IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Neural Networks in Time-Optimal Low-Thrust Interplanetary Transfers

HAIYANG LI[1,2], HEXI BAOYIN[1], AND FRANCESCO TOPPUTO[2]
[1]School of Aerospace Engineering, Tsinghua University, Beijing, 100084, China
[2]Department of Aerospace Science and Technology, Politecnico di Milano, Milan, 20156, Italy

Corresponding author: Hexi Baoyin (e-mail: baoyin@tsinghua.edu.cn).

Part of this research is supported by the Chinese National Natural Science Fund for Distinguished Young Scientists of China (No. 11525208). The first author would like to acknowledge the financial support provided by the China Scholarship Council.

ABSTRACT In this paper, neural networks are trained to learn the optimal time, the initial costates, and the optimal control law of time-optimal low-thrust interplanetary trajectories. The aim is to overcome the difficult selection of first guess costates in indirect optimization, which limits their implementation in global optimization and prevents on-board applications. After generating a dataset, three networks that predict the optimal time, the initial costate, and the optimal control law are trained. A performance assessment shows that neural networks are able to predict the optimal time and initial costate accurately, especially a 100% success rate is achieved when neural networks are used to initialize the shooting function of indirtect methods. Moreover, learning the state-control pairs shows that neural networks can be utilized in real-time, on-board optimal control.

INDEX TERMS indirect methods, low-thrust trajectory optimization, initial costates, neural networks

I. INTRODUCTION

SOLAR electric propulsion (SEP) is an ideal option for interplanetary missions, because of its high specific impulse and thus fuel saving capability. Since the successful test by Deep Space 1 [1], many interplanetary missions such as Hayabusa 1 and 2, Dawn, and BepiColombo [2] have used SEP as their primary propulsion system. However, the continuous action of SEP requires formulating the low-thrust trajectory design optimal control problems.

Methods for optimizing low-thrust trajectories are generally categorized as indirect methods, direct methods, and the hybridization of the two [3], [4], and indirect methods are utilized a lot because of its optimality. Indirect methods apply Pontryagin's maximum principle and formulate the original optimal control problem as a two-point boundary-value problem (TPBVP). The TPBVP is generally solved using shooting methods in which initial guesses of shooting variables are required. An accurate guess of these shooting variables can speed up the shooting process. However, due to the non-intuitive nature of the costates, which are also shooting variables, it is hard to provide accurate initial guesses. Therefore, the TPBVP may not converge easily. Some techniques are proposed to improving the convergence, such as homotopic approach [5]–[7], switching detection method [8], [9], analytic derivatives [10], [11], and expressing orbital states in modified equinoctial elements [12], [13]. Methods concerning the guess of initial costates include initial costates normalization [5], [14], costate transformation [15], providing initial guesses by solving simplified linear equations [16], using Particle Swarm Optimization [17], shape-based methods [18], pseudospectral methods [19], and k-nearest neighbor methods [20], just to mention few.

Even though indirect methods perform well in optimizing low-thrust trajectories, the optimization process is time-consuming, which involves two issues: global mission design and real-time on-board implementation. For global mission trades, especially in multi-target mission design, a huge number of trajectories needs to be optimized. However, sometimes only the optimum estimate is needed in preliminary stages, and thus a fast estimation is useful to reduce the computational time [21]–[23]. For real-time onboard implementation, due to the limitation of onboard computing resources, it is difficult to ensure the convergence and optimality of a real-time scheme.

In recent years, with the rapid development of Artificial Intelligence, neural networks (NNs) have attracted the interest of many researchers and have had applications in various fields, including astrodynamics [24]. NNs consist of connected layers that are composed of neurons and are able to exhibit desired behavior after learning [25]. In astrodynam-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2946657, IEEE Access

ics applications, NNs are trained mainly as predictors and optimal controllers. As predictors, NNs are trained to learn the optimal time and fuel of low-thrust transfers [26]–[28]. As controllers, NNs are trained to learn the optimal state-control pairs [29]–[32] or image-control relations [33]. In essence, it has been shown that NNs have good performances as predictors and optimal controllers.

In this paper, NNs are utilized as both predictors and optimal controllers in indirect optimization of low-thrust trajectories, that is, they are trained to learn the optimal time, the initial costate, as well as the optimal control of time-optimal problems (TOP). Costates are learned in [32] but they are not used to initialize the shooting function of indirect methods. Compared to the techniques mentioned above, predicting the optimal time and the initial costate can provide an accurate initial guess for the shooting in a simple and fast way. We show that NNs can predict the optimal time and initial costate accurately, so improving the efficiency in shooting convergence, and thus paving the way to real-time, on-board controller.

This paper is organized as follows: In Section 2, the indirect method for time-optimal low-thrust trajectory optimization as well as the dataset generation procedure are presented. In Section 3, architecture of NNs and random search of hyper-parameters are introduced. In Section 4, the evaluation methods of NN performance are detailed. In Section 5, numerical examples of two type of missions, the Earth to NEA and the Earth to Mars, are given. Conclusions are drawn in Section 5.

