Rapid and Uncertainty Quantified Orbital Propagation using Uncertainty-Aware AI

Kevin Vanslette, Alexander Lay
RTX BBN Technologies
David Kusterer, Kevin Schroeder
Virginia Tech National Security Institute

ABSTRACT

Due to the proliferation of Resident Space Objects (RSOs) in Low Earth Orbit (LEO), the task of real-time orbital tracking, propagation, conjunction prediction, and Space Domain Awareness (SDA) over the entire LEO belt is computationally expensive using physics-based methods. Alternatively, traditional data-driven Artificial Intelligence (AI) methods, while being fast and able to produce state-of-the-art metric performance over complex data, can produce overly confident, yet wrong, predictions without warning. We demonstrate the utility of an augmented, covariance predicting, AI framework for scalable orbital propagation of RSOs in LEO. The relative size of the predictive covariance is an indicator of how well this "uncertainty-aware" AI approach understands a given RSO's orbit, and in outlier situations, it overtly predicts large amounts of uncertainty rather than silently producing wrong results. This is a key feature for robust and performant SDA applications because it enables down-stream decision systems to choose whether a given AI prediction should be trusted and utilized or ignored and replaced. Trained over a relatively small dataset, the method forecasts the orbital trajectories of 34K RSOs one day into the future (5-minute intervals) on a single Graphical Processing Unit (GPU) in 5.5 seconds with 46.6 km Root Mean Squared Error (RMSE). Single orbit forecasts maintained RMSEs as low as 5.8 km and were able to propagate 1 million ephemerides in 12.1 seconds. Thus, the method offers practitioners a solution in the trade space between computational speed and accuracy while dynamically quantifying expected performance in terms of predicted uncertainty. This preliminary work begins exploring the utility of uncertainty predicting AI for scalable space domain awareness.

1. INTRODUCTION

It is no surprise that most Space Domain Awareness (SDA) applications have considerable functional dependence on the current and future position estimates of Resident Space Objects (RSOs). At the lowest level, and due to observation intermittence, RSO orbital track estimates need to be maintained so future observations can be properly correlated with them. Forecasts of these tracks are important decision variables in higher level SDA planning applications for space traffic management, conjunction and collision avoidance, higher level data fusion analytics, and space debris cleanup planning. Any error, uncertainty, and/or latency in the production of these track forecasts propagate to down-stream SDA applications, which diminishes their utility. The community has expended significant modeling effort and computational resources create high fidelity and scalable solutions for maintaining and forecasting RSO tracks with uncertainty.

For applications like space traffic management in the Low Earth Orbit (LEO) regime, tracks and track forecasts need to be maintained to identify potential conjunctions and collisions so avoidance maneuvers can be scheduled. ESA's 2024 Space Environment Report [1] estimates there are about 34K objects greater and 10cm in LEO and nearly 1 million between 1 and 10cm, both of which can create computational challenges for space traffic management. While physics-based parametric models for tracking and propagation have long-standing and proven high fidelity, reliability, explainability, and an ability to quantify and forecast track uncertainty, the methods themselves can require significant computational resources at this scale. Speed (or computational complexity) and accuracy can be traded off by choosing which special perturbations to use in a given model; however, there is a significant gap in the speed-accuracy trade space between simple Keplerian motion and J2 or other special perturbations.

Alternatively, there are many classes of non-parametric Artificial Intelligence (AI) models that may be able to populate useful regions within the speed-accuracy trade space to give more modeling flexibility to practitioners. AI models

move the bulk of the computational heavy lifting into offline training phases which generates models that can be evaluated quickly and with relatively low computational complexity due to Graphical Processing Unit (GPU) parallelization.

AI modeling approaches, while being known for being able to make fast and accurate predictions over complex datasets, often still fail to meet the high reliability and robustness standards that would allow for their adoption into critical space domain applications. In particular, in outlier scenarios, traditional AI methods often and without warning produce overly confident, yet wrong, predictions [2]. This is to say, it is not good enough to train models that make state-of-the-art accurate predictions over "inlier" training datasets if the model then fails to generalize appropriately in real world applications that may include "outliers". This is a major problem for AI-based SDA applications because not only is the prediction wrong, it is wrong without warning of possible errors, which could compromise the integrity of downstream processing and decision making.

Outlier scenarios arise when applying an AI model outside of the domain of the training data. Deviations from the domain of the training data may be driven dynamically by environmental non-stationary or may be due to induced scope/domain training and testing mismatches. For the problem of ephemeris propagation in LEO, the former would include things like increased drag in LEO [3] due to solar maximum (11 year cycle), while the later would include expecting training a model on one orbital regime to generalize to another or expecting models trained on low fidelity simulators that do not reflect reality well and testing on real data.

We believe a subfield of AI called Uncertainty-Aware AI (UA-AI) could be of considerable interest to the SDA community. UA-AI models differ from traditional AI models in that they are trained not only to make point predictions (i.e. a mean or classification probability vector), but also to predict distributional parameters (e.g. the variance around a classification probability vector [4]) as additional outputs from the neural network. While it is relatively straightforward for these models to learn the aleatoric (statistical) uncertainty of inlier dataset and model pair, an ongoing challenge is getting these models to reliably estimate their own predictive uncertainty when presented with outlier data, as is reviewed in [5]. However, training a small ensemble of uncertainty-aware networks (over the same data, about 5 models) and probabilistically combining their results has been shown to significantly improve uncertainty estimation in outlier data regions, that is, it reliably predicts large uncertainty there. This ensemble method exploits inter-model disagreement due to the differences in the learned high dimensional model parameters [6]. UA-AI approaches exist for regression, classification, object detection and segmentation based on these concepts and UA-AI attempts to tackle the problem of outliers directly through predictive uncertainty and a variety of UA-AI modeling approaches exist.

To demonstrate the feasibility of this approach, we apply UA-AI techniques to the problem of ephemeris forecasting in LEO. The dataset consists of the ephemerides of about 19K RSOs simulated over a 3 day period (5 minute intervals) generated using a high fidelity numerical propagator. The data includes cartesian position and velocity, their complementary orbital elements, and relative positions from the RSO to the Moon and Sun. We trained a small ensemble of UA-AI feed forward neural networks (with convolutional layers) that jointly forecast entire ephemeris trajectories in a single pass as well as their multi-time covariance matrices. We performed two different forecasting cases, the first predicts ephemeris up to 100 minutes ahead ("single orbit") and the second predicts ephemeris up to 1 day ahead along with their covariance matrices. We explored performance gains and losses of predicting fully specified multi-time covariance matrices vs block diagonal covariance matrices that had no inter-time correlations modelled. We demonstrate that the approach is robust to outliers by training the model on ephemeris with small eccentricity but tested them on ephemeris with the largest eccentricity in our dataset. We also explored using a multi-modal multivariate Gaussian mixture model, which alone was less robust to outliers than the ensemble of unimodal Gaussian mixtures.

