Synthetic Data Generation for Autonomic Computing

Catherine Saunders, Roy Sterritt, Peter Nicholl, Ian McChesney
School of Computing and Mathematics
Ulster University
Belfast, Northern Ireland
e-mail:{ce.saunders, r.sterritt, p.nicholl, ir.mcchesney}@ulster.ac.uk

***Abstract*—This paper discusses an approach that integrates data generation capabilities into the Autonomic Computing MAPE-K (Monitor Analyse Plan Execute and Knowledge Loop) to mitigate problems with data scarcity in autonomous space missions. The purpose of this work is to enhance the decision-making abilities of an Autonomic Manager by providing it with the ability to use simulation and data generation. A Conditional Tabular Generative Adversarial Network (CTGAN) is used to generate new synthetic datasets. Synthetic datasets are then evaluated to assess their utility. The evaluation results show that synthetic data can closely resemble the original data. However, this paper does not address the challenges of equipping a swarm with the necessary hardware, focusing instead on the feasibility of the proposed data generation pipeline.**

Keywords-Autonomic Computing; conditional generative adversarial networks; ctgan; autonomic manager; mape-k loop.

I. INTRODUCTION

Integrating data generation capabilities into the MAPE-K loop can address data scarcity challenges in space missions. The current trend in space exploration involves the development of autonomous swarms of small spacecraft that collaborate with each other to complete a common goal. NASA's Autonomous Nano Technology Swarm (ANTS) mission proposed using a swarm of a 1,000 small craft organized into 10 different classes depending on the instrument they carry [1][2]. Managing a large swarm requires a high degree of autonomy since human operators cannot manage each craft individually [3] .

The field of Autonomic Computing [4] aims to solve the complexity associated with managing a large swarm of autonomous craft. By incorporating Autonomic Computing concepts, such as the MAPE-K loop [5][6], each individual craft can self-manage its internal state and plan its actions. This work contributes to the concept of the MAPE-K control loop by adding simulation and data generation capabilities to enhance the analysis and planning stages. An in-built component that enables each swarm member to monitor and adapt its internal state helps decrease the complexity involved when designing a large swarm. Additionally, a Mission-level Autonomic Manager craft could be designated to oversee higher-level reasoning and planning tasks for the entire swarm. The MAPE-K loop could incorporate predictive analytics within the Analyse and Plan phases to improve decision-making. Prediction algorithms generally require substantial amounts of data to provide accurate results [7].

Recent advancements in Machine Learning (ML) have introduced techniques such as Generative Adversarial Networks (GANs) to help solve the issue of data scarcity [8] [9]. A generative model can be trained to produce new synthetic data that is statistically similar to the training dataset. This is particularly useful for ML prediction algorithms, as larger datasets can lead to better accuracy when identifying trends and relationships between features. However, the quality of the data is as important as the quantity; therefore, evaluation of the synthetic data is necessary to determine its usefulness. An area that could benefit from data generation is space exploration. Collecting sufficient data from space missions is a significant challenge due to high costs and risk associated with operating in a hazardous environments [10]. Deploying large swarms of spacecraft to unforgiving environments adds further complexity to mission management. Attempting to gather enough data to cover every possible scenario could prove costly and inefficient. Generative models address data scarcity by augmenting existing datasets with synthetic data that reflects the statistical properties of the real data.

By integrating data generation into the MAPE-K loop the issue of data scarcity can be mitigated, especially at the beginning of missions. A data generation pipeline could increase the dataset size so that there is enough data for prediction analytics. Additionally, the pipeline could enhance the dataset by interpolating new scenarios not captured by the swarm. This would allow the mission to gather comprehensive data at a lower cost. This synthetic data could then be used to inform more precise planning and deployment strategies. Enhancing real data is a step up from simulation and more cost effective than a full scale mission deployment. Synthetic data reduces reliance on real world data and can help augment the data and improve planning and prediction for future missions.

