AAS 21-558

SATELLITE DRAG COEFFICIENT MODELING AND ORBIT UNCERTAINTY QUANTIFICATION USING STOCHASTIC MACHINE LEARNING TECHNIQUES

Smriti Nandan Paul[*], Logan Sheridan[†], Piyush Mehta[‡], and S. Huzurbazar[§]

2021 AAS/AIAA Astrodynamics Specialist Conference

The rapidly increasing congestion in the low Earth environment makes the modeling of uncertainty in atmospheric drag force a critical task, affecting space situational awareness (SSA) activities like the probability of collision estimation. A key element in atmospheric drag modeling is the assessment of uncertainty in the atmospheric drag coefficient estimate. While atmospheric drag coefficients for space objects with known characteristics can be computed numerically, they suffer from large computational costs for practical applications. In this work, we use cost-effective data-driven stochastic methods for modeling the drag coefficients of objects in the low Earth orbit (LEO) region. The training data is generated using the numerical Test Particle Monte Carlo (TPMC) method. TPMC is simulated with Cercignani–Lampis–Lord (CLL) gas-surface interaction (GSI) model. Mehta et al. [1] use a Gaussian process regression (GPR) model to predict satellite drag coefficient, but the authors did not estimate the predictive uncertainty. The first part of this research extends the work by Mehta et al. [1] by fitting a GPR model to the training data and performing predictive uncertainty estimation. The results of the Gaussian fit are then compared against a deep neural network (DNN) model aided by the Monte Carlo dropout approach. To the best of our knowledge, this is the first study to use the aforementioned stochastic deep learning algorithm to perform predictive uncertainty estimation of the estimated satellite drag coefficient. Apart from the accuracy of the models, we also undertake the task of calibrating the models. Simulations are carried out for a spherical satellite followed by the Champ satellite. Finally, quantification of the effect of drag coefficient uncertainty on orbit prediction is carried out for different solar activity and geomagnetic activity levels.

INTRODUCTION

In the last few decades, there has been a dramatic proliferation in the population of objects in the LEO region partly propelled by ambitious programs from private space companies and easier access to space due to an increase in the number of launch providers (or decrease in launch cost [2]). From European Space Agency's Annual Space Environment Report 2020, there are currently

[*]Post Doctoral Researcher, Department of Mechanical and Aerospace Engineering, West Virginia University, Morgantown, WV 26505, smritinandan.paul@mail.wvu.edu.

[†]Graduate Research Assistant, Department of Mechanical and Aerospace Engineering, West Virginia University, Morgantown, WV 26505, pls0013@mix.wvu.edu.

[‡]Assistant Professor, Department of Mechanical and Aerospace Engineering, West Virginia University, Morgantown, WV 26505, piyush.mehta@mail.wvu.edu.

[§]Professor, School of Mathematical and Data Sciences, West Virginia University, Morgantown, WV 26505, snehalata.huzurbazar@mail.wvu.edu.

around 15,000 tracked objects in the LEO region; the U.S. company SpaceX's Starlink constellation alone will roughly double this population [3]. This unabated growth increases the collision risk among space objects, which demands better modeling and quantification of the orbital uncertainties. Uncertainty in the modeling of conservative perturbation forces is typically low. The orbit prediction errors for typical low area-to-mass ratio objects in the LEO region mainly emanate from the uncertainty in the non-conservative atmospheric drag force [1]. Difficulty in estimating the drag force arises from inadequate knowledge of the atmospheric density and the satellite drag coefficient. Effects of the uncertainties in the atmospheric density on orbital evolution have been investigated by a number of authors, including Bussy-Virat et al. [4], Gondelach and Linares [5], Schiemenz et al. [6], Sagnieres and Sharf [7], Emmert et al. [8]. This work will focus on modeling the uncertainties in the satellite drag coefficient and its impact on the prediction of orbit.

Accuracy and computational cost are often the main factors that determine the method one uses for the computation of satellite drag coefficient. For high-accuracy drag coefficient modeling, especially for complex geometrical shapes, computationally expensive numerical methods such as Panel Method, Ray-tracing Panel Method (RTP), Test Particle Monte Carlo (TPMC) method, and Direct Simulation Monte Carlo (DSMC) method are typically used [9]. Besides numerical techniques, analytical methods such as Schamberg's derivation and Schaaf and Chambre derivation are also commonly used for computing drag coefficients of simple convex geometrical shapes [9]. Another prevalent practice in literature, although inaccurate for geometries with a high aspect ratio, is to assume a constant value of 2.2 for the satellite drag coefficient [1, 9]. An alternative to the already mentioned methods is the machine learning techniques, which are computationally less expensive than numerical methods and applicable to complex geometrical shapes. Machine learning also allows for the estimation of uncertainty in the satellite drag coefficient computation and is the method of choice for this research work.

Machine learning techniques are increasingly used in space weather predictive modeling [1, 10, 11, 12] because of their fast evaluations of the surrogate models and the ability to make non-linear predictions. This paper is particularly interested in stochastic machine learning techniques. Mehta et al. [1] demonstrate the excellent performance of Gaussian processes in predicting satellite drag coefficients (root mean squared percentage errors $< 0.3\%$, 0.9%, 0.6%, 1.0% for sphere, GRACE satellite, Hubble Space Telescope, International Space Station, respectively) but the authors do not perform uncertainty quantification. As an extension to the work by Mehta et al. [1], a GPR model will be first designed for predictive uncertainty estimation. The GPR model will then be used as a benchmark against stochastic deep learning. In the stochastic deep learning category, this work will use a feed-forward deep neural network with the Monte Carlo dropout approach, which is a popular regularization technique that involves dropping out units in a neural network. Gal and Ghahramani [13] prove that the application of Monte Carlo dropout in a neural network can be construed as a Bayesian approximation for a Gaussian process.

In the current work, the drag coefficient training data for the machine learning algorithms will be created using the numerical TPMC method [14] using the CLL [15, 16] GSI models. In the TPMC method, the test particles, which represent actual molecules, are produced serially rather than simultaneously. The molecules do not collide with each other and, the method is well-suited for free-molecular flow. An advantage of the TPMC method is its ability to model flows with complex boundaries. The CLL model, a kernel-based rendition of the GSI, successfully simulates the quasi-specular reflection seen in molecular beam experiments [15]. The CLL GSI model allows for the handling of tangential and normal components of velocity independently, using different

scattering kernels for each direction. The two key parameters defining the CLL scattering kernels are the normal energy accommodation coefficient (α_n) and the tangential momentum accommodation coefficient (σ_t).

The remainder of this paper is organized as follows: section 2 provides details about the training data for the regression models. Section 3 discusses the stochastic machine learning models used for prediction in this work. Model calibration is discussed in section 4. Section 5 investigates the impact of drag coefficient uncertainty on orbit. Conclusions are drawn in the last section.

INPUT DATA FOR THE REGRESSION MODELS

Since it is difficult to obtain extensive experimental drag coefficient data for training predictive models, we use numerical simulations to generate the input training data. Drag coefficient values are computed for the primary LEO atmospheric constituents - atomic hydrogen (H), helium (He), atomic nitrogen (N), molecular nitrogen (N_2), atomic oxygen (O), and molecular oxygen (O_2) using the numerical TPMC method. During the training phase, the predictive models learn the relationship between independent variables and the drag coefficient. For the TPMC method with CLL GSI model, seven independent variables determine the value of the dependent drag coefficient - (i) relative velocity of the satellite, v_∞ (ii) satellite surface temperature, T_w (iii) atmospheric translational temperature, T_∞ (iv) normal energy accommodation coefficient, α_n (v) tangential momentum accommodation coefficient, σ_t (vi) satellite yaw, β (vii) satellite pitch, Φ. Since only a limited number of numerical simulations can be carried out due to computational constraints, input configurations for the training are carefully selected via the Latin Hypercube sampling (LHS) method [17].

In this study, the numerical TPMC simulations are carried out for a sphere and the Champ satellite. For both the objects, 1000 LHS design points are selected for each of the species H, He, N, N_2, O O_2. The upper and lower bounds defining the LHS design points are given in Table 1.

Table 1. Upper and Lower Bounds Defining the LHS Design Points

Independent Variables	Lower Bound	Upper Bound
v_∞	7250.0 m/s	8000.0 m/s
T_w	100.0 K	2000.0 K
T_∞	200.0 K	2000.0 K
α_n	0.0	1.0
σ_t	0.0	1.0
β	-10^0	10^0
Φ	-10^0	10^0

PROBABILISTIC MODELS FOR PREDICTION

As our primary goal is to provide uncertainty estimates, we use Gaussian process regression and a feed-forward neural network with the Monte Carlo dropout approach to predict drag coefficients for H, He, N, N_2, O, O_2.

