SOM-erizing Cislunar Orbits: Classification of Cislunar Orbits Using Self-Organizing Maps (SOMs)

Denvir Higgins
Lawrence Livermore National Laboratory
Kerianne Pruett
Lawrence Livermore National Laboratory

Travis Yeager
Lawrence Livermore National Laboratory

Michael Schneider
Lawrence Livermore National Laboratory

ABSTRACT

Cislunar orbits occupy an immense volume of phase-space, spanning from GEO out to where the Earth-Moon system is gravitationally dominant. Most existing cislunar models focus on small regions in this phase-space and target individual missions and assume simplified gravitational models (e.g., CR3BP). However, in this regime chaos can dominate dynamics, requiring n-body simulations for understanding and characterizing cislunar orbits (which becomes increasingly more important with increasing orbital timescales). Additionally, TLE's, which are commonly used as a method for summarizing orbital states (often used to understand similarity in close orbits), fail to properly capture orbital properties when 3- or n-body dynamics are important. Machine learning methods can be applied to complicated datasets to aid humans in understanding and characterizing these orbits. Due to the complexity of cislunar orbits, there are not obvious classes or labels that one could apply to similar orbits, making supervised learning (i.e., labeled) methods a non-viable solution. Unsupervised learning (i.e., unlabeled) will be the superior approach for understanding orbits in this space, potentially providing labels for things that were previously unknown, and that we may not know we are looking for. To begin understanding orbits in this complex space, we simulate one million orbits (considering n-body dynamics) over varying timescales utilizing high-performance computing resources at Lawrence Livermore National Laboratory (LLNL) and leveraging an LLNL-developed space situational awareness python package (SSAPy). Using this library of high-fidelity cislunar orbits we generate self-organizing maps (SOMs; an unsupervised machine learning method that creates a feature space where like features are grouped together) to characterize these orbits. Through these mappings we infer properties for different cislunar orbits (e.g., stability metrics, predicted lifetimes and end-of-life outcomes, identifying potentially interesting mission relevant orbits, etc.). Additionally, these methods can aid in identifying new families or classes of orbits and find new stable regions in cislunar phase-space, opening up new avenues for future cislunar mission orbits.

1. INTRODUCTION

The increase in planned commercial and military missions to the Moon over the next decade, including NASA's Lunar Gateway and Deep Space Network, increases the importance of cislunar space domain awareness (SDA) in particular. There have been marked efforts to create a constellation of sensors to detect and track objects in cislunar space [1] as well as efforts to create Moon-based sensors to track families of orbits [2]. While cislunar space is a regime of interest for current SDA missions, it is difficult to characterize due to its chaotic nature. The models used to develop the above missions are simplistic, using orbits such as a CR3BP model to generate a finite list of families. While a good starting point, this model fails to accurately describe dynamics in cislunar space; a high fidelity model is needed to fill this gap.

Due to the chaotic nature of this regime, machine learning algorithms are suitable and essential tools for prediction and classification of orbits and orbital parameters. The complexity of cislunar orbits means there are not obvious classes or labels that one can assign to similar orbits, making unsupervised (i.e., unlabled) methods the superior approach. The unsupervised self organizing map (SOM) method is uniquely suited for this challenge, and can be used to organize and try to make sense of the chaos. The SOM creates a feature space in which similarities are group together, enabling the injestion of large-volume n-dimensional datasets to discover similarities between thousands, and even millions, of orbits. Using competitive learning, each cell in a finite dimension map attempts to represent the input data the best; as a result, analysis can be performed to characterize families of orbits in individual cells.

Using a high fidelity n-body model, we generate one million orbits of variable lifetimes=, distributed throughout phase space by sampling initital Keplerian orbital elements in a sphere around the Earth [3, 4]. We then create maps generated by an unsupervised machine learning algorithm to try to identify and understand families of orbits in this regime. Though cislunar orbits are not characterized by Keplerian orbital elements, we can use each orbits' time series Keplerian orbital elements, and various labels including orbit lifetime (i.e., the time until an orbit is no longer stable) to identify clusters of potential orbit families. This method generates a database of stable orbits in cislunar space to anchor future missions.

2. DATA SIMULATION

Current accepted simulations of orbits in cislunar space use reductionist three body models that fail to accurately describe the dynamics of this space. In cislunar space, conic section type orbits are highly improbable; as time passes, the probability of perturbations altering the path of an orbit increases. These perturbations can be related first to gravitational forces impacting the trajectory of the orbit, which are effected by the positions of the objects in the Solar System, or other forces related to astronomical objects (solar radiation pressure, for example). They can also be related to non-gravitational forces, such as solar wind. To increase cislunar simulation fidelity, a Lawrence Livermore National Laboratory (LLNL) developed python package, the Space Situational Awareness Python package (SSAPy)[5][1] was developed for fast and precise modeling of orbits in cislunar space.

2.1 SSAPy

SSAPy is built on C and C++ languages, and implimented in python, enabling it to be fast, flexible, and efficient. It uses pythonic optimization of broadcastable computations, which allows for integrations to run in parallel. SSAPy enables the user to specify multiple parameters, including gravitational models for the Earth and Moon, radiation coefficients, etc. The orbit propagation done by SSAPy is executed in a modular fashion using variable time step integration, and supports multiple integration methods. It has support for multiple coordinate frames, including coordinate frame conversion, although the orbital plotting used in this paper uses a geocentric reference frame (GCRF). The orbital state representations include both Cartesian through position and velocity vectors as well as time varying Keplerian through the osculating Keplerian orbital elements.

2.2 Simulation Parameters

For this paper, we use the EGM 2008 model for Earth's gravitational harmonics and the GRGM1200A model for the lunar gravity field[6, 7]. For orbit propagation we use a 7/8 Runge-Kutta integrator. For simulating one million orbits, we randomly initialize orbits in phase space between GEO and two times the distance between the Earth and Moon (approximately 18x GEO). To place the orbits we randomly sample from the initialization parameters in Table 1, and that set of Keplerian elements becomes the first timestep of the orbit. Orbits are then propagated with all forces and parameters in Table 1, out to one year.

2.3 Dataset Population

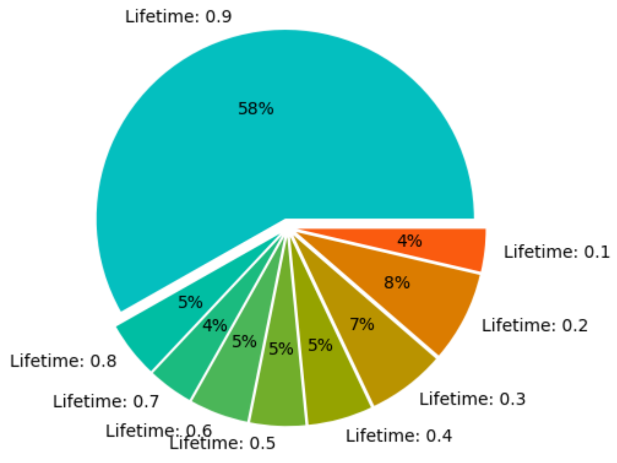
Most of the orbits (530,000) in this large, one million orbit dataset survive to a year. Because we propagate each orbit out to one year, we assign each orbit a lifetime (representing a fraction of a year) corresponding to how long it stayed stable in that one year timeframe. For our purposes, 'stable' is any orbit that does not come within GEO for a period of time, leave the Earth-Moon gravitational system, crash into the Earth, or crash into the Moon. Once an orbit becomes 'unstable' by reaching one of those criteria, it is assigned a lifetime. This majority population of lifetime 1.0 orbits (i.e., orbits that are still stable and bound to the Earth or Moon and located between GEO and 18x GEO after

[1]https://github.com/LLNL/SSAPy

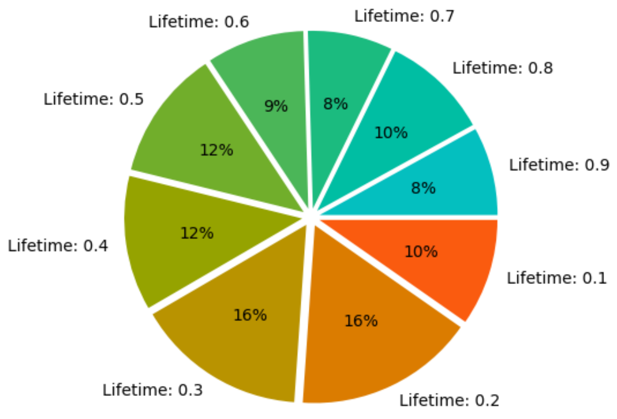
Table 1: SSAPy Simulation Parameters

Initialization	
Semi-major axis (a):	4.216×10^7 m (geosynchronous orbit) to 7.688×10^8 m (2 lunar distances)
Eccentricity (e):	0 to 1
Inclination (i):	0 to $\frac{\pi}{2}$
True Anomaly (TA):	0 to 2π
Argument of the Periapsis (PA):	0 to 2π
Right Ascension of the Acsending Node (RAAN):	0 to 2π
Propagation	
Integrator:	7/8 Runge-Kutta
Cross-sectional Area:	0.25 m^2
Mass:	250 kg
Drag coefficient:	2.3
Radiation pressure coefficient:	1.3
Timestep:	1 hour
Gravity Models	
Earth model:	EGM2008
Lunar harmonics model:	GRGM1200A

1 year of propagation) creates a recognizable pattern for the machine learning algorithm, meaning the accuracy of the model could be artificially increased because it can predict the highest lifetimes easier. Alongside this, a fraction of orbits survive for less than a week. Thus a dataset of both stable and unstable orbits represented in equal fractions is created for machine learning applications. Figure 1 provides a breakdown of lifetime values into ten bins, for both the original dataset and the adjusted dataset. We use the adjusted dataset, in which we have taken relative proportions of orbits in each lifetime bin, for the remainder of this work.