II. DATASET GENERATION OF TIME-OPTIMAL LOW-THRUST TRAJECTORIES

A. LOW-THRUST TRAJECTORY OPTIMIZATION

The orbital states are expressed in modified equinoctial elements (MEE; p, e_x, e_y, h_x, h_y, L) because they are non-singular, efficient, and robust. The relationship between MEE and classical orbital elements can be expressed as:

$$\begin{aligned} p &= a(1-e^2) \\ e_x &= e\cos(\omega+\Omega) \\ e_y &= e\sin(\omega+\Omega) \\ h_x &= \tan(i/2)\cos(\Omega) \\ h_y &= \tan(i/2)\sin(\Omega) \\ L &= \omega+\Omega+\theta \end{aligned} \tag{1}$$

where a is the semi-major axis, e the eccentricity, i the inclination, Ω the right ascension of the ascending node, ω the perigee anomaly, and f the true anomaly.

The dynamic equations of low-thrust propelled spacecraft in a two-body model are:

$$\begin{aligned} \dot{x} &= u\frac{T_{\max}}{m}M\boldsymbol{\alpha}+D, \\ \dot{m} &= -u\frac{T_{\max}}{I_{sp}g_0} \end{aligned} \tag{2}$$

where $x = [p, e_x, e_y, h_x, h_y, L]$, m is the instantaneous mass of the spacecraft, g_0 the standard value of gravitational acceleration, $T_{\max}$ the maximum achievable thrust magnitude, and I_{sp} the thruster specific impulse. M is a transformation matrix and D the gravity vector; their expression can be found in [13]. $u \in [0,1]$ is the engine thrust ratio, and $\boldsymbol{\alpha}$ is the thrust unit vector of the thrust

$$\boldsymbol{\alpha} = [\cos\beta\cos\alpha, \cos\beta\sin\alpha, \sin\beta]^\top \tag{3}$$

where α is the azimuth angle and β the elevation angle.

In TOP, the performance indexes take the forms:

$$J = \int_{t_0}^{t_f} \mathrm{d}t \tag{4}$$

Introducing the co-state vector $\boldsymbol{\lambda}(t)= [\boldsymbol{\lambda_x}, \lambda_m]$, the Hamiltonian is

$$H = u\frac{T_{\max}}{m}\boldsymbol{\lambda}_x^\top M\boldsymbol{\alpha} + \boldsymbol{\lambda}_x^\top D - u\frac{T_{\max}\lambda_m}{I_{sp}\,g_0} + 1 \tag{5}$$

and the dynamics of $\boldsymbol{\lambda}$ is given by

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \boldsymbol{x}} \tag{6}$$

According to Pontryagin's minimum principle, the direction of the optimal thrust is determined as

$$\boldsymbol{\alpha} = -\frac{M^\top\boldsymbol{\lambda_x}}{\|M^\top\boldsymbol{\lambda_x}\|} \tag{7}$$

while the optimal thrust magnitude is

$$\begin{aligned} u &= \begin{cases} 0, & if\ \rho > 0, \\ 1, & if\ \rho < 0, \\ [0,1], & if\ \rho = 0, \end{cases} \\ \rho &= -\frac{I_{sp}g_0\|M^\top\boldsymbol{\lambda_x}\|}{m} - \lambda_m \end{aligned} \tag{8}$$

Given the boundary condition, the optimal control problem can be transformed into a TPBVP and it can be solved by shooting methods solving shooting equations. The state variables of the spacecraft must satisfy the following boundary conditions:

$$x(t_0) = x_0, \quad m(t_0) = m_0, \quad x(t_f) = x_f. \tag{9}$$

The problem is to find $\{\boldsymbol{\lambda}_0, t_f\}$ such that [9], [12]:

$$\Phi(\boldsymbol{\lambda}_0, t_f) := \begin{bmatrix} x(t_f) - x_f \\ \lambda_m(t_f) \\ H(t_f) - \boldsymbol{\lambda_x}(t_f)\cdot\dot{x}_f \end{bmatrix} = 0 \tag{10}$$

We notice that $\lambda_m > 0$ since $\dot{\lambda}_m = -\partial H/\partial m < 0$ and $\lambda_m(t_f) = 0$. Therefore, ρ in Eq.(8) is always negative, and thus $u = 1$ for all times in TOP.

B. DATASET GENERATION

Given a departure celestial body (the Earth) and a target celestial body, a launch date (expressed in Modified Julian Date) mjd_1 is searched using Particle Swarm Optimization (PSO). In PSO, the transfer model is approximated by the two-impulse Lambert model and the performance index to be minimized is the total Δv. After finding the launch date mjd_1, a launch window set $\mathscr{L}$=$\{(x_i,\text{mjd}_i),i=1{:}n\}$, where n represents the length of the set, is created by increasing the launch date using a step of 1 day and calculating the corresponding initial states x_i.

Then the TOP of each element in $\mathscr{L}$ are optimized using the indirect method in Section II-A. The initial guess of the shooting variables, $\boldsymbol{\lambda}_0$ in Eq.(10), will affect the convergence. Therefore, the initial guess of the first element (x_1,mjd_1) is given randomly, while the following element $(x_{i+1},\text{mjd}_{i+1})$ takes the solution $\mathbf{S}_i$ of the previous element (x_i,mjd_i) as initial guess. The the launch window set is to incorporate the solution at each step: $\mathscr{L}$=$\{(x_i,\mathbf{S}_i,\text{mjd}_i),i=1{:}n\}$