This work focuses on enhancing the capabilities of a Mission-level craft that oversees the mission as a whole. By equipping this craft's Autonomic Manager with a simulation capability it could help improve planning by simulating the future mission data. This data can then be analysed and used to train prediction algorithms. In addition to prediction modelling, simulation data could be used to train a Conditional Tabular Generative Adversarial Network (CTGAN) [11] model that generates new synthetic data. The purpose of this step is to evaluate the ability of the Autonomic Manager's data generation pipeline to produce synthetic data that is a good proxy for the real data. By using simulation data to train the generative model, it functions as a first pass of the data generation pipeline.

Once generated, the synthetic data is evaluated to check whether it is similar to the original simulation data. The purpose of this is to provide assurance that any future real data gathered by the swarm and used for data generation will result in high quality synthetic data.

The data generation process could also include an interpolation feature that produces new data for unseen scenarios. This interpolated dataset could then be used to train the CTGAN to produce synthetic data for scenarios that aren't present in the original dataset. Data augmentation would help enhance real data that is scarce or incomplete. Enhancing limited datasets could reduce cost and unnecessary wastage of craft. For this work, we assume the AM has sufficient GPU resources to perform data processing. Hardware issues, such as power consumption, memory, and bandwidth in space missions are important considerations, however, these are outside the scope of this paper.

In Section II, we provide background information on previous work; Section III discusses the main contribution of Data Generation in the MAPE-K Loop; Section IV discusses the data evaluation experiments and results.

II. Background and related work

In our previous work [12], we developed a simulation tool for testing communication strategies for robot swarms, varying cooperation and cohesiveness. The ideas discussed in this paper build upon the previous work, the simulation output datasets were used to train a CTGAN model that produced synthetic data similar to the simulation output.

The Autonomic Computing concept of an Autonomic Manager (AM) that exists within each craft could be expanded so that the overseer craft processes the mission data and uses this to plan future tasks. Having an in-built component that enables each swarm craft to self-manage by monitoring and adapting their internal state helps to decrease the amount of complexity involved when designing a large swarm. The overseer AM could simulate mission data and use this simulated data to test a data generation pipeline that produces good quality synthetic data.

To prove that simulation and data generation are a useful addition to the MAPE-K loop, it is important to evaluate the synthetic data's quality [13]. The evaluation of the synthetic data is necessary so that there is confidence in the quality of the synthetic data produced. If the simulated data is evaluated to be of high quality, the AM could use the real data and generate synthetic data to increase the dataset size. This data could also be enhanced to include swarm configurations that were not gathered by the swarm. The purpose of this would be to save on the cost of sending a large swarm. A relatively small swarm could gather a sample of data. The small dataset could then be used to generate a large amount of synthetic data. If the real data is insufficient for training prediction algorithms, the AM would have the option to use the synthetic data to train machine learning prediction algorithms used by its planning component.

The purpose of this paper is to show that data generation could be used as a proxy for real data, to prove this we have performed several comparative tests that evaluate the quality of the synthetic data. This is an important step as the data must be an accurate representation of the original dataset in terms of statistical similarity and feature relationships [14] [15]. Section III discusses the data generation process using CTGAN, 20 models were trained with various parameters.

III. Data Generation in the Loop

Data generation is accomplished by training a neural network that can learn the statistical properties of the training dataset. The goal of synthetic data generation is to improve the accuracy of machine learning models by increasing the size and diversity of datasets. They can also be used to enhance the privacy of individuals by creating synthetic data that anonymizes personal information contained within the original dataset.

A Generative Adversarial Network (GAN) consists of two neural networks models that compete against each other during the training process. The Generator model creates new data and the Discriminator model acts as a binary classifier that scores the new data on its accuracy to the training set. The adversarial training process continues until the Generator can produce data that can fool the Discriminator into classifying it as real.