We introduce the data and simulation in Section 2, the UA-AI approached used in Section 3, experimental setup and results in Section 4, and we conclude with a discussion of applications and expansions of this work in Section 5.

2. DATA AND SIMULATION

The propagation of the orbital states of RSOs within the LEO regime was conducted using a robust and precise numerical integrator. The scope of the simulation encompassed a total of 19,210 RSOs, across various orbital altitudes for a span of 3 days. The numerical propagator used was extracted from the REsponsive Space ObservatioN Analysis and Autonomous Tasking Engine (RESONAATE) [7, 8], previously developed by researchers at the Virginia Tech National Security Institute (VTNSI).

The RESONAATE tool is a discrete time Space Object Surveillance and Identification (SOSI) modeling and simulation tool with a verified turnkey orbital propagator built-in. A key feature of the tool is the modularity of the spacecraft dynamics; therefore, the dynamics can be as precise as the scenario requires. The model can incorporate whichever gravity model (e.g. WGS84, EGM96, EGM2008) is needed with precise Earth procession and nutation parameters included. Sun and Moon, as well as other third body perturbations can be toggled on or off depending on the scenario. Solar radiation pressure, Earth albedo, and relativistic effects are also included as options to provide high-fidelity state estimation and covariance.

The propagation of the RSOs was performed using the Special Perturbations (SP) method, a high-fidelity approach that computes the trajectory of each object by numerically integrating the full set of perturbative forces acting on it. Among the perturbative forces accounted for were the gravitational influences of third bodies, specifically the Sun and Moon. These third-body perturbations are critical for accurately modeling the long-term evolution of orbits in the LEO regime, where gravitational interactions with these celestial bodies can induce significant deviations from purely Keplerian motion.

In addition to third-body perturbations, the Earth's gravitational field was modeled using the EGM96 geopotential model, truncated at degree 4 and order 4. This truncation was chosen to balance computational efficiency with the need for accuracy in modeling the Earth's gravitational perturbations. The EGM96 model, with its detailed representation of the Earth's gravitational anomalies, is particularly suited for precise orbit determination and prediction in the LEO environment.

Notably, the simulation did not account for the effects of drag, solar radiation pressure, or solid dynamic tides. The exclusion of these perturbation was deemed acceptable given the relatively short duration of the propagation and the focus on gravitational perturbations. While these perturbations can be important in certain high-precision scenarios, and are especially import for very low orbits, they were considered negligible for the scope of this simulation.

The numerical integration of the equations of motion was carried out using the 'RK45' solver, a member of the Runge-Kutta family, implemented in the 'scipy.integrate' package. The 'RK45' method was selected for its adaptive step size control, which dynamically adjusts the step size to maintain the specified error tolerances. This adaptive approach is particularly advantageous when dealing with the varying dynamical conditions encountered in LEO, where orbital velocities and perturbative forces can change rapidly. To ensure a high level of numerical precision, the integration was performed with a relative tolerance of 10^{-10} and an absolute tolerance of 10^{-12}. These stringent tolerances were essential for maintaining the accuracy of the propagation over the extended simulation period. The chosen tolerances also helped to mitigate the accumulation of numerical errors, which can become significant in long-term integrations of orbital systems.

The generated ephemerides were transformed and expressed using 6 cartesian coordinates, 6 classical orbital elements, plus 6 coordinates indicating relative position of the Moon and Sun:

$$\mathbf{x}_t^{(j)} = \Big[x, y, z, v_x, v_y, v_z \Big| a, e, i, \Omega, \omega, T \Big| x_m, y_m, z_m, x_s, y_s, z_s \Big]_t^{(j)}, \tag{1}$$

all of which are indexed by timestamp t and RSO identification number j.

The computational resources allocated to each data run consisted of 32 CPU cores, allowing for parallel processing to optimize the computational efficiency. The hardware utilized in this simulation was based on AMD EPYC™ 7542 processors, part of the EPYC 7002 Series designed specifically for server applications. Each processor features 32 CPU cores and 64 threads, operating at an overclock speed of up to 3.4 GHz. The processors are equipped with 128 MB of L3 cache, providing ample memory bandwidth for the intensive calculations required by the simulation.

The data run was distributed across four independent nodes. Each data run was tasked with propagating approximately 4,800 RSOs. Given this hardware configuration, each data run completed in approximately 115.28 hours. The processing rate was approximately 0.625 times real-time, indicating that the simulation covered 72 hours of orbital evolution in 115.28 hours of wall-clock time. This ratio underscores the computational demand of accurately propagating many RSOs over extended periods while maintaining high precision. While neither the hardware nor the software was fully optimized for this application, the described system is representative of the computational expense of the traditional physics-based orbital propagators. Overall, the combination of high-performance computing resources, high-fidelity perturbation modeling, and precise numerical integration enabled the accurate propagation of a large and diverse set of RSOs in the LEO environment. The results from this simulation provide a detailed and reliable basis for subsequent

analyses of orbital behavior and collision risk assessment in LEO.

3. UNCERTAINTY-AWARE AI APPROACH

We introduce the proposed UA-AI approach for ephemeris forecasting. In particular, we propose treating the problem like a multi-dimensional regression problem with fixed input and output sizes rather than as a sequential timeseries problem using recurrent neural networks (RNNs) or long-short term memory (LSTM) models.

Many physics-based solutions in this domain are sequential in nature, which results in error profiles that grow nonlinearly in the number of propagation steps. Further, the sequential nature of these approaches creates a computational bottleneck is that the previous step needs to be computed before the next step can be computed. While the proposed multi-dimensional regression method has fixed input and output sizes, it makes predictions in parallel across the ephemeris, increasing speed, while also not having the characteristic nonlinear error profile of sequential methods.