The dataset is generated based on the launch window set $\mathscr{L}$. To generate a data, at first an element in $\mathscr{L}$ is picked randomly, denoted as $(x_r,\mathbf{S}_r,\text{mjd}_r)$. Then the initial state x_{data} is obtained by perturbing x_r:

$$x_{data} = x_r + x_r \circ c\boldsymbol{\gamma} \tag{11}$$

where c=0.02, $\boldsymbol{\gamma}$ is a random vector of components between -1 to 1, and $\circ$ is the hadamard product. Then the solution $\mathbf{S}_{data}$ of the TOP with x_{data} is solved using $\mathbf{S}_r$ as initial guess. At last, the pair $(x_{data},\mathbf{S}_{data})$ is stored to the dataset. The procedure for dataset generation is shown in algorithm 1 and illustrated in Fig.1. In this work, for one dataset, 10,000 pieces of data are generated as training dataset, 1,000 pieces of data are generated as validation dataset, and 1,000 pieces of data are generated as test dataset. The training dataset is used to train the network. The validation dataset is used to stop training and prevent overfitting in the DNN model selection process. The test dataset is finally used to evaluate the performance of selected model. The validation dataset and test dataset should be two datasets when a model-selection process is considered. The optimal trajectories are found by using Low-Thrust Trajectory Optimizer (LT2O), a tool developed at Politecnico di Milano [34].

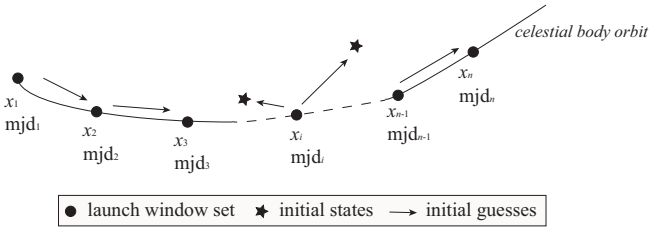

FIGURE 1: Illustration of dataset generation

Algorithm 1 Dataset generation

search for mjd_1 (Modified Julian Date) using PSO;
create launch window set $\{(x_i,\text{mjd}_i),i=1{:}n\}$ using mjd_{i+1}=mjd_i+1;
solve S_1 = TOP(x_1,initialguess=random);
for i=2:n **do**
  solve S_i = TOP(x_i,initialguess=S_{i-1})
end for
while **do**
  select one pair from $\{(x_i,S_i),i=1{:}n\}$ randomly
  perturb the initial state x = x_i+$x_i \circ c\gamma$;
  solve S = TOP(x,initialguess=S_i);
  store (x,S) to dataset
end while

III. NEURAL NETWORK DESIGN

Three NNs, an optimal time prediction network N_{t_f} that predicts the duration of the optimal control problem, an initial costate prediction network $N_{\boldsymbol{\lambda}_0}$ that predicts the initial costate of the indirect optimization method, and an optimal control prediction network $N_{\boldsymbol{\alpha}}$ that predicts the real-time optimal control, are designed.

A. ARCHITECTURE OF NEURAL NETWORK

The NN considered in this paper is essentially a feed-forward neural network with multiple hidden layers. The structure is determined by the number of layers n_{layer} and the number of neurons n_{neuron} at each hidden layer. At each layer, denoting the input l_i, the output l_{i+1} is calculated as follows:

$$l_{i+1} = f(wl_i + b) \tag{12}$$

where w is the weight matrix, b the bias vector, and f a nonlinear function named activation function. There are three most commonly used activation functions expressed in Eq.(13) for the hidden layers: the sigmoid function (sig), the hyperbolic tangent (tanh) function, and the rectified linear (relu) function.

$$\begin{aligned} f_{sig}(x) &= \frac{1}{1+e^{-x}}, \\ f_{tanh}(x) &= \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}, \\ f_{relu}(x) &= \max(0,x) \end{aligned} \tag{13}$$

The training is to adjust the value of the parameters of each layer to minimize the loss function, which is expressed in the form of mean squared error (MSE):

$$E = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}-y)^2, \tag{14}$$

where y is the actual value, $\hat{y}$ the estimated value, and n the number of data in the training iteration. Gradient descent (GD) algorithms are state-of-art in training the parameters, e.g.,

$$w' = w - \eta\frac{\partial E}{\partial w}, \tag{15}$$

where η is the learning rate. Some modified gradient descent algorithms are also very effective, such as momentum gradi-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2946657, IEEE Access

ent descent (MGD) [35] and Adam gradient descent (AGD) [36].

As shown in Eq.(15) the learning rate η is a important factor that affect the training. With a small learning rate, the NN may take a very long time to converge, while with a large learning rate the NN may oscillate around the optimal solution. Therefore, it is a good strategy to start from a large learning rate and decrease the learning rate gradually. Two decay schedules are applied in this paper, exponential decay (ED) and natural exponential decay (NED):

$$\begin{aligned} \eta'_{ED} &= \eta c^t, \\ \eta'_{NED} &= \eta e^{-ct}, \end{aligned} \tag{16}$$

where c is the decay rate and t is the step

The initialization of weights is also a factor that can affect the training. Two initialization schemes are used: Fan_in initializer and Fan_avg:

$$\begin{aligned} x_{in} &= \sqrt{\frac{2}{in}}, \\ x_{avg} &= \sqrt{\frac{6}{in+out}} \end{aligned} \tag{17}$$

where in and out are the number of units of the previous and following layers.

Besides the hyper-parameters mentioned above, the batch size B is also a parameter that must be chosen. Early stopping is utilized here; the training will stop when there is no improvement on the validation dataset in the last N epochs. The usage of the validation dataset can help prevent overfitting.