Gaussian Process Regression Model

Gaussian process regression is a Bayes' theory based approach to supervised machine learning. Unlike traditional regression approaches that fit a single function through the observed training

data, GPR models the probability distribution over the functional space conditional on the observed data. In GPR, a Gaussian process (GP) prior is first defined over the functional space $\mathbf{x} \in \mathbb{R}^{D_i} \rightarrow \mathbf{f}(\mathbf{x}) \in \mathbb{R}^{D_o}$. GP is a stochastic process such that for any set of inputs $\{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}$, the random variables $\{\mathbf{f}(\mathbf{x_1}), \mathbf{f}(\mathbf{x_2}), ..., \mathbf{f}(\mathbf{x_n})\}$ are jointly Gaussian. GP prior is defined by a mean function (assumed $\mathbf{0}$ in this research work) and a covariance matrix:

$$p(\mathbf{F}|\mathbf{X}) = \mathcal{N}(\mathbf{F}|\mathbf{0}, \mathbf{K}) + \mathcal{N}(\epsilon|\mathbf{0}, n_l^2\mathbf{I}) \tag{1}$$

where $\mathbf{F} = [\mathbf{f}(\mathbf{x_1}), \mathbf{f}(\mathbf{x_2}), ..., \mathbf{f}(\mathbf{x_n})]^T$, $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}]^T$, and $\mathcal{N}$ represents the normal distribution. The variable ϵ represents the training data noise with noise-level n_l. The elements of the covariance matrix $\mathbf{K}$ are defined by the "kernel" functions. This work makes use of the Matern kernel, which is given as:

$$K_{ij} = \kappa(\mathbf{x_i}, \mathbf{x_j}) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}}{l} d(\mathbf{x_i}, \mathbf{x_j}) \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{l} d(\mathbf{x_i}, \mathbf{x_j}) \right) \tag{2}$$

where $\Gamma(\cdot)$ represents the gamma function, $d(\cdot, \cdot)$ represents the Euclidean distance, and $K_\nu(\cdot)$ is a modified Bessel function [18]. The parameter ν controls the smoothness of the learned function. In this work, we use $\nu = 2.5$, a common choice for functions that are at least twice differentiable. l is a positive length-scale parameter with the same dimension as that of the input $\mathbf{x}$.

The posterior predictive distribution is generated by conditioning on the prior and the observations using Bayes' theorem. For the observation dataset $\{\mathbf{X}_o, \mathbf{F}_o\}$, the posterior predictive distribution corresponding to test inputs $\mathbf{X}_\star$ is an updated Gaussian that can be derived (see [19]) as:

$$p(\mathbf{F}_\star|\mathbf{X}_\star, \mathbf{X}_o, \mathbf{F}_o) = \mathcal{N}(\mathbf{F}_\star|\mu_\star, \mathbf{\Sigma}_\star) \tag{3}$$

$$\mu_\star = \mathbf{K}_\star^T(\mathbf{K}_o + n_l^2\mathbf{I})^{-1}\mathbf{F}_o \tag{4}$$

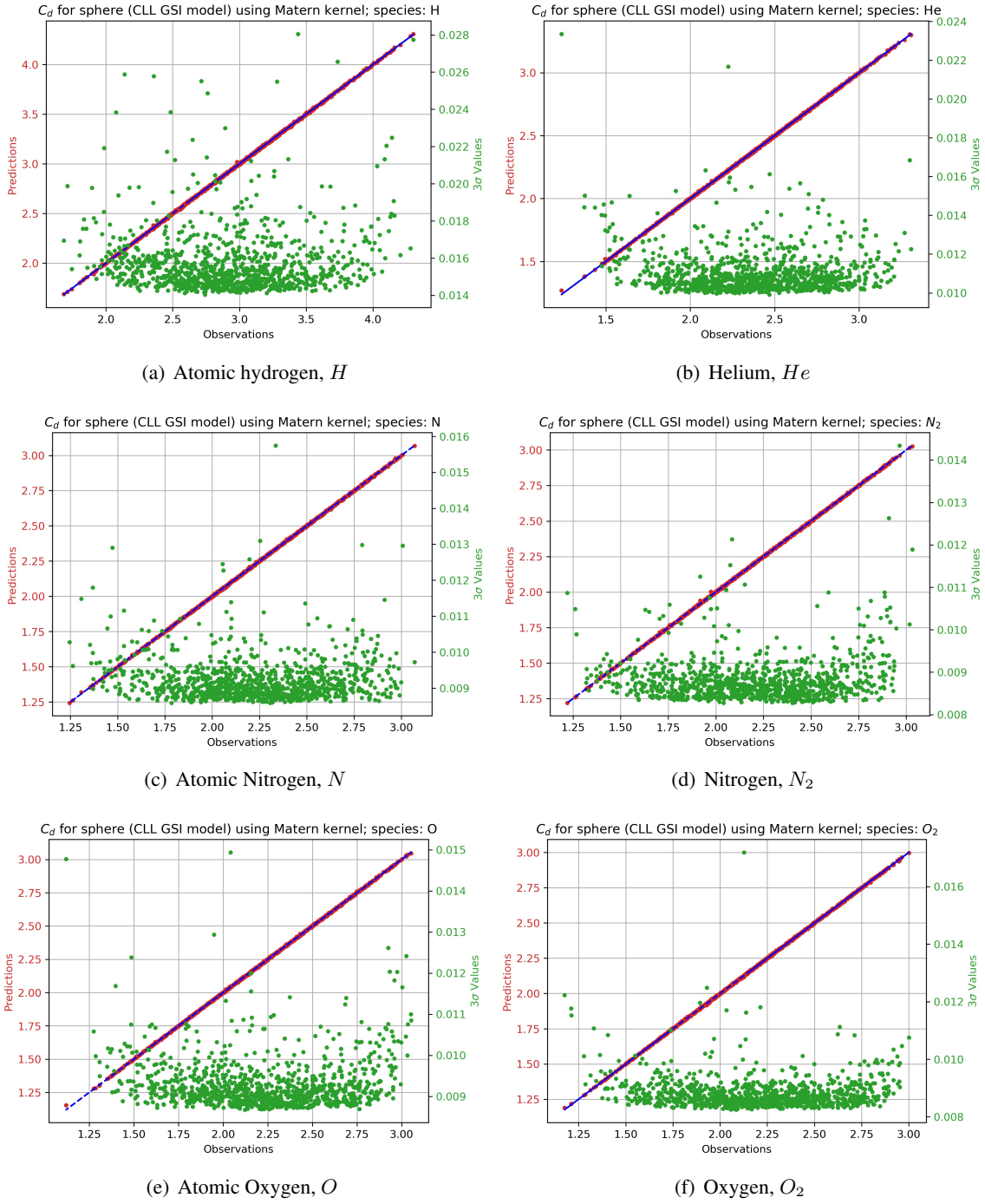
$$\mathbf{\Sigma}_\star = \mathbf{K}_{\star\star} - \mathbf{K}_\star^T(\mathbf{K}_o + n_l^2\mathbf{I})^{-1}\mathbf{K}_\star \tag{5}$$

where, $\mathbf{K}_\star = \mathbf{K}(\mathbf{X}_o, \mathbf{X}_\star)$, $\mathbf{K}_o = \mathbf{K}(\mathbf{X}_o, \mathbf{X}_o)$, and $\mathbf{K}_{\star\star} = \mathbf{K}(\mathbf{X}_\star, \mathbf{X}_\star)$.

The hyperparameters l, n_l appearing in Eqs. (1), (2) are obtained by maximizing the log marginal likelihood of the training data.

For implementing the GPR, we use the "GaussianProcessRegressor" module from the Python scikit-learn library [20]. The GPR models are trained on 1000 LHS samples and their corresponding drag coefficients calculated using TPMC. After that, the trained models are put to the test on a new set of 1000 data. The comparison between GPR predictions and test data for a sphere is shown in Fig. 1. In each plot, the x-coordinates of the red markers correspond to TPMC drag coefficients, and the y-coordinates of the red markers correspond to mean drag coefficients predicted by GPR. The y-coordinates of the green markers correspond to the 3σ values predicted by the GPR. The blue line corresponds to the ideal case of perfect calibration. The root mean squared error (RMSE) between the predicted and observed values for H, He, N, N_2, O, O_2 are 0.005397, 0.003799, 0.003136, 0.003207, 0.003214, 0.002975, respectively. Similarly, Fig. 2 shows comparison of GPR predictions and test data for Champ satellite. The RMSE between the predicted and observed values for Champ for H, He, N, N_2, O, O_2 are 0.033636, 0.019427, 0.013985, 0.013991, 0.014714,

0.013326, respectively. When compared to the sphere, the RMSE values are higher for Champ because Champ has a more complex geometry and, it would require a higher number of LHS training samples for Champ to achieve the same accuracy.



