktor, J. *et al.* Robust vision-based multi-spacecraft guidance navigation and control using cnn-based pose estimation. In *2022 IEEE Aerospace Conference (AERO)* 1–10 (IEEE, 2022).
12. Petit, A., Marchand, E. & Kanani, K. Vision-based detection and tracking for space navigation in a rendezvous context. In *Int. Symp. on Artificial Intelligence, Robotics and Automation in Space, i-SAIRAS* (2012).
13. Kaluthantrige, A., Feng, J. & Gil-Fernández, J. CNN-based Image Processing algorithm for autonomous optical navigation of Hera mission to the binary asteroid Didymos. *Acta Astronaut.* **211**, 60–75. https://doi.org/10.1016/j.actaastro.2023.05.029 (2023).
14. Stevenson, E., Martinez, R., Rodriguez-Fernandez, V. & Camacho, D. Predicting the effects of kinetic impactors on asteroid deflection using end-to-end deep learning. In *2022 IEEE Congress on Evolutionary Computation (CEC)* 1–8 (2022). https://doi.org/10.1109/CEC55065.2022.9870215.
15. Stevenson, E., Rodriguez-Fernandez, V., Urrutxua, H. & Camacho, D. Benchmarking deep learning approaches for all-vs-all conjunction screening. *Adv. Space Res.* **72**, 2660–2675. https://doi.org/10.1016/j.asr.2023.01.036 (2023).
16. Sánchez, L. & Vasile, M. Intelligent decision support for collision avoidance manoeuvre planning under uncertainty. *Adv. Space Res.* **72**, 2627–2648. https://doi.org/10.1016/j.asr.2022.09.023 (2023).
17. Raissi, M. & Karniadakis, G. E. Hidden physics models: Machine learning of nonlinear partial differential equations. *J. Comput. Phys.* **357**, 125–141. https://doi.org/10.1016/j.jcp.2017.11.039 (2018).
18. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707. https://doi.org/10.1016/j.jcp.2018.10.045 (2019).
19. Karniadakis, G. E. *et al.* Physics-informed machine learning. *Nat. Rev. Phys.* **3**, 422–440 (2021).
20. Goswami, S., Bora, A., Yu, Y. & Karniadakis, G. E. Informed deep neural operator networks. *Physics* **2207**, 05748 (2022).
21. Brake, M. R. W., Reu, P. L. & Aragon, D. S. A comprehensive set of impact data for common aerospace metals. *J. Comput. Nonlinear Dyn.* **12**, 145. https://doi.org/10.1115/1.4036760 (2017).
22. Schwager, T. & Poeschel, T. Coefficient of restitution and linear dashpot model revisited (2007).
23. Vallado, D. A. *Fundamentals of Astrodynamics and Applications* Vol. 12 (Springer Science & Business Media, Uk, 2001).

24. Price-Whelan, A. M. *et al.* The Astropy project: Sustaining and growing a community-oriented Open-source Project and the Latest Major Release (v5.0) of the Core Package (2022). https://doi.org/10.3847/1538-4357/ac7c74.
25. Rodríguez, J. L. C., Eichhorn, H. & McLean, F. Poliastro: An astrodynamics library written in python with fortran performance. In *6th International Conference on Astrodynamics Tools and Techniques* (2016).
26. Chen, T. & Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining* 785–794 (2016).
27. Mhaskar, H. N. & Poggio, T. Deep vs shallow networks. An approximation theory perspective. *Anal. Appl.* **14**, 829–848 (2016).
28. Lundberg, S. M. & Lee, S.-I. A unified approach to interpreting model predictions. *Adv. Neural Inf. Process. Syst.* **30**, 563 (2017).
29. Fang, Z., Wang, S. & Perdikaris, P. Ensemble learning for physics informed neural networks: A gradient boosting approach. arXiv preprint arXiv:2302.13143 (2023).

Acknowledgements
We acknowledge the National Supercomputing Mission (NSM) for providing computing resources of 'PARAM PRAVEGA' at SERC Building IISc Main Campus Bangalore, which is implemented by C-DAC and supported by the Ministry of Electronics and Information Technology (MeitY) and Department of Science and Technology (DST), Government of India. We also acknowledge the support of the Infosys Foundation for conducting our research. The authors would like to acknowledge the support of U R Rao Satellite Centre, Indian Space Research Organisation (ISRO).

Author contributions
S.K.B. conceived the research idea, S.K.B., A.B.B., and V.K. contributed to developing the methodology, H.M., G.S. and S.K.B. conducted the simulation experiments and analysed results. All authors reviewed the manuscript.

Funding
This research is supported by National Super Computing Mission (NSM) - HPC Applications, Grant no. DST/NSM/R &D_HPC_Applications/2021/03.17.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-51897-9.

Correspondence and requests for materials should be addressed to H.M.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.