(a) Original Dataset



(b) Adjusted Dataset

Fig. 1: Dataset Lifetime Populations

3. METHODOLOGY

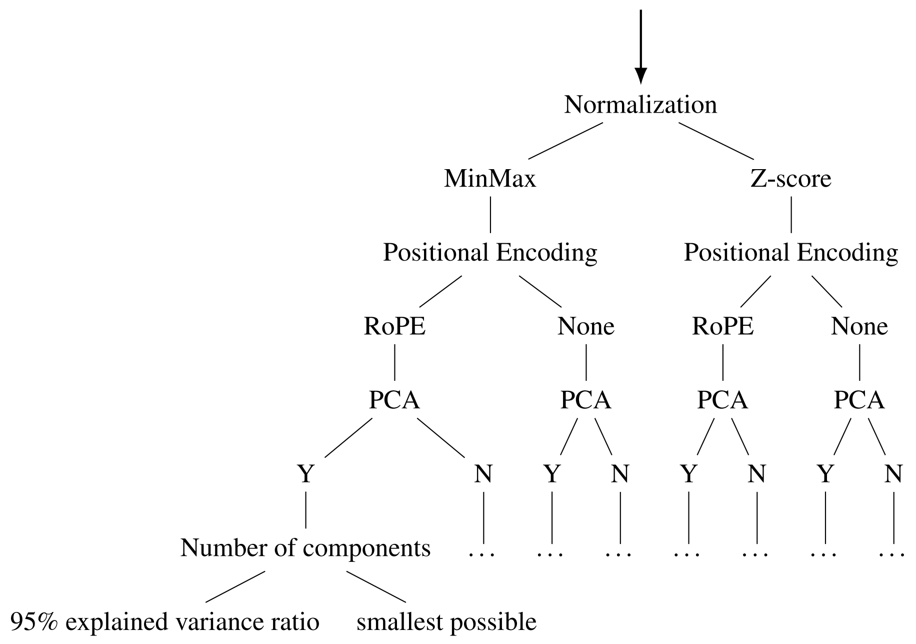
The purpose of a machine learning model is to extract knowledge from data. In the case of cislunar data, an unsupervised machine learning model is an important tool to characterize and classify orbits. Since the desired output, for example the number of families of orbits, is unknown, the unsupervised model is the best fit for this data. Unsupervised machine learning can reveal hidden structure (i.e., families of orbits) in unlabeled data. The Self Organizing Map (SOM) method is an unsupervised machine learning technique capable of finding clusters of data points in large datasets. Although the SOM is not the only dimensionality reducing machine learning technique that exists, it is built on a competitive learning model where each neuron in the map competes to represent the input data. We utilize the SOM method to take in high dimensional time series data as input, and output a two dimensional grid where each data point is organized into respective neurons. This technique is a unique method of exploratory data analysis as there are no hypotheses that are validated in the process.

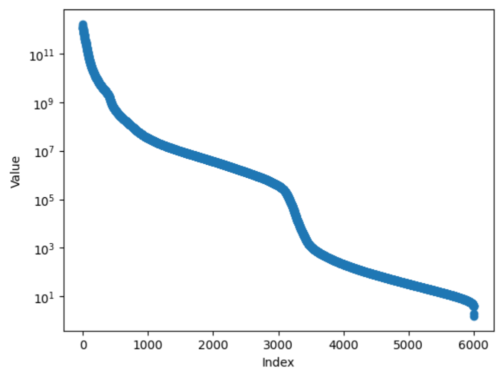
3.1 Data Preprocessing

To prepare the data to input into the machine learning model, it must go through several preprocessing steps. We sweep over a variety of normalization techniques and hyperparameters until the combination that yields the best outcome (described later in this section) is determined. The GCRF time series state vectors are used as inputs.

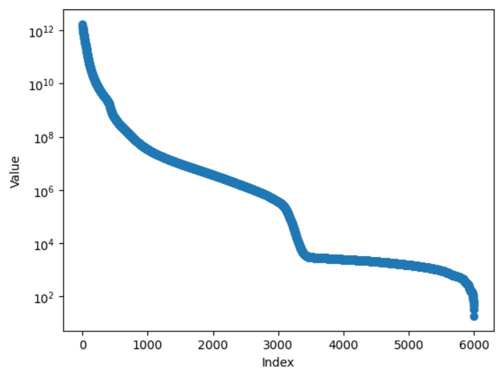
First each orbit is normalized to a unit hypercube using either sklearn's MinMax scaler or using sklearn's StandardScaler. To communicate the time series nature of this data to the machine learning model, Rotary Positional Embedding (RoPE) is optionally used [8]. The RoPE method encodes the position of the tokens, here each $[x, y, z, v_x, v_y, v_z]$ vector, by rotating it based on its position in the sequence. Then, to ensure that each orbit has the same amount of features, or data points, a B-spline interpolation is performed on the N-dimensional data. As a result, each orbit has 1000 time steps of each position and velocity vector regardless of overall lifetime. This ensures that this model is not biased to quickly identify short or long lifetime orbits based on entry length alone.

After normalization, optional RoPE, and interpolation, the data can be reduced using Principle Component Analysis (PCA) techniques, or continue into the SOM without PCA. If PCA is used, there is a sweep over the number of PCA components, in which the number of compoents can be determined in two different ways. Reducing the number of features in this way can decrease SOM map generation time, decrease plotting time, and increase accuracy in the model. The diagram below describes our full pre-processing work flow. Figure 2 shows the number of components needed in PCA to be representative of the data. In each case, the order of magnitude of the components hits an inflection point around 1000 components, meaning that 1000 components should be needed to fully represent the data. However, in subsequent tests, as little as 10 PCA components are tested and yield the highest SOM accuracy.

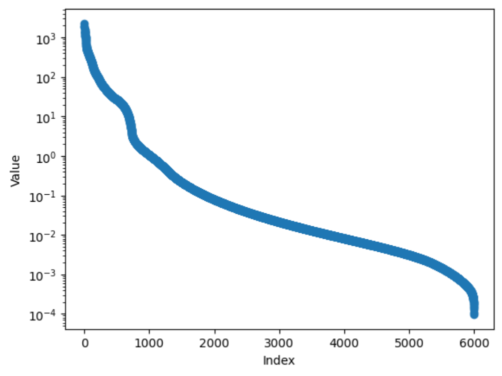




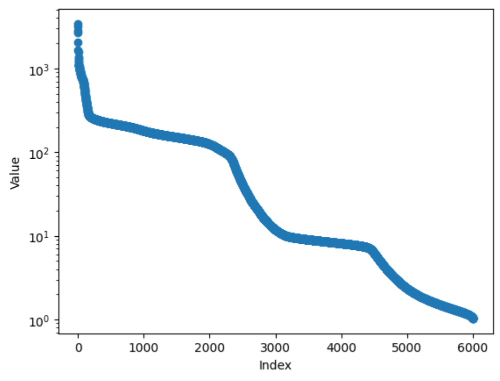
(a) No Normalization



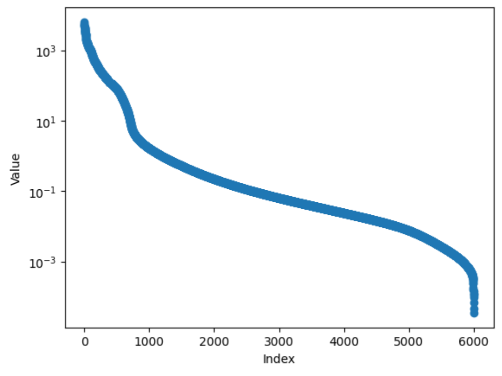
(b) No Normalization with RoPE



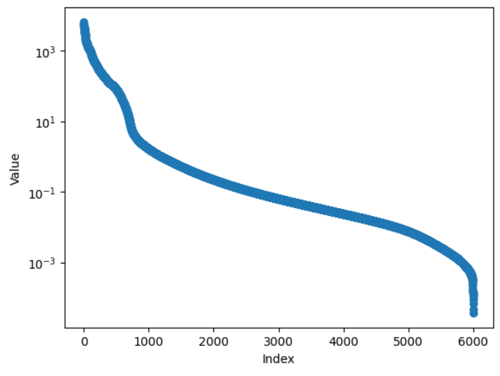
(c) MinMax Normalization



(d) MinMax Normalization with RoPE



(e) Z-Score Normalization



(f) Z-Score Normalization with RoPE

Fig. 2: Value and Index of PCA Components for Various Preprocessing Schemes

3.2 Self Organizing Maps

A SOM clusters input data, in this case cislunar orbital data, using synaptic connections (weights) into a lattice of post synaptic neurons. There are three stages to this clustering methodology: first, each neuron competes to represent the input data in the competition stage. In this stage, only one winning neuron is activated at a time.

$$i(x) = arg_j min||x - w_j||^2 \qquad (1)$$
$$j \in [1, 2, \ldots, m]$$

The winning neuron or best matching unit (BMU) has the smallest euclidean ditsance between the input vector and the weighting vector. Next, the SOM enters the cooperation stage, where the BMUs form neighborhoods with adjacent neurons determined by a Gaussian function.

$$h_{ij}(d_{ij}) = e^{\frac{-d^2{}_{ij}}{2\sigma^2}} \tag{2}$$

Finally, the SOM adjusts the weights based on these neighborhoods, essentially sharing the wealth, and starts over in the adaptation stage. The weights of the chosen BMU taken on the values of the input data.

$$w_j(n+1) = w_j(n) + \mu(n)h_{ij}(n)[x - w_j(n)] \tag{3}$$

The number of times the algorithm iterates through the data is specified by the user alongside a number of other parameters. As the algorithm iterates over the input data, the neurons become selectively tuned to patterns in the features of the input. As a result, the winning neurons create a meaningful coordinate system for the input features on the lattice of the SOM. This coordinate system can then be used to identify clusters in the data. The stages of the SOM are described by the equations (1), (2), and (3), where m is the number of units in the map and x is the input data. The typical starting point for the dimensions of the map are described by equation (4). Conventionally, the starting point for m is related to the length of the dataset being input into the SOM.

$$m = \sqrt{5\sqrt{x}} \tag{4}$$

To create this lattice, a number of parameters that effect how the input data is assigned to neurons on the lattice of the map must be specified. This organization is primarily effected by the choice of neighborhood function, in this case Gaussian, but is also impacted by user selected radius of the Gaussian neighborhood (σ_0), learning rate (μ_0), number of iterations (n), and the dimensions of the map m^2.

$$\mu = \mu_0 e^{\frac{n}{-T}} \in [0,1] \tag{5}$$

$$\sigma = \sigma_0 e^{\frac{n}{-T}} \tag{6}$$

The learning rate and radius decrease asymptotically as a function of the number of iterations and a constant, T. Convergence takes many iterations, typically several thousand times the number of units. The larger the dimensionality of the map, the higher the number of iterations needed to achieve convergence. The SOM algorithm stops when there is no noticeable change in the feature map. We use the MiniSOM[2] python implementation of the SOM method for analysis in this paper [9]. MiniSOM is a numpy application of Self Organizing Maps, and gets its name from its minimalistic approach.