(a) Atomic hydrogen, H

(b) Helium, He

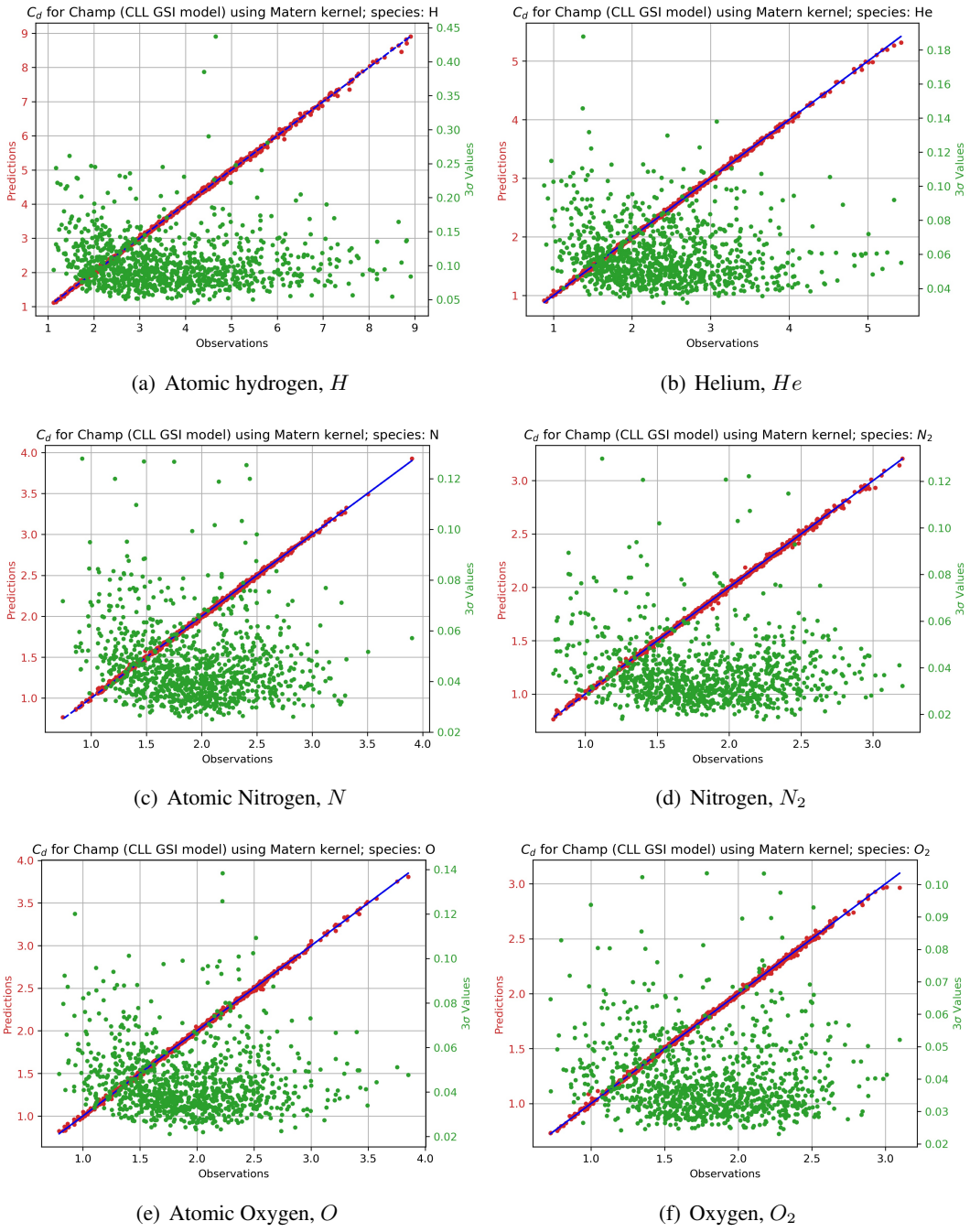
(c) Atomic Nitrogen, N

(d) Nitrogen, N_2

(e) Atomic Oxygen, O

(f) Oxygen, O_2

Figure 1. Comparison of GPR and TPMC Drag Coefficients for Sphere Test Dataset



(a) Atomic hydrogen, H

(b) Helium, He

(c) Atomic Nitrogen, N

(d) Nitrogen, N_2

(e) Atomic Oxygen, O

(f) Oxygen, O_2

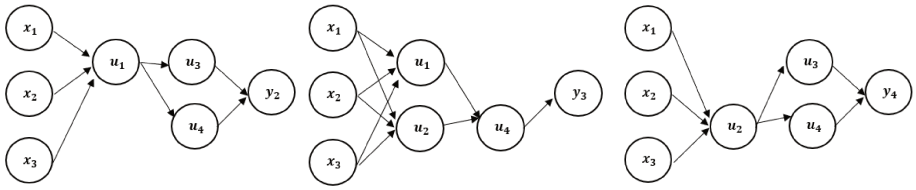
Figure 2. Comparison of GPR and TPMC Drag Coefficients for Champ Test Dataset

Deep Neural Network Regression With Monte Carlo Dropout

Feed-forward deep neural network, an artificial neural network with more than one layer of hidden units between its input and output layers [21], has found use across a wide range of predictive applications in recent years. DNN consists of a large number of parameters ("weights" and "biases") that control the function mapping from one layer to the next. One problem in machine learning algo-

rithms with many parameters and relatively small training datasets is the overfitting of the training data. This problem is commonly overcome by using L^1 and L^2 regularization techniques [22], which add an additional term to the cost function that penalizes large weights. Another simple and yet powerful regularization technique introduced by Srivastava et al. [23] in 2014 is the dropout. In dropout, each node is associated with an independent Bernoulli distribution (probability p for value 1). During each pass of a set of inputs through the network during training, samples are drawn from each Bernoulli distribution. If a "0" is sampled, the associated node is dropped along with all its incoming and outgoing connections. The output of each node is then scaled by a factor of $(1/p)$ to maintain the expected value.

Usually, dropout is kept on only during the training phase and disabled during the testing phase, resulting in a deterministic output. However, when dropout is enabled during the testing phase, multiple passes of the same input can produce different predictions depending upon the set of neurons dropped, as illustrated in Fig. 3. A large number of passes for any input will produce a range of outputs, from which one can compute the mean and variance. This method of inferring the uncertainty distribution is known as Monte Carlo dropout.



(a) Run 1: u_2 dropped (b) Run 2: u_3 dropped (c) Run 3: u_1 dropped

Figure 3. Same Input Producing Different Outputs With Dropout Enabled During Testing Phase

To implement Monte Carlo dropout in this research, we employ Keras' "Dropout" layer feature, where Keras is a Python-based deep learning API that runs on top of open-source machine learning platform Tensorflow [24].

Tuning of Deep Neural Network Hyper-Parameters A DNN can learn the values of its parameters (weights and biases) directly from the training process. Certain high-level parameters, on the other hand, cannot be learned from the training process. These high-level parameters are known as hyper-parameters [25]. In this work, we consider tuning of the following hyper-parameters: number of hidden layers, number of neurons in each hidden layer, activation functions, dropout rates, network optimizer, batch size. Because of knowledge limitations, a heuristic or manual approach to picking these hyper-parameters is not always feasible, and it is difficult to ensure that the chosen design is optimal or sub-optimal. For the current work, we conduct hyper-parameter tuning using the KerasTuner library [26]. KerasTuner has three different optimizers for finding the optimal hyper-parameters: random search, Bayesian optimization, and hyperband. We use Bayesian optimization because of its ability to quickly reach a solution that is close to optimal.

MODEL CALIBRATION

Calibration is the requirement in stochastic modeling that the predicted probabilities give an approximation of the probability of true events [27]. A well-calibrated model (assuming Gaussian distribution), for example, should have around 68 $\%$ true observations within one standard devia-

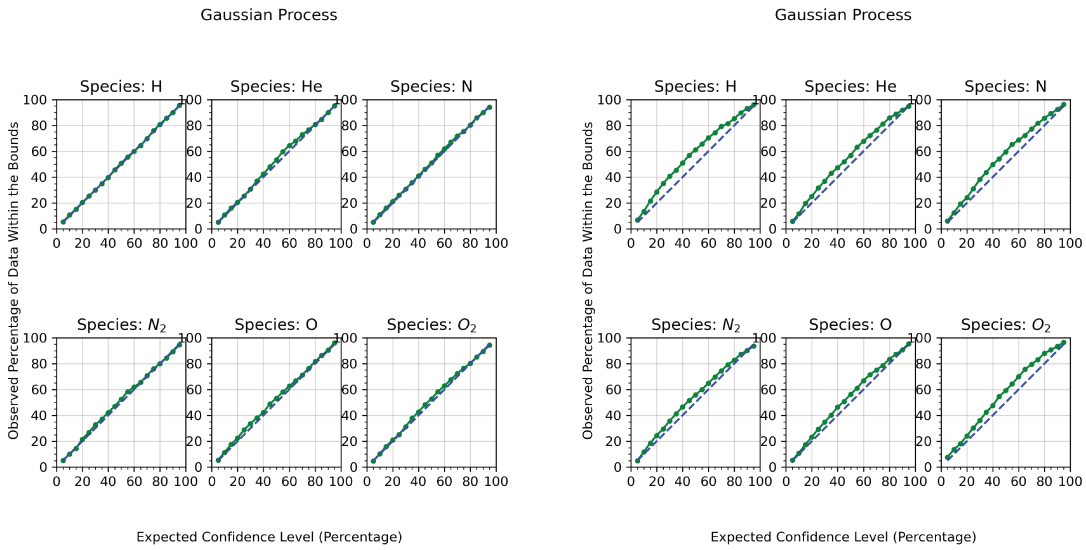
tion, 95% true observations within two standard deviations, and 99.7 % true observations within three standard deviations. Uncalibrated models tend to be over-confident or under-confident in their predictions, and one should not trust their inferences. A convenient way to check how well a model is calibrated is by looking at its "consistency curve".

