Intelligent Systems with Applications 15 (2022) 200092

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

Design of deep neural networks for transfer time prediction of spacecraft electric orbit-raising

Ali Hassaan Mughal [1,a], Pardhasai Chadalavada [2,b], Arslan Munir [3,*,a], Atri Dutta [4,b], Mahmood Azhar Qureshi [5,a]

[a] Kansas State University, Manhattan, Kansas, USA
[b] Wichita State University, Wichita, Kansas, USA

ARTICLE INFO

Keywords:
Deep neural networks
Spacecraft orbit-Raising
Solar-Electric propulsion
Deep reinforcement learning
Orbit transfer

ABSTRACT

Recently, there has been a surge in use of electric propulsion to transfer satellites to the geostationary Earth orbit (GEO). Traditionally, the transfer times to reach GEO using all-electric propulsion are obtained by solving challenging trajectory optimization problems that naturally do not lend themselves to incorporation within deep reinforcement learning (DRL) framework to solve trajectory planning problems in near real-time. The operation of DRL, as typically used in trajectory planning, relies on a Q-value. In the electric orbit-raising problem under consideration in this paper, this Q-Value requires computation of transfer time in near real-time to have practical DRL training times. This work proposes to design and evaluate a machine learning (ML) framework, focusing on deep neural networks (DNNs), to predict the transfer time to assist in Q-value determination instead of solving traditional orbit-raising optimization problems. To this end, we investigate different architectures for DNNs to determine a suitable DNN configuration that can predict the transfer time for each of the mission scenarios with high accuracy. Experimental results indicate that our designed DNNs can predict the transfer time for different scenarios with an accuracy of over 99.97%. To verify the efficacy of our designed DNNs for predicting transfer time that is required for Q-value estimation, we also compare the results from our designed DNNs with the contemporary ML algorithms, such as support vector machines, random forests, and decision trees for regression. Experimental results indicate that our best-performing DNNs can provide an improvement in the mean error of transfer time prediction by up to 14.05× for non-planar transfers and up to 254× for planar transfers as compared to contemporary ML algorithms.

1. Introduction

In the past decade, telecommunication satellite operators with assets stationed in the geosynchronous equatorial orbit (GEO) have demonstrated an increased interest in deploying their satellites into orbit using solar-electric propulsion. Such satellites are initially launched into a geosynchronous transfer orbit (GTO) or similar orbits (such as Sub-GTO or Super-GTO) before the satellite uses its onboard electric propulsion to reach the GEO, which is a circular equatorial orbit of altitude 35,786 km. The resulting transfer from GTO, Sub-GTO or Super-GTO to GEO is referred to as an orbit-raising maneuver, which if done using electric thrusters, takes a significantly long time to accomplish, often being in the order of several months owing to the low-thrust generation capability of electric thrusters. However, the enhanced transfer time is more than compensated by the tremendous propellant savings and the resulting design of smaller and lighter satellites. The mass and volume reduction lead to stacking of multiple satellites in the launch vehicle, facilitating sharing of launch costs, which mean significant cost savings for the telecommunication operators. Compared to traditional chemical orbit-raising maneuver, the electric orbit-raising maneuver is fairly

* Corresponding author.
E-mail address: amunir@ksu.edu (A. Munir).
[1] MSc Student at Kansas State University.
[2] PhD Student at Wichita State University.
[3] Associate Professor at Kansas State University.
[4] Associate Professor at Wichita State University.
[5] Mahmood is currently working as a Senior IP Design Engineer at Intel Corporation

https://doi.org/10.1016/j.iswa.2022.200092
Received 21 February 2022; Received in revised form 13 May 2022; Accepted 2 June 2022
Available online 12 June 2022

complex because the maneuver requires the thruster operation over prolonged periods, disrupted intermittently when the spacecraft is in the shadow of the Earth (note that the electric thrusters derive power from the satellite solar panels). Additionally, the passage through the Van Allen radiation belts, particularly the inner one, damages the solar panels and leads to degradation of power availability for the remaining transfer (and the remainder of the mission as well). As more spacecraft adopt solar-electric propulsion for conducting orbit-raising maneuver, the future mission operations will likely benefit from an enhanced autonomy in mission planning.

A key step in mission planning for the orbit-raising maneuver is the determination of low-thrust multi-revolution trajectory, which requires the solution of a long time scale, multi-phase, nonlinear, non-convex optimal control problem that is challenging to solve. The trajectory to be computed is over a long time scale because of the low magnitude of the propulsion system thrust, resulting in a slow spiraling transfer around the Earth. The problem is multi-phase because the spacecraft passes through multiple eclipses during its transfer, and the dynamics are different in Sun-lit parts of the trajectory and in the shadow of the Earth. The absence of the Sun during eclipses prohibit the spacecraft from thrusting because electric propulsion systems derive power from the solar arrays. The problem is nonlinear owing to the dynamics being primarily governed by the inverse-square gravitational force. Finally, the problem is non-convex because of the nature of underlying spacecraft dynamics and the objective functions (transfer time or fuel expenditure) that are of interest to mission designers. Nevertheless, this challenging problem has been studied in the astrodynamics literature for many decades, and a number of methods have been developed. These methods can be largely classified into direct and indirect optimization techniques, depending on whether calculus of variations is used to determine the necessary conditions of optimality (direct methods bypass this step). Numerical schemes based on these approaches need good quality initial guesses for demonstrating good convergence, and determining the good initial guesses depend on a human expert as well.

A number of other approaches have also been considered, such as shape-based methods or guidance-like schemes, in order to address the challenges of underlying trajectory optimization. In this paper, we leverage a recently developed optimization scheme that can yield fast and robust computations of the low-thrust orbit-raising trajectory without the need for user-provided initial guesses. This optimization scheme relies on two innovations: the use of a set of regularized elements suitable for the application, and an optimization scheme that poses the long time-scale problem as a sequence of unconstrained optimization sub-problems that are easier to solve. We refer to this optimization scheme as sequential low-thrust orbit-raising (SLTOR) optimization problem. Note here that the advantages of fast, robust and automated trajectory computation come at the cost of a sub-optimality of the resulting solution.

Deep reinforcement learning (DRL), which combines reinforcement learning (RL) and deep learning (DL), has emerged as a promising approach for optimization of guidance-based systems in recent years (He et al., 2021; Kolosa, 2019; Sutton and Barto, 2018; Willis et al., 2016). DRL has shown significant improvements in ever-challenging domains despite several uncertainties in the environment. Markov decision processes (MDPs) are one of the most common underlying mathematical frameworks leveraged by DRL. MDPs comprise an agent, an environment, and a given set of goals. Continuous-state MDPs are required for solving many problems, but present various challenges, in particular, state space explosion. A framework for faster computation of continuous-state MDPs by discretizing the space has been presented by) Marecki (Koenig). In MDPs, the agent in a given state takes an action in the environment to reach a new state. The agent starts at an initial point and is given a set of heuristics to reach the end goal (Sutton and Barto, 2018). The heuristic, which is the adaptation of the policy for a particular environment or performance of an agent, improves with time. In MDP, the reward is provided to the agent based on action(s) taken either at the final state or intermediate states, depending on the environment design. A satellite takes a series of actions to reach the GEO, and these actions need to be optimized in order to minimize a certain objective function, such as minimizing the transfer time and/or minimize the fuel consumption, en route to GEO. DRL is typically seen as an enhanced version of the more traditional RL. While the RL is more dynamic and uses trial and error to maximize the outcome, a DRL agent can use the existing knowledge and apply it to a new problem.

While the recent innovations in trajectory determination (such as in Sreesawet and Dutta, 2018) result in fast and robust computation of electric orbit-raising trajectory (at the cost of sub-optimality of the computed trajectories), these cannot be readily integrated within a DRL scheme. The operation of DRL, as typically used in trajectory planning, relies on a Q-value. In the electric orbit-raising problem, this Q-Value requires computation of a metric of interest (such as transfer time or fuel expenditure) in near real-time to have practical DRL training times. We define the prediction of metric of interest (e.g., transfer time) is in *real-time* if the time taken to predict the metric of interest is negligible (i.e., orders of magnitude less) as compared to the time for each discretized segment within a planning horizon. For instance, the SLTOR optimization problem uses 72 segments over a revolution, whose time period ranges from a few hours to sidereal day. Assuming a revolution takes two hours, then for each segment, the decision for spacecraft thrust and direction needs to be taken in $120/72 = 1.67$ minutes. Thus, for a prediction of transfer time to be considered as *real-time*, the prediction should be obtained in a fraction of a second. The expeditious prediction of transfer time can enable the mission designer to make further fine-grained designs regarding thrust and direction of spacecraft (say 360 decisions in a revolution), thus giving the designer more control authority on spacecraft path planning. If the prediction decisions can be taken close to *real-time*, as defined earlier, but not as fast to be considered real-time, then these decisions can be said to be taken in *near real-time*. We also define that trajectory planning problems can be solved in *near real-time* if the metric of interest (e.g., transfer time) that needs to be assessed at each decision-making instant (such as at each segment within a revolution) can be predicted in real-time.

The main challenge here is that the computation of a metric of interest (such as transfer time or fuel expenditure) at any point using SLTOR optimization requires the solution of all optimal control sub-problems corresponding to all future revolutions. While this process is not a challenge for the computation of a single orbit-raising trajectory, this is not an efficient mechanism to evaluate the Q-value, which essentially signifies the maximum expected reward an agent can reach by taking a given action a from the state s, and rewards, which depend on the metrics of interest (such as transfer time or fuel expenditure), associated with all state-action pairs within a DRL scheme. This paper addresses this challenge by designing and evaluating a machine learning (ML) framework, focusing on deep neural networks (DNNs), that facilitate the straight-forward prediction of the metric of interest without having to solve numerous optimization problems corresponding to forthcoming revolutions as in SLTOR optimization. While there have been some works in the literature on training artificial neural networks (ANNs) for the electric orbit-raising problem (Arora and Dutta, 2020; Kwon et al., 2021), there has been a lack of a comprehensive assessment of neural network architectures, and our present paper fills in this void. Our main contributions in this paper are as follows:

- Designing and evaluating an ML framework, focusing on DNNs, for accurate prediction of metric of interest (transfer time in this work) to assist in Q-value determination instead of solving traditional orbit-raising optimization problems for six different planar and non-planar mission scenarios for transfer to GEO.
- Exploring the design space of DNNs to determine a suitable setting of hyperparameters of DNNs for each orbit-raising mission scenario that provides the transfer time prediction with at least 99.97% accuracy,

- Evaluating and comparing the transfer time prediction results from our optimized DNNs with the contemporary ML algorithms, such as support vector machines (SVMs), random forests (RFs), and decision trees (DTs).

To the best of knowledge of authors, there does not exist a comprehensive study that designs and evaluates different ML frameworks for prediction of the metric of interest in missing planning for the orbit-raising scenarios. To this end, we design and evaluate different ML frameworks, focusing on DNNs, for prediction of the metric of interest in mission planning for the following orbit-raising mission scenarios:

- Planar and non-planar Super-GTO to GEO
- Planar and non-planar GTO to GEO
- Planar and non-planar Sub-GTO to GEO

In this paper, we refer to our designed DNNs with a suitable setting of hyperparameters that can provide the transfer time prediction with at least 99.97% accuracy as *optimized DNNs*. While this paper has been primarily motivated by a future application of these artificial neural networks to DRL orbit-raising framework (which is not the focus of this paper), these networks also have potential applications for an adaptive weight modification strategy as studied by Arora (2020), and the six-state targetted orbit-raising maneuver as studied by Chadalavada et al. (2022).

The remainder of this paper is organized as follows. Section 2 discusses prior works in literature related to spacecraft orbit-raising. Section 3 presents the problem formulation for DRL-based orbit-raising, and clarifies where our current work fits in DRL-based orbit-raising. Section 4 details the dataset generation for orbit-raising scenarios. Section 5 elaborates the approach for design of optimized DNNs for transfer time prediction. Experimental results showing the accuracy of the designed DNNs for transfer time prediction for different orbit-raising scenarios with respect to SLTOR optimization are presented in Section 6. Section 7 presents the comparison of our designed DNNs with the contemporary ML algorithms in terms of accuracy of transfer time prediction. Finally, Section 8 concludes this work.