For the problem considered in this paper, the inputs are or include orbital states. The form of input, which is also known as features, is also an important factor that affects the performance of NNs. Therefore, two forms of orbital states are considered: *mee* (p, e_x, e_y, h_x, h_y, L), and Cartesian form *eci* (x, y, z, v_x, v_y, v_z), and the corresponding input forms are denoted F_{mee}, and F_{eci}, respectively. Therefore, there are altogether six NNs $\{N_{t_f,mee}, N_{t_f,eci}, N_{\lambda_0,mee}, N_{\lambda_0,eci}, N_{\boldsymbol{\alpha},mee}, N_{\boldsymbol{\alpha},eci}\}$ that need to be trained.

For N_{t_f}, the inputs are initial orbital states (a six-dimensional vector) and the output is the predicted optimal transfer time; for $N_{\boldsymbol{\lambda}_0}$, the inputs are also initial orbital states, and the output is the predicted costate (a seven-dimensional vector); for $N_{\boldsymbol{\alpha}}$, the inputs are states that consist of orbital states and mass, and the output is the predicted optimal control angles α and β that make up a two-dimensional vector. The combination of the state and optimal control is called the state-action pair. All inputs are normalized to make their average zero and their standard deviation one. The control angles α and β are scaled to the range [-1 1].

B. SELECTION OF NN MODELS

Random search [37] is applied in this paper to find hyper-parameters [26]. The hyper-parameters that must be defined are: number of layers (n_{layer}), number of neurons at each hidden layer (n_{neuron}), activation function (f), initial learning rate (η), optimizer (opt), batch size (B), weights initializer (ini), decay model (dm), decay step (ds), and decay rate (c). For $N_{\boldsymbol{\alpha}}$, the activation function of the output layer f_{out} is selected from a linear function and tanh. The search space of hyper-parameters is listed in Table 1. These parameters are uniformly random, except that the learning rate is uniform in the log-domain. The step for early stopping is set to 10 for the search, and after the model is selected the NN is retrained setting the early stop step to 50. It should be noted that the minimum layer number can be one, which makes up a shallow network. To evaluate the performance of the shallow network with many neurons, when the layer number is one, the neuron number is set to be twice the value selected from the search space. It should also be noted that the selection of NN models is based on the validation dataset. In this work, DNNs are trained using the python package tensorflow.

TABLE 1: Search space of random search

Hyper-parameters	Search space
n_{layer}	1–7
n_{neuron}	32–512
f	sigmoid, tanh, relu
$f_{out}(N_{\boldsymbol{\alpha}}\ only)$	linear, tanh
η	0.1–0.0001
opt	GD, MGD, AGD
B	100–1000
ini	Fan_in, Fan_avg
dm	ED, NED
ds	100–500
c	0.8–1

IV. EVALUATION OF NN PERFORMANCE

Besides the loss function in the form of MSE in Eq.(14), some other approaches should be proposed to evaluate the performance of NNs.

A. EVALUATION OF N_{T_F} AND $N_{\boldsymbol{\lambda}_0}$

Absolute percentage error (APE) is used to evaluate the performance of N_{t_f}:

$$APE = \frac{|t_{nn} - t_{opt}|}{t_{opt}} \tag{18}$$

where t_{opt} is the optimal value, t_{nn} is the predicted value given by N_{t_f}.

The solution of the shooting function in Eq.(10) consists of time of flight and initial costate, and their initial guess can be given by N_{t_f} and $N_{\boldsymbol{\lambda}_0}$. The shooting success rate and iteration number can be used to evaluate how accurate the initial guess is. As comparison, the success rate and iteration number of random initial guess and initial guess from the launch window set $\mathscr{L}$ are also investigated.

It is also of interest to study how the error of initial guesses given by N_{t_f} and $N_{\boldsymbol{\lambda}_0}$ will be propagated by integrating Eq.(2) and Eq.(6). Therefore, the orbital distance consisting of δa, δe, δi, and δL is utilized to evaluate how accurate the final states are compared to the target states at the final time.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2946657, IEEE Access

B. EVALUATION OF N_α

Trajectories controlled by $N_{\boldsymbol{\alpha}}$, instead of the optimal control, are integrated. The time of flight is given by N_{t_f}. The orbital distance introduced above is also used here to evaluate the performance of $N_{\boldsymbol{\alpha}}$.

The optimality of $N_{\boldsymbol{\alpha}}$ should also be evaluated. It is fair to evaluate the optimality of $N_{\boldsymbol{\alpha}}$ using the same final state [29]. For one $N_{\boldsymbol{\alpha}}$-controlled trajectory, assume that there is a virtual celestial body that has the same states as the final states x_f of a $N_{\boldsymbol{\alpha}}$-driven trajectory at the final time t_f. Then the trajectory from the same initial states at the same initial time to this virtual celestial body is optimized using indirect methods, and the optimal time $t_{f,opt}$ can be acquired. The optimality error is then calculated by comparing t_f and $t_{f,opt}$.