Let the expected confidence interval levels be: $C = [5\%, 10\%, 15\%, ..., 95\%]$. The corresponding coefficients defining the uncertainty bounds are: $\zeta[j] = \sqrt{2}erf^{-1}(C[j]/100)$. Let, $(\mathbf{x_{o_j}}, \mathbf{y_{o_j}})_{j=1,...,m}$ be the observation dataset and let the corresponding predictions be $(\mu_\mathbf{j}, \sigma_\mathbf{j})_{j=1,...,m}$, where μ_j represents the mean and σ_j represents the standard deviation. Then, the percentage of observed dataset within the lower and upper uncertainty bounds associated with $C[j]$ is obtained as [28]:

$$P[j] = \left[\sum_{j=1}^{m} \frac{\mathcal{I}\big((\mu_j - \zeta[j]\sigma_j) < y_{o_j} < (\mu_j + \zeta[j]\sigma_j)\big)}{m} \right] \times 100 \tag{6}$$

where $\mathcal{I}$ is the indicator function.

The consistency curve mentioned earlier is the plot of P versus C. The proximity of the consistency curve to the $y = x$ line (i.e., a straight line with a slope of 45^0 and passing through the origin) is used to measure calibration in this study. The consistency curve will perfectly overlap the $y = x$ line in a perfectly calibrated system. The consistency plots for sphere and Champ drag coefficient predictions using GPR for the test dataset are shown in Fig. 4. The blue dotted line corresponds to the ideal case of perfect calibration, and the green curve corresponds to the GPR predictions. The predictions for the sphere are better calibrated than the predictions for the Champ satellite.



(a) Sphere Consistency Curve

(b) Champ Consistency Curve

Figure 4. Consistency Plots for Sphere and Champ for Drag Coefficient Prediction Using Gaussian Process Regression for the Test Dataset

The output of a deep learning model with Monte Carlo dropout is not guaranteed to be calibrated [28]. If the consistency curve indicates an uncalibrated model, there are two popular methods for

improving the calibration: (a) post-processing of the calculated uncertainty (b) utilizing a training loss function with improved calibration properties. Although we explore both options, the primary focus is on the latter method.

The usual practice in stochastic deep learning is to use mean squared error (MSE) as a training loss function:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{y_i} - \hat{\mathbf{y}}_\mathbf{i})^2 \tag{7}$$

where $(\mathbf{x_i}, \mathbf{y_i})_{i=1,...,n}$ represent the training dataset and $\hat{\mathbf{y}}_\mathbf{i}$ represents the prediction mean. The MSE loss function, however, ignores the predicted variance, possibly resulting in poor calibration. MSE can also be interpreted as a loss function that assumes a homoscedastic uncertainty, i.e., an uncertainty that is independent of the input [13].

In order to examine the calibration performance of training using the MSE loss function, a feed-forward DNN (with Monte Carlo dropout) is trained to predict the drag coefficient of a sphere for each of the constituent species H, He, N, N_2, O, O_2. KerasTuner, which is used to obtain the appropriate architecture for each of the six species, is trained over the following hyper-parameter search space: (i) number of hidden layers $\in [1, 2, 3, .., 10]$ (ii) number of neurons in each hidden layer $\in [32, 64, 96, ..., 128]$ (iii) activation function applied after every hidden layer $\in$ ['relu', 'tanh', 'sigmoid', 'softsign', 'selu', 'elu', 'linear'] (iv) dropout rate for the dropout regularization applied to each hidden layer $\in [.08, .16, .24, ..., 0.8]$ (v) optimizer for training the network $\in$ ['rmsprop', 'adagrad', 'adam', 'nadam'] (vi) batch size $\in [16, 32, 48, ..., 128]$. Table 2 lists other essential parameters utilized in the tuning process.

Table 2. Parameters Used in Hyper-Parameter Optimization Using KerasTuner

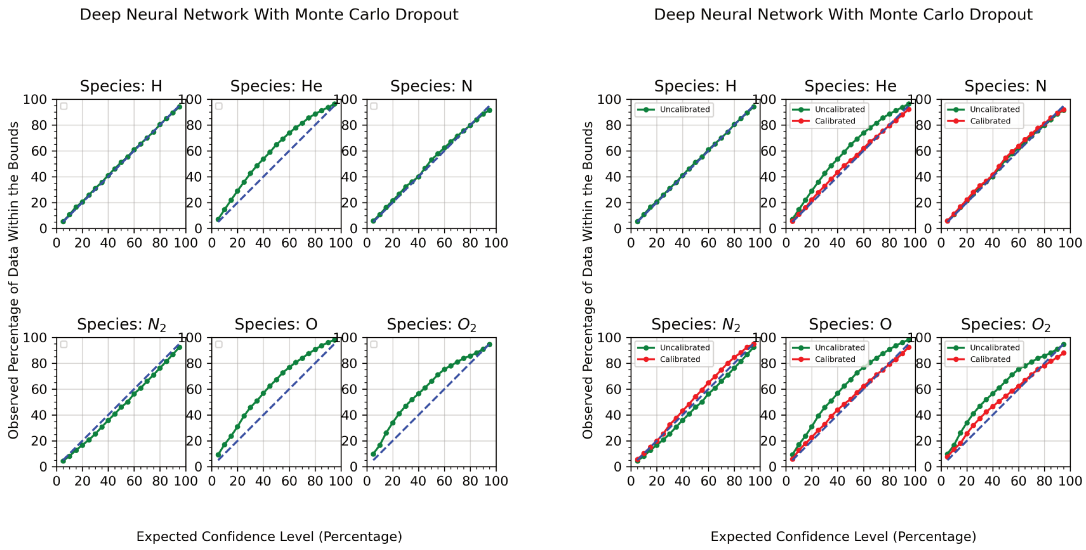
Parameter	Value
Maximum number of trials	150
Executions per trial	7
Number of initial points	50
Early stop regularization patience	50
Validation split	0.15
Number of epochs	200

The DNNs corresponding to the six species are trained using the best architectures output by the KerasTuner, and then predictions are made on the test datasets, resulting in the consistency curves shown in Fig. 5(a). The drag coefficient predictions for H are the best calibrated of the six species. To improve the lesser calibrated species He, N, N_2, O, O_2, the predicted standard deviation values are re-scaled using the following factor [29]:

$$s = \sqrt{\frac{1}{m_v}\sum_{i=1}^{m_v}(\sigma_i)^{-2}||\mathbf{y_i} - \hat{\mathbf{y}}_\mathbf{i}||^2} \tag{8}$$

where m_v is the number of validation samples, $\mathbf{y_i}$ is the TPMC drag coefficient, $\hat{\mathbf{y}}_\mathbf{i}$ is the predicted drag coefficient, and σ_i is the predicted standard deviation for the ith validation sample. Fig. 5(b) shows the consistency plots obtained after re-scaling the predicted uncertainties. In Fig. 5, the green

and red plots correspond to the calibration plots before and after $\sigma-$scaling, respectively. The red curves are better calibrated than the green curves, with notable improvements in He, O, and O_2.



(a) Uncalibrated

(b) Post-Training Calibration

Figure 5. Consistency Plot for Sphere for Drag Coefficient Prediction Using DNN (With Monte Carlo Dropout) for the Test Dataset. The Trained Network is Based on MSE Loss Function.

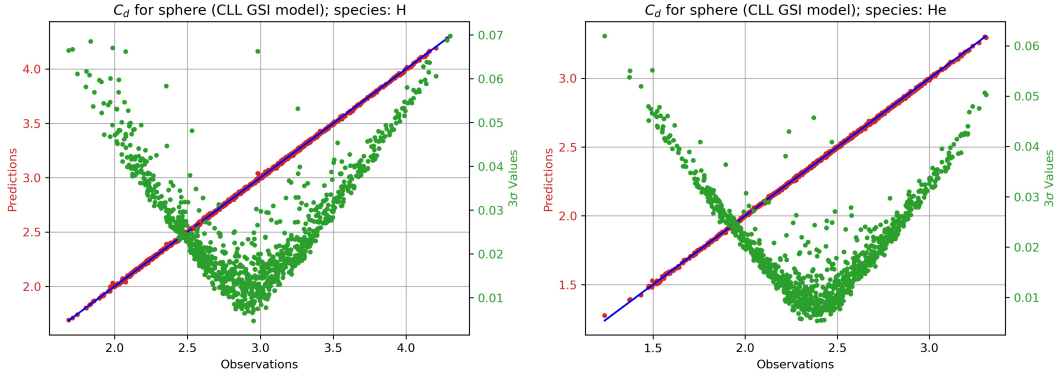
Although the post-training scaling method yields better calibration results in the above case, it is not always guaranteed to work. Instead of using the post-training calibration technique, it is preferable to employ a loss function other than MSE in this study. A suitable candidate for the training loss function that does not ignore either the mean or the variance and assumes a heteroscedastic uncertainty is based on the negative logarithm of the probability density (NLPD) and is given as:

$$NLPD = \frac{1}{n}\sum_{i=1}^{n}\left[\log\sigma_i^2 + \frac{||\mathbf{y_i} - \hat{\mathbf{y}}_\mathbf{i}||^2}{\sigma_i^2} + \log 2\pi\right] \tag{9}$$

where $(\mathbf{x_i}, \mathbf{y_i})_{i=1,\ldots,n}$ represent the training dataset, $\hat{\mathbf{y}}_\mathbf{i}$ represents the prediction mean, and σ_i^2 represents the prediction variance.