2. Related work

In recent years, researchers have been studying optimal trajectories associated with electric low-thrust orbit raising maneuvers to reach GEO (Betts, 1998; Kluever and Oleson, 1998; Sackett et al., 1975). The low thrust as mentioned previously is the primary concern for using a spacecraft that is all-electric, and that causes a long transfer time taken for the spacecraft to reach the GTO. In addition, the spacecraft encounters multiple eclipses in the route, this further prolongs the transfer time unless the spacecraft has the capability of storing electric energy for use during the eclipses (Marasch and Hall, 2000). This creates a reason to investigate closely the variety of crucial factors affecting the electric orbit-raising problem. These factors include the initial orbit of the spacecraft, dry mass i.e. the mass that is delivered, the mass of propellant, the capacity solar panel arrays have to generate thrust during transfer and the type of thrusters to be used in the spacecraft (Hall, Magneto Plasma Dynamic (MPD) or Ion Thrusters). Hence, mission designers have to investigate in detail various scenarios to obtain the least time taking trajectory.

In order to overcome the above stated issue, direct optimization methods are used. The state and control variables can be discretized with respect to time and apply quadrature rules across the discretized segments in order to set up a parameter optimization problem that will be a nonlinear programming problem (Betts, 2000; Herman and Conway, 1996), that are solved by softwares like Sparse Nonlinear OPTimizer (SNOPT), Interior Point OPTimizer (IPOPT) (Dutta and Arora, 2019; Graham and Rao, 2016; Wang and Grant, 2017; 2018). The rate of convergence for low-thrust optimization using direct methods are better than indirect methods. However, direct methods need initial guesses which are unknown prior. This lack of knowledge about initial guesses has led to a number of studies such as shape based methods (De Pascale and Vasile, 2006; Novak and Vasile, 2011; Petropoulos and Longuski, 2004; Taheri and Abdelkhalik, 2012; Vasile and Casotto, 2007; Wall and Conway, 2009) and a number of guidance-based schemes (Kluever, 1998a; 1998b; Petropoulos, 2004) to approximately solve low-thrust optimization problems. The electric orbit-raising problem has been studied using a variety of dynamic models, to specifically, study the influence on the convergence of low-thrust trajectory through numerical optimization schemes (Caruana and Niculescu-Mizil, 2006; Junkins and Taheri, 2018). Also, among all the regularized elements, five remain constant for time and change slowly under the perturbations in case of Keplerian motion. Hence, these are suitable for use in trajectory optimization schemes that have long-time-scale transfers.

Many prior works have explored the use of artificial intelligence (AI), in particular, RL and DRL, for trajectory optimization and planning of spacecraft. Shirobokov et al. (2021) have presented a survey of ML techniques in spacecraft control detailing the work on formulation control and design of control laws in spacecraft flight and landing. An actor critic-based RL model consisting of two neural networks has been explored in Kolosa (2019) by Kolosa. Kolosa has also analyzed the effect of gravity in maneuvering near small bodies. Kolosa has used an RL model to solve low-thrust trajectory optimization problems using two neural networks, an actor network and a critic network. Willis et al. (2016) have studied the spacecraft near a small celestial body with unknown gravity using RL. Shaoming et al. (He et al., 2021) have used a heuristic way to shape a proper reward function for the RL agent where the heuristic is shaped into a command for the missile guidance. For the heuristic function, they have taken into account the energy consumption and guidance accuracy, and their trade-off. They have examined two types of learning agents, learning from scratch and learning with prior knowledge (with a guidance command provided at each step). They have validated the effectiveness of the proposed approach using extensive numerical simulations. LaFarge (2020) has explored a controller, trained via RL considering multibody perturbations. Xiao et al. (Wang et al., 2020) have developed a framework to find suitable parameters for numerical optimization of such RL models. Xiao et al. (Wang et al., 2020) have used an actor-critic algorithm where each satellite cluster is assigned to three sensing zones. To tune the flight parameters, they have used an actor-critic network and have shown to successfully tune flight parameters with lower deviation in trajectory for a cluster of flights.

A guidance strategy for spacecraft proximity operations using DRL has been explored by Hovell and Ulrich (2021). Hovell and Ulrich have utilized a control theory approach alongside DRL to aid the transfer of the learned behavior from simulation to reality. They have demonstrated their approach via a proof-of-concept spacecraft pose tracking and docking scenario. Broida and Linares (2019) have presented a Rendezvous guidance technique for spacecraft in a cluttered environment using RL. They have implemented and evaluated a proximal policy optimization (PPO) to develop a control policy. This policy has been used to move a satellite relative to an orbit reference frame into a docking position with another. Li et al. (2021) have used a deep deterministic policy gradient (DDPG) framework to target and avoid obstacles for unmanned aerial vehicles (UAVs). Oestreich et al. (2021) have looked into the problem of six degrees of freedom docking via RL. The policy implemented by Oestreich et al. (2021) allows the docking maneuvers as a feedback control law under uncertain environments. A safe landing site selection considering divert maneuvers using DRL has been proposed by Iiyama et al. (2021).

Some prior works have explored other ML techniques in conjunction with DRL for space and other domains. Haj-Ali et al. (2020) have used RF in DRL to assist in high-level synthesis (HLS) phase orderings. The state space that compilers use is huge, and the authors have used RF to reduce the state space. The authors have implemented an RF based RL algorithm called AutoPhase, that can generate an efficient program for

compilers. Gaudet et al. (2022) have utilized a meta-RL technique (a technique to train an agent that generalizes across multiple tasks via summarizing experience over those tasks (Zhao et al., 2021)) to optimize an adaptive guidance system for the approach phase of a gliding hypersonic vehicle. Their guidance technique learns to induce target trajectories to reach a target location while satisfying constraints such as terminal speed, heating rate, load, and dynamic pressure. Cheng et al. (2021) have presented and trained a DNN based re-entry guidance algorithm. The DNN takes the current state as input and predicts the longitudinal downrange and the lateral crossrange of a flight. Yuexuan et al. (2018) have implemented an SVM in conjunction with advantage actor-critic (A2C) RL algorithm. The actor takes the actions and gets feedback from the environment. Then the critic network evaluates the results of actions in the environment and optimizes an SVM network according to the evaluation result. Then the critic uses this optimized result to instruct the actor network to improve its behavioral policy, and hence complete a step in an iteration of the A2C network training. Hyeokjoon et al. (Kwon et al., 2021) have presented a soft actor-critic (SAC) algorithm for trajectory-raising optimization. The authors have considered SAC for only low Earth orbit (LEO) to GEO, and GTO to GEO mission scenarios.

Although AI has been applied to spacecraft trajectory optimization, prior works in the area did not design and/or evaluate a machine learning framework for transfer time prediction for different orbit-raising scenarios. Prior works on AI-based trajectory optimization are complementary to this work. The proposed framework can be utilized in other AI-based trajectory optimization techniques, in particular DRL-based trajectory optimization, as it predicts the transfer time given a state with very high accuracy instead of using SLTOR optimization equations, and is thus suitable for automating trajectory optimization solutions.

3. Problem formulation and relevance to DRL

This work proposes to design and evaluate an ML framework, focusing on DNNs, that can be integrated with other AI-based trajectory optimization techniques to predict the metric of interest (transfer time in this work) from a given state in trajectory planning of spacecraft in an orbit-raising scenario. This metric of interest is traditionally computed by solving challenging SLTOR trajectory optimization problems, whose solution rely on compute-intensive numerical schemes that lack automated implementation capabilities. In context of DRL, this prediction of transfer time metric can assist in Q-value determination instead of solving SLTOR optimization problems.

This section succinctly summarizes RL and illustrates where our proposed optimized DNNs fit in a DRL-based trajectory optimization framework. In the same manner, our proposed optimized DNNs can be integrated with other AI-based trajectory optimization techniques. We clarify that DRL-based trajectory planning is not focus of this work, and this section only illustrates DRL-based trajectory planning to show that how our proposed DNNs fit in a DRL-based framework.

RL is one of the three basic machine learning paradigms, with the other two being supervised and unsupervised learning. In RL-based approaches, agents, in an environment, take actions to maximize the cumulative reward. Unlike dynamic programming, RL does not assume knowledge of the mathematical model and generally targets large MDPs where the exact mathematical solution becomes infeasible. While greedy approaches follow a fixed set of paths for finding the optimal path, RL allows an agent to take actions for exploration and not follow a fixed set of paths, but a continuous one. The agent in an RL model is typically trained using Q-Learning (Sutton and Barto, 2018), as shown in the following equation.

$$\underbrace{Q(s,a)'}_{\text{New Q-Value}} = \overbrace{Q(s,a)}^{\text{Old Q-Value}} + \underbrace{\alpha}_{\text{Learning Rate}} \left[\underbrace{R(s)}_{\text{Reward}} + \underbrace{\gamma}_{\text{Discount rate}} \overbrace{\max Q'(s',a')}^{\text{Maximum predicted reward, given new state and all possible actions}} - Q(s,a) \right] \tag{1}$$

In Eq. 1 Q' is the updated Q value for a state s for an action a, that is, take action a in state s and update the Q value using the old Q value for the state and action. R is the reward an agent receives when it reaches a new state s' after taking an action a in state s. γ is a discount factor that determines how much future rewards are worth and α is the learning rate at which we update the Q value and get Q'. The reward has to be maximized over a set of possible actions $a \in A$ for a state s and can be formulated as:

$$R = \max_{a \in A} F(s,a), \tag{2}$$

where R is a reward function, F is a function that depends on the predicted transfer time for each state s given an action a from a set of possible actions in that state s. We note that the state s is referred to as x in the dataset generation section (Section 4). Each a in A will generate a different possible next state. In orbit-raising, the action a at each segment in the revolution is a set of thrust force ($\mathscr{T}$), and two control angles χ (in-plane angle) and β (out-of-plane angle).

As can be seen from Eq. 1, this Q value determination needs calculation of a reward function for every single action given a state. This reward function calculation depends on the estimation of a metric of interest (e.g., transfer time or fuel expenditure). The computation of this metric (transfer time) at any point using SLTOR optimization requires the solution of all optimal control sub-problems corresponding to all future revolutions. Thus, using SLTOR optimization becomes highly computationally expensive for assisting in Q-value determination for every state-action pair. This is where our proposed optimized DNNs come handy, which can predict the transfer time in real-time for any given state, and thus can be used in estimation of Q-value for all state-action pairs.