3.3 Model Accuracy

The accuracy of the SOM can be measured through a variety of techniques; most of these methods use scikit-learn's train test split function. This partitions the input data into a training and testing set in fractions set by the user. MiniSOM's built in accuracy model measures the fraction of test data for which the SOM correctly predicted the label. This technique works well for discrete labels, but doesn't translate well to continuous labels. In the case of orbit lifetimes, we use the absolute mean and median deviation between the test orbits and the average label of the training orbits in each cell to quantify the precision of the SOM's lifetime estimation. For each test orbit, we identify the winning neuron (i.e., the SOM cell that the orbit was organized into), then we get the average or median lifetime value of all training orbits that fell inside that cell, then we calculate the difference between the truth label of the test orbit and that cell average or median. Then, the average or median of the absolute deviations is determined. In the case of lifetime, the output is a fraction of a year, the result can be converted to a number of days.

3.4 Parameter Sweeps

Alongside topographic and quantization errors, MiniSOM includes a built-in method to determine the accuracy, or the ability for the SOM to determine the correct label for new input data. As mentioned above, this method is effective in discriminating between classes of data, but not for continuous labeling methods such as orbit lifetime. The median absolute deviation technique is therefore leveraged to determine the optimal learning rate ($\boldsymbol{\mu_0}$), neighborhood radius ($\boldsymbol{\sigma_0}$), and number of iterations (n). The convential area described in equation (4) is used as a starting point; for

[2]https://github.com/JustGlowing/minisom

a 100,000 orbit dataset, the side length of the convential area is set to m=39. The dimensions of the SOM are also shrunk to determine the impact of SOM area on accuracy. Alongside this, increasing the area of the SOM allows the BMUs to be more spread out, potentially allowing for clusters to begin to form. These clusters can later be used for classification of families of orbits. The neighborhood radius is initially set to be the hypotenuse of the map. The orbit's lifetime is used as a label to determine ideal clustering. A variety of pre-processing techniques, such as MinMax and Z-score normalization, RoPE, and PCA are used to find the combination of pre-processing and SOM parameters that yield the lowest absolute deviation. Table 2 shows every value that we use in our hyperparameter sweep, in which we try every combination of parameters listed.

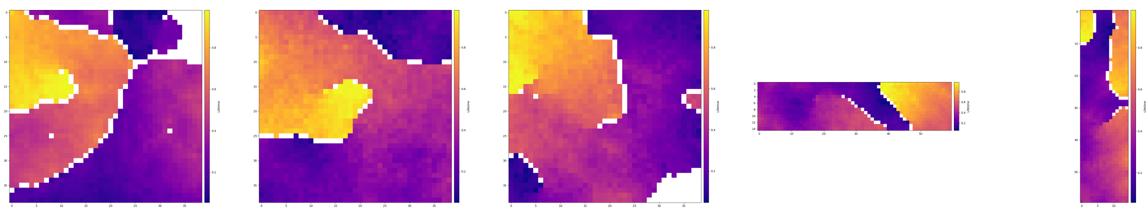
Table 2: Sweep Parameters

Preprocessing method:	MinMax + RoPE, MinMax, Z-score + RoPE, Z-score
Number of PCA components:	1, 10, 50, 1000
Dimensions:	15 by 15, 9 by 25, 25 by 9, 20 by 20, 16 by 25, 25 by 16, 30 by 30, 60 by 15, 15 by 60, 39 by 39
Initial learning rate (μ_0):	0.01, 0.1, 1.0, 10
Initial neighborhood radius (σ_0):	0.1, 1.0, 10, 30
Number of iterations:	1,000, 10,000, 100,000, 1,000,000

As a result of this sweep, SOMs with the lowest median deviation (measured in days) between the truth lifetime and median estimated lifetime of the cell, are deemed more accurate. Below, we list parameters (Table 3) and plots (Figure 3) for the five most accurate SOMs after our sweeps.

Table 3: Sweep Results

Preprocessing	**Dimensions**	σ_0	μ_0	**n**	**Median Absolute Deviation (in days)**	**Generation Time (in seconds)**	**Location**
MinMax + RoPE + 10 PCA Components	39 by 39	1.0	0.1	1000000	3.698201	246.027566	a
MinMax + RoPE + 10 PCA Components	39 by 39	1.0	1.0	1000000	3.760531	245.611797	b
MinMax + RoPE + 10 PCA Components	39 by 39	1.0	1.0	100000	3.802083	23.462315	c
MinMax + RoPE + 10 PCA Components	15 by 60	1.0	0.1	1000000	4.030624	149.641436	d
MinMax + RoPE + 10 PCA Components	60 by 15	1.0	1.0	1000000	4.072177	148.402357	e



(a) (b) (c) (d) (e)

Fig. 3: Highest Accuracy SOMs

In some cases, the preprocessing method, specifically the use of PCA, can increase the computation costs through longer SOM generation times. Aside from lifetime recovery, the computation costs mentioned above should be considered in relation to generating and plotting the SOM. Figure 4 shows the breakdown of SOM dimension to SOM processing time, as a function of different pre-processing techniques.

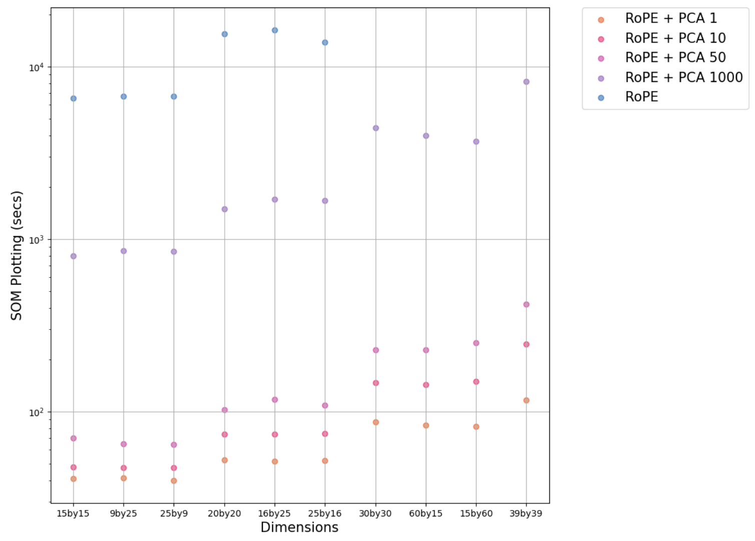


Fig. 4: Effect of Preprocessing Method and Dimension on Generation Time

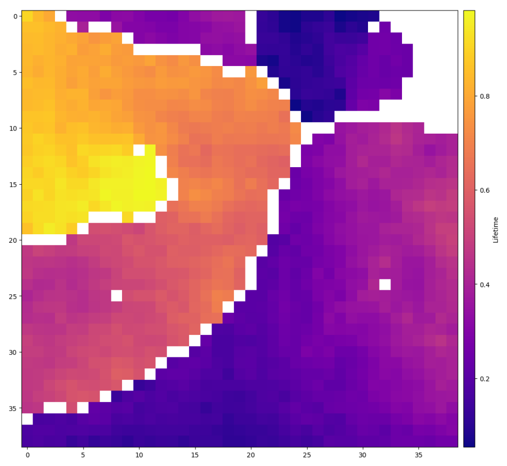
From the sweep results, the optimal dataset that will be analyzed throughout the remainder of this paper uses MinMax normalization, RoPE, and 10 PCA components in its preprocessing. Additionally, we can visually confirm from figure 3-d (using MinMax normalization with RoPE) had an elbow at the smallest number of indexes, which appears to be roughly 10 PCA components. Additionally, the learning rate, neighborhood radius, and number of iterations are fixed at 0.1, 1.0, and 1,000,000 respectively (which we found to be the optimal parameters for this pre-processed dataset with our best fitting dimensions of 39x39).

4. SELF ORGANIZING MAP ANALYSIS

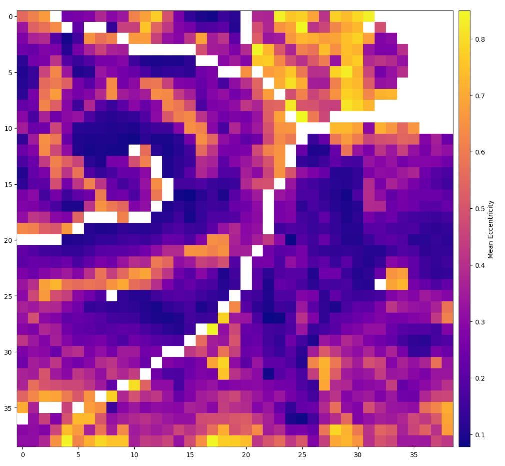
A visual inspection of a SOM is not enough to determine how it has classified the data. Beyond that, the cislunar dataset is hard to understand. As a feature reducing method, the generated SOMs allow for clustering and classification of orbital data based on the chosen label. Alongside this, the neurons inside the SOM containing the orbits are each distinct and reflective of their contents, allowing for more understanding on SOM clustering. For mission oriented applications, the most accurate SOM is also tested on limited data for lifetime recovery. Finally, families of orbits are identified and plotted on the SOM for further cluster understanding.

4.1 Map Labeling

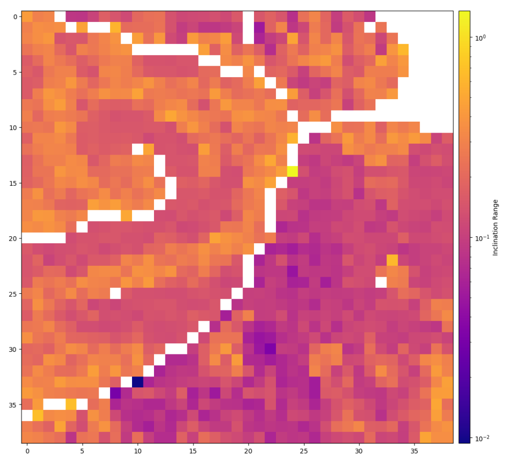
The motivation behind using the SOM is to characterize families of cislunar orbits by interpreting how the SOM has determined that orbits are similar. The SOM has organized orbits based on pre-processed state vectors, and after running the SOM, we can use other orbital parameters (not used or seen by the SOM) to try and understand how the SOM did this organization. Visually inspecting the SOM for clusters is advised against; two orbits may exist in the same BMU and have radically different labels, however, these labels reveal underlying links between orbits clustered in the same cells and orbits in the same BMU. We can use these additional labels, such as orbit lifetime, average change over various Keplerian orbital elemetns (KOEs), maximum and minimum velocity, and number of revolutions over an entire orbit, to understand additional details on how the SOM organizaed these orbits. These labels can all be interpreted by laying a color map over the SOM, where the color of each cell is the average value of that label for all orbits that fall in that cell. For this work, we focused on building a SOM to precisely recover orbital lifetimes (see the Figure 5-a).