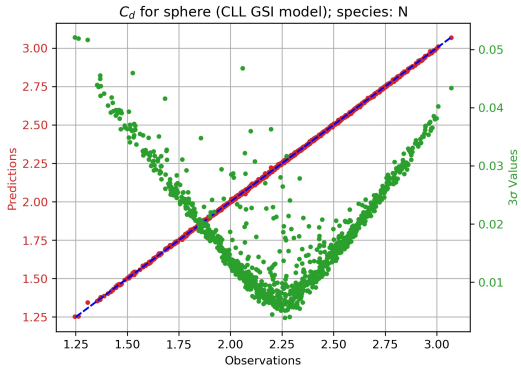
Using the NLPD loss function, a Monte Carlo dropout aided feed-forward DNN is trained to predict drag coefficients for a sphere for the different species. Like earlier, the hyper-parameters for the training architectures are obtained using KerasTuner. The implementation of the mean $\hat{\mathbf{y}}_\mathbf{i}$ and the standard deviation σ_i appearing in Eq. 9 needs some discussion. The neural network does not directly output $\hat{\mathbf{y}}_\mathbf{i}$ or σ_i. As such, 100 duplicates of each training sample are created. $\hat{\mathbf{y}}_\mathbf{i}$ and σ_i are taken as the mean and standard deviation of the stochastic forecasts of these 100 duplicates, respectively. Fig. 6 shows the drag coefficient of the sphere predicted using DNN with NLPD as the training loss function. The RMSE between the predicted and TPMC values for H, He, N, N_2, O, O_2 are 0.007912, 0.005833, 0.004484, 0.004781, 0.004794, 0.004581, respectively. Although dropout is theoretically a Bayesian approximation of the Gaussian Process, in practice, the uncertainties estimated by the two methods can be significantly different, as demonstrated in

Figs. 6 and 1.

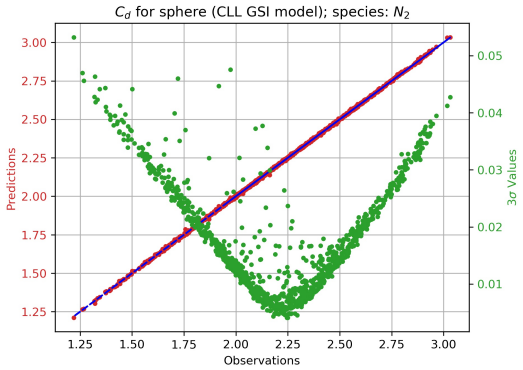


(a) Atomic hydrogen, H

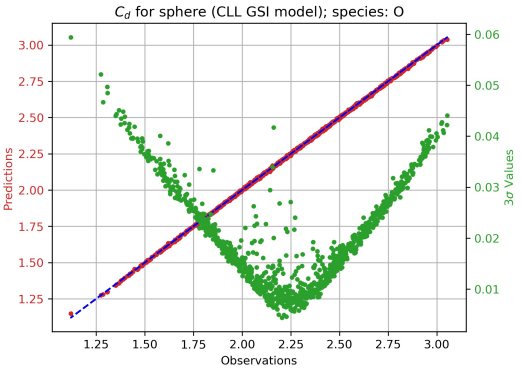
(b) Helium, He



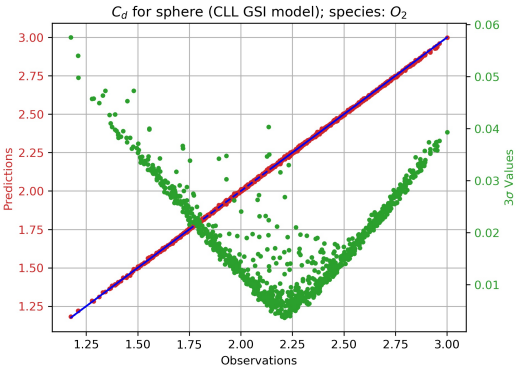
(c) Atomic Nitrogen, N



(d) Nitrogen, N_2



(e) Atomic Oxygen, O



(f) Oxygen, O_2

Figure 6. Comparison of Feed-Forward DNN (With Monte Carlo Dropout) and TPMC Drag Coefficients for Sphere Test Dataset for Different Species.

Fig. 7 shows the consistency plots for the sphere drag coefficient prediction using DNN. In Fig. 7, the black curves correspond to the NLPD loss function, the green curves correspond to the MSE loss function (also shown earlier in Fig. 5), and the blue dotted line represents the hypothetical case of perfect calibration. Compared to the MSE curves, the NLPD curves are better calibrated for He,

O, O_2, similarly calibrated for H, N_2, and poorly calibrated for N. Overall, NLPD is preferred over MSE loss function.

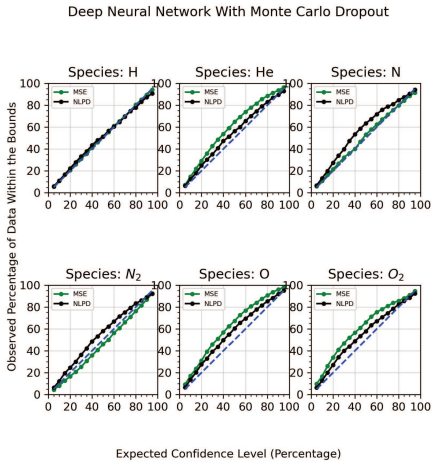
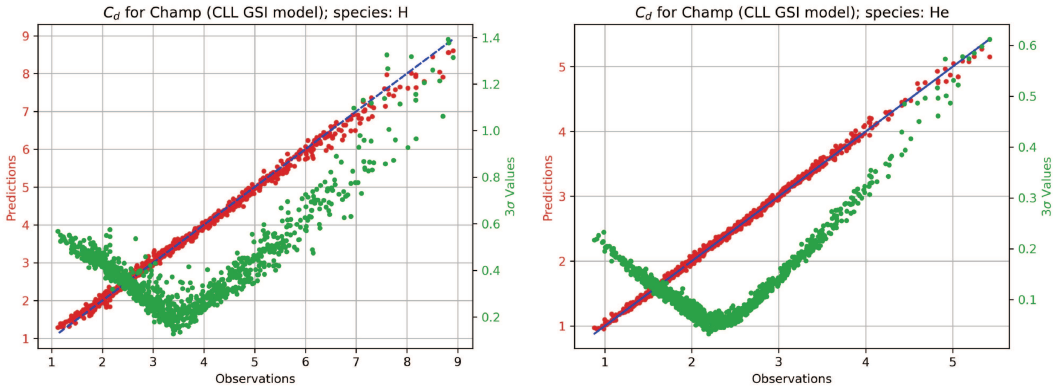


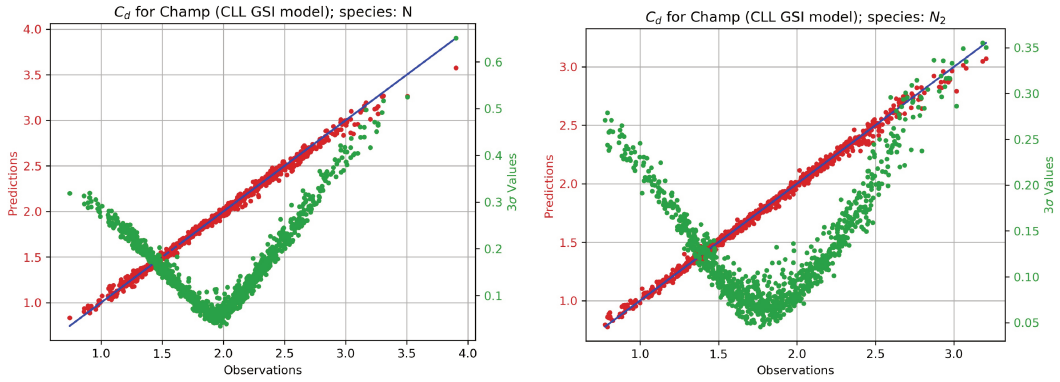
Figure 7. Consistency Plot for Sphere for Drag Coefficient Prediction Using Feed-forward DNN for the Test Dataset. Performance of the NLPD Training Loss Function is Compared to the MSE Training Loss Function.

Next, a feed-forward DNN with Monte Carlo dropout approach and NLPD loss function is trained to predict the drag coefficient of Champ satellite corresponding to different species. Fig. 7 shows the predicted drag coefficients. The red markers compare the predicted means and the numerical TPMC values, while the green markers show the 3σ uncertainty values. The RMSE between the predicted and TPMC drag coefficient values for H, He, N, N_2, O, O_2 are 0.109611, 0.037761, 0.037959, 0.031282, 0.039637, 0.033111, respectively. Similar to the case of sphere, the DNN uncertainty predictions for Champ (Fig. 7) are larger than the Gaussian Process uncertainty predictions for Champ (Fig. 2). The corresponding consistency plots are shown in Fig.8.