For trajectory optimization, the trained DRL model computes the total transfer time required for each orbit-raising scenario to reach the GEO. Thus, there can be different transfer times corresponding to different actions taken in the environment. For trajectory optimization, the trained DRL model prescribes a policy that can minimize the transfer time required for each orbit-raising scenario to reach the GEO. At each state, the DRL model selects an action which can minimize the transfer time. A set of all such actions given each state form a policy that is prescribed by a DRL model. Fig. 1 depicts DRL model for spacecraft orbit-raising. In Fig. 1, deep Q-network (DQN) is used to estimate Q-value $Q(s,a)$ for each action a that can be taken in state s to reach a new state s'. The DRL model relies on DQN, which serves as the predictor function of the Q-value. The model passes a state to the DQN, which then provides a set of (action, Q-value) pairs corresponding to the possible actions in the state and the respective Q-value for taking the action in

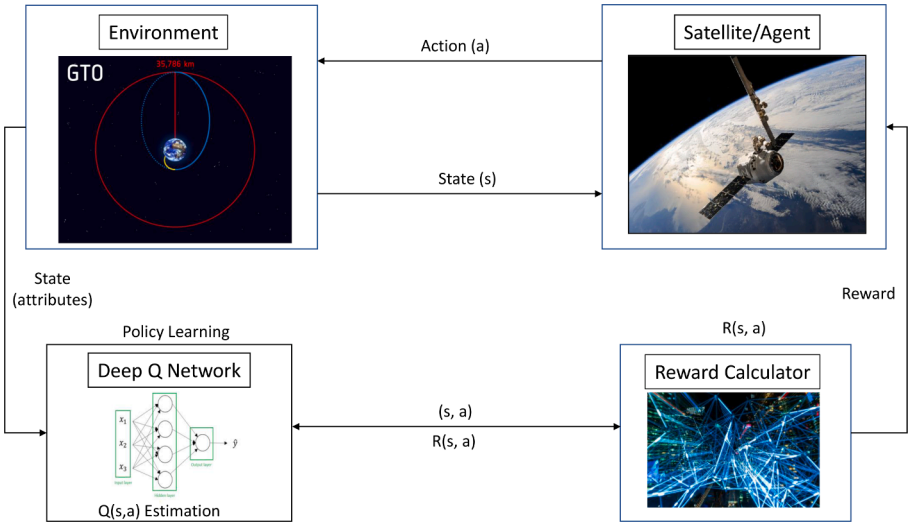

Fig. 1. Deep reinforcement learning model for spacecraft orbit-raising.

that state. The pair with the highest Q-value is selected by the model. Our proposed optimized DNNs can assist DQN in estimating Q-value by predicting the transfer time from each state, which is used in reward function estimation, which in turn is used for Q-value calculation (Eq. 1).

Our optimized DNN model's predicted time values can be used to in the DRL framework to take an action, that can serve as a directional input for the DRL agent and give the raising trajectory to GEO with the least time. This set of actions serves as the policy that provides minimal time transfer for a specific orbit-raising. To train our optimized DNNs, we need to generate data for each of the mission scenarios mentioned in Section 1. The dataset generation for orbit-raising scenarios is discussed in the following section.

4. Generating dataset for orbit-raising scenarios

In this section, we discuss the procedure that we have followed to generate the data to train the neural networks. To this end, we compute the transfer times required from a given orbit of the spacecraft to the final GEO by solving a sequence of optimization sub-problems, each over one revolution of the satellite around the Earth. As already mentioned, in this work, we describe the states of the spacecraft using a set of regularized elements introduced by Sreesawet and Dutta (2018). The states are described by the following: the magnitude of the specific angular momentum vector h, the components of specific angular momentum vector h_X and h_Y in the X-Y plane of Earth-centered inertial reference frame, the components e_x and e_y of the eccentricity vector in the x-y plane of the reference frame obtained after 2-1 rotation of the inertial frame, and an angle ϕ that locates the spacecraft in orbit. The first five coordinates used in this model are slow varying for an electric orbit-raising problem. Each optimization sub-problem aims to bring the spacecraft in close proximity of GEO by minimizing the following objective function:

$$J = w_h(h - h_{GEO})^2 + w_{hxy}\left(h_X^2 + h_Y\right)^2 + w_e\left(e_x^2 + e_y^2\right)^2, \tag{3}$$

where h_{GEO} is the specific angular momentum of the spacecraft in the GEO, w_h, w_{hxy}, and w_e are weights (positive scalars) associated with the three components of the objective function satisfying the constraint $w_h + w_{hxy} + w_e = 1$. Clearly, the scalar weights play a crucial role in determining the transfer time to the GEO. To this end, we generate data using different weight combinations.

To solve the optimization problem, we discretize each revolution of the spacecraft into N segments. Additionally, we determine the variation of five slow varying states (collectively represented as $\boldsymbol{x}$) of the spacecraft with respect to the angle ϕ. The differential state equations depicting the variation of the slow variables $\boldsymbol{x}$ with respect to the fast variable ϕ are detailed in Sreesawet and Dutta (2018). Using these differential equations, we can determine the state transition over each segment in the revolution using the equation below:

$$\widetilde{\boldsymbol{x}}_{i\to i+1} = \frac{\Delta\phi}{2}\left(\frac{d\boldsymbol{x}_i}{d\phi} + \frac{d\boldsymbol{x}_{i+1}}{d\phi}\right), \tag{4}$$

where $\boldsymbol{x}$ is $[h\ h_X\ h_Y\ e_x\ e_y]^T$ and index i represents a node on the trajectory, that is, i belongs to the set {0,1,....,N-1}. To set up the objective function for the optimization sub-problem corresponding to the $k+1$th revolution, we sum these change of states over each segment and add them to the end states of the previous revolution (k^{th} revolution) and determine the states of the spacecraft at the end of the $(k+1)^{th}$ revolution as shown below:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \sum_{i=0}^{N-1} \widetilde{\boldsymbol{x}}_{i\to i+1}. \tag{5}$$

The optimization problem's decision variables are the α (in-plane) and β (out-of-plane) angles in which the spacecraft thrust. As we have N nodes over each revolution, each optimization problem has 2N variables: N for the in-plane thrust angle α and N for the out-of-plane thrust angle β. Additionally, we have normalized the states of the spacecraft that helps with the fast computation of the optimization problems. For normalization, we consider the distance unit (DU) as the radius of the GEO orbit. We choose the time unit (TU) such that the Earth's gravitational constant is 1 DU^3/TU^2. We consider the mass unit (MU) as the initial mass of the spacecraft. Each optimization problem determines the normalized states at the end of the revolution by choosing these 2N variables such that the objective function is minimized. It is important to note that the coordinates used in this work can easily be converted to other coordinates such as Cartesian and Keplerian orbital elements; please see Ref. Marquardt (1963) for details. The optimization process is continued until stopping conditions determining the proximity of the spacecraft to GEO is detected. To this end, the terminal conditions for the sequence of the optimization problems are described in the Keplerian elements that are semi-major axis, inclination, and eccentricity. The terminal orbit corresponds to a semi-major axis of 1 DU, zero eccentricity and zero inclination. We enforce these terminal constraints as inequality constraints, with upper and lower bounds being determined, with the tolerance on the semi-major axis being $\pm$ 0.00001 DU and the tolerance on the eccentricity being 0.00001. The tolerance on the

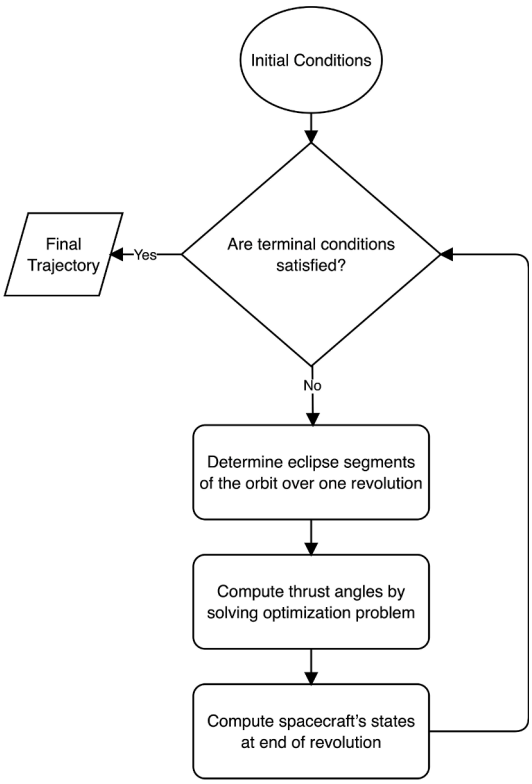

Fig. 2. Data generation flowchart.

inclination is 0.01 deg.

As shown in Fig. 2, we start with a set of given initial conditions that includes the six states of the spacecraft and the three weights for the objective function of the optimization problem. During the transfer of the spacecraft from the initial orbit to the GEO, it may encounter an eclipse each time it passes through the Earth's shadow. In the absence of power from solar array during eclipses, we consider zero thrust availability. To this end, we determine the nodes that are under the shadow of the Earth using a non-rotating cylindrical shadow model. Once the eclipse part of the trajectory during a revolution is selected, we determine the optimal thrust angles that brings the spacecraft closest to GEO at the end of the revolution. After each revolution, we check the terminal or stopping conditions to determine the need for solving additional optimization sub-problems. The optimization problems are solved in Matlab using the "Fminunc" solver. We also provide gradients of the objective function with respect to the thrust angles to the solver, which makes it more robust and computationally efficient. The implementation details of the dynamic model used to propagate the states of the spacecraft, the Earth's shadow model, and the optimization problem setup is discussed in detail in the works of Sreesawet and Dutta (2018), and Chadalavada et al. (2020).

We generate data from trajectories starting from six different initial orbits and all the possible weight combinations. The six initial orbits consist of three planar and three non-planar GTO, Sub-GTO, and Super-GTO cases. The generated data set has **18.6k** points in planar GTO data, **483.4k** points in non-planar GTO data, **23.3k** points in planar Sub-GTO data, **567.9k** points in non-planar Sub-GTO data, **14.3k** points in planar Super-GTO data and **388.3k** points in non-planar Super-GTO, totalling to a **1495.8k** training points (1 for each revolution). Each training pair has seven inputs and one output. The seven inputs are the semi-major axis, eccentricity, inclination, the spacecraft's mass, and the three scalar weights of the optimization function's objective function. The output in the training pairs is the transfer time corresponding to the inputs in the respective training pairs. We consider the spacecraft's mass as one of the input to account for the electric propulsion characteristics of the spacecraft as the change of mass of the spacecraft is directly proportional to the thrust of the spacecraft and inversely proportional to the specific impulse (I_{sp}).

The computational time to generate data set for each transfer is: **56.64 sec** for planar GTO data, **2.03 hr** for non-planar GTO data, **61.53 sec** for planar Sub-GTO data, **2.20 hr** for non-planar Sub-GTO data, **63.09 sec** for planar Super-GTO data and **1.40 hr** for non-planar Super-GTO, totalling to a **5.69 hr** data generation computational time. The data generation algorithm is run on Matlab installed on a machine using an Apple M1 chip with 8 Cores and 16 GB RAM. In this work, we compute the trajectories for the planar cases using the same formulation as in the non-planar instances in contrast to the results of Sreesawet and Dutta (2018). Using the planar case formulation presented in Reference Sreesawet and Dutta (2018) will generate the same trajectory in a fast manner and thus reduce the data generation times for the planar trajectories.

Fig. 3 shows the optimal all-electric orbit raising trajectory to the GEO from the GTO launched by the Ariane-5 launch vehicle. This trajectory is the solution obtained for the objective function weights w_h as 0.5, w_{hxy} as 0.4, and w_e as 0.1. The satellite is considered to have an initial mass of 5000 kg and has a thrust of 1.17 N and I_{sp} of 1700 s.

For a given trajectory, the data set contains each revolution's end states, the spacecraft's mass, and the time required to reach GEO from the current states as one data point and solve trajectories for different weight combinations to get the entire data set. It is important to note that the data generation steps detailed in this section must be repeated if the propulsion system characteristics vary drastically from the mission scenario considered for the data generation. However, one can envision a neural network with additional inputs for the propulsion system characteristics, such as thrust magnitude or the specific impulse that can be used to train DNNs that can be utilized for mission scenarios that consider different propulsion systems.

5. Designing the optimized neural networks for transfer time prediction

In this section, the proposed problem-solving methodology is discussed. It starts with an overview and continues to detail the tuning of neural networks trained for orbit-raising mission.

5.1. Overview

The dataset has trajectory data for the six mission scenarios referred in Section 1 and discussed in detail in Section 4. Our goal is to be able to predict, accurately, the time from a current state represented using the 7 variables (a, e, i, w_h, w_{hxy}, w_e, m) to a final state in GEO. These variables are detailed in Section 4. The predicted time is a continuous value and hence a linear regression model is used (Cass, 1983; Marriott, 1985).