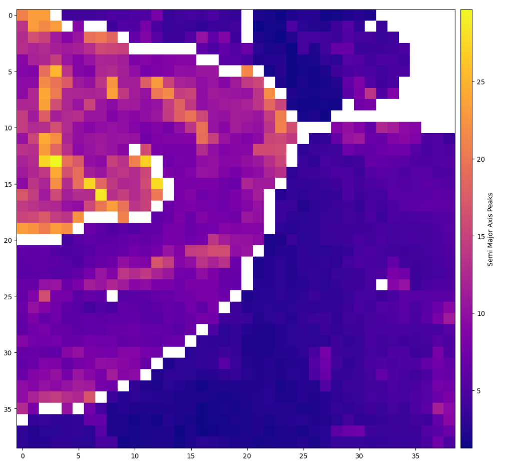
(a) Lifetime



(b) Mean Eccentricity



(c) Inclination Range



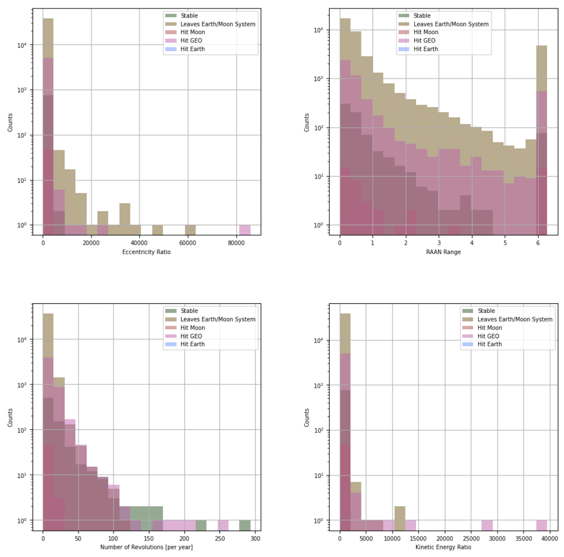
(d) Semi Major Axis Peaks

Fig. 5: Different Labels

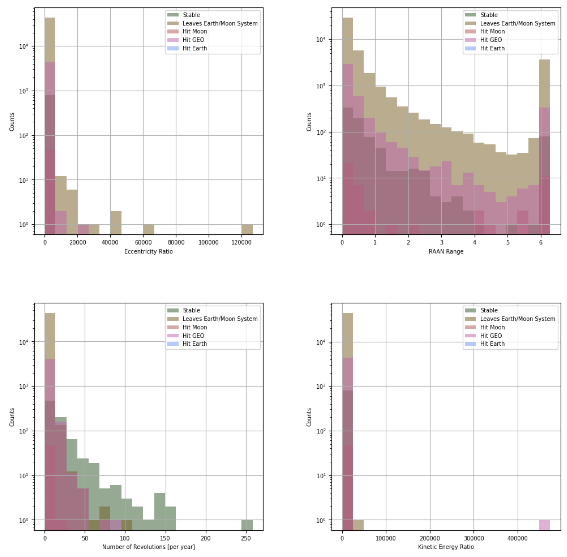
These additional statistics reveal texture within the distinct clusters formed by the lifetime labeling method. For example, looking at the lifetime labels in Figure 5-a you can see there are two main regions of low lifetime orbits - the dark purple patch near the top middle, and the dark purple patch at the bottom, spanning from left to middle. If you look at those same regions in the mean eccentricity map in Fig 5-b you can notice that these two regions vary in mean eccentricity (e.g., the low lifetime population in the top middle correcponds to high mean eccentricity values). Moreover, statistics done on the number of peaks in the time series semi major axis values highlight similarities within the high lifetime cluster. The neurons closer to the center of the cluster that looks like a snail shell has more peaks in this parameter.

Alongside visually comparing these different labeling techniques, the dynamics of neuron contents warrant analysis as well. Statistical analysis on these elements is imperative to characterize families of orbits in cislunar space. First, the high and low lifetime clusters are analyzed to discover trends. The large high lifetime and low lifetime cluster are initially scrutinized.

The main difference between the two clusters shown in Figure 6 seems to reside in the kinectic energy ratio - the higher lifetime orbits have a lower ratio on average, indicating that the orbits might be less chaotic. The split populations where two clusters exist that share the same label also warrant analysis - visually, these neurons should all be clustered together. Underlying reasons for this split may lie in the KOEs and other orbital dynamics that makes each subcluster a distinct family.

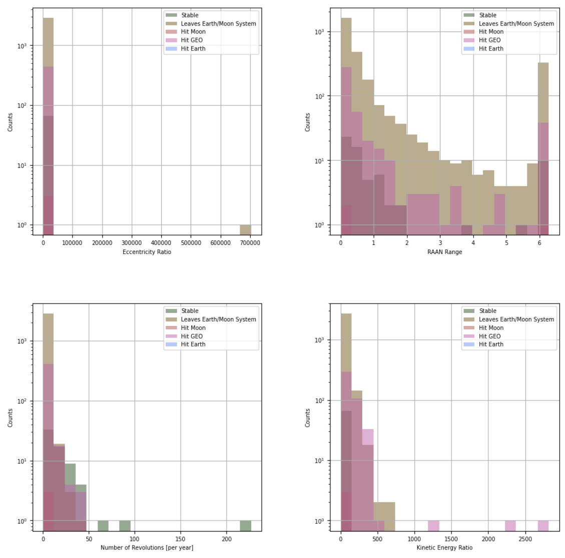


(a) High Lifetime Cluster

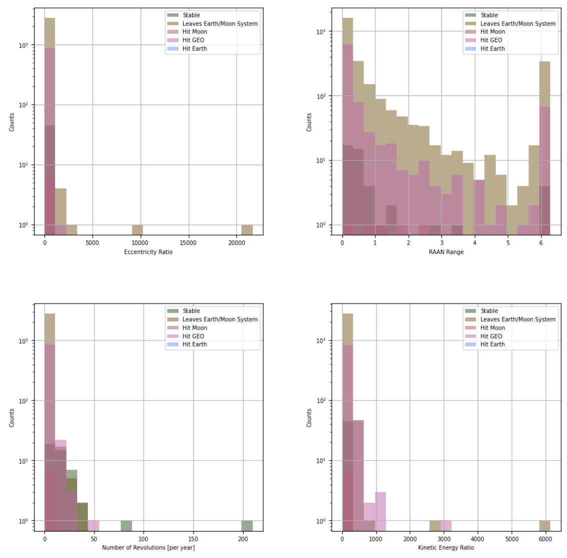


(b) Low Lifetime Cluster

Fig. 6: Cluster Histograms



(a) Pink Island



(b) Purple Island

Fig. 7: Island Histograms

The more pink regions of Figure 5-a (left side, between the bottom and middle) are comprised of orbits that could have potentially been clustered in the high lifetime cluster. Here, we see that the eccentricity ratio of orbits on this pink ‘island” is far higher than that of the larger cluster, which could be why the orbits were sequestered into their own cluster. On the other hand, both the eccentricity and kinetic energy ratio of the orbtis on the more ‘purple' island (spanning the bottom) are lower than that of the low lifetime cluster.

4.2 Neuron Weights and Contents

Each neuron in the SOM has weights associated to it that are the same length as the input data. Ideally, each set of neuron weights should roughly be representative of the orbits that fall within that neuron (i.e., cell). Although the weights are initialized through a stochastic process, if a neuron is chosen as the BMU of the input data during the competition stage, the weight takes on the values of the input data. The weights surrounding the BMU are subsequently adjusted according to the chosen neighborhood function and learning rate, so even empty neurons can have weights that reflect the data that fell into the neurons around it. Visualizing the neuron weights and associated contents enables understanding for the logic the SOM uses to sort the input data.

Although preprocessing allows the data to be clustered more effectively by the SOM, visualizing the weights from the raw data is difficult, as orbit state vectors have potentially gone through normalization, ROPE, and PCA, and the data and neurons will look unintuitive. For this reason, visually understanding the raw state vector data can aid in first understanding the association between the weight and the cluster contents. Below are a sample of weights from a low lifetime neuron, mid lifetime neuron, empty neuron, and high lifetime neuron, respectively.

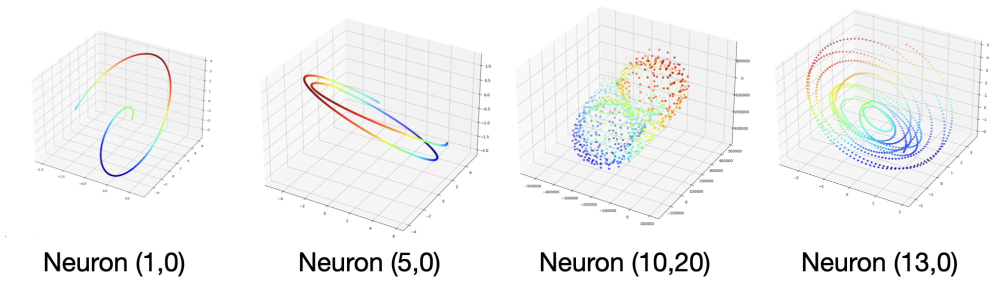


Fig. 8: Sample of Four Weights from SOM made from Raw Data

From this sample of four weights, the most chaotic weight (which comes from a high lifetime cell) is chosen for additional content investigation in Figure 9. From inspecting Figure 9, which only includes 12 orbits within that given cell, there is clearly a correlation between the shape of the weight and its content (i.e., the orbits that fall within that cell). While these orbits may be grouped together in the SOM generated from raw data, the clustering will be refined through subsequent preprocessing steps, where orbits that look similar in normalization, ROPE, and PCA space will end up grouped together.