3.1 Deep Multivariate Gaussian Model

Let a sample of input feature data (past multi-dimensional window of orbital data) be represented by a k_x dimensional vector $\mathbf{x} \in \mathbf{R}^{k_x}$ and its corresponding label (future multi-dimensional window of positional ephemeris) be a represented by a k_y dimensional vector $\mathbf{y} \in \mathbf{R}^{k_y}$. We model a deep multivariate Gaussian distribution following [9],

$$\rho(\mathbf{y}|\mu(\mathbf{x}),\Sigma(\mathbf{x})) = \frac{1}{\sqrt{(2\pi)^{k_y}|\Sigma(\mathbf{x})|}}\exp\Big(-\frac{1}{2}(\mathbf{y}-\mu(\mathbf{x}))^T\Sigma(\mathbf{x})^{-1}(\mathbf{y}-\mu(\mathbf{x}))\Big), \tag{2}$$

where $\mu(\mathbf{x})$ is $\mu\colon \mathbf{R}^{k_x} \to \mathbf{R}^{k_y}$ and $\Sigma(\mathbf{x})$ is $\Sigma\colon \mathbf{R}^{k_x} \to \mathbf{R}^{k_y \times k_y}$ are outputs from a neural network. The data $(\mathbf{x},\mathbf{y})$ is treated like a completely certain quantity and the mean and covariance parametrically model the expected uncertainty, or error, in the neural prediction as a multivariate Gaussian. The neural network is trained to minimize the negative log likelihood,

$$\begin{aligned}\mathscr{L}(\mathbf{x},\mathbf{y}) &= -\ln(\rho(\mathbf{y}|\mu(\mathbf{x}),\Sigma(\mathbf{x}))) \\ &= \frac{1}{2}(\mathbf{y}-\mu(\mathbf{x}))^T\Sigma(\mathbf{x})^{-1}(\mathbf{y}-\mu(\mathbf{x})) + \frac{1}{2}\ln(|\Sigma(\mathbf{x})|) + \text{const.},\end{aligned} \tag{3}$$

which is averaged over batches of data samples. The covariance matrix $\Sigma(\mathbf{x})$ is forced to be is symmetric positive definite through Cholesky decomposition,

$$\Sigma(\mathbf{x}) = L(\mathbf{x})L(\mathbf{x})^T, \tag{4}$$

where $L(\mathbf{x})$ is a real positive lower triangular matrix, which we guarantee through the choice of positive activation functions along the diagonal elements of L. It should be noted that the predictive covariance in this case represents the AI model's uncertainty in its prediction of the ephemeris (most similar to a quantification of expected error) and is not equal to the process noise driven state covariance matrices that would be generated as part of a Kalman Filter paradigm orbital propagator.

Because the data is normalized between $[-1,1]$ prior to training[1] and because k_y is relatively large in this orbital propagation problem, direct computations of the determinant and matrix inverse are not numerically stable. This is because the computation of the determinant involves computing k_y products of values < 1, which can become exponentially small. We largely circumvent this issue by using logarithm tricks while also forcing the neural model to directly predict the inverse covariance matrices,

$$L(\mathbf{x})L(\mathbf{x})^T \longrightarrow \Sigma(\mathbf{x})^{-1} = L(\mathbf{x})L(\mathbf{x})^T, \tag{5}$$

where we redefine $L(\mathbf{x})$ to be the Cholesky decomposition of the (also symmetric positive semidefinite) inverse covariance matrix $\Sigma(\mathbf{x})^{-1}$ instead, for notational simplicity in all future equations. In addition to helping with numerical stability, this also speeds up training time by avoiding inverse matrix evaluations and back propagation gradients. The log of the determinant of the inverse covariance is then,

$$\ln(|\Sigma^{-1}(\mathbf{x})|) = 2\ln(|L(\mathbf{x})|) = 2\ln(\prod_{i=1}^{k_y} L_{ii}(\mathbf{x})) = 2\sum_{i=1}^{k_y}\ln(L_{ii}(\mathbf{x})), \tag{6}$$

[1] Standard AI optimization algorithms are designed to work with data that are normalized prior to training.

where $L_{ii}(\mathbf{x})$ is the ith diagonal element of L due to the eigenvalue product determinant relationship $|L| = \prod_i \lambda_i$ and because the eigenvalues of L, $\lambda_i = L_{ii}(\mathbf{x})$, are equal to the diagonal elements of positive lower triangular matrices. Using these relations, and the identity $\ln(|\Sigma(\mathbf{x})|) = -\ln(|\Sigma^{-1}(\mathbf{x})|)$, we cast the negative log likelihood into a more numerically stable form in terms of L,

$$\mathscr{L}(\mathbf{x},\mathbf{y}) = \frac{1}{2}(\mathbf{y}-\mu(\mathbf{x}))^T(L(\mathbf{x})L(\mathbf{x})^T)(\mathbf{y}-\mu(\mathbf{x})) - \sum_{i=1}^{k_y}\ln(L_{ii}(\mathbf{x})) + \text{const.} \tag{7}$$

We implemented a custom neural network layer to output $\mu(\mathbf{x})$ and $L(\mathbf{x})$ given data $\mathbf{x}$ or neural transformations of $\mathbf{x} \rightarrow f(\mathbf{x})$ from upstream neural layers. This approach is particularly well suited when applications only require inverse covariance matrices (such as for computing Mahalanobis distances), because computing numerically stable inverses of $\Sigma^{-1}(\mathbf{x})$ require using matrix exponentials and logarithms $\Sigma(\mathbf{x}) = \exp(-\ln(\Sigma^{-1}(\mathbf{x})))$ in extreme cases.

We perform a non-exhaustive random hyperparameter search as part of the training process. The table below summarizes the search space: Because the negative log likelihood does not directly try to minimize the error between $\mu(\mathbf{x})$ and $\mathbf{y}$, the optimization would sometimes get "lazy" in its prediction of $\mu(\mathbf{x})$ and use the prediction of large covariance matrices to "hide" what would be considered large predictive errors $|\mu(\mathbf{x})-\mathbf{y}|$ in this domain. To combat this "laziness", we included hyperparameters that scale L, which generally encouraged the optimizer to find more precise $\mu(\mathbf{x})$ predictions while improving negative log likelihood. We also searched over a few activation functions, learning rates, and batch sizes. In addition to implementing a method for predicting fully populated covariance matrices, we also implemented a block diagonal covariance model using sparse tensors. As a final post processing step, covariance matrices are rescaled to reproduce mean squared errors in the training set.