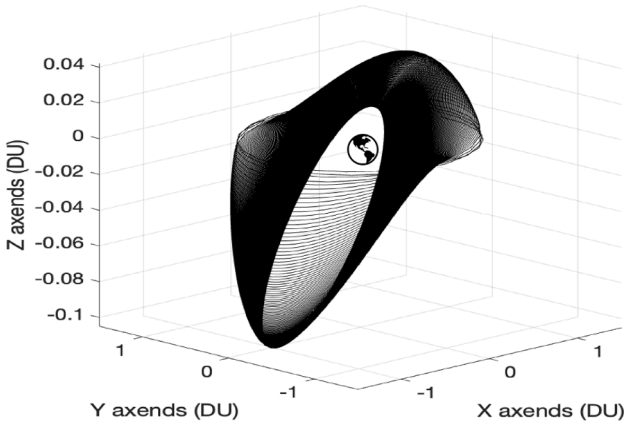

Fig. 3. Optimal trajectory to GEO from GTO.

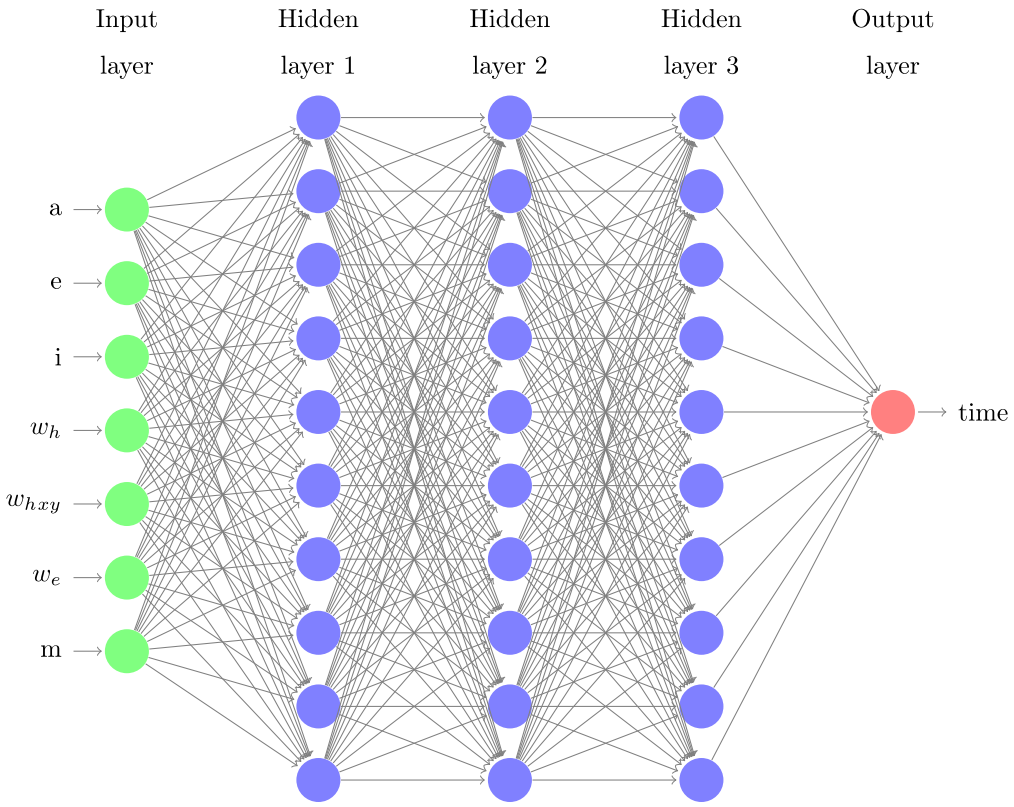

Fig. 4. Example of DNN-3 with 30 neurons (a 10-10-10 configuration).

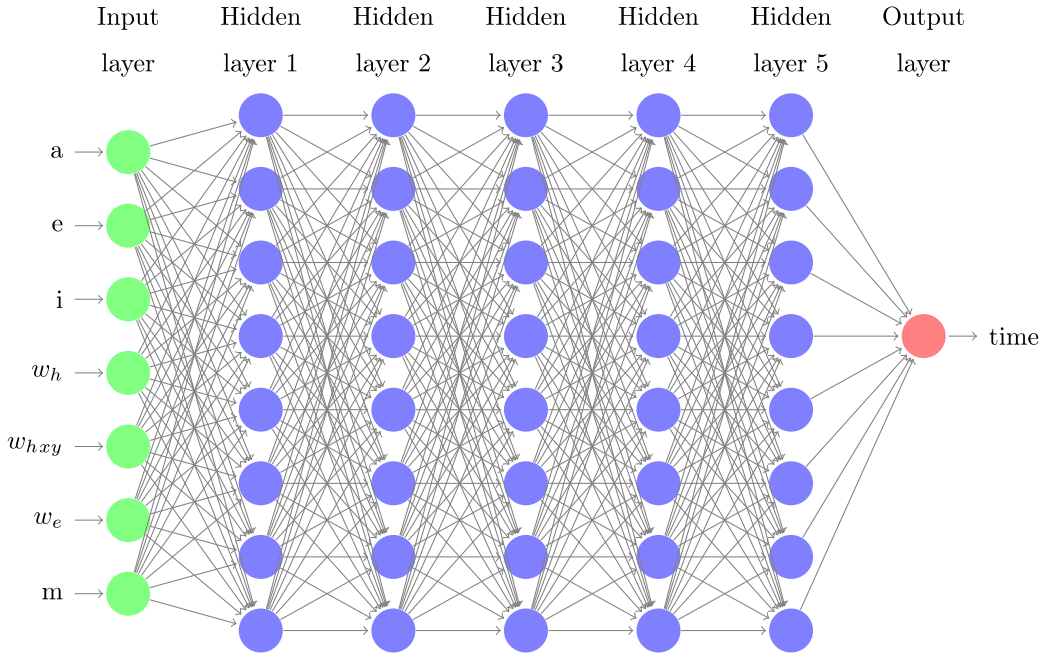

Fig. 5. Example of DNN-5 with 40 neurons (an 8-8-8-8-8 configuration).

Figs. 4 and 5 depict DNN-3 with 30 neurons and DNN-5 with 40 neurons respectively.

For our experiments, we have explored a total of 40 configurations with two varying hyperparameters: number of layers and number of neurons. We have explored the total neurons in DNN ranging from 10 to 110 with eight (8) values being 10, 20, 30, 40, 50, 70, 90, and 110. We have explored varying number of DNN layers ranging from one layer to five layers. Thus, in total, we have 8 x 5 = 40 configurations to explore for each orbit-raising scenario. A total of $5 \times 8 = 40$ combinations for each orbit raising problem were considered. The naming convention for the sake of this paper goes as follows. DNN-X with N neurons means a DNN with X number of layers, with each layer having N/X neurons. For example, DNN-5 with N = 50 neurons implies a DNN that has a total of 5 layers, with each layer having N/X = 50/5 = 10 neurons. For N and X,

where N is not divisible by X, the floor value of N/X is taken, and the excess neurons are distributed equally among the initial layers. For example, when N and X are 10 and 4, respectively would be calculated as:

$$\lfloor 10/4 \rfloor . = 2$$

Each layer is assigned 2 neurons initially, then the remaining 2 neurons are distributed to layer 1 and layer 2, thus, the layers configuration becomes 3-3-2-2, where the first two layers have three neurons each and the last two layers have two neurons each.

5.2. Approach

We initially used Adam optimizer Kingma and Ba (2015) by Kingma and Ba. Our results for transfer time prediction and mean squared error (MSE) obtained through the Adam optimizer. The results obtained through the Adam optimizer resulted in an unacceptably high MSE. Furthermore, the time of prediction is of essence in evaluation of spacecraft orbit-raising scenarios, and hence, keeping a notch on the number of neurons in the DNN design is crucial. One such technique that is relatively faster is the Levenberg Marquardt (LM) algorithm (Gavin, 2020) by Gavin. We, therefore, have used the LM algorithm for training the DNN to achieve a faster and more accurate solution when compared against the Adam optimizer solution. The LM algorithm finds the optimal solution, but its effectiveness comes with a drawback of high training time/compute power and memory usage. It uses a Hessian matrix that is a matrix of second derivative. This matrix makes it guaranteed to find the minima of the loss function. But computing a Hessian matrix becomes an in-efficient solution in terms of memory and computations for large datasets and DNNs with a large number of neurons. The training for the LM algorithm was first done using a Python implementation, but that was inefficient in terms of memory usage. The MATLAB implementation had more options to limit the amount of data it processes in each chunk of iteration of the algorithm. Therefore, MATLAB implementation was faster and had better attributes for the network to analyze and test. The results in Section 6. are generated using MATLAB implementation of the LM algorithm.

Initially, the Sub-GTO to GEO transfer dataset was used. Then, we tested different numbers of neurons and various configurations on this dataset to achieve a good fit for the problem. Through these experiments on Sub-GTO to GEO transfer dataset, we analyzed and found the range for maximum neurons to be 110 and a total of 5 layers to be enough to fit the dataset. We then performed a design space exploration to explore various DNN configurations with varying layer sizes and total number of neurons.

6. Experimental results for optimized DNNs vs SLTOR optimization

The section discusses results from 40 different DNN configurations for each of the 6 orbit raising mission scenarios. The results are presented as a set of both figures and tables, from figures we can analyze the pattern of DNN performance. The tables come into play for specific value comparisons. We have discussed the best neural networks overall first, then explore the best DNN configuration corresponding to the number of neurons and the number of layers the neurons are distributed into.

6.1. Non-planar Super-GTO to GEO transfer

In our first set of experiments, we aim to design a DNN that accurately predicts the spacecraft's transfer time for non-planar transfer from Super-GTO to GEO. Fig. 6 depicts mean error in days of the spacecraft's transfer time predicted by different DNN configurations. Table 1 presents the mean error in days for 40 different DNN configurations with varying number of layers and the number of neurons for non-planar Super-GTO to GEO transfer.

The results show that the DNN-3 configuration with 110 neurons achieves the lowest mean error of 0.0256 days (37.44 minutes). DNN-2 and DNN-4 configurations with 110 neurons exhibit mean errors of 0.0294 days (41.76 minutes) and 0.0400 days (57.60 minutes), respectively. This represents a 11.5% and a 53.8% increase in the mean error over the DNN-3 configuration with 110 neurons. DNN-2 configuration with 90 neurons has a mean error of 0.0891 days (128.30 minutes).

One key observation from Fig. 6 is that by dividing a limited number of neurons into multiple layers, the error becomes higher. However, to achieve a good regression fit, more layers need to be added. Therefore, having more layers and a greater number of neurons results in a

Table 1
Mean error in days (square root of MSE rounded to 4 decimal points) for 5 different neural network configurations for non-planar Super-GTO to GEO transfer.

No. of Neurons	DNN-1	DNN-2	DNN-3	DNN-4	DNN-5
10	0.6660	0.6259	2.2304	2.1148	3.5103
20	0.5943	0.5849	0.5868	0.5891	0.6234
30	0.5764	0.5082	0.5308	0.6680	0.5652
40	0.5750	0.4504	0.4867	0.5196	0.5379
50	0.5578	0.3528	0.3800	0.5772	0.4905
70	0.5381	0.2106	0.2329	0.2964	0.3198
90	0.5200	0.0891	0.1091	0.1998	0.2370
110	0.5309	0.0294	0.0256	0.0400	0.1105

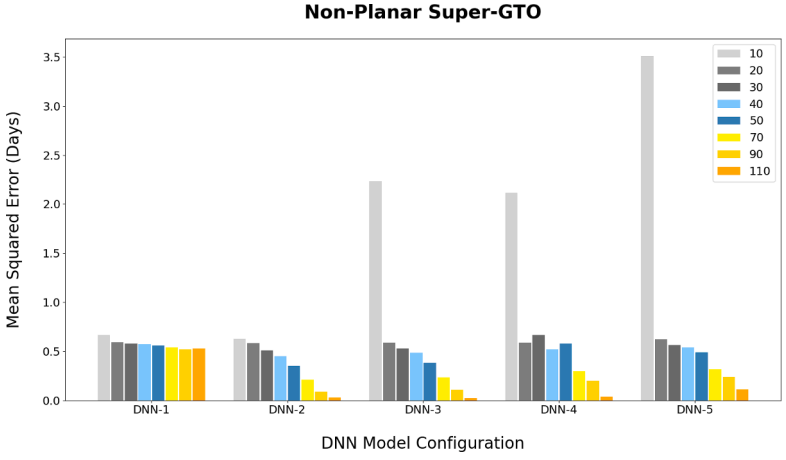

Fig. 6. Mean error in days (square root of MSE) for 5 different neural network configurations for non-planar Super-GTO to GEO transfer.

significant decrease in mean error. DNN-2 and DNN-3 with 30 neurons result in a mean error of 0.5082 days (731.81 minutes) and 0.5308 days (764.35 minutes), respectively. With 40 neurons, the DNN-2 has a mean error of 0.4504 days. DNN-2 with 90 neurons and DNN-3 with 110 neurons exhibit mean errors of 0.0891 and 0.0256 days (128.30 and 36.86 minutes), respectively.