Traditionally, GANs perform best with image data, however, a CTGAN was designed specifically for tabular data. It can work with categorical and numerical data it is also capable of learning the relationships between the features/columns. The GAN architecture consists of two models, each with an input layer, several hidden layers and an output layer. The input layer of the Generator takes a random noise vector and transforms it into output data resembling the training dataset. The Discriminator's classification of the output data is then used to calculate the loss function, a backward pass is performed through the Discriminator's network to update it's internal weights and improve its predictions.

The Discriminator is also used to help guide the Generator, a backward pass through the Generator network uses the Generator's loss value to determines how much the Generator's internal weights need to be adjusted in order to improve the data quality during the next iteration. The Generator never sees the training data; it relies on the feedback from the Discriminator. This learning technique is known as backpropagation and continues iteratively until the training process ends.

The training duration can be adjusted by modifying the number of Epochs. The Batch Size determines how many training samples are processed during one pass through the model. For this paper, we trained 20 CTGAN models and varied the Epochs and Batch Size parameters. In future work, additional hyperparameters may be considered to improve training performance. This could involve adjusting the Generator and Discriminator learning rates.

The tabular dataset outputted by the simulation includes many features, such as: 'Simulation Time', which robot discovered and found each item, and what time the items were discovered and found. The rules of the simulation stipulated that items could only be analysed by a robot of the same type as the item. This was simulated by rules that state that only a robot that is the same colour as the item can analyse that item. However, items could be discovered by any type of robot, messages would then be sent to the rest of the swarm asking for help from other robots. The swarm simulation output included all of the information discussed above, the dataset was compiled from 270 simulation runs and consists of ~50,000 rows. This was reduced to 4,600 rows for training, a sub-set of the data was used to train the CTGAN as this improved the chance of learning the distribution of the data. The subset also significantly reduced the amount of time required to train the CTGAN. The subset of data consisted of only the simulation runs in which the signal range was set to the lowest range and the robot swarm was split unevenly with 90% one type of robot and only 10% of the other type.

The flowchart in Figure 1 outlines the stages of the control loop for data generation using a CTGAN. The processing, training and evaluation code was written in Python. An AM could implement this pipeline to prepare real or simulated data for data generation. The pipeline performs data pre-processing to convert the time features to seconds so that there is consistency when performing calculations on the data. The original dataset is then filtered to a subset of the data based on key attributes such as robot split, signal type or type of communication protocol. Unique IDs were added to help maintain data integrity.

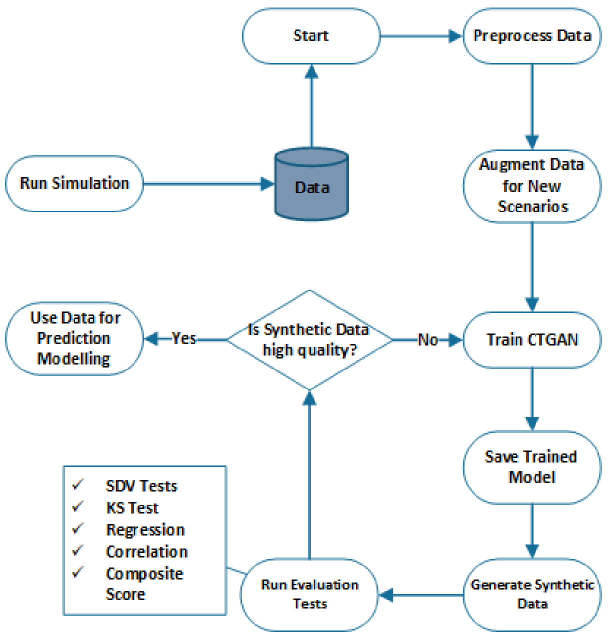

Figure 1. Flowchart of simulation and data generation pipeline.

Constraint rules are applied to ensure that the model learns the relationships between certain features and respects the rules of the original simulation. The constraint rules are essential to ensure that the synthetic data generated by the model follows the rules and relationships defined in the original simulation that produced the training dataset. In the swarm simulation, the following rules were defined:

- Time Relationship Rule: the Found (analysed) Time of the item must be greater than or equal to its Discovered Time.
- Robot and Item Matching Rule: this dictates that items can only be found/analysed by a robot of the same type (e.g., Colour).