V. NUMERICAL EXAMPLES

The initial mass of the spacecraft is 1500 kg. The maximum thrust T_{max} is 0.3 N and the I_{sp} is 3000 s. Two types of missions, the Earth to Near Earth Asteroids (NEA) mission and the Earth to Mars mission are considered. The length of launch window set is 20 days. The NEA 2016HO3 is selected as the target. The launch window for NEA mission is [2026-11-1 2026-11-21], and for Mars mission is [2026-6-8 2026-6-28]. The orbit elements of celestial bodies are calculated using the data given by JPL [38]

Altogether 12,000 trajectories for each mission are generated, and there are 548,710 state-action pairs in NEA mission and 701,756 state-action pairs in Mars mission. The datasets are visualized in Fig.2 for NEA mission and in Fig.3 for Mars mission, respectively. It can be seen that the dataset covers a large range along the transfers from the Earth to NEA or Mars. The histograms of t_f and $\boldsymbol{\lambda}_0$ of the NEA mission dataset and Mars mission dataset are shown in Fig. 4 and Fig. 5, respectively.

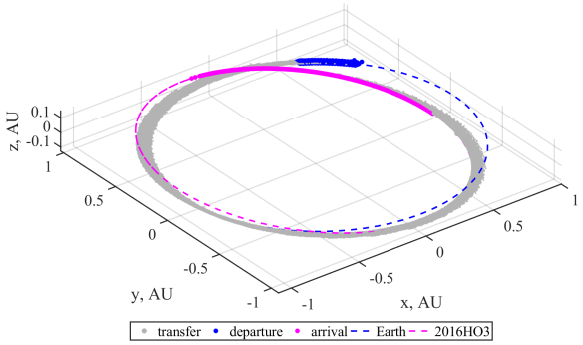

FIGURE 2: Visualization of the NEA mission dataset

A. NN TRAINING RESULTS

Random search is run 100 times for each dataset to select the hyper-parameters. Since the initialization of weights and bias are random, each complete training process can lead to

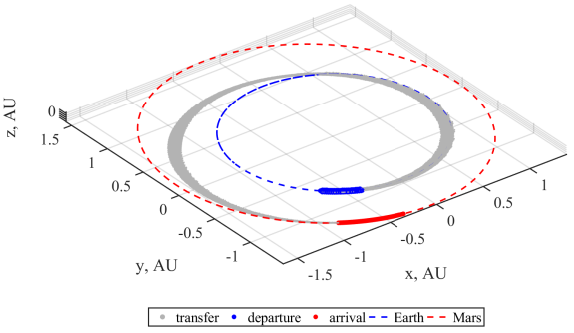

FIGURE 3: Visualization of the Mars mission dataset

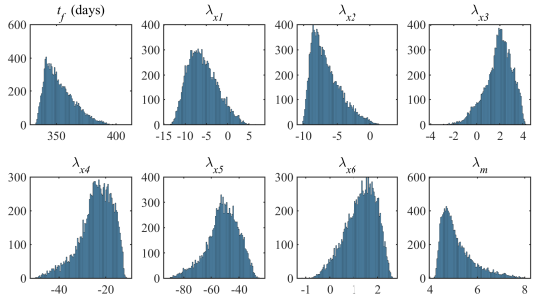

FIGURE 4: Histogram of t_f and $\boldsymbol{\lambda}_0$ of the NEA mission dataset

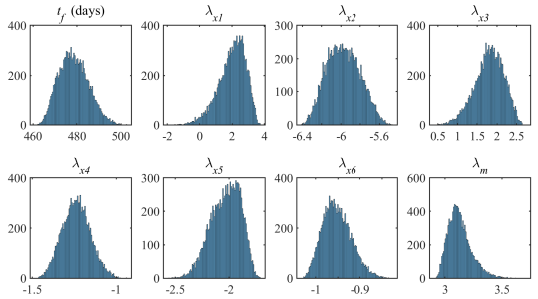

FIGURE 5: Histogram of t_f and $\boldsymbol{\lambda}_0$ of the Mars mission dataset

different results. Therefore, The NN with the selected hyper-parameters is retrained three times using 50 as the early stop step, and the one that has the best performance on the test dataset is used for our following analysis. The selected hyper-parameters and MSE on the test dataset are listed in Table 2 for NEA mission and in Table 3 for Mars mission. It can be found that all NNs are deep networks, and AGD is used by most NNs. In NEA mission, mee shows better performance than eci in general, while they are competitive in Mars mission. In the following part that will evaluate the NN performance, $\{N_{t_f,mee}, N_{\boldsymbol{\lambda}_0,eci}, N_{\boldsymbol{\alpha},mee}\}$ are used for NEA mission, and $\{N_{t_f,mee}, N_{\boldsymbol{\lambda}_0,eci}, N_{\boldsymbol{\alpha},eci}\}$ are used for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2946657, IEEE Access

TABLE 2: Hyper-parameters and MSE of NEA mission.

	n_{layers}	$n_{neurons}$	f	f_{out}	η	opt	B	ini	dm	ds	c	MSE
$N_{tf,mee}$	4	254	sigmoid	linear	0.0057	AGD	500	Fan_{avg}	ED	110	0.94	0.00003973
$N_{tf,eci}$	3	460	relu	linear	0.000034	AGD	150	Fan_{avg}	ED	160	0.83	0.00005037
$N_{\lambda_0,mee}$	4	452	tanh	linear	0.000034	AGD	200	Fan_{avg}	ED	170	0.84	0.01125374
$N_{\lambda_0,eci}$	4	350	tanh	linear	0.00021	AGD	500	Fan_{in}	ED	150	0.84	0.01123626
$N_{\boldsymbol{\alpha},mee}$	3	382	relu	tanh	0.00042	AGD	650	Fan_{in}	NED	130	0.84	0.00107357
$N_{\boldsymbol{\alpha},eci}$	6	390	relu	linear	0.00077	AGD	100	Fan_{in}	NED	80	0.86	0.00138577