6.2. Non-planar GTO to GEO transfer

In our second set of experiments, we aim to design a DNN that accurately predicts the spacecraft's transfer time for non-planar transfer from GTO to GEO. Fig. 7 depicts mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

The results show that the DNN-4 configuration with 110 neurons achieves the lowest mean error of 0.0435 days (62.49 minutes). DNN-2 and DNN-3 configurations with 110 neurons exhibit mean error of 0.0586 days (84.24 minutes) and 0.0670 days (96.48 minutes), respectively. This represents a 34.79% and a 54.38% increase in the mean error over the DNN-4 configuration with 110 neurons. DNN-2 and DNN-3 configurations with 90 neurons have a mean error of 0.1227 days (176.69 minutes) and 0.1230 days (177.12 minutes), respectively. Table 2 details mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

For the DNN containing 10 neurons, the most optimal performance is exhibited by DNN-2 with a mean error of 1.1831 days (1703.664 minutes). For 20 and 30 neurons, DNN-2 has mean errors of 0.7780 days (1120.32 minutes) and 0.4999 days (719.86 minutes), respectively. DNN-3 with 90 neurons and DNN-4 with 110 neurons exhibit mean errors of 0.1227 days (176.69 minutes) and 0.0435 days (62.64 minutes), respectively.

6.3. Non-planar Sub-GTO to GEO transfer

In our third set of experiments, we aim to design a DNN that accurately predicts the spacecraft's transfer time for non-planar transfer from Sub-GTO to GEO. Fig. 8 depicts mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

The results in Fig. 8 that the DNN-3 with 110 neurons achieves the best mean error of 0.0294 days (42.19 minutes). DNN-2 and DNN-4 configurations with 110 neurons exhibit mean errors of 0.0326 days (46.80 minutes) and 0.0423 days (60.77 minutes), respectively. This represents a 10.92% and 44.03% increase in the mean error over the best DNN-3 configuration with 110 neurons. DNN-2 with 90 neurons has a mean error of 0.0728 days (104.83 minutes), which is nearest to the mean errors of DNN-3 with 90 neurons and DNN-5 with 110 neurons 0.0802 days (115.488 minutes) and 0.0945 days (136.08 minutes), respectively. This represents an increase of 10.31% and 29.85% mean error over DNN-2 with 90 neurons. Table 3 details mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

Table 2
Mean error in days (square root of MSE rounded to 4 decimal points) for 5 different neural network configurations for non-planar GTO to GEO transfer.

No. Neurons	DNN-1	DNN-2	DNN-3	DNN-4	DNN-5
10	1.2230	1.1831	1.3164	1.3979	2.3424
20	1.1077	0.8336	0.7787	0.8202	1.2013
30	1.0355	0.4999	0.5080	1.1626	0.5381
40	0.6237	0.4677	0.4608	0.4499	0.4789
50	1.0026	0.3470	0.3970	0.4176	0.4509
70	0.5188	0.2349	0.2443	0.3692	0.3692
90	0.7230	0.1227	0.1230	0.1501	0.2159
110	0.4961	0.0586	0.0670	0.0435	0.1854

For a network containing 10 neurons, the most optimal performance was exhibited by DNN-2 with a mean error of 0.6099 days (878.26 minutes). For 20 and 30 neurons, DNN-2 has mean error of 0.5555 days (799.92 minutes) and 0.5083 days (731.95 minutes), respectively. DNN-2 with 90 neurons and DNN-3 with 110 neurons exhibit mean errors of 0.0728 days (104.83 minutes) and 0.0294 days (42.34 minutes), respectively.

6.4. Planar Super-GTO to GEO transfer

In our fourth set of experiments, we aim to design a DNN that accurately predicts the spacecraft's transfer time for planar transfer from Super-GTO to GEO. Fig. 9 depicts mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

The results show that the DNN-3 with 70 neurons achieves the best mean error of 0.0028 days (4.03 minutes). DNN-4 with 70 neurons and DNN-3 with 110 neurons configurations exhibit mean errors of 0.0034 days (4.90 minutes) and 0.0036 days (5.18 minutes), respectively. This represents a 21.4% and 28.6% increase, respectively, in the mean errors over the best DNN-3 configuration with 70 neurons. Table 4 details mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

DNN-2 with 50 neurons has a mean error of 0.0042 days (6.05 minutes) which is nearest to the mean error of DNN-5 with 70 neurons of 0.0053 days (7.63 minutes). Hence, DNN-2 with 50 neurons represents an increase of 23.8% mean error over DNN-2 configuration with 90 neurons.

For a network containing 10 neurons, the most optimal performance is exhibited by DNN-2 with a mean error of 0.4059 days (584.50 minutes). For 20 and 30 neurons, DNN-2 and DNN-5 have mean errors of 0.2385 days (343.44 minutes) and 0.2686 days (386.78 minutes),

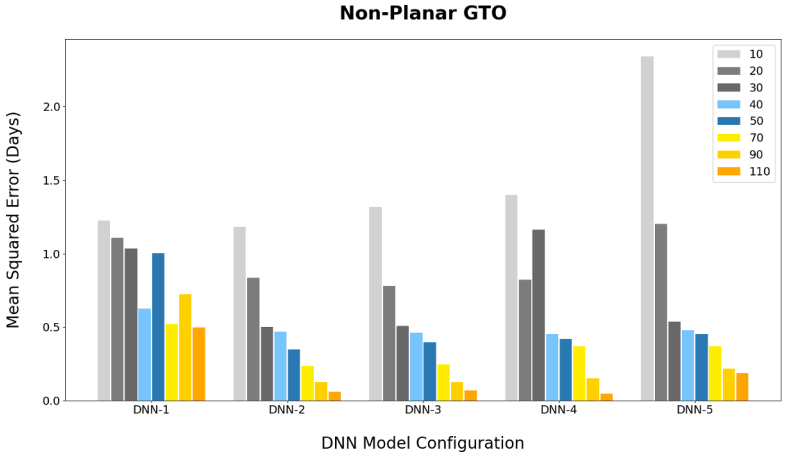

Fig. 7. Mean error in days (square root of MSE) for 5 different neural network configurations for non-planar GTO to GEO transfer.

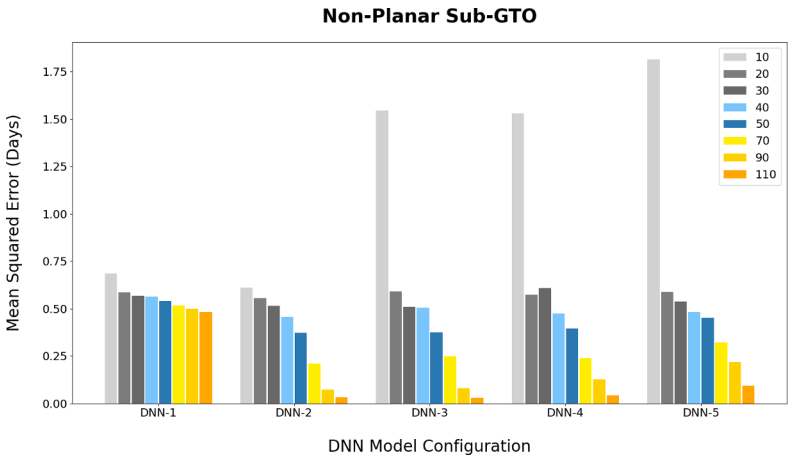

Fig. 8. Mean error in days (square root of MSE) for 5 different neural network configurations for non-planar Sub-GTO to GEO transfer.

Table 3
Mean error in days (square root of MSE rounded to 4 decimal points) for 5 different neural network configurations for non-planar Sub-GTO to GEO transfer.

No. of Neurons	DNN-1	DNN-2	DNN-3	DNN-4	DNN-5
10	0.6849	0.6099	1.5446	1.5298	1.8147
20	0.5865	0.5554	0.5913	0.5724	0.5888
30	0.5685	0.5137	0.5083	0.6087	0.5366
40	0.5625	0.4568	0.5053	0.4744	0.4818
50	0.5399	0.3715	0.3746	0.3943	0.4509
70	0.5182	0.2111	0.2472	0.2379	0.3206
90	0.4999	0.0728	0.0802	0.1273	0.2166
110	0.4826	0.0326	0.0294	0.0423	0.0945

respectively. DNN-3 with 90 neurons and 110 neurons exhibit mean errors of 0.0196 days (28.22 minutes) and 0.0036 days (5.18 minutes), respectively.

6.5. Planar GTO to GEO transfer

In our fifth set of experiments, we aim to design a DNN that accurately predicts the spacecraft's transfer time for planar transfer from GTO to GEO. Fig. 10 depicts mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

The results show that the DNN-3 with 90 neurons achieves the best mean error of 0.0030 days (4.32 minutes). DNN-3 with 110 neurons exhibits mean error of 0.0032 days (4.61 minutes). This represents a 3.33% increase in mean error over the mean error of DNN-3 with 90 neurons. DNN-5 with 70 neurons has a mean error of 0.0050 days (7.20 minutes) that is closest to the mean error of DNN-2 with 50 neurons of 0.0063 days (9.07 minutes) that is 26.0% greater than the mean error of DNN-5 with 70 neurons. Table 5 details mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

For a network containing 10 neurons, the most optimal performance is exhibited by DNN-2 with a mean error of 0.4062 days (584.93 minutes). For 20 and 30 neurons, DNN-2 and DNN-3 have mean errors of 0.3495 days (503.28 minutes) and 0.1162 days (167.33 minutes), respectively. DNN-3 with 90 neurons and 110 neurons exhibit mean errors of 0.0030 days (4.32 minutes) and 0.0032 days (4.61 minutes), respectively.

Table 4
Mean error in days (square root of MSE rounded to 4 decimal points) for 5 different neural network configurations for planar Super-GTO to GEO transfer.

No. of Neurons	DNN-1	DNN-2	DNN-3	DNN-4	DNN-5
10	0.4245	0.4059	0.4249	0.4213	2.7789
20	0.3914	0.2385	0.4208	0.3537	0.3985
30	0.4062	0.3984	0.4145	0.4183	0.2686
40	0.4105	0.3777	0.0097	0.4069	0.2789
50	0.4051	0.0042	0.3117	0.0096	0.0114
70	0.4084	0.4008	0.0028	0.0036	0.0053
90	0.3488	0.0212	0.0196	0.1259	0.2629
110	0.1731	0.1564	0.0036	0.1457	0.0785

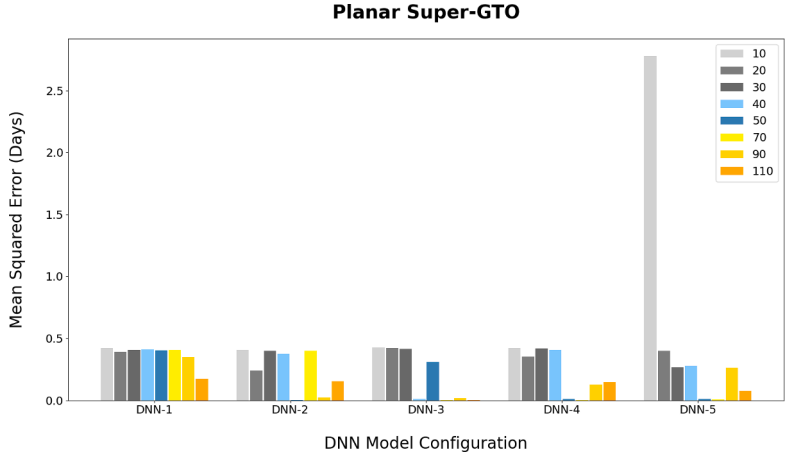

Fig. 9. Mean error in days (square root of MSE) for 5 different neural network configurations for planar Super-GTO to GEO transfer.