Table 1: Hyperparameter Search Space

Hyperparameter	Values
# of convolution layers	$[0,8]$
# of filters per convolutional layer	$[32,256]$
# of dense neural layers	$[0,15]$
# of neurons per dense layer	$[75,1500]$
Scale L	[1,200]

3.2 Ensembling UA-AI Models for Outlier Robustness

We use a deep ensembles approach [6, 10], but generalized to the multivariate domain through the construction of "ensemble covariance" via the law of total covariance. Like to the law of total variance, the ensemble covariance decomposes into the inlier (learnable, aleatoric) covariance and the outlier (estimated, epistemic) covariance, respectively,

$$\begin{aligned}\text{Cov}[y^i,y^j] &= \text{E}\Big[\text{Cov}[y^i,y^j|g]\Big] + \text{Cov}\Big[\text{E}[y^i|g],\text{E}[y^j|g]\Big] \\ &= \underbrace{\text{E}\Big[\Sigma_g^{i,j}(\mathbf{x})\Big]}_{\textit{inlier covariance}} + \underbrace{\text{Cov}\Big[\mu_g^i(\mathbf{x}),\mu_g^j(\mathbf{x})\Big]}_{\textit{outlier covariance}} \equiv \Sigma_T(\mathbf{x}).\end{aligned} \tag{8}$$

Here, the superscripts i, j are the i, jth components of $\mathbf{y}, \Sigma$, and μ and G are the total number of models in the ensemble. We use uniform $1/G$ ensemble member mixing, and empirically, ensembling $G \sim 5$ models is typically sufficient. Letting the ensemble mean be $\mu_T(\mathbf{x}) = \frac{1}{G}\sum_{g=1}^{G}\mu_g(\mathbf{x})$ and the ensemble (total) covariance be $\Sigma_T(\mathbf{x})$, the resultant ensemble model is a unimodal multivariate Gaussian distribution $\rho(\mathbf{y}|\mu_T(\mathbf{x}),\Sigma_T(\mathbf{x}))$.

Even if the G models are trained over the same data and have the same hyperparameters, the approach (tuned weights/parameter values sets) taken by stochastic training algorithms is to randomly initialize a set of parameters. This means, with great certainty, the learned solutions end up very far away from one another due to the very high dimensional parameter spaces of neural networks. However, in regions with sufficient training data, the training process causes the ensemble member μ predictions to corroborate with one another, leading to small (co)variances of the

μ's. Alternatively, far outside this sufficient data region, and because the models are so far away from one another in parameter space, the ensemble members have no reason to agree with one another and tend to disagree, which leads to an increase in ensemble covariance.

3.3 Deep Multivariate Gaussian Mixture Model

As an alternative to ensembling unimodal deep multivariate Gaussian models, we considered a deep multivariate Gaussian mixture model,

$$\rho(\mathbf{y}|\mu_{1:M}(\mathbf{x}),\Sigma_{1:M}(\mathbf{x}),\alpha_{1:M}(\mathbf{x})) = \sum_{m=1}^{M} \alpha_m(\mathbf{x})\rho(\mathbf{y}|\mu_m(\mathbf{x}),\Sigma_m(\mathbf{x})), \tag{9}$$

where M is the number mixture modes and $\alpha_{1:M}(\mathbf{x})$ are positive mode weights, which normalize to 1 through the choice of an appropriate activation function. The negative log likelihood of the deep multivariate Gaussian mixture takes the form of the log-sum-exponential (LSE) function $\text{LSE}(z_1,...,z_M) = \ln(\sum_{m=1}^{M}\exp(z_m))$, which is a standard function in most scientific computing libraries and is numerically stable. The negative log likelihood of the mixture distribution is

$$\mathbf{L}(\mathbf{x},\mathbf{y},M) = -\text{LSE}(\mathscr{L}'_1(\mathbf{x},\mathbf{y}),...,\mathscr{L}'_M(\mathbf{x},\mathbf{y})), \tag{10}$$

where $\mathscr{L}'_m(\mathbf{x},\mathbf{y}) = -\mathscr{L}_m(\mathbf{x},\mathbf{y}) + \ln(\alpha_m(\mathbf{x}))$ is the negative log likelihood of the mth mode from (7) , $\mathscr{L}_m(\mathbf{x},\mathbf{y})$, plus the log of its corresponding mode weight, $\ln(\alpha_m(\mathbf{x}))$. The number of mixture modes M is a free variable in our custom neural network layer, where $M = 1$ reproduces the unimodal case from Section 3.1, that is, $\mathbf{L}(\mathbf{x},\mathbf{y},M=1) = \mathscr{L}(\mathbf{x},\mathbf{y})$.

As we are interested in comparing this model's performance on outlier data to the ensemble method from the previous subsection, we compute an analogous total covariance quantity from (8), but instead replace ensemble members with mixture modes. This "mixture total covariance" quantity is (8) given that ensemble members are replace by modes $g \to m$ and uniform ensemble averaging is replace by mode weight averaging. For instance, the predictive mixture mean is instead $\mu_T(\mathbf{x}) = \sum_{m=1}^{M}\alpha_m(\mathbf{x})\mu_m(\mathbf{x})$.

4. EXPERIMENTS AND RESULTS

We summarize the experimental setup and results of our approach.

4.1 Data and Processing

The dataset was split multiple ways to test the approach against both inlier and outlier data. The dataset was first split into "outlier" testing data, which consists of orbits having the largest 1% eccentricity values, $e_{1\%} \in [0.061, 0.22]$, which we use for testing the method is robust to outliers. The remaining "inlier" data was randomly split into training and testing datasets such that 15% of this dataset is reserved for testing the fidelity and compute speed of the models. It should be noted that, in a sense, what makes the "outlier" testing data an outlier in the context of AI modeling is that these large eccentricity values were not included in the training data. If instead they were included, then the AI training method would be forced to learn how to handle these cases statistically.