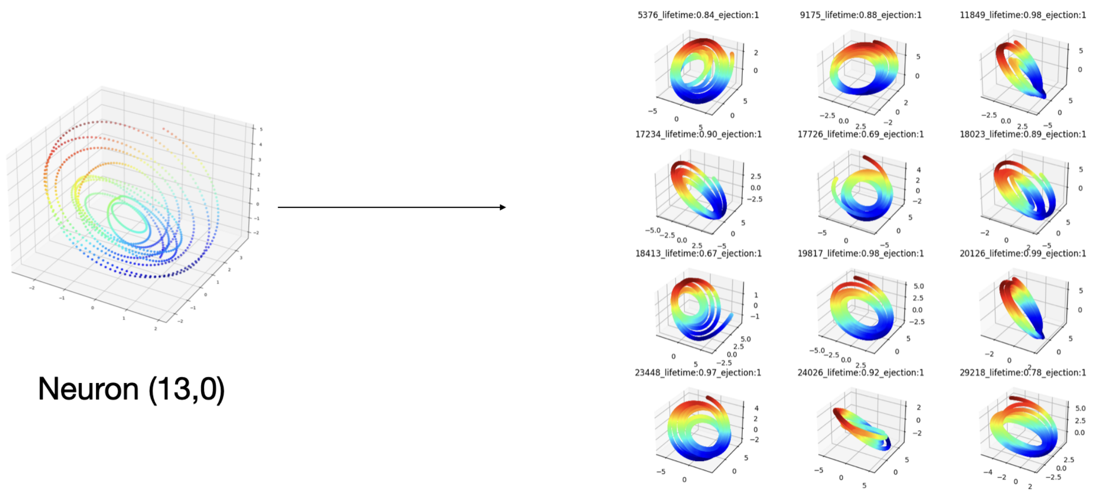


Fig. 9: Neuron Weight and Contents

The empty neuron (the third subfigure in Figure 8) had no orbit fall into it, but nonetheless, it ended up with an orbit-like shape. During the SOM generation stage, there was potentially an orbit that fell into this neuron that ended up getting organized into a different neuron in the end, which may have left an impression on the weights of that neuron. Alternatively, since the neighborhood function is dependent on the chosen neighborhood radius, which is blank, the weights of this empty neuron could've been adjusted according to this function if a neighboring cell contained an orbit. Regardless, some empty neurons may appear as noise, while some may have an orbit-like shape. Figure 10 shows a 'neighborhood' of 9 cells, and what each of the neurons looks like in that neighborhood. The bottom two cells have orbits inside them, while the rest are empty. By investigating the surrounding neurons, both empty and not empty, the role of the neighborhood function in adjusting the weights is clearly identified, and results in orbit-shaped neurons despite there being no data inside.

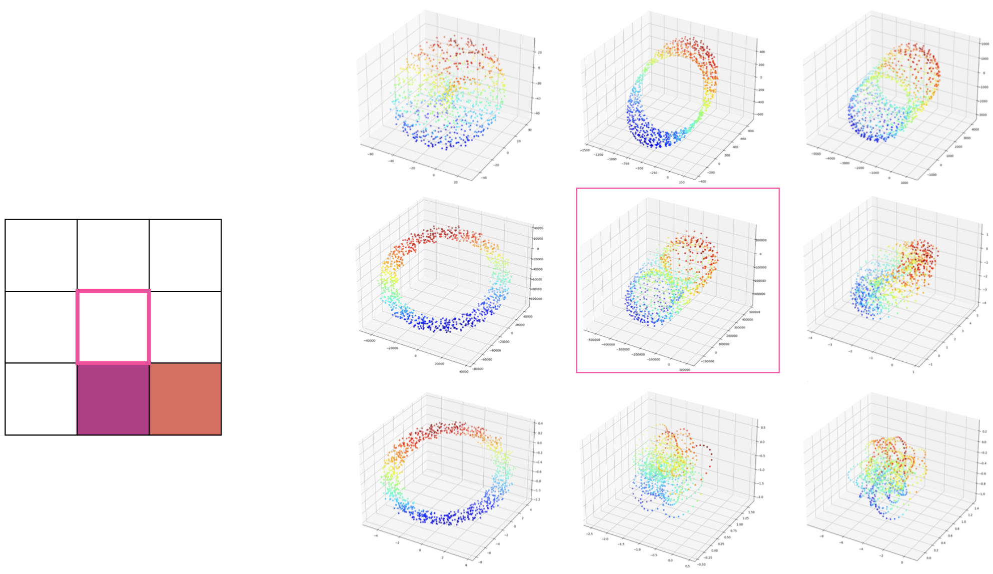


Fig. 10: Neighborhood of Empty Neuron

As shown in the pre-processing parameter sweep, the raw orbital data does not produce optimal clustering; the combination of PCA, MinMax normalization, and RoPE does. Inverting the effects of RoPE after the data has been interpolated is difficult, therefore interpreting the neuron weights, and what orbits might fall in an empty cell, is difficult. Shown in Figure 11 are neuron weights from a PCA and MinMax SOM. Although the SOM is worse at recovering lifetimes without RoPE, the following neurons are visualized to show how PCA effectively works as a dimensionality reduction technique.

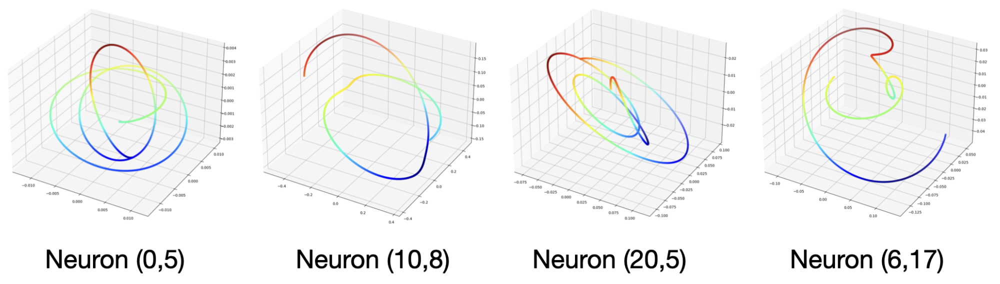


Fig. 11: Sample of Four Weights from SOM made from PCA Data

The data with PCA removed can no longer be described as an orbit, but rather a sort of motif for the input data. These motifs look starkly similar to the weight motif, moreso than the neuron contents compared to the neuron weights in the case of the raw data. The resemblance between the weight and contents in this case is striking. Figure 12 shows a neuron weight from a single SOM cell, with 12 orbits that fall inside that cell, in which all normalization techniques have been applied. This resemblance demonstrates how the preprocessing allowed for the SOM to sort through this large dataset more effectively. Ultimately, the best combination uses both RoPE and PCA, but the process of undoing RoPE after interpolation is still being examined.

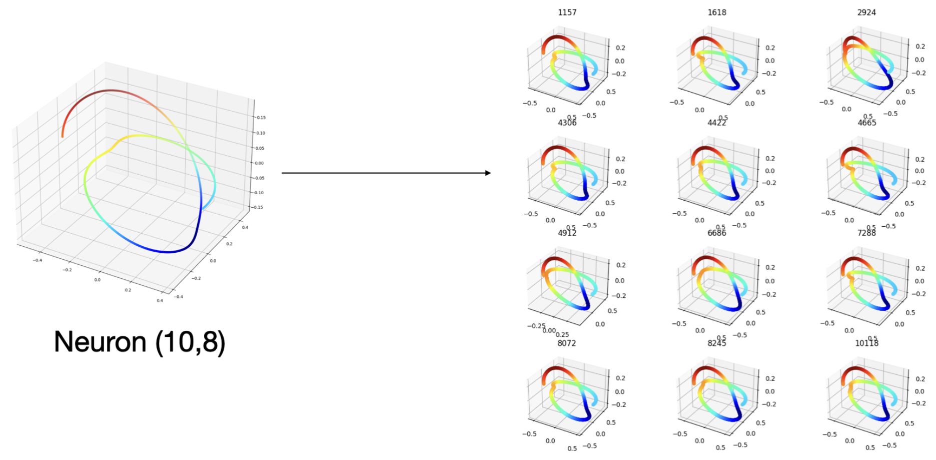


Fig. 12: Neuron Weight and Contents

4.3 Anomalous Neurons and Outliers

As a feature reduction tool, the SOM is exceptionally good at grouping together orbits that are similar. Conversely, it also places anomalous orbits in neurons separate from the rest, resulting in low frequency neuron 'islands'.

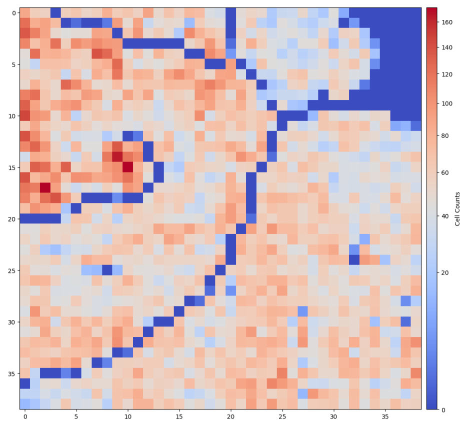


Fig. 13: SOM Density Plot

Low frequency neurons harbor orbits that are distinct. These neurons typically reside at cluster peripheries near chasms, mostly near clusters of similar labels. The hole in the medium lifetime cluster at (cell [8, 25] in Figure 13) and in the low lifetime cluster (cell [32, 25] in Figure 13) are of interest, particularly of interest are the orbits that fell into the surrounding neurons. First we investigate the randomly selected anomalous orbits in the high lifetime cluster (see Figure 14). Next, random orbits that fell around the hole in the low lifetime cluster are plotted (see Figure 15).