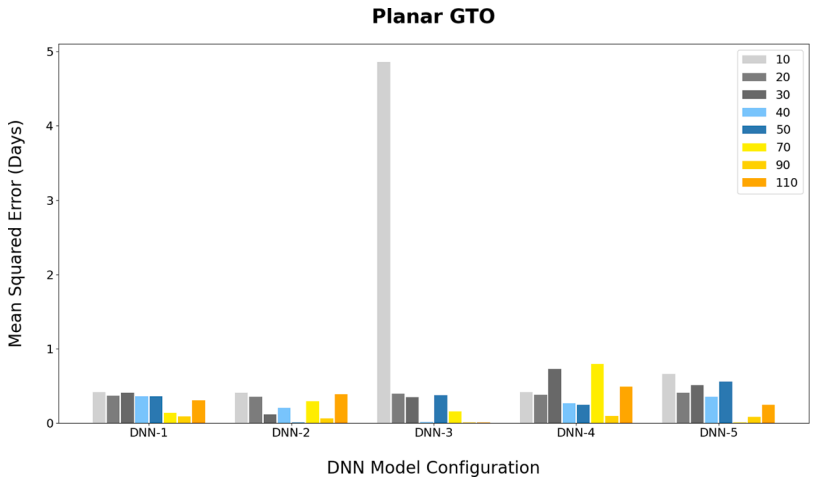

Fig. 10. Mean error in days (square root of MSE) for 5 different neural network configurations for planar GTO to GEO transfer.

Table 5
Mean error in days (square root of MSE rounded to 4 decimal points) for 5 different neural network configurations for planar GTO to GEO transfer.

No. of Neurons	DNN-1	DNN-2	DNN-3	DNN-4	DNN-5
10	0.4098	0.4063	4.8613	0.4163	0.6590
20	0.3625	0.3495	0.3915	0.3811	0.4061
30	0.4030	0.1162	0.3441	0.7234	0.5067
40	0.3619	0.2027	0.0133	0.2655	0.3535
50	0.3607	0.0063	0.3702	0.2411	0.5537
70	0.1373	0.2901	0.1523	0.7932	0.0050
90	0.0900	0.0623	0.0030	0.0928	0.0770
110	0.3030	0.3829	0.0032	0.4858	0.2406

6.6. Planar Sub-GTO to GEO transfer

In our sixth set of experiments, we aim to design a DNN that accurately predicts the spacecraft's transfer time for planar transfer from Sub-GTO to GEO. Fig. 11 depicts mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

The results show that the DNN-4 with 90 neurons achieves the best mean error of 0.0030 days (4.32 minutes). Other DNN configurations that achieve relatively low mean error are DNN-2 with 70 neurons, DNN-5 with 70 neurons, and DNN-2 with 50 neurons. DNN-2 with 70 neurons configuration exhibits mean error of 0.0039 days (5.47 minutes). This represents a 26.66% increase in mean error over DNN-3 with 90 neurons. DNN-5 with 70 neurons has a mean error of 0.0080 days (11.52 minutes) that is closest to the mean errors of DNN-2 with 50 neurons 0.0081 days (11.66 minutes) and DNN-3 with 40 neurons with a mean error of 0.0093 days (13.39 minutes); that are 1.25% and 16.25% greater than the mean error of DNN-5 with 70 neurons. Table 6 details mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

For a network containing 10 neurons, the most optimal performance is exhibited by DNN-1 with a mean error of 0.4740 days (682.56 minutes). For 20 and 30 neurons, DNN-1 and DNN-2 have mean errors of 0.3869 days (557.14 minutes) and 0.1924 days (277.06 minutes), respectively. DNN-4 with 90 neurons and DNN-2 with 110 neurons exhibit mean errors of 0.0030 days (4.32 minutes) and 0.1088 days (156.67 minutes), respectively.

Table 6
Mean error in days (square root of MSE rounded to 4 decimal points) for 5 different neural network configurations for planar Sub-GTO to GEO transfer.

No. of Neurons	DNN-1	DNN-2	DNN-3	DNN-4	DNN-5
10	0.4740	0.4806	0.4976	0.4866	1.6699
20	0.3869	0.5051	0.4739	0.4299	1.4739
30	0.4556	0.1924	0.4264	0.4606	0.4917
40	0.6603	0.4737	0.0093	0.4382	0.5111
50	0.3379	0.0081	0.4331	0.4427	0.0585
70	0.8507	0.0039	0.4107	0.4177	0.0080
90	0.3788	0.7039	0.5346	0.0030	0.6164
110	0.1668	0.1088	1.812	0.5709	0.4390

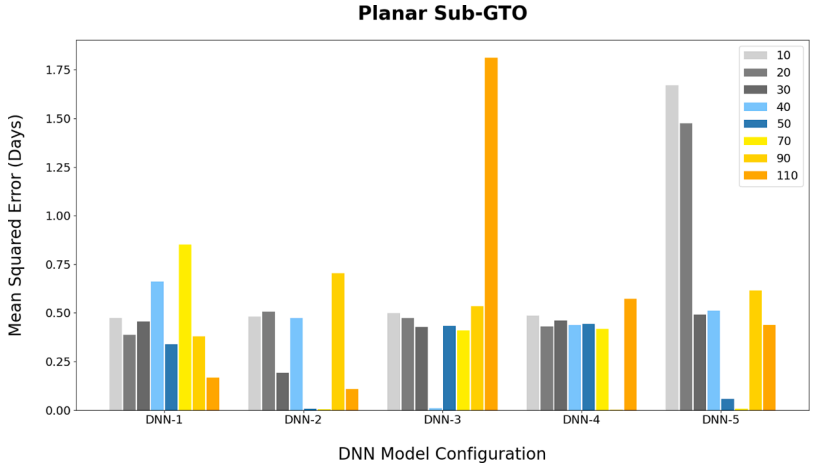

Fig. 11. Mean error in days (square root of MSE) for 5 different neural network configurations for planar Sub-GTO to GEO transfer.

An important point to note while examining the mean error values is that the mean error values for the best-performing DNN configurations are not as huge as they seem to be. The complete trajectory is over several months, ranging to over 240 days. The DNN's have an accuracy of over 99% for the best networks for each of the trajectories shown above. Secondly, over-fitting these neural networks would not be a good idea. These networks are expected to give a direction to the reinforcement learning framework in exploring an unexplored part of space.

6.7. SLTOR optimization vs optimized DNN-Predicted trajectories

Figs. 12–14 provide a comparison of actual versus the predicted time for Super-GTO to GEO DNN-3 with 110 neurons, GTO to GEO DNN-4 with 110 neurons and Sub-GTO to GEO DNN-3 with 110 neurons respectively, in non-planar orbits. Results show that the predicted time follows closely to the optimal time obtained through solutions of optimization equations.

6.8. Computational time of SLTOR optimization vs optimized DNNs

In Table 7, we compare the computational times for our designed DNNs and the optimization approach to determine the transfer times from a given orbit to GEO. Both the SLTOR optimization approach and our designed optimized DNNs are executed on a machine running Windows 11 operating system and equipped with Intel Core i7-11370H processor operating at 3.30 GHz 4 core CPU with 16 GBs of random access memory (RAM). Results indicate that our designed DNNs can predict the transfer time to reach GEO from a given orbit in a fraction of second whereas the optimization approach takes tens of seconds to compute the transfer time. Results indicate that on average our designed DNNs provide an improvement of 1600× in computation time as compared to the tradition optimization approach. These results verify the effectiveness of our DNNs providing the inference of metric of interest (i.e., transfer time here) in real-time, and thus the suitability of our DNNs for Q-value estimation and usage in DRL frameworks in an automated manner.

7. Comparison of optimized DNNs vs contemporary ML algorithms

This section compares the results from our designed DNNs using the Levenberg Marquardt (LM) algorithm with the contemporary Machine Learning (ML) algorithms. The algorithms we provide a comparison with are: Support Vector Machine (SVM), Random Forest (RF), and Decision Tree (DT) for regression. We only provide comparison results of non-planar SuperGTO to GEO transfers to demonstrate the effectiveness of our designed DNN versus contemporary ML algorithms as other mission transfers will follow similar comparison trends.

7.1. Non-planar Super-GTO

In this subsection, first we obtain the results of mean error for transfer time prediction for non-planar Super-GTO to GEO transfers using SVM, RF, and DT. After that, we provide a comparison of our designed DNNs using the LM algorithm with SVM, RF, and DT.

7.1.1. Support vector machine

Table 8 contains the results (mean error in days) for SVM algorithm with values of C and ϵ varying from 0.1 - 0.9 with an increment of 0.2 in each subsequent SVM trained. In total we have 25 configurations of SVMs that we have reported in this paper.

As it can be seen in bold text in the Table 8, the best performance for the mission scenario is coming from the SVM with $C = 0.9$ and $Epsilon = 0.1$ with a mean error of 0.6032 days. The second best performer is SVM with $C = 0.9$ and $Epsilon = 0.3$ with mean error of 0.6039 days that is 0.12% higher than the SVM with $C = 0.9$ and $Epsilon = 0.1$. The third best in performance is SVM with $C = 0.9$ and $Epsilon = 0.5$ with a mean error of 0.6100 days that is 1.01% higher than the second best performing SVM with $C = 0.9$ and $Epsilon = 0.3$.

7.1.2. Random forest

Table 9 contains the results (mean error in days) for Random Forest algorithm with a varying number of estimators from 10 to 110. The first row in the table (header) displays the total number of estimators (#Est) we have used in Random Forest. The second row is the results presented as mean error (ME) in days.

As it can be seen in bold text in the Table 9, the best performance for planar SuperGTO to GEO mission scenario is achieved with the RF algorithm with 30 estimators. The mean error of RF algorithm with 30 estimators is 0.4041 days. The second best performer, is the RF algorithm with 50 estimators with a mean error of 0.4050 days which is 0.22% higher than the mean error of RF algorithm with 30 estimators. The third best performance is achieved with 90 estimators with a mean error of 0.4060 days that is 0.25% higher than the mean error of RF algorithm with 80 estimators.

7.1.3. Decision tree

Table 10 contains the results (error in days) for Decision Tree (DT) algorithm with a varying random state from 0 to 9. The first row in the table (header) displays the random state value (RSV) used to train DT algorithm. The second row is the results presented as mean error (ME) in days.

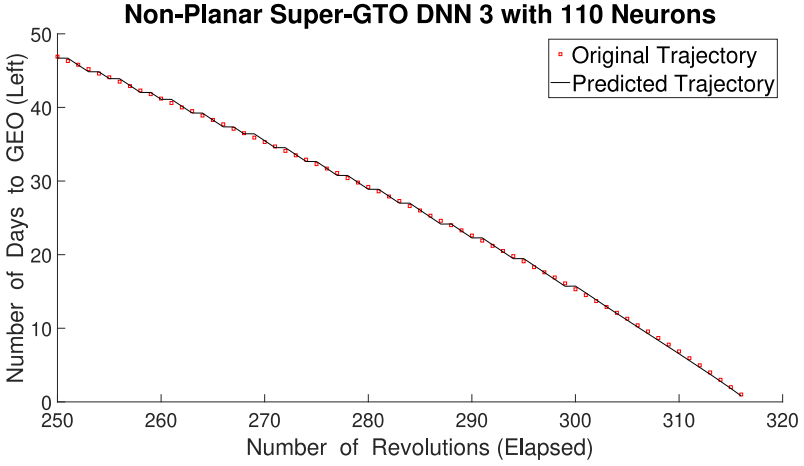

Fig. 12. Comparison of original and predicted transfer time for non-planer Super-GTO to GEO transfer using DNN-3 with 110 neurons.