In our problem, we let the input features be a concatenation of T_x time steps of ephemeris data from equation (1),

$$\mathbf{x} = [\mathbf{x}^{(j)}_{t=1},...,\mathbf{x}^{(j)}_{t=T_x}],$$

which is $k_x = 18\,T_x$ dimensional, and let the labels to be predicted be the concatenation of T_y three future dimensional position coordinates of the RSO,

$$\mathbf{y} = [(x,y,z)^{(j)}_{t=T_x+1},...,(x,y,z)^{(j)}_{t=T_x+T_y}],$$

which is $k_y = 3\,T_y$ dimensional. We consider two scenarios, the first for forecasting a ephemeris a single orbit ($\sim$ 100 minutes) into the future and the second for forecasting ephemeris one day into the future (5 minute time intervals). The number of timesteps and the number dimensions of the feature and label data used in our experiments, as well as the number of rows available for training, are summarized below: The single orbit prediction experiment has more rows available for training than the single day prediction experiment because there are more non-overlapping single orbit input windows available (due to the dimensions above) than single day windows in this 3-day, 19K RSO, simulated

Table 2: Feature and Label Data Dimensions

Scenario	T_x	k_x	T_y	k_y	# training rows
Single Orbit	10	180	20	60	448K
Single Day	30	540	288	864	161K

dataset. Because k_y is relatively high dimensional in the single day case, we also explored using sparse block diagonal matrices (no inter-time correlations) to reduce the number of covariance elements that need be predicted by the model from $(k_y^2 + k_y)/2$ (because the matrix is symmetric) to $6T_y$ (as there is one 3×3 lower triangular Cholesky matrix per timestep, which has 6 elements). For the single day model, this reduces the number of covariance related outputs from $(864^2 + 864)/2 = 373,680$ to $6 \times 288 = 1,728$.

We trained models on an A6000 GPU with 48 GB of GPU memory and 4 CPU cores having 10 GB of RAM per CPU, which run at 3.8 GHz. The models used in the experiments were trained upwards of 1,000 epochs, which took between 3.5 hour 7.5 hours. The single orbit model sizes varied from 5.5 MB to 82 MB, while the single day models used were larger, about 375 MB. The best candidate models we found through hyperparameter search tended to have 1 or 2 temporal convolutional layers with a medium number of filters (convolutional layers, if present, are prepended to the network), 4-9 dense layers with a medium number of neurons, and a medium L scale factor.

We performed speed tests at two different scales τ_{34K}, which is the amount of time to propagate 34K RSOs (hypothetically all objects greater than 10cm in LEO), and τ_{1M}, which is the amount of time to propagate 1 million objects in LEO (hypothetically all objects between 1-10cm). The times recorded are averaged over 20 trials, and include the time it takes to push data in and out of the GPU. Bold values indicate perceived good performance.

4.2 "Inlier" Experiments

These experiments review the performance of the individual UA-AI models and their ensemble against the held out test dataset.

4.2.1 Single Orbit Forecasts

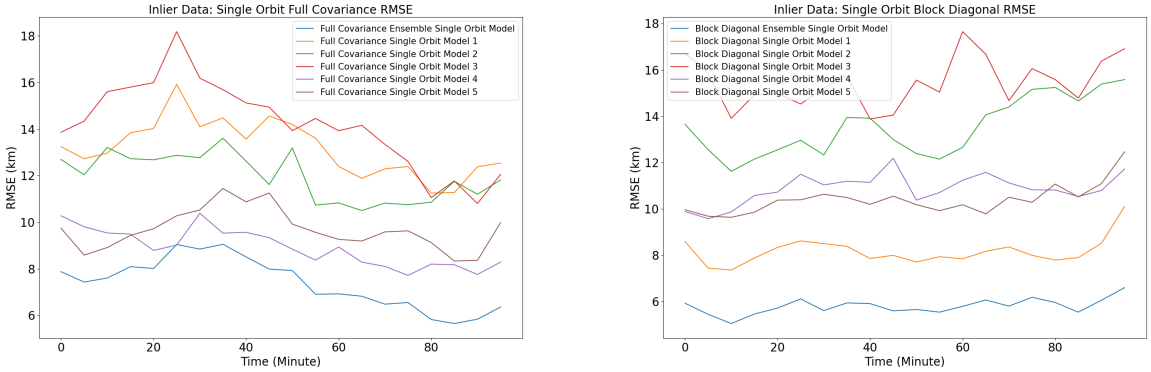
We compare the performance of two single orbit UA-AI models, a fully populated $(60, 60)$ predictive covariance matrix and a sparse block diagonal implementation that ignores inter-time correlations. Each ensemble consists of $G = 5$ ensemble members and the multivariate Gaussian mixture model is set to $M = 5$ modes for comparison purposes.

We calculate the following quantities and record them in the table: We compute the (positional) Root Mean Squared Errors (RMSEs) between the predicted means $\mu_t(\mathbf{x})$ at time t and its corresponding true value $\mathbf{y}_t = (x, y, z)_t$ per timestep which we then average over time. Because each ensemble is comprised of $G = 5$ models, we recorded their lowest and highest individual model RMSEs in square brackets. The RMSE for the multivariate Gaussian mixture model is computed using the $M = 5$ mixture weights before averaging over time, i.e. it involves $\sqrt{\sum_{m=1}^{M} \alpha_{m,t}(\mathbf{x})(\mathbf{y}_t - \mu_{m,t}(\mathbf{x}))^2}$. Below are the results. RMSE is recorded in kilometers and τ is recorded in seconds:

Table 3: Inlier Data: Single Orbit Model

Model Type	RMSE	τ_{34K}	τ_{1M}
Full Cov. Individuals	[9.2 km, 14.5 km]	0.0215 s	0.62 s
Full Cov. Ensemble	**7.6** km	**0.254** s	**7.38** s
Block Cov. Individuals	[8.1 km, 15.4 km]	0.0129 s	0.173 s
Block Cov. Ensemble	**5.8** km	**0.446** s	**12.1** s
Full Cov. Mixture Model Individual	22 km	0.093 s	2.77 s

The RMSE for these models remain relatively flat over the period of a single orbit, as shown in Figure 3.



(a) Full covariance

(b) Block Diagonal

Fig. 1: Single orbit RMSE on inlier data.