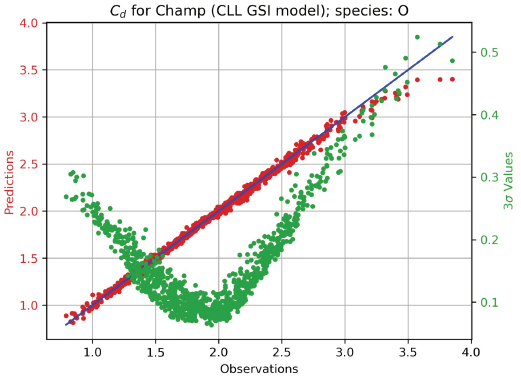
(a) Atomic hydrogen, H

(b) Helium, He

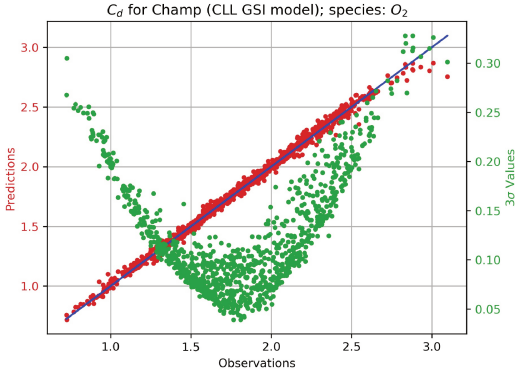


(c) Atomic Nitrogen, N

(d) Nitrogen, N_2



(e) Atomic Oxygen, O



(f) Oxygen, O_2

Figure 7. Comparison of Feed-Forward DNN (With Monte Carlo Dropout) and TPMC Drag Coefficients for Champ Test Dataset for Different Species.

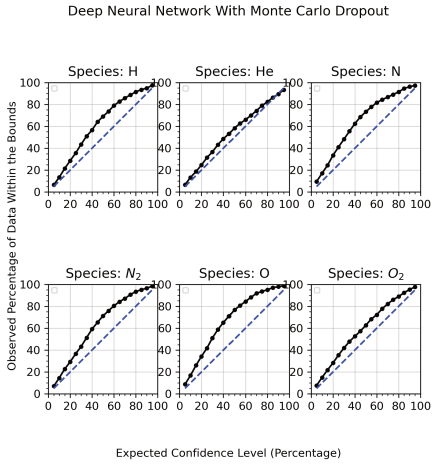


Figure 8. Consistency Plot for Champ for Drag Coefficient Prediction Using Feed-forward DNN for the Test Dataset.

ORBIT UNCERTAINTY QUANTIFICATION DUE TO UNCERTAINTY IN DRAG COEFFICIENT UNDER VARIOUS SOLAR AND GEOMAGNETIC ACTIVITY LEVELS

This work relies on Monte Carlo simulations to study the effect of uncertainty in drag coefficient on orbital state uncertainties. The details of the simulations for a spherical satellite and the Champ satellite are described in the underneath points:

1. Starting from the initial position described in Table 3, objects are propagated for a duration of 4 days. The initial epoch is taken as 21:49:18.64 UT, March 4, 2010.
2. For orbit propagation, we consider only the Earth's central gravitational term and the dominant perturbation forces. At 300 km altitude, the dominant perturbation forces are the atmospheric drag and the J_2 perturbation resulting from Earth's oblateness. The atmospheric drag is modeled using the NRLMSISE-00 density model [30]. The NRLMSISE-00 density model is a function of the geomagnetic Ap-index and the 10.7 cm solar radio flux (F10.7). Orbits are propagated under three different $\{Ap, F10.7\}$ combinations, as given in Table 4.
3. Orbit integration is performed using a modified version of Dormand and Prince's Runge-Kutta Method [31] (also referred to as 'RK45' integrator in Python's scipy.integrate package). The modified integrator uses a constant integration step size of 10 seconds rather than striving for specified absolute and relative tolerances. This modification was made because a variable step size integrator takes a long time to converge in presence of stochastic drag coefficient, whose value changes in every internal adjustment of a single call of the step size.
4. Mass of both the spherical object and the Champ satellite is taken as 489.166 kg. The cross-sectional area for the sphere is taken as 0.770981 m^2. The cross-sectional area for the Champ satellite is dependent on satellite attitude and is obtained by using 2-D linear interpolation on a look-up table. The attitude dynamics of the Champ satellite is taken as: satellite pitch $= \sin(100t)^0$, satellite yaw $= 5\cos(100t)^0$, where t is the time in days since initial epoch.
5. There are, in total, twelve cases that are simulated consisting of two objects (sphere/Champ), 3 geomagnetic-solar conditions (high Ap-high F10.7/medium Ap-high F10.7/low Ap-low F10.7), and 2 different stochastic models for computing drag coefficient (Gaussian process/ DNN with Monte Carlo dropout). For each of these cases, 500 Monte Carlo runs are simulated. For example, let's consider the case of the Champ Satellite under low solar and geomagnetic activities and drag coefficient modeling using Gaussian process regression. An instance of the Champ satellite is generated at its initial position at the initial epoch. For each of the species H, He, N, N_2, O, O_2, a drag coefficient value is sampled from the corresponding normal distribution predicted by the Gaussian process. The individual drag coefficient values are combined (discussed in the next point) to compute the total drag coefficient. The object is propagated for 10 seconds, and the whole task of sampling drag coefficients is repeated. The propagation continues for a total of 4 days. The whole simulation, starting from generating an instance of the Champ satellite, is repeated 500 times.
6. The total drag coefficient referred to in the previous point is obtained by using [32, 33]:

$$C_D = f_{sc} C_{D_{ads}} + (1 - f_{sc}) C_{D_{surf}} \tag{10}$$

where $C_{D_{ads}}$ is the total drag coefficient based on a satellite completely covered by the adsorbate (atomic oxygen), and $C_{D_{surf}}$ is the total drag coefficient based on a clean satellite surface. The weight f_{sc} is given as [33]:

$$f_{sc} = \frac{K_{CLL} P_o}{1 + K_{CLL} P_o} \tag{11}$$

where K_{CLL} is the Langmuir adsorbate constant for the CLL model (= 2.89×10^6) and P_O is the partial pressure of atomic oxygen. The adsorbate and the surface drag coefficients are obtained from the drag coefficients of constituent species (H, He, N, N_2, O, O_2) using [33]:

$$C_{D_{ads/surf}} = \left(\frac{1}{\sum_{k=1}^{6}(\chi_k m_k)} \right) \sum_{k=1}^{6} (\chi_k m_k C_{D_{ads/surf},k}) \tag{12}$$

where χ_k is the mole fraction of species k, m_k is the mass of species k, and $C_{D_{ads/surf},k}$ is the drag coefficient for species k. The adsorbate drag coefficient corresponding to species k, i.e., $C_{D_{ads},k}$, is obtained by sampling from the distribution predicted by the Gaussian process or the DNN model with inputs: [v_∞, 400K, T_∞, 1, 1, β, ϕ]. Similarly, the surface drag coefficient corresponding to species k, i.e., $C_{D_{surf},k}$, is obtained by sampling from the distribution predicted by the Gaussian process or the DNN model with inputs: [v_∞, 400K, T_∞, $\alpha_{n_{sub}}$, 1, β, ϕ]. The atmospheric translational temperature, T_∞ is obtained from the NRLMSISE-00 model. The parameter $\alpha_{n_{sub}}$ is obtained using:

$$\alpha_{n_{sub}} = max\left\{ 2\left(\frac{3\mu_n}{(1+\mu_n)^2} \right) - 1, 0 \right\} \tag{13}$$

$$\mu_n = \frac{\sum_{k=1}^{6}(\chi_k m_k)}{m_{surf}} \tag{14}$$

where m_{surf} is the mass of a particle that composes the surface lattice (=28 amu).

7. For each of the twelve cases, orbital state uncertainties are characterized in terms of the radial, along-track, and cross-track errors [34] of the 500 Monte Carlo runs with respect to a reference orbit at the end of the 4-day propagation period. The reference orbit is obtained by propagating the object using mean drag coefficient values.

Table 3. Keplerian Elements Defining the Initial Position of the Satellites

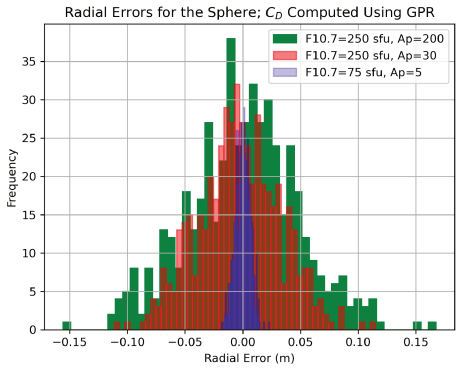
Orbital Element	Values
Semi-major Axis, a	6674127.099236 m
Eccentricity, e	.000221
Inclination, i	87.2193^0
True Anomaly, ν	274.488696^0
Argument of Perigee, ω	85.6397^0
RAAN, Ω	206.9785^0

Table 4. Space Weather Scenarios With Different Geomagnetic and Solar Activity Levels

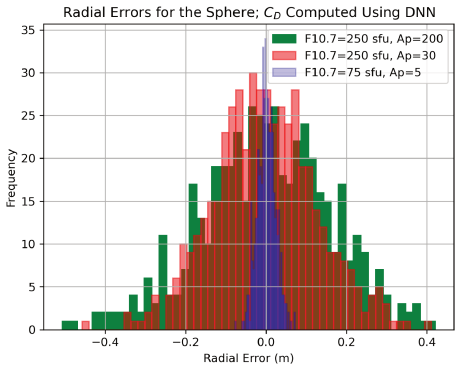
Scenario	Values
Scenario 1: High Geomagnetic Activity, High Solar Activity	$Ap = 200$, $F10.7 = 250$ sfu
Scenario 2: Medium Geomagnetic Activity, High Solar Activity	$Ap = 30$, $F10.7 = 250$ sfu
Scenario 3: Low Geomagnetic Activity, Low Solar Activity	$Ap = 5$, $F10.7 = 75$ sfu