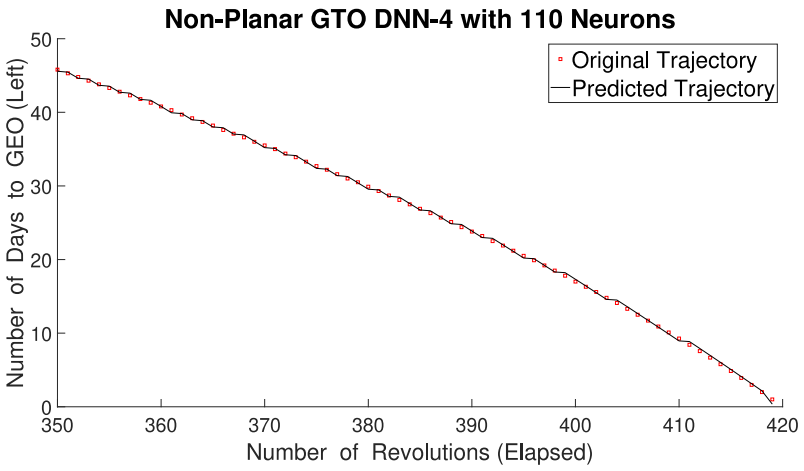

Fig. 13. Comparison of original and predicted transfer time for non-planer GTO to GEO transfer using DNN-4 with 110 neurons.

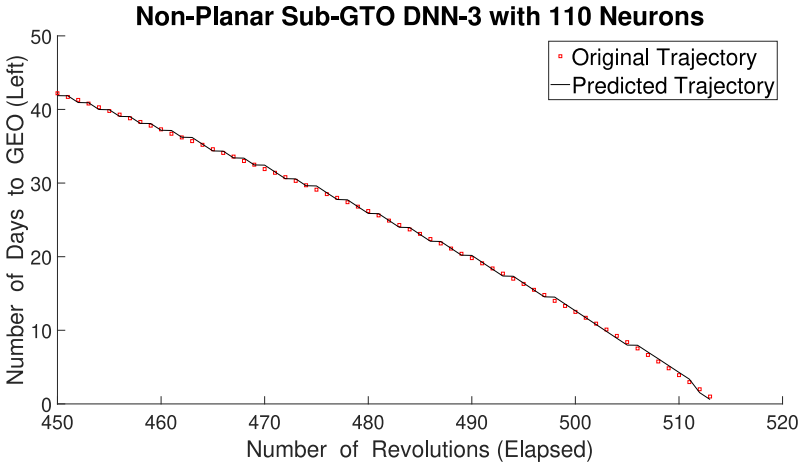

Fig. 14. Comparison of original and predicted transfer time for non-planer Sub-GTO to GEO transfer using DNN-3 with 110 neurons.

Table 7
Comparison of computational times for DNNs and optimization approach to determine the transfer times from given orbit to GEO.

Current Orbit	Optimization approach	DNNs	Otimal DNN configuration	Improvement Afforded by DNNs
Non-planar GTO	26.03 sec	0.016 sec	DNN-4 with 110 neurons	1626.88 ×
Planar-GTO	15.48 sec	0.009 sec	DNN-3 wih 90 neurons	1720.00 ×
Non-Planar Sub-GTO	35.51 sec	0.016 sec	DNN-3 with 110 neurons	2219.38 ×
Planar Sub-GTO	18.08 sec	0.010 sec	DNN-4 with 90 neurons	1808.00 ×
Non-planar Super-GTO	20.50 sec	0.014 sec	DNN-2 with 110 neurons	1464.28 ×
Planar Super-GTO	11.43 sec	0.015 sec	DNN-3 with 110 neurons	762.00 ×
Average	21.17 sec	0.013 sec	– –	1600.1 ×

Table 8
Mean error in days (square root of MSE rounded to 4 decimal points) for each subsequent configuration for planar Super-GTO to GEO transfer.

ϵ	0.1	0.3	0.5	0.7	0.9
C					
0.1	0.9338	0.9284	0.9531	0.9909	1.0166
0.3	0.7065	0.7157	0.7319	0.7577	0.7937
0.5	0.6417	0.6494	0.6668	0.6905	0.7152
0.7	0.6179	0.6274	0.6410	0.6592	0.6849
0.9	**0.6032**	0.6039	0.6100	0.6345	0.6713

Table 9
Mean error in days (square root of MSE rounded to 4 decimal points) for 11 different configurations of support vector machine (SVM) from 10 to 110 estimators (#Est) with an increment of 10 estimators in each subsequent configuration for non-planar Super-GTO to GEO transfer.

#Est	10	20	30	40	50	60
ME	0.4078	0.4082	**0.4041**	0.4070	0.4050	0.4072
#Est	**70**	**80**	**90**	**100**	**110**	
ME	0.4065	0.4065	0.4060	0.4066	0.4093	

Table 10
Mean error in days (square root of MSE rounded to 4 decimal points) for 10 different configurations of DT algorithm from 0 to 9 RSV for non-planar Super-GTO to GEO transfer.

RSV	0	1	2	3	4
ME	0.6139	**0.6133**	0.6182	0.6183	0.6153
RSV	**5**	**6**	**7**	**8**	**9**
ME	0.6161	0.6177	0.6141	0.6196	0.6174

As it can be seen in bold text in the Table 10, the best performance for non-planar SuperGTO to GEO mission scenario is achieved with the DT algorithm with random state = 0. The mean error of DT algorithm with random state = 1 is 0.6133 days. The second best performer, is the DT algorithm with random state = 0 with a mean error of 0.6139 days which is 0.10% higher than the mean error of DT algorithm with random state = 1. The third best performance is achieved with random state = 7 with a mean error of 0.6141 days that is 0.03% higher than the mean error of RF algorithm with random state = 0.

7.1.4. Comparison of optimized DNNs versus SVM, RF, and DT

Table 11 compares the mean error for our best-performing DNNs trained using LM optimizer versus SVM, RF and DT algorithms. DNN-4 with 110 neurons is the best performing algorithm with a ME of 0.0435. RF with 30 estimators is the second best performer with a ME of 0.4041 that is 9.29× better than the ME of DNN. SVM with C = 0.9 and Epsilon = 0.1 has a ME of 0.6032 that is 13.87× higher than the ME of the best performing DNN. DT with random state = 1 has a ME of 0.6113 that is 14.05× higher than the ME of the best performing DNN.

7.2. Planar Super-GTO

In this subsection, first we obtain the results of mean error for transfer time prediction for planar SuperGTO to GEO transfers using SVM, RF, and DT. After that, we provide a comparison of our designed DNNs using the LM algorithm with SVM, RF, and DT is provided.

7.2.1. Support vector machine

Table 12 contains the results (mean error in days) for SVM algorithm with values of C and ϵ varying from 0.1 - 0.9 with an increment of 0.2 in each subsequent SVM trained. In total we have 25 configurations of SVMs that we have reported in this paper.

As it can be seen in bold text in the Table 12, the best performance for the mission scenario is coming from the SVM with $C = 0.9$ and *Epsilon* = 0.1 with a mean error of 0.7115 days. The second best performer is SVM with $C = 0.9$ and *Epsilon* = 0.5 with mean error of 0.7183 days that is 0.96% higher than the SVM with $C = 0.9$ and *Epsilon* = 0.1. The third best in performance is SVM with $C = 0.9$ and *Epsilon* = 0.3 with a mean error of 0.7305 days that is 1.70% higher than the second best performing SVM with $C = 0.9$ and *Epsilon* = 0.5.

7.2.2. Random forest

Table 13 contains the results (error in days) for Random Forest algorithm with a varying number of estimators from 10 to 110. The first row in the table (header) displays the total number of estimators (#Est) we have used in Random Forest. The second row is the results presented as mean error (ME) in days.

As it can be seen in bold text in the Table 13, the best performance for planar SuperGTO to GEO mission scenario is achieved with the RF algorithm with 70 estimators. The mean error of RF algorithm with 70 estimators is 0.5022 days. The second best performer, is the RF algorithm with 80 estimators with a mean error of 0.5035 days which is 0.25% higher than the mean error of RF algorithm with 70 estimators. The third best performance is achieved with 40 estimators with a mean error of 0.5065 days that is 0.59% higher than the mean error of RF algorithm with 80 estimators.

7.2.3. Decision tree

Table 14 contains the results (error in days) for Decision Tree (DT) algorithm with a varying random state from 0 to 9. The first row in the table (header) displays the random state value (RSV) used to train DT algorithm. The second row is the results presented as mean error (ME) in days.

As it can be seen in bold text in the Table 14, the best performance for planar Super-GTO to GEO mission scenario is achieved with the DT algorithm with random state = 0. The mean error of DT algorithm with random state = 0 is 0.6074 days. The second best performer, is the DT algorithm with random state = 9 with a mean error of 0.6131 days which is 0.94% higher than the mean error of DT algorithm with random state = 0. The third best performance is achieved with random state = 4 with a mean error of 0.6134 days that is 0.05% higher than the mean error of RF algorithm with random state = 9.

7.2.4. Comparison of optimized DNNs versus SVM, RF, and DT

Table 15 compares the mean error for best DNNs trained using LM optimizer, SVM, RF and DT algorithms.

DNN-3 with 70 neurons is the best performing algorithm with ME 0.0028. RF with 70 estimators is the second best performer with a ME of 0.5022 that is 179.39× better than ME of DNN. DT with random state = 0 has a ME of 0.6074 that is 216.92× higher than ME of the best per-

Table 11
Comparison of mean error (actual values and times improvement) for the best DNN trained using LM optimizer, SVM, RF and DT for non-planar SuperGTO to GEO transfer. $\mathscr{I}$ denotes the improvement provided by our best-performing DNN.

Algorithm	DNN	RF	SVM	DT
ME in Days	**0.0435**	0.4041	0.6032	0.6113
$\mathscr{I}$	**N/A**	9.29×	13.87×	14.05×

Table 12
Mean error in days (square root of MSE rounded to 4 decimal points) for each subsequent configuration for planar Super-GTO to GEO transfer.

ϵ	0.1	0.3	0.5	0.7	0.9
C					
0.1	2.1524	2.2899	2.3458	2.2086	2.2637
0.3	1.1975	1.1996	1.2167	1.2181	1.2990
0.5	0.9009	0.8971	0.9450	0.9462	0.9854
0.7	0.7912	0.7891	0.7916	0.8567	0.8979
0.9	**0.7115**	0.7305	0.7183	0.7602	0.8097

Table 13
Mean error in days (square root of MSE rounded to 4 decimal points) for 11 different configurations of RF algorithm from 10 to 110 estimators (#Est) with an increment of 10 estimators in each subsequent configuration for planar Super-GTO to GEO transfer.

#Est	10	20	30	40	50	60
ME	0.5147	0.5086	0.5205	0.5065	0.5131	0.5141
#Est	**70**	**80**	**90**	**100**	**110**	
ME	**0.5022**	0.5035	0.5172	0.5112	0.5158	

Table 14
Mean error in days (square root of MSE rounded to 4 decimal points) for 10 different configurations of DT algorithm from 0 to 9 RSV for planar Super-GTO to GEO transfer.