4.2.2 Single Day Forecasts

We compare single and ensemble performance similarly for the single day forecast experiment. Due to memory limitations and the effectiveness of the block diagonal covariance models in the single orbit experiment, we only trained sparse block diagonal single day models. We did not implement a sparse multivariate Gaussian mixture model, so it is omitted:

Table 4: Inlier Data: Single Day Model

Single Day Model	RMSE	τ_{34K}	τ_{1M}
Block Cov. Individuals	[82.3 km, 91.4 km]	0.0849 s	2.46 s
Block Cov. Ensemble	46.6 km	5.53 s	162.4 s

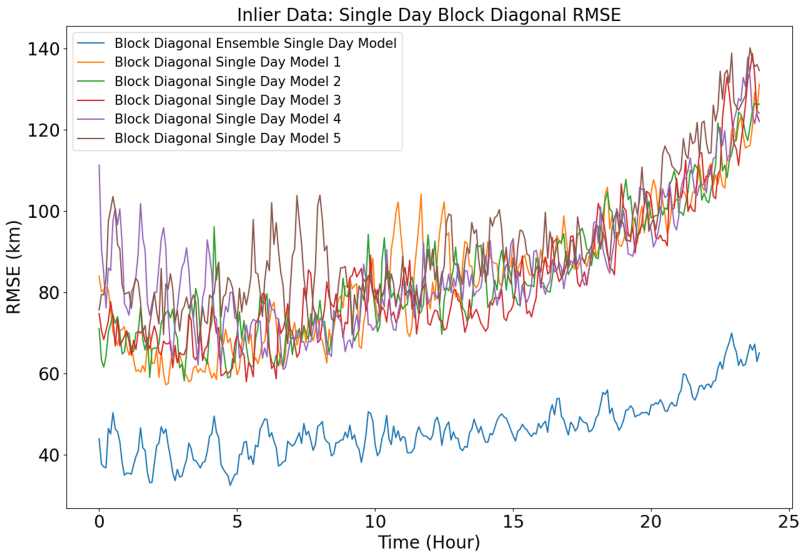


Fig. 2: Single day model RMSE over inlier data.

While Figure 2 shows the RMSE for these models grows seemingly nonlinearly, we believe with more training or rescaling covariance elements later in the orbit could lessen or remove the nonlinear growth in RMSE.

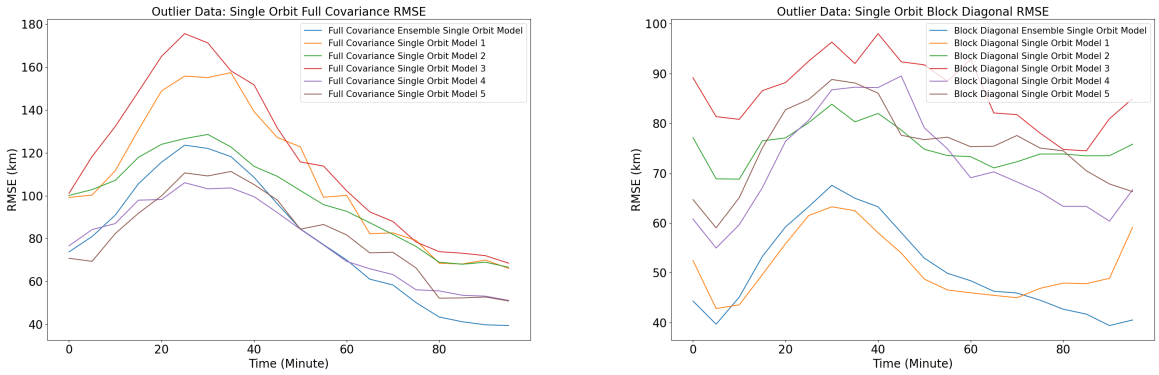
4.3 "Outlier" Experiments

These experiments review the performance of the UA-AI models against the eccentricity outlier dataset. We record the relative multiplier of how much larger the outlier RMSE and the time averaged UA-AI model predicted positional displacement standard deviation, $\overline{\sigma}_D = \frac{1}{T}\sum_{t=1}^{T}\sigma_{D,t} = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\sigma_{x,t}^2+\sigma_{y,t}^2+\sigma_{z,t}^2}$, are compared to their inlier counterparts.

Table 5: Outlier Data: Uncertainty Awareness

Prediction Period	Model Type	RMSE Multiplier	$\overline{\sigma}_D$ Multiplier
Single Orbit	Full Cov. Individuals	[7.81, 8.38]	[0.96, 1.02]
Single Orbit	Full Cov. Ensemble	10.3	**4.25**
Single Orbit	Block Cov. Individuals	[5.58, 7.25]	[0.97, 0.99]
Single Orbit	Block Cov. Ensemble	8.71	**4.11**
Single Orbit	Full Cov. Mixture Model Individual	11.6	2.01
Single Day	Block Cov. Individuals	[3.02, 3.47]	[1.00, 1.02]
Single Day	Block Cov. Ensemble	4.5	**1.98**

Relative to the increase in RMSE, the ensemble models were more indicative of being in an outlier regime than were the individual models, including the mixture model.



(a) Full covariance

(b) Block Diagonal

Fig. 3: Single orbit RMSE on outlier data. We believe RMSE rises and falls because the model learned to exploit the cyclic nature of orbits.

While it is unreasonable to expect good RMSE over the outlier data, the UA-AI ensemble models do predict large covariances in high RMSE situations. Note how the individual model covariances not changing much in Table 5 correspond to the "inlier" covariance term in (8), which indicates that the ensemble covariances grew due to the outlier term in (8). The RMSEs are plotted over time in Figures 3 & 4.

Finally, we also compared the growth in RMSE to the ensemble model predicted $\overline{\sigma}_D$'s over eccentricity in Figures 5, 6, & 7. We see the anticipated growth in RMSE as the 0.089 inlier/outlier eccentricity boundary is approached and surpassed. Further, the key desirable feature that the predicted $\overline{\sigma}_D$'s generally trend upward as the RMSE and eccentricity grows is indicative of the ensemble models' ability to understand and predict their own uncertainty. The single day model did not have as stark an increase in the predicted $\overline{\sigma}_D$ as compared to the single orbit models, but this can likely be explained by the fact that the single day outlier RMSE did not increase as much either.