TABLE 3: Hyper-parameters and MSE of Mars mission.

	n_{layers}	$n_{neurons}$	f	f_{out}	η	opt	B	ini	dm	ds	c	MSE
$N_{tf,mee}$	4	264	sigmoid	linear	0.0041	AGD	400	Fan_{in}	ED	150	0.96	0.00000675
$N_{tf,eci}$	5	386	relu	linear	0.0020	MGD	550	Fan_{avg}	ED	190	0.98	0.00003590
$N_{\lambda_0,mee}$	5	444	sigmoid	linear	0.0015	AGD	200	Fan_{avg}	NED	120	0.88	0.00003804
$N_{\lambda_0,eci}$	3	298	tanh	linear	0.000067	AGD	100	Fan_{avg}	ED	130	0.81	0.00002933
$N_{\boldsymbol{\alpha},mee}$	4	392	relu	linear	0.00089	AGD	550	Fan_{in}	ED	140	0.85	0.00409602
$N_{\boldsymbol{\alpha},eci}$	3	436	tanh	tanh	0.00044	AGD	650	Fan_{in}	NED	130	0.90	0.00320074

Mars mission. Meanwhile, the generalization capability is also evaluated by changing the factor c in Eq.(11) to 0.03 $\mathscr{A}_1$ and 0.05 $\mathscr{A}_2$ to generate data that outside the original initialization area $\mathscr{A}_0$.

B. PERFORMANCE OF N_{T_F} AND $N_{\boldsymbol{\lambda}_0}$

N_{t_f} is a network that predicts the optimal time of an optimal control problem, and $N_{\boldsymbol{\lambda}_0}$ is a network that predicts the initial costates of indirect optimization methods. From each initialization area $\mathscr{A}_0$, $\mathscr{A}_1$, $\mathscr{A}_2$, 100 initial states are randomly generated respectively, and the corresponding predicted optimal time and initial costate are found by evaluating N_{t_f} and $N_{\boldsymbol{\lambda}_0}$. Then the indirect problem is solved by using three different initial guess methods: random initial guess (Rand, initial guess from $\mathscr{L}$, and NN prediction.

Results are given in Table 4. As for $\mathscr{A}_0$, in both missions, the error of optimal time prediction is less than 0.1%. $\mathscr{L}$ and NN can both ensure 100% success rate. However, compared to $\mathscr{L}$, NN is simpler to generate the initial guesses without looking up from a set. As for the average iteration number, the one of NN is about half than that in $\mathscr{L}$, which means NNs improve the convergence efficiency. The orbital distances in both missions are quite small. As the initialization area getting large from $\mathscr{A}_0$ to $\mathscr{A}_2$, the error of optimal time prediction increases but it is still less than 2%. The success rate of $\mathscr{L}$ decreases but the success rate of NN is still 100%, which shows the excellent generalization capability of NNs. The iteration number is still about half in NN than in $\mathscr{L}$. The orbital distances are also acceptable. These results show that NNs have excellent performance and generalization capability in predicting the optimal time and initial costates.

C. PERFORMANCE OF N_{α}

The $N_{\boldsymbol{\alpha}}$ is a network that predicts the optimal law to implement real-time onboard optimal control. From each initialization area $\mathscr{A}_0$, $\mathscr{A}_1$, $\mathscr{A}_2$, 100 initial states are randomly generated respectively. Then $N_{\boldsymbol{\alpha}}$ controlled trajectories are integrated. The examples of optimal control, NN prediction and NN control are shown in Fig.6. The NN prediction is the control that $N_{\boldsymbol{\alpha}}$ predicts given the states from the optimal trajectory, while the NN control is the control from a $N_{\boldsymbol{\alpha}}$-controlled trajectory. It can be found that NN prediction is very close to optimal control and NN control also coincides well with the optimal control. The orbital distance and optimality error are given in Table.5. In NEA mission, the orbital distance and optimality error are small in $\mathscr{A}_0$ and $\mathscr{A}_1$ but become quite large in $\mathscr{A}_2$. In Mars mission, the results are not as good as that in NEA mission but are still acceptable when considering $N_{\boldsymbol{\alpha}}$ as a long-distance guidance and control law.

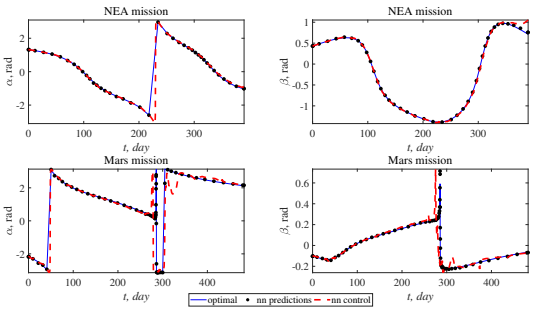

FIGURE 6: Comparison of different control

VI. CONCLUSION

In this paper, three neural networks, N_{t_f}, $N_{\boldsymbol{\lambda}_0}$, and $N_{\boldsymbol{\alpha}}$, are trained to learn the optimal time, initial costates, and optimal control law of time-optimal low-thrust indirect trajectory optimization, respectively. Hyper-parameters of NNs are searched by random search, and evaluation methods of N_{t_f}, $N_{\boldsymbol{\lambda}_0}$, and $N_{\boldsymbol{\alpha}}$ are also proposed. Results show that NNs have the capability to provide an accurate initial guess of optimal time and initial costates that are shooting variables in indirect optimization methods, even if outside the original initialization area. Compared to random initial guess and initial guess from the launch window set $\mathscr{L}$, NN prediction is superior in aspects of success rate and iteration number.