An item can be discovered by robots that are unequipped to analyse it, but can only be found/analysed by robots that have the correct instrument. A robot that discovers but cannot analyse the item sends a help request to find a robot that has the correct instrument. If an item is discovered at time=10 seconds, it cannot have a found time less than this value. A Time Constraint check is added so that item 'Found Time' is always greater than or equal to 'Discovered Time'. The Matching Constraint rule is added so that items can only ever be found and analysed by a robot of the same type (e.g., Colour). If a blue item is discovered by a red robot, the robot must send a help request to find a red robot that can analyse the item. Applying a constraint that enforces this rule ensures that the relationships between robots and items are preserved in the new data. The constraint step is necessary to maintain the fundamental rules of the simulation and ensure the synthetic data is realistic.

The 'Train CTGAN' step uses the Synthetic Data Vault's (SDV) [16] implementation of CTGAN to train the model. The trained model can then be used to produce data that is similar in structure to the original data. The 'Evaluate Data Quality' stage uses a variety of metrics to assess the similarity of the generated data to the original test dataset. An overall composite score was then calculated from the key metrics were identified as most important. The composite score assigns a weighting to the key metrics tests.

IV. EXPERIMENTS

To assess the ability of the CTGAN to generate synthetic data similar to the original data, we conducted 20 training experiments. For each experiment, we varied the Epochs training time (100, 500, 1000, 1500, 2000), and Batch Size (50, 100, 250, 500), all other parameters stayed consistent. The Generator Learning rate was set to 0.0001 and the Discriminator Learning rate was 0.0002. The learning rate controls how much the models can learn within an iteration, with a lower value allowing the model to learn at a gradual rate. A lower batch size results in a longer training time as a smaller number of samples are viewed within each iteration. The entire dataset must be covered per epoch, therefore a lower batch size results in more iterations per epoch. Using a large batch size results in faster processing times and fewer iterations per epoch, however this can lead to less accurate weight updates and less accurate synthetic data. A larger batch size may require more epochs to reach the same results as a lower batch size and less epochs.

The CTGAN model training script was implemented in Python, it used PyTorch [17] to enable GPU acceleration. An NVIDIA RTX A500 GPU was used for training the GANs. The training script used an instance of SDV's 'CTGAN Synthesizer', this is an implementation of Conditional Tabular GAN (CTGAN) [11] which was designed to work with tabular data. The script saves the trained CTGAN model as a pickle (.pkl) file and generates a new synthetic dataset for comparison. The model .pkl file can be reused to generate more data if necessary.

A. *Evaluation Metrics*

To evaluate the synthetic data generated by the CTGAN, we used the open-source SDV library that provides a suite of evaluation metrics. These SDV metrics assess the quality of the Synthetic Data in terms of its similarity to the original data. In addition to the SDV tests, we also implemented tests that check for similarity. These tests used Python libraries such as Scikit-learn, SciPy and Pandas to perform regression, statistical and correlation tests.

The primary objective was to generate synthetic data that closely mimics the real data, especially in terms of how the simulation time varies with different communication protocols. The goal was to have the CTGAN capture the same feature relationships and distributions.

- ***SDV Evaluation Metrics***

The SDV metrics verify that the synthetic data adheres to the schema as the original data. The schema refers to the data types, categories, and numerical ranges. The Validity Score result should always be 100%, indicating the data adheres to the schema but is not a similarity check. For all tests shown below the Validity Score was 100%. The results for each test are shown in Table 1, along with the number of Epochs and Batch Size.