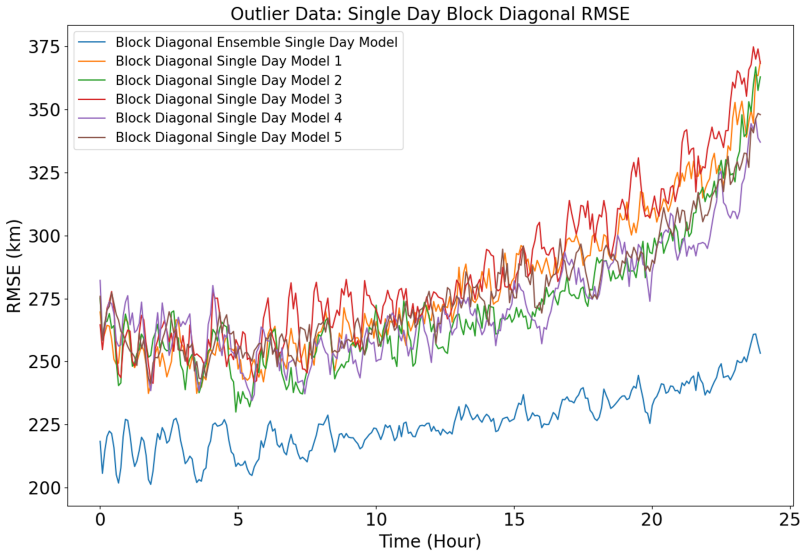


Fig. 4: Single day model RMSE over outlier data.

5. DISCUSSION

The UA-AI LEO ephemeris prediction models we found sit in the middle of the accuracy-speed trade space between highly accurate, but slow, special perturbation models and inaccurate, but fast, simple 2-body Keplerian motion models. Due to J2 and other perturbations in LEO, 2-body Keplerian motion models can easily pick up between 2-5 degrees of angular error over the course of a day (somewhere around 250km-625km). Our single day ensemble model maintained RMSEs around 45 km and our single orbit ensemble model maintained RMSEs around 6 km. In terms of speed, these models were able to forecast 1 million ephemerides in the LEO belt on the order of seconds or minutes rather than days or weeks.

While we were able to find some relatively performant UA-AI models, our hyperparameter search ($\sim$ 250 shallowly trained models) only just started to explore the large set of possible UA-AI models that could be trained or designed. As an example, our choice of the number of timesteps in the input dimension to be 10 and 30 for the single orbit and single day models, respectively, was relatively arbitrary and not explored nor optimized. The large space of possible UA-AI models can potentially densely populate new/sparse regions in the accuracy-speed modeling trade space.

Further, although not shown explicitly in our experiments above, the amount of data available for training has a considerable impact on performance. Prior to the 19,210 RSO 3-day dataset, we had a similar but smaller dataset consisting of about 750 RSOs. Even with hyperparameter search, we had difficulty breaking an RSME of about 60km in the single orbit experiment with this smaller dataset. However, taking that model, and without changing its hyperparameters, the model reached a RMSE of about 15km when training on the larger dataset. While there are almost certainly diminishing returns in performance as the amount of data increases, it is encouraging that with more data the models could continue to improve. Further, it should be noted that we had to stop the single day training short, which was still continuing to improve at the time of submission.

The ability of these neural models to make rapid predictions, while also maintaining accuracy, implies this approach can handle the increase in complexity required to propagate ephemeris uncertainty along with mean ephemeris by pushing thousands of representative Monte Carlo samples (particle filter propagation) through the UA-AI model. Similarly, or perhaps in addition, this method could be used for "quick and dirty" space traffic control/conjunction screening to rapidly rule out large swaths of infeasible conjunctions, in an uncertainty quantified manner. Alternatively, if ephemeris training data comes with heteroskedastic uncertainties, these uncertainties could likely be learned outright and predicted by the UA-AI model. It would be interesting to see how well this approach performs on other

orbital regimes or realistic, non-simulated, orbits with maneuvers.

An essential feature of this uncertainty-aware AI approach is that its predicted uncertainty measures can be used to overcome some critical limitations that prevent AI's wide adoption into SDA applications. Because our UA-AI ensemble models were able to correctly identify, predict, and indicate when their performance is expected to diminish due to the presence of outliers, this information can be used to build more efficient decision systems. Rather than running computationally expensive propagators for all RSOs in LEO, one could instead opt to only run these propagators if the UA-AI exceeds a prediction uncertainty tolerance, or in a conjunction screening scenario, if the system similarly exceeds a miss distance and uncertainty tolerance. Such an approach would drive down computational cost and free up additional resources.

6. ACKNOWLEDGEMENTS

Fotis Barlos, Sean Colbath, Zac Dutton, Ian Emmons, Greg Joiner, Jonathan Lasko, Max Leamer, Anthony Mendoza, Plamen Petrov, Lauren Richardson, John Sustersic, Aisha Yousuf, & Jiangying Zhou.

7. REFERENCES

[1] ESA Space Debris Office. Esa's annual space environment report. Technical Report GEN-DB-LOG-00288-OPS-SD, European Space Agency, Darmstadt, Germany, 2024.

[2] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks, 2018.

[3] Nick Ratajczyk, Sierra Flynn, Gabe Bershenyi, Scott Gennari, Adrienne Pickerill, Brian Boyle, Dave Harber, and Lucia Witikko. Smallsat end-of-life operations: Opportunities and challenges. In *Small Satellite Conference 2024*, 2024.

[4] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. *Advances in neural information processing systems*, 31, 2018.

[5] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

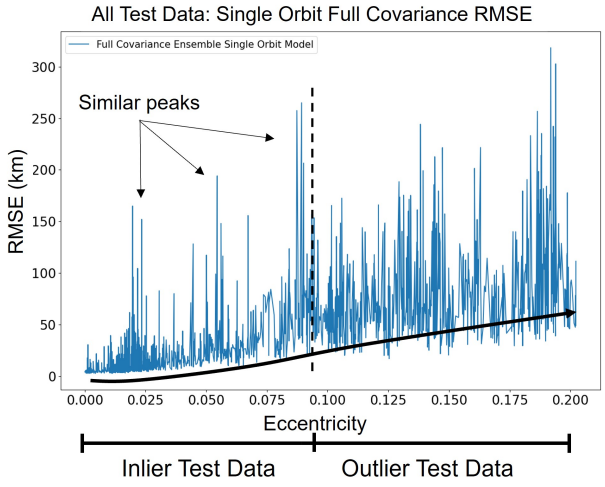
[6] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[7] Cameron Harris, Dylan Thomas, Jonathan Kadan, DK Schroeder, and Jonathan Black. Expanding the space surveillance network with space-based sensors using metaheuristic optimization techniques. In *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS), Maui, HI, USA*, 2021.