RSV	0	1	2	3	4
ME	**0.6074**	0.6168	0.6260	0.6184	0.6134
RSV	**5**	**6**	**7**	**8**	**9**
ME	0.6157	0.6191	0.6249	0.6254	0.6131

Table 15
Comparison of mean error (actual values and times improvement) for the best DNN trained using LM optimizer, SVM, RF and DT for planar Super-GTO to GEO transfer. $\mathscr{I}$ denotes the improvement provided by our best-performing DNN.

Algorithm	DNN	RF	DT	SVM
ME	**0.0028**	0.5022	0.6074	0.7115
$\mathscr{I}$	**N/A**	179.39 ×	216.92 ×	254.11 ×

forming DNN. SVM with C = 0.9 and Epsilon = 0.1 has a ME of 0.7112 that is 254.11× higher than the ME of the best performing DNN.

8. Conclusions

In recent years, there has been a surge in use of electric propulsion to transfer satellites to the geostationary Earth orbit (GEO). Traditionally, the transfer times to reach GEO using all-electric propulsion are obtained by solving challenging trajectory optimization problems, whose solution rely on numerical schemes that are not only computationally intensive, but also lack automated implementation capabilities, which is an impediment to their incorporation within a deep reinforcement learning (DRL) framework to solve trajectory planning problems in near real-time. This work designs and evaluates a machine learning (ML) framework, focusing on deep neural networks (DNNs), to predict the transfer time in near real-time to assist in Q-value determination instead of solving traditional sequential low-thrust orbit-raising (SLTOR) optimization problems. This paper investigates different architectures for DNNs to determine a suitable setting of hyperparameters of DNNs for transfer time prediction for six orbit-raising trajectories involving transfers from planar and non-planar geostationary transfer orbit (GTO), Sub-GTO, and Super-GTO to geostationary Earth orbit (GEO). Since spacecraft orbit-raising trajectory problem is a continuous-space problem, our designed DNNs can both interpolate and extrapolate to states in the space not present in the training set.

Experimental results indicate that our designed DNNs can predict the transfer time for different orbit-raising scenarios with an accuracy of over 99.97% with respect to solving an SLTOR optimization problem. Comparison of prediction time of our designed DNNs with the SLTOR optimization problem reveals that our designed DNNs provide an improvement in computation time of 1600 × , on average, over the SLTOR optimization, thus verifying the suitability of our designed DNNs to be used in DRL frameworks in an automated manner. To verify the efficacy of our designed DNNs for predicting transfer time that is required for Q-value estimation, we have also compared the results from our designed DNNs using the Levenberg Marquardt (LM) algorithm with the contemporary ML algorithms, such as support vector machines (SVM), random forests (RFs), and decision trees (DTs) for regression. Experimental results indicate that our best-performing DNN provides an improvement of mean error of 9.29 × , 13.87 × , and 14.05× over RF, DT, and SVM, respectively, for non-planar Super-GTO to GEO transfers. Experimental results indicate that our best-performing DNN provides an improvement of mean error of 179.39 × , 216.92 × , and 254.11× over RF, DT, and SVM, respectively, for planar Super-GTO to GEO transfers.

In our future work, we plan to design DNNs for predicting other metrics of interest (such as fuel expenditure) from a given orbit or state for different planar and non-planar GTO, Super-GTO, and Sub-GTO orbits to GEO. We further plan to use our designed DNNs in a DRL framework for predicting the transfer time and assisting in Q-value determining for determining an optimal trajectory for transfer from different planar and non-planar GTO, Super-GTO, and Sub-GTO orbits to GEO.

CRediT authorship contribution statement

Ali Hassaan Mughal: Software, Validation, Formal analysis, Writing – original draft, Visualization. **Pardhasai Chadalavada:** Software, Validation, Data curation, Writing – original draft, Writing – review & editing. **Arslan Munir:** Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Atri Dutta:** Conceptualization, Methodology, Investigation, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition, Project administration. **Mahmood Azhar Qureshi:** Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by the National Aeronautics and Space Administration (NASA), through the grant NASA-20-2020EPSCoR-0017. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NASA.

References

Arora, L. (2020). *Reinforcement learning framework for spacecraft low-thrust orbit raising*.
Arora, L., & Dutta, A. (2020). *Reinforcement learning for sequential low-thrust orbit raising problem* (p. 2186).
Betts, J. T. (1998). Survey of numerical methods for trajectory optimization. *21*(2), 193–207.
Betts, J. T. (2000). Very low-thrust trajectory optimization using a direct sqp method. *120*(1–2), 27–40.
Broida, J., & Linares, R. (2019). Spacecraft rendezvous guidance in cluttered environments via reinforcement learning. *29th AAS/AIAA Space Flight Mechanics Meeting*, 1–15.
Caruana, R., & Niculescu-Mizil, A. (2006). *An empirical comparison of supervised learning algorithms* (pp. 161–168).
Cass, J. M. (1983). *Journal of the Royal Statistical Society. Series C (Applied Statistics), 32* (1), 94–95.
Chadalavada, P., Dutta, A., & Ghosh, P. (2022). An efficient algorithm for the longitude-targeted ascent of all-electric satellites. *Aiaa scitech 2022 forum* (p. 2473).
Chadalavada, P., Farabi, T., & Dutta, A. (2020). Sequential low-thrust orbit-raising of all-electric satellites. *Aerospace, 7*(6), 74.
Cheng, L., Jiang, F., Wang, Z., & Li, J. (2021). Multiconstrained real-time entry guidance using deep neural networks. *IEEE Transactions on Aerospace and Electronic Systems, 57* (1), 325–340.
De Pascale, P., & Vasile, M. (2006). Preliminary design of low-thrust multiple gravity-assist trajectories. *43*(5), 1065–1076.
Dutta, A., & Arora, L. (2019). *Objective function weight selection for sequential low-thrust orbit-raising optimization problem* (pp. 13–17).
Gaudet, B., Drozd, K., & Furfaro, R. (2022). Adaptive approach phase guidance for a hypersonic glider via reinforcement meta learning. *Aiaa scitech 2022 forum* (p. 2214).San Diego
Gavin, H. P. (2020). *The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems*. Duke University Press.
Graham, K. F., & Rao, A. V. (2016). Minimum-time trajectory optimization of low-thrust earth-orbit transfers with eclipsing. *53*(2), 289–303.
Haj-Ali, A., Huang, Q. J., Xiang, J., Moses, W., Asanovic, K., Wawrzynek, J., & Stoica, I. (2020). Autophase: Juggling hls phase orderings in random forests with deep reinforcement learning. In I. Dhillon, D. Papailiopoulos, & V. Sze (Eds.), *vol. 2. Proceedings of machine learning and systems* (pp. 70–81).
He, S., Shin, H.-S., & Tsourdos, A. (2021). Computational missile guidance: A deep reinforcement learning approach. *Journal of Aerospace Information Systems, 18*(8), 571–582.
Herman, A. L., & Conway, B. A. (1996). Direct optimization using collocation based on highorder gauss-lobatto quadrature rules. *19*(3), 592–599.
Hovell, K., & Ulrich, S. (2021). Deep reinforcement learning for spacecraft proximity operations guidance. *Journal of Spacecraft and Rockets, 58*, 254–264.
Iiyama, K., Tomita, K., Jagatia, B. A., Nakagawa, T., & Ho, K. (2021). Deep reinforcement learning for safe landing site selection with concurrent consideration of divert maneuvers. *arXiv preprint arXiv:2102.12432*.
Junkins, J. L., & Taheri, E. (2018). Exploration of alternative state vector choices for low-thrust trajectory optimization. *42*(1), 47–64.
Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio, & Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR*.
Kluever, C. (1998a). *Simple control laws for low-thrust orbit transfers (aas 98–203). (vol.99* pp. 1455–1468)). UNIVELT INC.

Kluever, C. A. (1998b). Simple guidance scheme for low-thrust orbit transfers. *21*(6), 1015–1017.
Kluever, C. A., & Oleson, S. R. (1998). Direct approach for computing near-optimal lowthrust earth-orbit transfers. *Journal of Spacecraft and Rockets, 35*(4), 509–515.
Kolosa, D. S. (2019). *A reinforcement learning approach to spacecraft trajectory optimization* (p. 3542)). Western Michigan University Press.
Kwon, H., Oghim, S., & Bang, H. (2021). Autonomous guidance for multi-revolution low-thrust orbit transfer via reinforcement learning. *AAS 21–315*.
LaFarge, N. B. (2020). *Autonomous guidance for multi-body orbit transfers using reinforcement learning*. Purdue University Graduate School.
Li, B., peng Yang, Z., qing Chen, D., yang Liang, S., & Ma, H. (2021). Maneuvering target tracking of uav based on mn-ddpg and transfer learning. *Defence Technology, 17*(2), 457–466.
Marasch, M. W., & Hall, C. D. (2000). Application of energy storage to solar electric propulsion orbital transfer. *37*(5), 645–652.
Marecki, J., Koenig, S., & Tambe, M. ( ). A fast analytical algorithm for solving markov decision processes with real-valued resources. IJCAI, 2017,.
Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *11*(2), 431–441.
Marriott, F. H. C. (1985). *Biometrics, 41*(2).596–596
Novak, D. M., & Vasile, M. (2011). Improved shaping approach to the preliminary design of low-thrust trajectories. *34*(1), 128–147.
Oestreich, C. E., Linares, R., & Gondhalekar, R. (2021). Autonomous six-degree-of-freedom spacecraft docking with rotating targets via reinforcement learning. *Journal of Aerospace Information Systems*, 1–12.
Petropoulos, A. (2004). *Low-thrust orbit transfers using candidate Lyapunov functions with a mechanism for coasting* (p. 5089)).
Petropoulos, A. E., & Longuski, J. M. (2004). Shape-based algorithm for the automated design of low-thrust, gravity assist trajectories. *41*(5), 787–796.
Sackett, L. L., Malchow, H. L., & Delbaum, T. N. (1975). *Solar electric geocentric transfer with attitude constraints: analysis. Final technical report*.
Shirobokov, M. G., Trofimov, S. P., & Ovchinnikov, M. (2021). Survey of machine learning techniques in spacecraft control design. *Acta Astronautica, 186*, 87–97.
Sreesawet, S., & Dutta, A. (2018). Fast and robust computation of low-thrust orbit-raising trajectories. *Journal of Guidance, Control, and Dynamics, 41*(9), 1888–1905.
Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd). The MIT Press.
Taheri, E., & Abdelkhalik, O. (2012). Shape based approximation of constrained low-thrust space trajectories using fourier series. *49*(3), 535–546.
Vasile, D. P. P. M., & Casotto, S. (2007). On the optimality of a shape-based approach based on pseudo-equinoctial elements. *61*(1–6), 286–297.
Wall, B. J., & Conway, B. A. (2009). Shape-based approach to low-thrust rendezvous trajectory design. *32*(5), 95–101.
Wang, X., Shi, P., Wen, C., & Zhao, Y. (2020). *An algorithm of reinforcement learning for maneuvering parameter self-tuning applying in satellite cluster* (vol. 2020). Hindawi.
Wang, Z., & Grant, M. J. (2017). Constrained trajectory optimization for planetary entry via sequential convex programming. *Journal of Guidance, Control, and Dynamics, 40*(10), 2603–2615.
Wang, Z., & Grant, M. J. (2018). Optimization of minimum-time low-thrust transfers using convex programming. *Journal of Spacecraft and Rockets, 55*(3), 586–598.
Willis, S., Izzo, D., & Hennes, D. (2016). Reinforcement learning for spacecraft maneuvering near small bodies. *AAS/AIAA Space Flight Mechanics Meeting*, 14–18.
Yuexuan, A., Shifei, D., Shi, S., & Li, J. (2018). Discrete space reinforcement learning algorithm based on support vector machine classification. *Pattern Recognition Letters, 111*, 30–35.
Zhao, Z., Sun, M., & Ma, X. (2021). Meta-Reinforcement Learning for Mastering Multiple Skills and Generalizing across Environments in Text-based Games. *Meta learning for nlp workshop at annual meeting of the association for computational linguistics (acl)*. Bangkok, Thailand