TABLE I. SDV EVALUATION METRICS

Test	*Eps*	*Batch Size*	*Data Quality*	*Column Shapes*	*Pair Trends*
1	100	50	91%	91%	91%
2	100	100	91%	92%	90%
3	100	250	90%	90%	90%
4	100	500	90%	90%	89%
5	500	50	91%	91%	91%
6	500	100	92%	93%	91%
7	500	250	92%	93%	92%
8	500	500	91%	91%	91%
9	1000	50	93%	92%	93%
10	1000	100	92%	92%	93%
11	1000	250	92%	92%	91%
12	1000	500	91%	91%	90%
13	1500	50	92%	92%	92%
14	1500	100	93%	93%	92%
15	1500	250	91%	91%	91%
16	1500	500	92%	93%	92%
17	2000	50	93%	93%	93%
18	2000	100	91%	91%	91%
19	2000	250	92%	92%	92%
20	2000	500	93%	94%	92%

The Data Quality metric is a composite score calculated from the 'Column Shapes' and 'Pair Trends' values. Column Shapes measures how well the distribution of features in the data matches those in the real data. The Pair Trends metrics analyses the relationships between columns. All tests scored similar results, it was therefore necessary to perform additional tests to gain more insight into the quality of the data.

- ***Statistical Similarity Metrics***

This Kolmogorov-Smirnov (KS) test checks whether the distributions of continuous variables are similar. The variables checked were 'Discovered Time', 'Found Time', 'Simulation Time' and 'Time Difference'. The 'Time Difference' variables gives the time between an item being discovered and found. The results for 'Simulation Time' and 'Time Difference' are shown in Table II.

The KS Test consists of the KS Statistic and the KS Complement, it measures the difference between the numerical values in the two datasets. It helps assess whether the distributions are similar and if the synthetic data is a reliable replacement for the real data.

The KS Statistic measures the maximum difference between the two datasets, a smaller KS Statistic results indicates that the datasets are similar. The KS Complement test transforms the KS Statistic into a score that is more intuitive for comparison purposes, the closer the score is to 1 the higher the similarity between the datasets.

Model 20 had the best KS Complement result for 'Time Difference'. Models that were trained with higher epochs generally show higher KS Complement scores. This suggest that longer training times are better at capturing the distribution of the data. For 'Simulation Time', Model 18 performed best as it had the highest KS Complement score of 0.950. This indicates that the synthetic data closely matches the simulation times of the original data. Figure 2 shows the distribution for 'Simulation Time' for Model 18, the Synthetic data approximately matches the pattern of the real data.

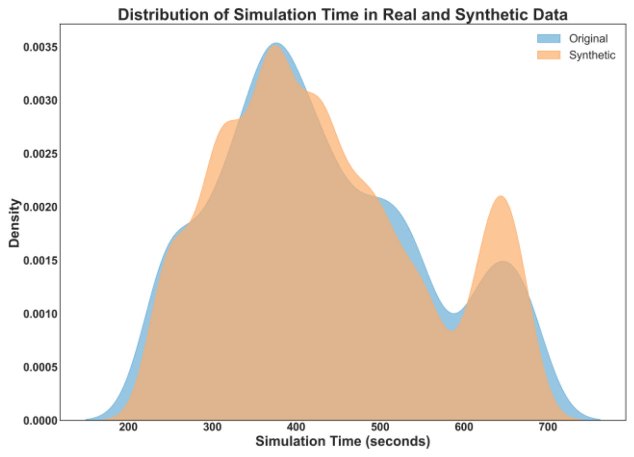

Figure 2. Simulation Time numerical distribution.

TABLE II. STATISTICAL SIMILARITY TEST

Test	*Eps*	*Batch Size*	*Simulation Time KS Stat*	*Simulation Time KS Comp*	*Time Diff KS Stat*	*Time Diff KS Comp*
1	100	50	0.148	0.851	0.396	0.603
2	100	100	0.096	0.903	0.309	0.691
3	100	250	0.091	0.908	0.372	0.627
4	100	500	0.087	0.912	0.367	0.632
5	500	50	0.122	0.878	0.355	0.645
6	500	100	0.069	0.931	0.205	0.785
7	500	250	0.069	0.931	0.331	0.669
8	500	500	0.107	0.893	0.358	0.642
9	1000	50	0.098	0.902	0.296	0.704
10	1000	100	0.073	0.926	0.265	0.735
11	1000	250	0.118	0.882	0.259	0.741
12	1000	500	0.106	0.894	0.311	0.689
13	1500	50	0.072	0.928	0.290	0.710
14	1500	100	0.069	0.930	0.282	0.718
15	1500	250	0.079	0.921	0.330	0.670
16	1500	500	0.087	0.913	0.345	0.655
17	2000	50	0.075	0.925	0.356	0.644
18	**2000**	**100**	**0.050**	**0.950**	0.325	0.675
19	2000	250	0.061	0.939	0.438	0.562
20	**2000**	**500**	0.069	0.931	**0.174**	**0.826**