TABLE 4: Performance of N_{t_f} and $N_{\boldsymbol{\lambda}_0}$.

		APE$_{t_f}$, %	success rate			iterations			orbital distance			
			Rand	$\mathscr{L}$	NN	Rand	$\mathscr{L}$	NN	δa, AU	δe	δi, deg	δL, deg
NEA	$\mathscr{A}_0$	0.0802	43	100	100	84.51	7.96	4.84	0.003002	0.001557	0.006040	0.430759
	$\mathscr{A}_1$	0.3241	36	98	100	70.13	9.01	5.07	0.008708	0.004210	0.024537	1.082738
	$\mathscr{A}_2$	1.9032	34	90	100	84.60	13.60	7.03	0.025053	0.016950	0.205458	4.889173
Mars	$\mathscr{A}_0$	0.02111	70	100	100	36.34	7.45	4.41	0.004314	0.003143	0.001842	0.122544
	$\mathscr{A}_1$	0.1069	82	100	100	31.49	8.62	4.59	0.007950	0.005900	0.003652	0.272159
	$\mathscr{A}_2$	1.7038	76	93	100	34.98	11.72	7.20	0.028195	0.020773	0.027707	1.660816

TABLE 5: Performance of $N_{\boldsymbol{\alpha}}$.

		orbital distance				Optimality error, %
		δa, AU	δe	δi, deg	δL, deg	
NEA	$\mathscr{A}_0$	0.008522	0.006014	0.055284	0.411350	0.834291
	$\mathscr{A}_1$	0.009873	0.007281	0.150389	0.850601	2.246412
	$\mathscr{A}_2$	0.028864	0.022576	1.089927	11.995903	18.656750
Mars	$\mathscr{A}_0$	0.033320	0.022061	0.074826	2.777529	2.137036
	$\mathscr{A}_1$	0.087914	0.037381	0.162581	8.391229	9.881813
	$\mathscr{A}_2$	0.197655	0.056714	0.390094	24.657194	19.847472

Moreover, NN prediction ensures 100% success rate in all cases. Results of $N_{\boldsymbol{\alpha}}$ show that NNs can be utilized in real-time onboard optimal control. However, more precise design and training of NNs should be further investigated to improve the terminal accuracy of NN control.

REFERENCES

[1] M. D. Rayman, P. A. Chadbourne, J. S. Culwell, and S. N. Williams, "Mission design for deep space 1: a low-thrust technology validation mission," Acta astronautica, vol. 45, no. 4-9, pp. 381–388, 1999.

[2] J. Benkhoff, J. Van Casteren, H. Hayakawa, M. Fujimoto, H. Laakso, M. Novara, P. Ferri, H. R. Middleton, and R. Ziethe, "BepicolomboâĂŤcomprehensive exploration of mercury: Mission overview and science goals," Planetary and Space Science, vol. 58, no. 1-2, pp. 2–20, 2010.

[3] J. T. Betts, "Survey of numerical methods for trajectory optimization," Journal of guidance, control, and dynamics, vol. 21, no. 2, pp. 193–207, 1998.

[4] Y. Gao and C. Kluever, "Low-thrust interplanetary orbit transfers using hybrid trajectory optimization method with multiple shooting," in AIAA/AAS Astrodynamics Specialist Conference and Exhibit, p. 5088, 2004.

[5] F. Jiang, H. Baoyin, and J. Li, "Practical techniques for low-thrust trajectory optimization with homotopic approach," Journal of Guidance, Control, and Dynamics, vol. 35, no. 1, pp. 245–258, 2012.

[6] Z. Chi, H. Yang, S. Chen, and J. Li, "Homotopy method for optimization of variable-specific-impulse low-thrust trajectories," Astrophysics and Space Science, vol. 362, no. 11, p. 216, 2017.

[7] H. Yang, S. Li, and X. Bai, "Fast homotopy method for asteroid landing trajectory optimization using approximate initial costates," Journal of Guidance, Control, and Dynamics, pp. 1–13, 2018.

[8] G. Tang and F. Jiang, "Capture of near-earth objects with low-thrust propulsion and invariant manifolds," Astrophysics and Space Science, vol. 361, no. 1, p. 10, 2016.

[9] Z. Chi, H. Li, F. Jiang, and J. Li, "Power-limited low-thrust trajectory optimization with operation point detection," Astrophysics and Space Science, vol. 363, no. 6, p. 122, 2018.

[10] C. Zhang, F. Topputo, F. Bernelli-Zazzera, and Y.-S. Zhao, "Low-thrust minimum-fuel optimization in the circular restricted three-body problem," Journal of Guidance, Control, and Dynamics, vol. 38, no. 8, pp. 1501–1510, 2015.