Fig. 9 shows the distribution of radial, along-track, and cross-track errors between the Monte Carlo runs and the corresponding reference orbits for the sphere. The green histogram corresponds to high solar and geomagnetic activities, the red histogram corresponds to high solar and medium geomagnetic activities, and the blue histogram corresponds to low solar and geomagnetic activities. The spread in the radial, in-track, and cross-track errors are much larger under high solar and

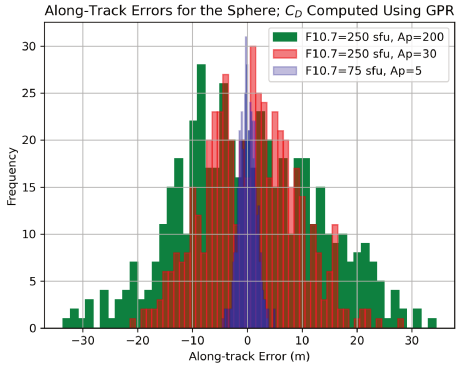
geomagnetic activities (red histograms) as compared to low solar and geomagnetic activities (blue histograms). The skewness of the distributions is minimal, and the error distributions are nearly centered at zero. The error spread with the DNN-based drag coefficients is around three times greater than the error spread with the GPR-based drag coefficients. Along-track errors are significantly larger than radial errors, which are significantly larger than cross-track errors.



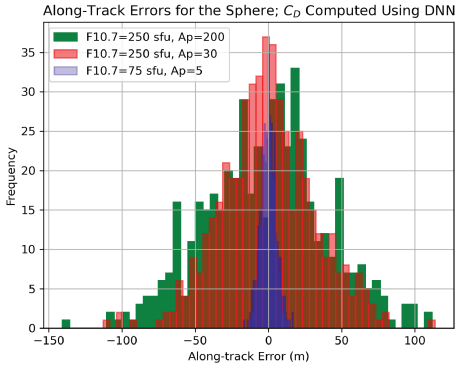
(a) Radial Errors for Sphere (GPR)



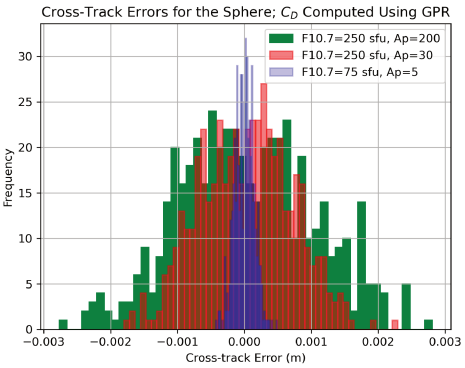
(b) Radial Errors for Sphere (DNN)



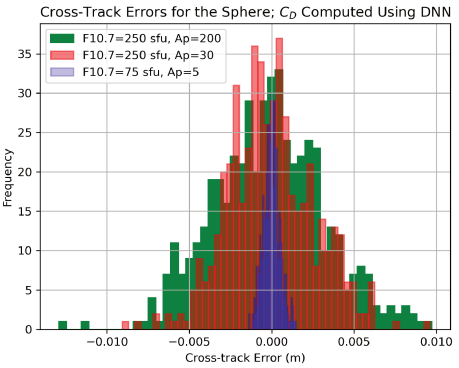
(c) Along-Track Errors for Sphere (GPR)



(d) Along-Track Errors for Sphere (DNN)

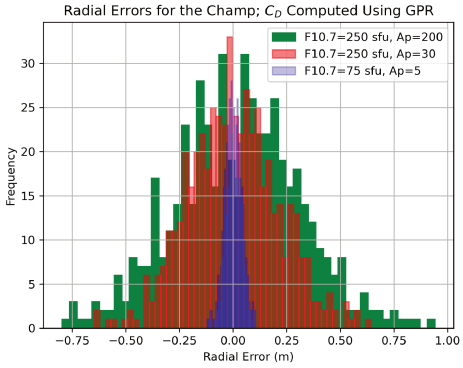


(e) Cross-Track Errors for Sphere (GPR)

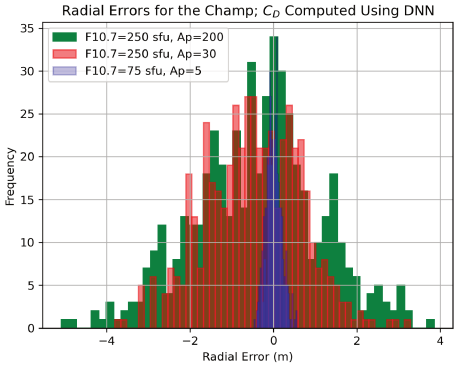


(f) Cross-Track Errors for Sphere (DNN)

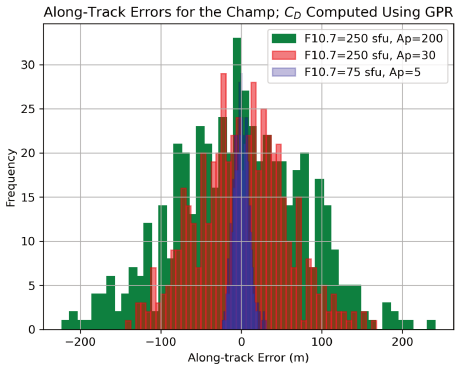
Figure 9. Radial, Along-Track, and Cross-Track Errors for the Sphere Monte Carlo Simulations.



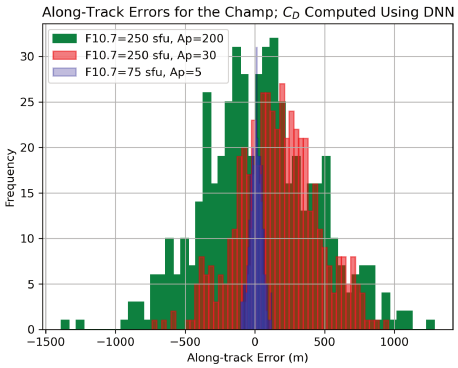
(a) Radial Errors for Champ (GPR)



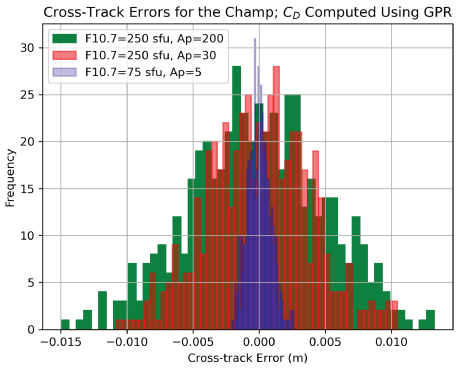
(b) Radial Errors for Champ (DNN)



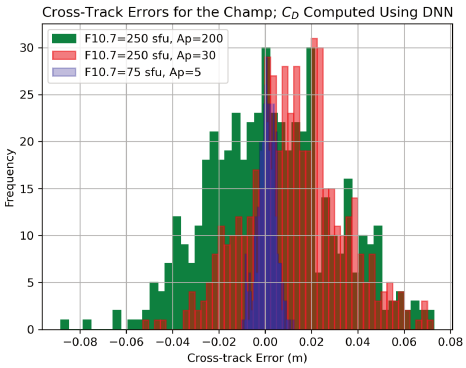
(c) Along-Track Errors for Sphere (GPR)



(d) Along-Track Errors for Champ (DNN)



(e) Cross-Track Errors for Champ (GPR)



(f) Cross-Track Errors for Champ (DNN)

Figure 10. Radial, Along-Track, and Cross-Track Errors for the Champ Monte Carlo Simulations.

Fig. 10 shows the distribution of radial, along-track, and cross-track errors between the Monte Carlo runs and the corresponding reference orbits for the Champ satellite. The green histogram corresponds to high solar and geomagnetic activities, the red histogram corresponds to high solar

and medium geomagnetic activities, and the blue histogram corresponds to low solar and geomagnetic activities. When compared to the case of the sphere, the spread in the distributions is larger for the Champ satellite. This is because of the larger drag coefficient uncertainty predictions in the case of Champ. The error spread with the DNN-based drag coefficients is roughly five times greater than the error spread with the GPR-based drag coefficients. Similar to the case of the sphere, the along-track errors are larger than the radial errors, which are larger than cross-track errors.

CONCLUSIONS

In this paper, we make probabilistic predictions of the drag coefficients of a spherical satellite and the Champ satellite in the low Earth orbit region. Estimates of the drag coefficients corresponding to H, He, N, N_2, O, O_2 species are obtained using the following stochastic machine learning models: (i) Gaussian process regression (ii) deep neural network with Monte Carlo dropout approach. Numerical test Particle Monte Carlo (TPMC) method with Cercignani–Lampis–Lord gas-surface interaction model is used to generate the training data. For the training dataset, the independent variables are the relative velocity of the satellite, the satellite surface temperature, the atmospheric translational temperature, the normal energy accommodation coefficient, the tangential momentum accommodation coefficient, satellite yaw, and satellite pitch. The stochastic models are trained on 1000 samples (1000 samples for Gaussian process; 850 training samples plus 150 validation samples for deep neural network), which are carefully selected using Latin Hypercube Sampling. The trained models are then tested on a separate dataset of 1000 samples.