- ***Regression Results for Simulation Time***

The Regression Analysis Test compares how well a Random Forest Regressor model that's trained on the synthetic data performs against the real data when predicting 'Simulation Time'. This test demonstrates the utility of the synthetic data and how useful it is as a proxy for the real data. We trained Random Forest models on both the real and synthetic data. A combined model (Model B) was also created by augmenting the real data with synthetic data.

Table III shows the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 scores for models trained on the real data, synthetic data and the combined dataset. The MSE is the average of the error rate between actual values and those predicted by the model. The RMSE value gives the root of the MSE, it shows how much the predictions deviate from the actual values, this value is in seconds and is easier to interpret. A lower MSE and RMSE indicates that the model's predictions are closer to the actual real values. The R^2 value indicates how much the changes in the 'Simulation Time' variable can be explained by the independent variables – 'Discovered Time', 'Found Time', 'Communication Protocol'.

The range for the 'Simulation Time' variable is ~90-700 seconds. The results for the real data are, MSE 12,495 sec², RMSE 111.78 sec, and R^2 0.26. The model trained on the real data has a prediction error rate of 111.78 seconds, this is high and suggests that the data may be too variable for the Random Forest to learn effectively. However, the error rates for the models trained using the Synthetic data are similar to those for the real data, demonstrating that the synthetic data is a good proxy for the real data in predictive modelling.

The model trained in Test 5 performs best and has a lower RMSE than the real data. Of the combined models, Test 14 gives the lowest RMSE result. Several models trained on the synthetic data and the combined data outperform the model trained on the real data.

The limited number of independent variables may explain the poor R^2 result. The CTGAN training dataset was reduced from having all signal ranges and robot swarm splits to just one signal range and one type swarm split. This may have hindered the Random Forest from learning as it cannot use the signal and robot split as independent variables. The lack of dataset variation means both variables are constants in the dataset and do not contribute to explaining the variance in 'Simulation Time'. Despite the low R^2 values, the results are consistent across models trained on both real and synthetic data. This suggests that the CTGAN has been able to capture the relationships within the subset of data.

The results show that models trained on the synthetic data perform similar to, and in some cases better than the models trained on real data. This demonstrates the utility of synthetic data in improving predictive performance.

TABLE III. REGRESSION RESULTS FOR SIMULATION TIME

Test	*MSE (sec²)*	*RMSE (sec)*	*R²*	*Comb. MSE*	*Comb. RMSE*	*Comb. R²*
1	10,320	101.59	0.25	11,325	106.42	0.33
2	10,969	104.74	0.22	12,051	109.78	0.28
3	12,608	112.28	0.04	12139	110.18	0.28
4	18,654	136.58	-0.20	13,947	118.09	0.17
5	**9,623**	**98.10**	**0.31**	11,990	109.50	0.29
6	10,894	104.38	0.32	11,217	105.91	0.37
7	10,270	101.34	0.28	11,473	107.12	0.32
8	9,902	99.51	0.28	12,495	111.78	0.26
9	11,041	105.08	0.21	11,401	106.78	0.33
10	10,316	101.57	0.25	11,800	109.09	0.30
11	10,534	102.64	0.18	12,070	109.86	0.29
12	10,208	101.04	0.25	11,080	105.26	0.35
13	11,088	105.30	0.23	11,742	108.37	0.31
14	10,220	101.09	0.24	**10,426**	**102.11**	**0.38**
15	10,084	100.42	0.22	11,954	109.34	0.29
16	10,015	100.08	0.29	11,553	107.49	0.32
17	10,584	102.88	0.35	11,477	107.13	0.32
18	10,473	102.34	0.29	11,252	106.08	0.33
19	9,777	98.88	0.29	10,858	104.20	0.36
20	11,682	108.09	0.24	11,973	109.42	0.29