[11] J. L. Gonzalo, C. Bombardelli, and F. Topputo, "Unified formulation for element-based indirect trajectory optimization," in 26th International Symposium on Space Flight Dynamics, pp. 1–6, 2017.

[12] H. Li, S. Chen, and H. Baoyin, "J2-perturbed multitarget rendezvous optimization with low thrust," Journal of Guidance, Control, and Dynamics, vol. 41, no. 3, pp. 802–808, 2017.

[13] J. L. Gonzalo, F. Topputo, and R. Armellin, "Indirect optimization of end-of-life disposal for galileo constellation using low thrust propulsion," in Proc. 26th International Symposium on Space Flight Dynamics, 2017.

[14] X. Zeng and X. Liu, "Searching for time optimal periodic orbits near irregularly shaped asteroids by using an indirect method," IEEE Transactions on Aerospace and Electronic Systems, vol. 53, no. 3, pp. 1221–1229, 2017.

[15] S. Chen, H. Li, and H. Baoyin, "Multi-rendezvous low-thrust trajectory optimization using costate transforming and homotopic approach," Astrophysics and Space Science, vol. 363, no. 6, p. 128, 2018.

[16] F. Jiang and G. Tang, "Systematic low-thrust trajectory optimization for a multi-rendezvous mission using adjoint scaling," Astrophysics and Space Science, vol. 361, no. 4, p. 117, 2016.

[17] M. Pontani and B. Conway, "Optimal low-thrust orbital maneuvers via indirect swarming method," Journal of Optimization Theory and Applications, vol. 162, no. 1, pp. 272–292, 2014.

[18] F. Jiang, G. Tang, and J. Li, "Improving low-thrust trajectory optimization by adjoint estimation with shape-based path," Journal of Guidance, Control, and Dynamics, vol. 40, no. 12, pp. 3282–3289, 2017.

[19] X. Bai, J. Turner, and J. Junkins, "Bang-bang control design by combing pseudospectral method with a novel homotopy algorithm," in AIAA Guidance, Navigation, and Control Conference, p. 5955, 2009.

[20] G. Tang and K. Hauser, "A data-driven indirect method for nonlinear optimal control," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4854–4861, IEEE, 2017.

[21] L. Casalino, "Approximate optimization of low-thrust transfers between low-eccentricity close orbits," Journal of Guidance, Control, and Dynamics, vol. 37, no. 3, pp. 1003–1008, 2014.

[22] D. Hennes, D. Izzo, and D. Landau, "Fast approximators for optimal low-thrust hops between main belt asteroids," in Computational Intelligence (SSCI), 2016 IEEE Symposium Series on, pp. 1–7, IEEE, 2016.

[23] A. Mereta, D. Izzo, and A. Wittig, "Machine learning of optimal low-thrust transfers between near-earth objects," in International Conference on Hybrid Artificial Intelligence Systems, pp. 543–553, Springer, 2017.

[24] D. Izzo, M. Märtens, and B. Pan, "A survey on artificial intelligence trends in spacecraft guidance dynamics and control," arXiv preprint arXiv:1812.02948, 2018.

[25] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural networks, vol. 61, pp. 85–117, 2015.

[26] H. Li, S. Chen, D. Izzo, and H. Baoyin, "Deep networks as approximators of optimal transfers solutions in multitarget missions," arXiv preprint arXiv:1902.00250, 2019.

[27] Y. Song and S. Gong, "Trajectory design of multiple near earth asteroids exploration using solar sail based on deep neural network," arXiv preprint arXiv:1901.02172, 2019.

[28] H. Peng and X. Bai, "Artificial neural network–based machine learning approach to improve orbit prediction accuracy," Journal of Spacecraft and Rockets, pp. 1–13, 2018.

[29] C. Sánchez-Sánchez and D. Izzo, "Real-time optimal control via deep neural networks: study on landing problems," Journal of Guidance, Control, and Dynamics, vol. 41, no. 5, pp. 1122–1135, 2018.
[30] L. Cheng, Z. Wang, F. Jiang, and C. Zhou, "Real-time optimal control for spacecraft orbit transfer via multi-scale deep neural networks," IEEE Transactions on Aerospace and Electronic Systems, 2018.
[31] D. Izzo, C. I. Sprague, and D. V. Tailor, "Machine learning and evolutionary techniques in interplanetary trajectory design," in Modeling and Optimization in Space Engineering, pp. 191–210, Springer, 2019.
[32] D. Izzo, E. Öztürk, and M. Märtens, "Interplanetary transfers via deep representations of the optimal policy and/or of the value function," arXiv preprint arXiv:1904.08809, 2019.
[33] R. Furfaro, I. Bloise, M. Orlandelli, P. Di Lizia, F. Topputo, and L. Richard, "Deep learning for autonomous lunar landing," in 2018 AAS/AIAA Astrodynamics Specialist Conference, pp. 1–22, 2018.
[34] F. Topputo and S. Ceccherini, "A catalogue of parametric time-optimal transfers for all-electric geo satellites," in Modeling and Optimization in Space Engineering, pp. 459–478, Springer, 2019.
[35] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in International conference on machine learning, pp. 1139–1147, 2013.
[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
[37] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," Journal of Machine Learning Research, vol. 13, no. Feb, pp. 281–305, 2012.
[38] E. M. Standish, "Keplerian elements for approximate positions of the major planets," available from the JPL Solar System Dynamics web site (http://ssd. jpl. nasa. gov/), 2006.