To ensure that our models produce meaningful uncertainty estimates, we investigate methods to calibrate the models. Both the post-training calibration method and the method of using a loss function with better calibration properties are investigated. We focus on the latter method. The negative logarithm of the probability density (NLPD) loss function is preferred over the commonly used mean squared error loss function because of better calibration properties and because it assumes heteroscedasticity.

The root mean squared error between the Gaussian process predictions and observed test values for H, He, N, N_2, O, O_2 for the sphere are 0.005397, 0.003799, 0.003136, 0.003207, 0.003214, 0.002975, respectively. And, the root mean squared error between the Gaussian process predictions and observed values for Champ for H, He, N, N_2, O, O_2 are 0.033636, 0.019427, 0.013985, 0.013991, 0.014714, 0.013326, respectively. The uncertainty estimates for the sphere are better calibrated than the uncertainty estimates of the Champ satellite.

For deep neural network predictions, the KerasTuner library is used to obtain the near-optimal hyper-parameters in this work. The networks for drag coefficient predictions are trained using the Keras-Tuner provided architectures and the NLPD loss function. The Monte Carlo dropout approach, which is typically used only in the training phase, is kept 'on' during the testing in this work to produce stochastic predictions. The root mean squared error between the deep neural network predictions and TPMC values for H, He, N, N_2, O, O_2 for the sphere are 0.007912, 0.005833, 0.004484, 0.004781, 0.004794, 0.004581, respectively. The root mean squared error between the predicted and TPMC drag coefficient values for H, He, N, N_2, O, O_2 for the Champ are 0.109611, 0.037761, 0.037959, 0.031282, 0.039637, 0.033111, respectively. The deep neural network uncertainty estimates are larger than the uncertainty estimates from the Gaussian process regression.

The effect of drag coefficient uncertainty on the orbital state uncertainties is investigated by performing Monte Carlo simulations under three different space weather scenarios: (i) high solar and

geomagnetic activities, (ii) high solar and medium geomagnetic activities, (ii) low solar and geomagnetic activities. For both sphere and the Champ satellite, for each of the space weather scenarios, and for each of the stochastic methods of drag coefficient predictions, 500 Monte Carlo objects are propagated for a duration of 4 days. The radial, along-track, and cross-track errors between the propagated Monte Carlo orbits and reference orbit (obtained from orbit propagation using mean drag coefficient estimates) are investigated. It is observed that the spread in the distributions of the errors is much larger under high solar and geomagnetic activity levels as compared to low solar and geomagnetic activity levels. The largest errors are found in the along-track direction followed by the radial direction followed by the cross-track direction. Compared to the Gaussian process predictions, neural network predictions resulted in more spread in the error distributions.

We observe a parabolic trend in the deep neural network predicted uncertainties for the sphere/ Champ test dataset. No such trend was observed in the case of Gaussian process predictions. In the future, we intend to investigate the reason for this observation.

REFERENCES

[1] P. M. Mehta, A. Walker, E. Lawrence, R. Linares, D. Higdon, and J. Koller, "Modeling Satellite Drag Coefficients With Response Surfaces," *Advances in Space Research*, Vol. 54, No. 8, 2014, pp. 1590–1607.

[2] H. W. Jones, "The Recent Large Reduction in Space Launch Cost," *48th International Conference on Environmental Systems*, 2018.

[3] J. C. McDowell, "The Low Earth Orbit Satellite Population and Impacts of the SpaceX Starlink Constellation," *The Astrophysical Journal Letters*, Vol. 892, No. 2, 2020.

[4] C. D. Bussy-Virat, A. J. Ridley, and J. W. Getchius, "Effects of Uncertainties in the Atmospheric Density on the Probability of Collision Between Space Objects," *Space Weather*, Vol. 16, No. 5, 2018, pp. 519–537.

[5] D. J. Gondelach and R. Linares, "Atmospheric Density Uncertainty Quantification for Satellite Conjunction Assessment," *AIAA Scitech 2020 Forum*, 2020.

[6] F. Schiemenz, J. Utzmann, and H. Kayal, "Propagation of Grid-scale Density Model Uncertainty to Orbital Uncertainties," *Advances in Space Research*, Vol. 65, No. 1, 2020, pp. 407–418.

[7] L. Sagnieres and I. Sharf, "Uncertainty Characterization of Atmospheric Density Models for Orbit Prediction of Space Debris," *7th European Conference on Space Debris*, 2017.

[8] J. Emmert, H. Warren, A. Segerman, J. Byers, and J. Picone, "Propagation of Atmospheric Density Errors to Satellite Orbits," *Advances in Space Research*, Vol. 59, No. 1, 2017, pp. 147–165.

[9] D. M. Prieto, B. P. Graziano, and P. C. Roberts, "Spacecraft Drag Modelling," *Progress in Aerospace Sciences*, Vol. 64, 2014, pp. 56–65.

[10] L. Weng, J. Lei, J. Zhong, X. Dou, and H. Fang, "A Machine-Learning Approach to Derive Long-Term Trends of Thermospheric Density," *Geophysical Research Letters*, Vol. 47, No. 6, 2020.

[11] R. J. Licata, P. M. Mehta, and W. K. Tobiska, "Impact of Space Weather Driver Forecast Uncertainty on Drag and Orbit Prediction," *Astrodynamics Specialist Conference*, 2020.

[12] R. J. Licata and P. M. Mehta, "Physics-informed Machine Learning with Autoencoders and LSTM for Probabilistic Space Weather Modeling and Forecasting," *100th American Meteorological Society Annual Meeting*, 2020.

[13] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," *arXiv:1506.02142*, 2015.

[14] D. H. Davis, "Monte Carlo Calculation of Molecular Flow Rates through a Cylindrical Elbow and Pipes of Other Shapes," *Journal of Applied Physics*, Vol. 31, No. 7, 1960, pp. 1169–1176.

[15] C. Cercignani and M. Lampis, "Kinetic Models for Gas-surface Interactions," *Transport Theory and Statistical Physics*, Vol. 1, No. 2, 1971, pp. 101–114.

[16] R. G. Lord, "Some Extensions to the Cercignani–Lampis Gas–surface Scattering Kernel," *Physics of Fluids A: Fluid Dynamics*, Vol. 3, No. 4, 1991, pp. 706–710.

[17] M. D. McKay, R. Beckman, and W. Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, Vol. 21, No. 2, 1979, pp. 239–245.

[18] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions, Section 9.6*. 1965.

[19] C. E. Rasmussen and K. I. Williams, *Gaussian Processes for Machine Learning*. Massachusetts Institute of Technology: MIT Press, 2006.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, Vol. 12, No. 85, 2011, pp. 2825–2830.

[21] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *arXiv:1404.7828*, 2014.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, Vol. 15, No. 56, 2014, pp. 1929–1958.

[24] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. Software available from tensorflow.org.

[25] P. Kumar, S. Batra, and B. Raman, "Deep neural network hyper-parameter tuning through twofold genetic approach," *Soft Computing*, Vol. 25, 2021, p. 8747–8771.

[26] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, *et al.*, "Keras Tuner," `https://github.com/keras-team/keras-tuner`, 2019.

[27] E. Camporeale and A. Carè, "Estimation of Accurate and Calibrated Uncertainties in Deterministic models," *arXiv:2003.05103*, 2020.

[28] G. J. Anderson, J. A. Gaffney, B. K. Spears, P.-T. Bremer, R. Anirudh, and J. J. Thiagarajan, "Meaningful uncertainties from deep neural network surrogates of large-scale numerical simulations," *arXiv:2010.13749*, 2020.

[29] M.-H. Laves, S. Ihler, J. F. Fast, L. A. Kahrs, and T. Ortmaier, "Recalibration of Aleatoric and Epistemic Regression Uncertainty in Medical Imaging," *arXiv:2104.12376*, 2021.

[30] J. M. Picone, A. E. Hedin, D. P. Drob, and A. C. Aikin, "NRLMSISE-00 empirical model of the atmosphere: Statistical comparisons and scientific issues," *Journal of Geophysical Resesarch*, Vol. 107, No. A12, 2002, pp. SIA 15–1–SIA 15–16.

[31] J. R. Dormand and P. J. Prince, "A family of embedded Runge-Kutta formulae," *Journal of Computational and Applied Mathematics*, Vol. 6, No. 1, 1980, p. 19–26.

[32] P. Mehta, A. Walker, C. A. McLaughlin, and J. Koller, "Comparing Physical Drag Coefficients Computed Using Different Gas-Surface Interaction Models," *JOURNAL OF SPACECRAFT AND ROCKETS*, Vol. 51, No. 3, 2014, pp. 873–883.

[33] A. Walker, P. Mehta, and J. Koller, "Drag Coefficient Model Using the Cercignani–Lampis–Lord Gas–Surface Interaction Model," *JOURNAL OF SPACECRAFT AND ROCKETS*, Vol. 51, No. 5, 2014, pp. 1544–1563.

[34] D. A. Vallado, *Fundamentals of Astrodynamics and Applications, Fourth Edition*. Hawthorne, CA: Microcosm Press, 2013.