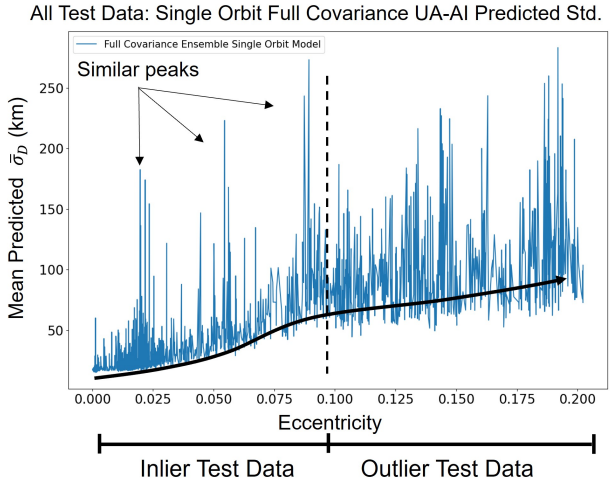
[8] Jonathan Kadan, Dylan Thomas, Amit Bala, Kevin Schroeder, and Jonathan Black. Application of novel filtering approaches to modern space domain awareness. In *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS), Maui, HI, USA*, 2021.

[9] Rebecca L Russell and Christopher Reale. Multivariate uncertainty in deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):7937–7943, 2021.

[10] Romain Egele, Romit Maulik, Krishnan Raghavan, Bethany Lusch, Isabelle Guyon, and Prasanna Balaprakash. Autodeuq: Automated deep ensemble with uncertainty quantification. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 1908–1914. IEEE, 2022.

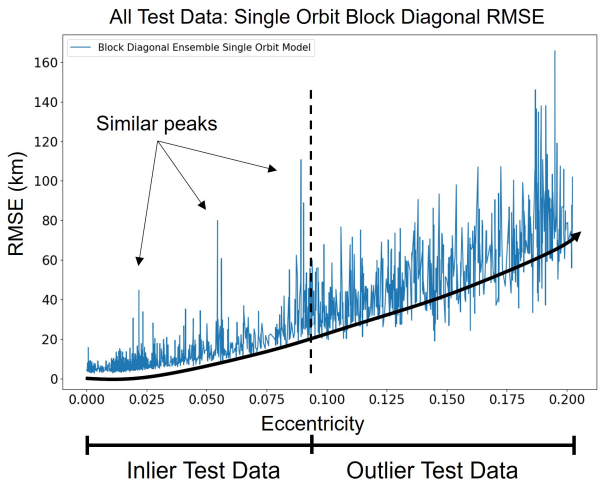


(a) RMSE: Full Covariance Single Orbit Model

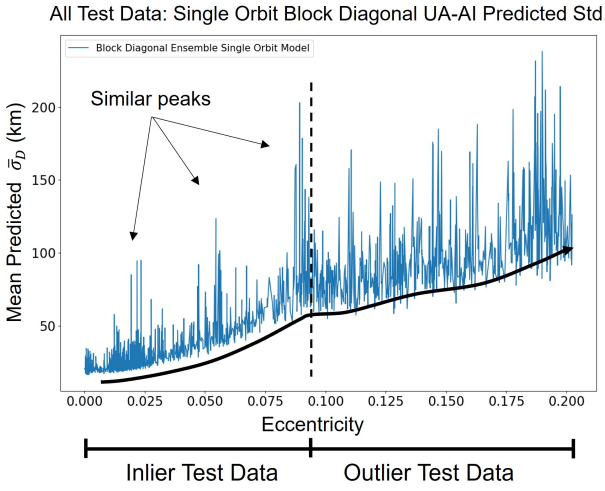


(b) $\overline{\sigma}_D$: Full Covariance Single Orbit Model

Fig. 5: The RMSE and predicted $\overline{\sigma}_D$'s follow a similar upward trend indicating uncertainty-awareness. They share similar peaks on the inlier testing data, which means both that all statistical outliers in this data cannot be prescribed to large eccentricity alone and that the ensemble model recognizes its own inability to make accurate predictions on those cases.

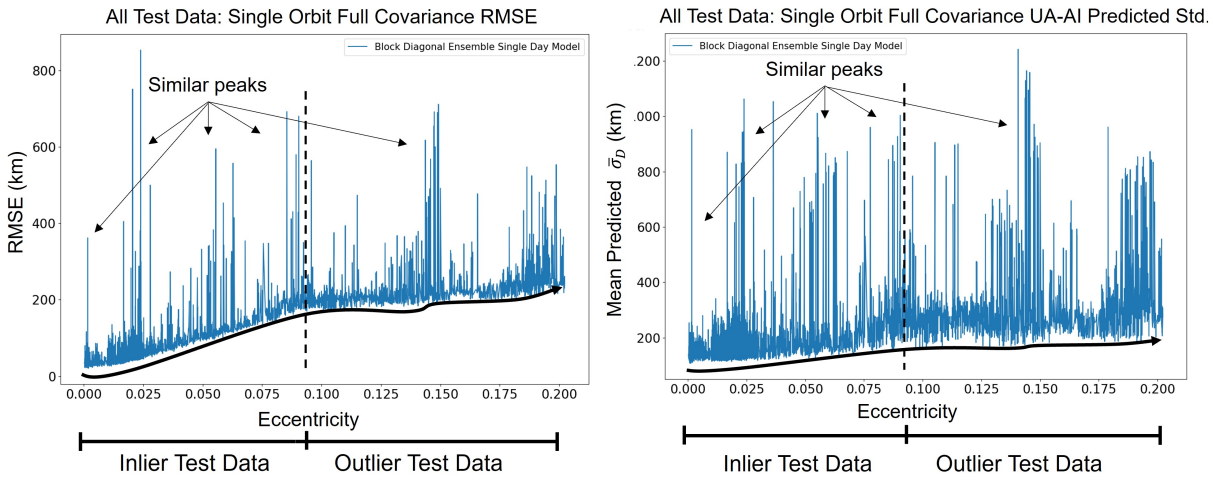


(a) RMSE: Block Diagonal Single Orbit Model



(b) $\overline{\sigma}_D$: Block Diagonal Single Orbit Model

Fig. 6: The block diagonal single orbit model shares the same uncertainty-awareness features as the full covariance single orbit model from Figure 5.



(a) RMSE: Block Diagonal Single Day Model

(b) $\overline{\sigma}_D$: Block Diagonal Single Day Model

Fig. 7: The block diagonal single day model is similarly uncertainty-aware, but the increase in the outlier domain is less pronounced, likely due to the RMSE there not increasing as much as the single orbit models. There are many similarities in the peaks, some of which are overly pronounced in the single orbit model, but the predictions still reliably indicate the large outlier cases.