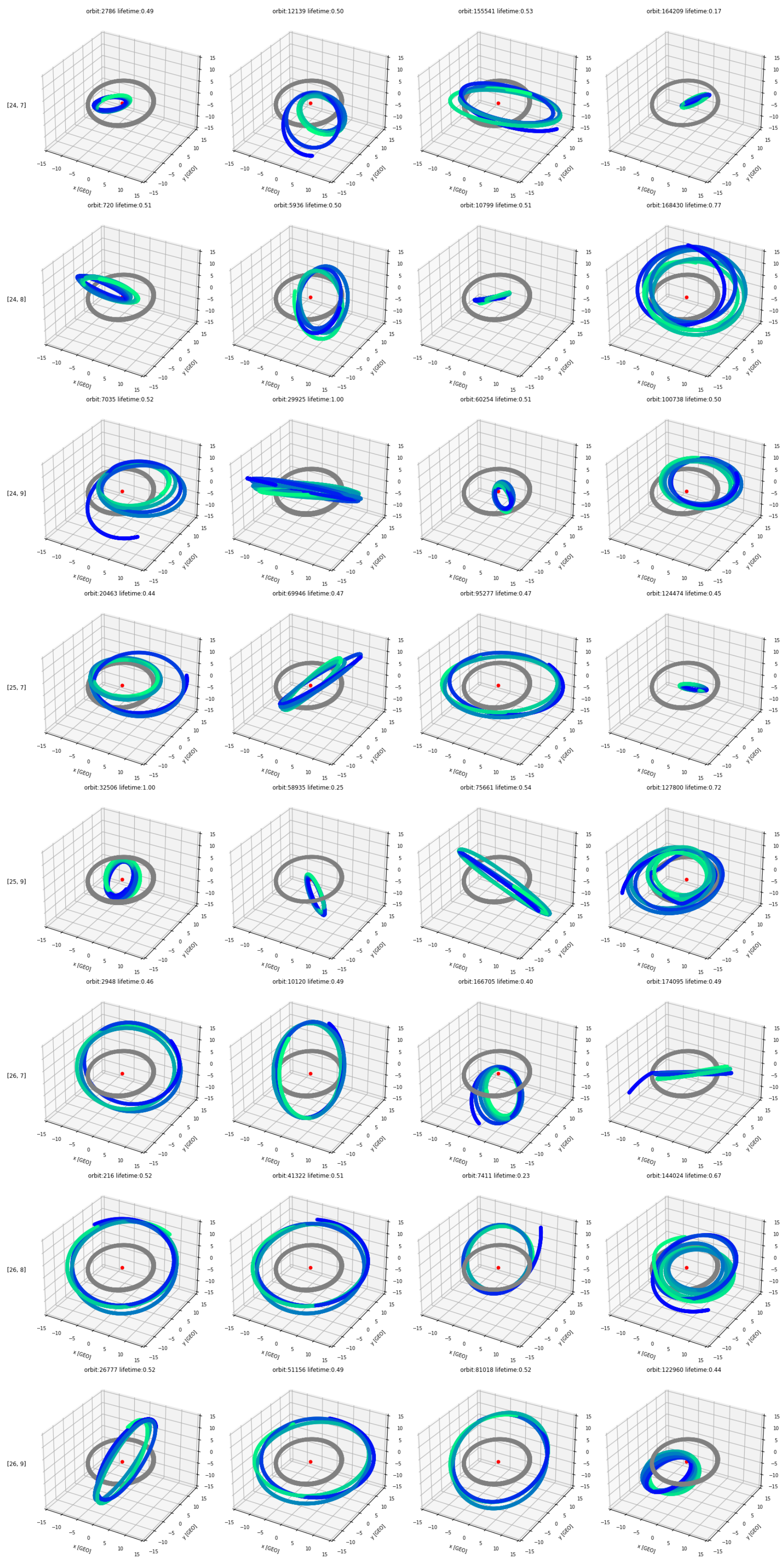


Fig. 14: Orbits in Cells around [8, 25]

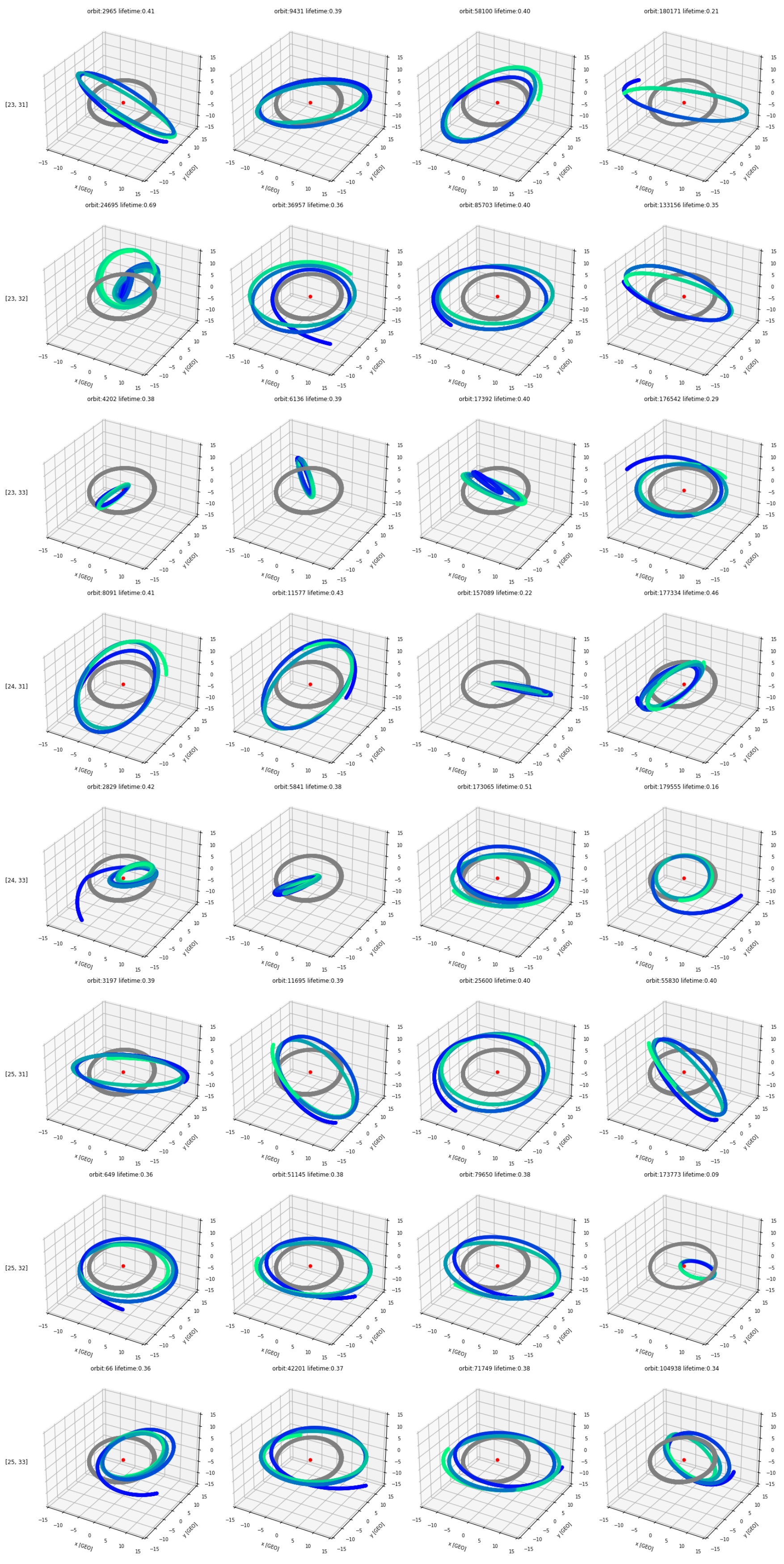
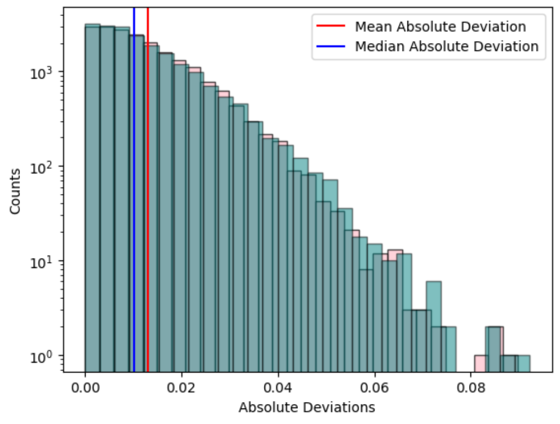


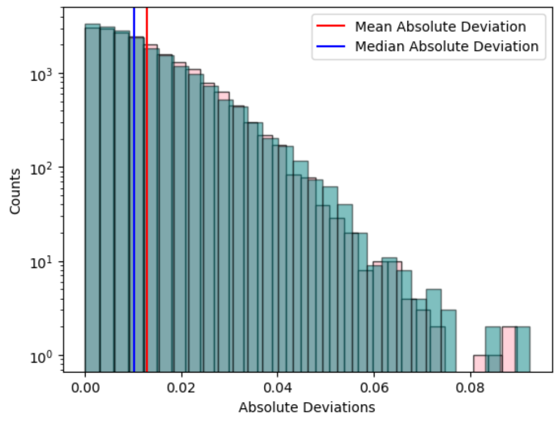
Fig. 15: Orbits in Cells around [32, 24]

4.4 Mission Oriented Applications

The rich, high fidelity simulation data is a great starting point to create a machine learning model for mission relevant applications. For real world applications, you want to simulate a planned orbit and understand the forces that will act on that orbit over its lifetime. The accurate time series information of the entire orbit is not necessarily known ahead of time. For this reason, we create a subset of test data using just one week of orbital data, then test using a SOM that was trained on full orbital data over 1 year. Using this SOM we try to predict the lifetime of an orbit using just one week of data, where we know the truth lifetime over the 1 year, and have that to compare to for accuracy. We then repeat this using the full orbital data. Figure 16 shows the absolute deviations for these test orbits, using the same methods described earlier, where we compare the lifetime of the test orbit to the mean or median of the lifetimes in the neuron the orbit fell into, then the mean or median of all deviations is determined. On average, this method can predict orbit lifetimes to approximately 3.5 days, where the SOM recovers the lifetime of the full orbit equally as well from one week of data as it does from the full orbit.



(a) Full Orbit Data



(b) One Week of Data

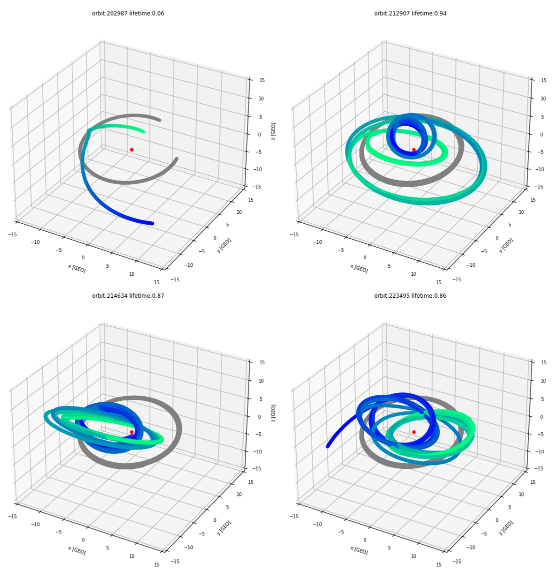
Fig. 16: Mean and Median Absolute Deviations

Worth noting is that the outliers between the two histograms in Figure 16 are not the same orbits. Figure 17 shows orbits that appear as outliers in the histograms in Figure 16. As illustrated by the left panel of Figure 17, the SOM has more difficulty with complex, chaotic orbits using the full orbital data. From the right panel of Figure 17, we see that with one week of data, the sparse, simple orbits pose more of an issue to its understanding.

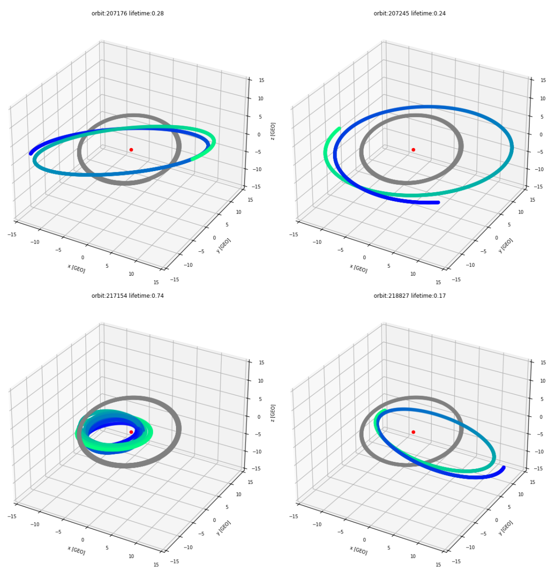
4.5 Orbital Families

To apply this machine learning method to ‘families’ of orbits in cislunar space, we must first construct and define what families are. We create various ‘flags’ that query the osculating KOEs and construct subsets of orbits within the full dataset; then, ‘families’ are created using a combinatorics method on the flags. Each family is comprised of one or more flags. The rules and populations of the flags are outlined in Table 4.

A first category of flags is comprised of general orbital mechanics – is the orbit periodic? Circular or eccentric? How does it rotate? The periodic orbits are made up of 6,541 orbits whose ratio of minimal and maximal semi-major axis values stays around 1.0 throughout its lifetime; alongside this, its ratio of total Energy must remain around 1.0, which attempts to constrain this family to only non-chaotic orbits. The highly eccentric orbits, made up of 3,663 orbits, are those whose eccentricity remains above 0.8 throughout the orbit lifetime. The simply eccentric orbits, the largest flag describing orbit shape with 54,195 orbits, constrain the eccentricity to be between 0.2 and 0.8. Circular orbits have an eccentricity that reamins below 0.2; there are 8,312 orbits in this flag. There are orbits whose eccentricity changes drastically over its lifetime, like those who begin circularly and then become elliptical – the changing eccentricity flag attempts to describe these with the range of eccentricity being above 4, with 47,027 orbits matching this criteria. Beyond orbit shape, polar and equatorial orbits are also considered. A polar orbit is defined as one whose incliantion



(a) Full Orbit Outliers



(b) One Week Outliers

Fig. 17: Histogram Outliers

stays between 85° and 95° over its lifetime, comprising 5,233 orbits. Equatorial orbits on the other hand have an inclination below 10° over its entire orbit, of which 9,990 do. Orbits whose inclination changes more than 90° during its lifetime are flagged as having a large plane change, and 472 do.

In the next category, the location of the orbit is used as a flag. Orbits are referred to as anterior if they stay between the moon and the Earth during its lifetime; there are 2,804 of these. Conversely, orbits are superior if they stay beyond the moon but remain in cislunar space in their lifetime, of which there are 48,941. We define an orbit to be centric of some region in space (the moon or a lagrange point) if the distance between their respective position vectors remains below 1.5 GEOs. An orbit is defined as crashing into Earth if it comes with 36,000 km; 10,644 in the sample do. The orbits leave cislunar space if they go beyond 18 GEOs, or 648,000 km.

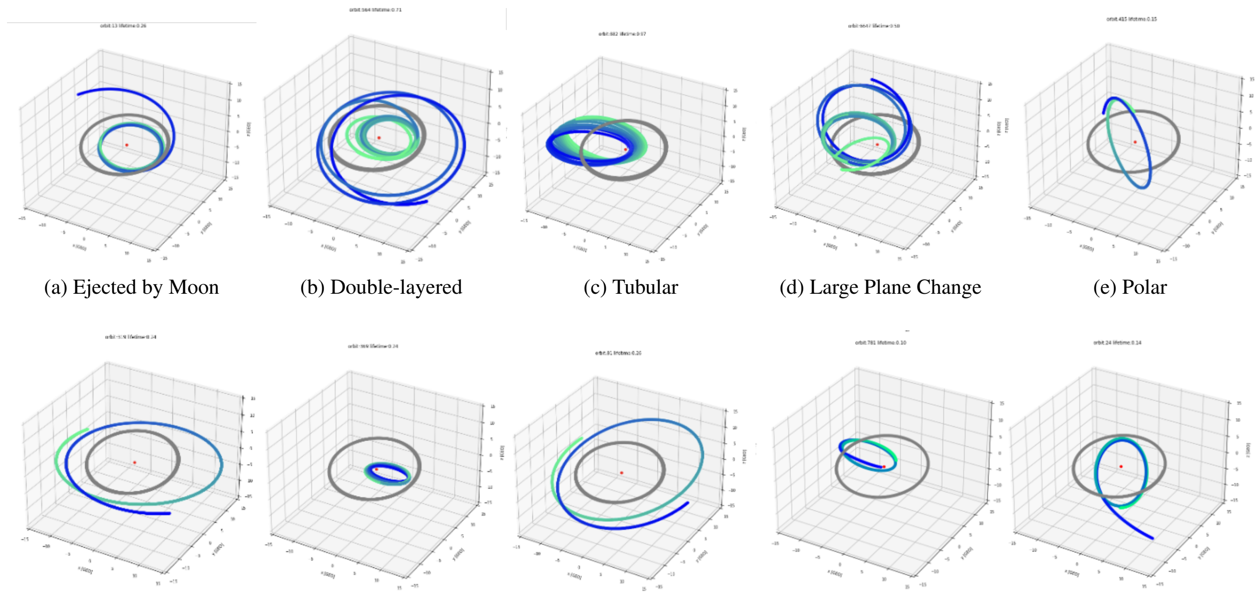
More specific flags are defined in a third category. In the process of creating SOMs and inspecting their contents, groups of orbits with distinct shapes have made themselves known. A tubular orbit is a periodic orbit that is rotating through space, carving out a tube. Cuts on RAAN and PA describe this tube, whereas the number of revolutions places a constraint on the size of the tube. The semi-major axis ratio also needs to remain near 1.0 to describe this orbit as periodic or near periodic. Next, the double-layered family is defined, where an orbit rotates in a certain way and then increases its semi major axis to rotate slightly differently throughout its orbital lifetime. Here we also place a restriction on the number of revolutions that an orbit needs to complete to fall into this class. Next, some orbits are ejected by the moon and produce a distinct tail, meaning both its semi major axis and eccentricity increase. Finally, some orbits can be classified as transfer orbits, such as eccentric orbits that stay between the Earth and the Moon.

Families are then constructed by combining common orbits from all of the flags listed in Table 4. The fifteen largest families are subsequently investigated. Table 5 shows the combination of flags the result in the highest number of families. Figure 19 shows where on the SOM that each orbit within a family falls, plotted over the SOM u-matrix which visually shows the SOM weighting and helps see the structure and empty regions. From Figure 19 you can see that some families occupy most of SOM space, such as subfigures a, b, c, and f, while some families have very distinct regions and cling to SOM cluster peripheries, such as subfigures i, l, and n.

Some orbits are not labeled by any of the current flags, warranting investigation for further labels to add to the dataset. Figure 20 shows the only 12 unlabeled orbits resulting after all flags have been applied.

Table 4: Family Flags

Flag Name	Flag	Rule	Time	Population
Periodic	Pe	$0.9 < a_{max} : a_{min} \leq 1.1$ $0.9 < E_{max} : E_{min} \leq 1.1$	orbit lifetime	6541
Highly Eccentric	hEcc	$e > 0.8$	orbit lifetime	3663
Eccentric	Ecc	$0.2 < e \leq 0.8$	orbit lifetime	54195
Changing Eccentricity	delE	$\Delta e > 4$	orbit lifetime	47027
Circular	C	$e \leq 0.2$	orbit lifetime	8312
Polar	Po	$85^\circ < i < 95^\circ$	orbit lifetime	5233
Equatorial	Eq	$i < 10^\circ$	orbit lifetime	9990
Large Plane Change	B	$\Delta i \geq 90^\circ$	orbit lifetime	472
Anterior	Sub	$r_{orbit} < r_{moon}$	orbit lifetime	2804
Superior	Sup	$r_{orbit} > r_{moon}$	orbit lifetime	48941
Moon centric	MC	$r_{orbit} - r_{moon} < 1.5GEOs$	two weeks	56
L2 centric	L2C	$r_{orbit} - r_{L_2} < 1.5GEOs$	3 days	0
L3 centric	L3C	$r_{orbit} - r_{L_3} < 1.5GEOs$	3 days	48
L4 centric	L4C	$r_{orbit} - r_{L_4} < 1.5GEOs$	3 days	27
L5 centric	L5C	$r_{orbit} - r_{L_5} < 1.5GEOs$	3 days	32
Crashes into Earth	Cr	$r_{orbit} < 36,000km$	at any point	10644
Leaves Cislunar Space	L	$r_{orbit} > 648,000km$	at any point	87914
Tubular	F	$\Delta RAAN > 0.2$ $\Delta PA > 0.4$ $revolutions > 8$ $0.8 < a_{max} : a_{min} < 1.2$	orbit lifetime	1302
Double-layered	DL	$a_{max} : a_{min} > 1.7$ $revolutions > 8$	orbit lifetime	5229
Ejected by Moon	T	$a_{max} : a_{min} > 2$ $e_{max} : e_{min} > 1.5$ $r_{orbit} - r_{moon} < 5GEOs$ for one day	orbit lifetime	8147
Transfer	Tr	$e > 0.5$ $r_{orbit} < r_{moon}$ $r_{orbit} - r_{moon} < 5GEOs$ for one day	orbit lifetime	1667



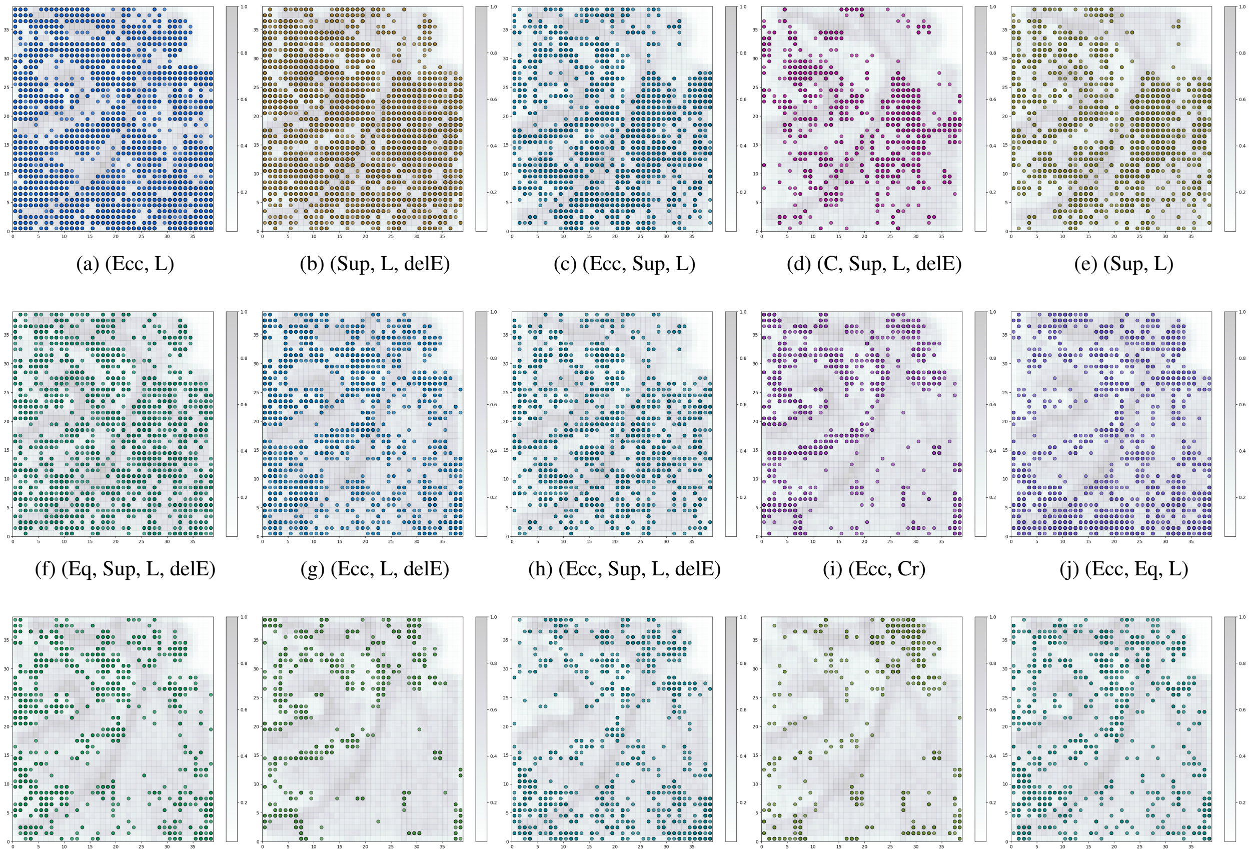
(a) Ejected by Moon (b) Double-layered (c) Tubular (d) Large Plane Change (e) Polar

(f) Equatorial (g) Anterior (h) Superior (i) Crashes into Earth (j) Leaves Cislunar Space

Fig. 18: Examples Orbits for a Subset of Flags

Table 5: Largest Families

Flags	Population	Location
(Ecc, L)	19433	(a)
(Sup, L, delE)	19005	(b)
(Ecc, Sup, L)	7105	(c)
(C, Sup, L, delE)	5724	(d)
(Sup, L)	3961	(e)
(Eq, Sup, L, delE)	3482	(f)
(Ecc, L, delE)	3019	(g)
(Ecc, Sup, L, delE)	2395	(h)
(Ecc, Cr)	2327	(i)
(Ecc, Eq, L)	2280	(j)
(L, delE)	1629	(k)
(Pe, Ecc, Sub, Cr)	1400	(l)
(Ecc, T, L)	1246	(m)
(Pe, hEcc, Cr)	1134	(n)
(Ecc, T, L, delE)	1133	(o)


(a) (Ecc, L) (b) (Sup, L, delE) (c) (Ecc, Sup, L) (d) (C, Sup, L, delE) (e) (Sup, L)

(f) (Eq, Sup, L, delE) (g) (Ecc, L, delE) (h) (Ecc, Sup, L, delE) (i) (Ecc, Cr) (j) (Ecc, Eq, L)

(k) (L, delE) (l) (Pe, Ecc, Sub, Cr) (m) (Ecc, T, L) (n) (Pe, hEcc, Cr) (o) (Ecc, T, L, delE)

Fig. 19: U-Matrix Plots of Largest Families

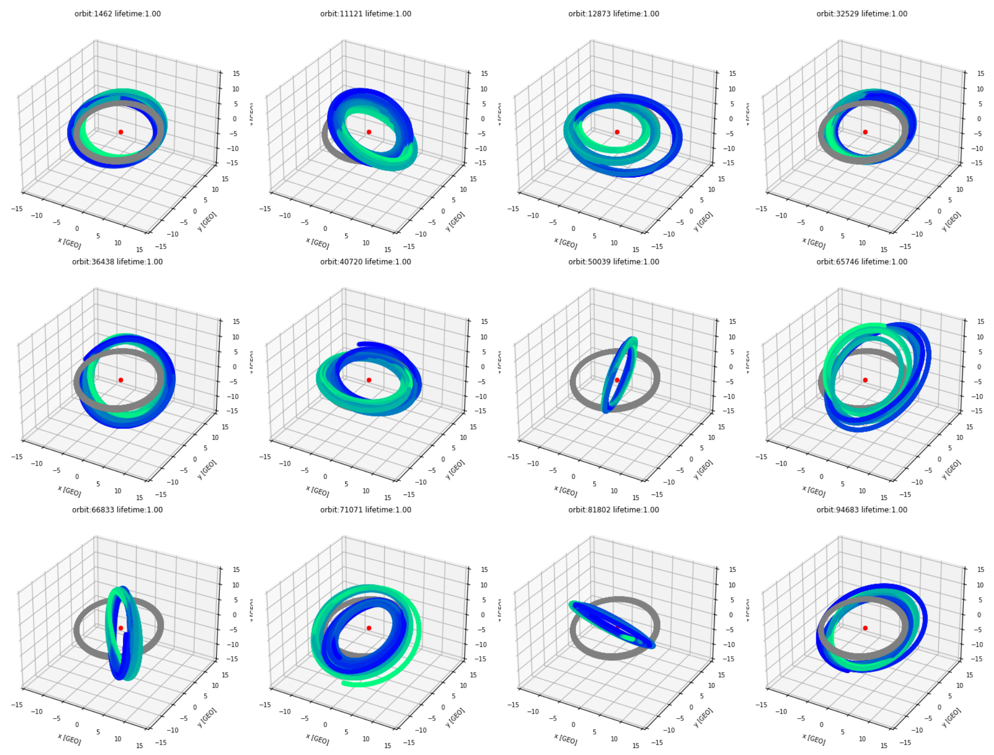


Fig. 20: Orbits with No Flags

We then analyze the number of orbits that are deemed lunar centric as a function of the number of days spent around the moon. Figure 21 illustrates that many orbits spend non-negligable time around the Moon. Note that our analysis consists of un-aided orbits that are not undergoing any kind of delta-v for station keeping or maneuvering of any kind. The orbits that are found in Figure 21 can be investigated further to understand how stationkeeping can be used to turn these into stable orbits that remain around the Moon.

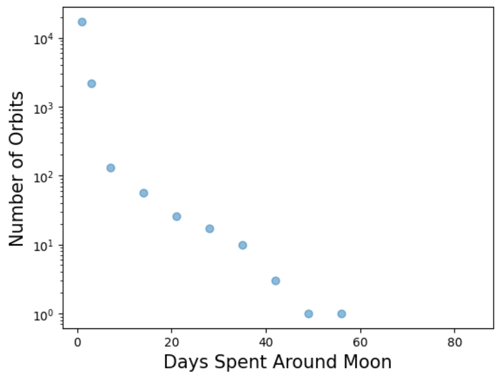


Fig. 21: Number of Lunar Centric Orbits by Days Spent Around Moon

5. CONCLUSION

Understanding cislunar space is imperative for the future of SDA. We explore the use of SOMs as a method to cluster and characterize families of orbits in this chaotic regime; through these methods, we recover lifetimes of potential orbits with a precision down to approximately 3.5 days. The dynamics of the KOEs are analyzed to create families of low and high lifetime orbits. We have recently expanded our one million orbit simulations out to six years and intend to expand our one year analysis to the six year data. Additionally, we plan to create new labels for distinct orbital shapes (eccentric, periodic, Moon centric, Earth centric, etc...). These discrete labels will lend to further exploration of potential cislunar families.

6. ACKNOWLEDGEMENTS

We would like to thank Tara Grice, James Buchanan, and Alexx Perloff for thoughtful and meaningful discussions on orbital dynamics and statistics. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the LLNL LDRD Program under Project Number 22-ERD-054. LLNL-PROC-XXXXXX.

REFERENCES

[1] E. Gorman et al. “QuantumNet: A scalable cislunar space domain awareness constellation”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*. Ed. by S. Ryan. Sept. 2023, 109, p. 109.

[2] Erin Jarrett-Izzi et al. *Investigation on Moon-based Sensor Placement for Cislunar Orbit Determination with Exclusion Zones*. Feb. 2024.

[3] Travis Yeager, Kerianne Pruett, and Michael Schneider. “Long-term N-body Stability in Cislunar Space”. In: 2023.

[4] Travis Yeager et al. “A Dataset of One Million Cislunar Orbits: Generating Statistics on the Stability of Cislunar Orbits”. In: *Advances in Space Research (In Prep)* (2024).

[5] Edward Schlafly et al. *Space Situational Awareness for Python*. Version 1.0. Nov. 2023. DOI: 10.11578/dc.20240417.1. URL: https://github.com/LLNL/SSAPy.

[6] Nikolaos K. Pavlis et al. “The development and evaluation of the Earth Gravitational Model 2008 (EGM2008)”. In: *Journal of Geophysical Research: Solid Earth* 117.B4 (2012). DOI: https://doi.org/10.1029/2011JB008916. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2011JB008916. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2011JB008916.

[7] Frank G. Lemoine et al. “High–degree gravity models from GRAIL primary mission data”. In: *Journal of Geophysical Research: Planets* 118.8 (2013), pp. 1676–1698. DOI: https://doi.org/10.1002/jgre.20118. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/jgre.20118. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/jgre.20118.

[8] Nipun Wijerathne Varuna Jayasiri. *labml.ai Annotated Paper Implementations*. 2020. URL: https://nn.labml.ai/.

[9] Giuseppe Vettigli. *MiniSom: minimalistic and NumPy-based implementation of the Self Organizing Map*. 2018. URL: https://github.com/JustGlowing/minisom/.