- ***Correlation of Feature Importances***

The purpose of this test was to determine the degree to which the CTGAN preserved the relationships between features. To do this, we compared the results from a Feature Importances Test to assess the correlation between feature importances values for real and synthetic data. The correlation of Feature Importances results are shown in Table IV.

Figure 3 shows the Feature Importances results for Test 10, the blue bars show the amount of importance assigned to that feature when predicting the 'Simulation Time' feature. The red bars show the importance assigned to the synthetic data features. The Feature Importances Test measures how much each feature contributes to predicting the target variables ('Simulation Time' and 'Time Difference'). Two Random Forest Regressor models were trained on both real and synthetic data, and the importance of each feature calculated.

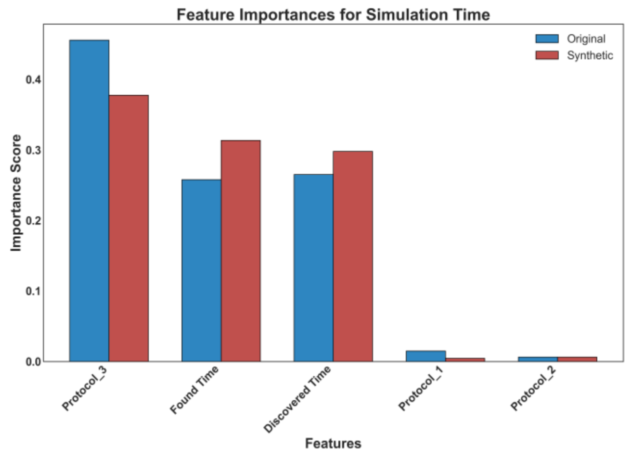

Figure 3. Feature Importances for Simulation Time.

To calculate the correlation of Feature Importances, we used the Pearson correlation coefficient. This compares the importances scores for both datasets and outputs a Correlation of Features result. A high correlation close to 1 indicates that the relationships between the features are preserved in the synthetic data. If the models disagree on which features are the most important then the correlation result will be low and closer to zero. The Correlation Similarity Score is a composite score for the Correlation of Features for both 'Simulation Time' and 'Time Difference'.

TABLE IV. CORRELATION OF FEATURE IMPORTANCES

Test	*Eps*	*Batch Size*	*Corr. of Features (Sim. Time)*	*Corr. of Features (Time Diff.)*	*Corr. Similarity Score*
1	100	50	0.950	0.996	0.960
2	100	100	0.910	0.996	0.959
3	100	250	0.683	0.990	0.919
4	100	500	0.442	0.982	0.880
5	**500**	**50**	0.981	**0.999**	**0.970**
6	500	100	0.980	0.995	0.968
7	500	250	0.980	0.999	0.923
8	500	500	0.985	0.999	0.966
9	1000	50	0.943	0.987	0.958
10	**1000**	**100**	**0.997**	0.973	0.967
11	1000	250	0.941	0.994	0.952
12	1000	500	0.974	0.999	0.966
13	**1500**	**50**	0.966	**0.999**	**0.970**
14	1500	100	0.960	0.999	0.964
15	1500	250	0.945	0.999	0.967
16	1500	500	0.984	0.993	0.957
17	2000	50	0.995	0.977	0.964
18	2000	100	0.993	0.995	0.968
19	2000	250	0.972	0.998	0.961
20	2000	500	0.964	0.982	0.962

Most tests show a high correlation above 0.9, this suggests that the synthetic data has preserved the relationships between features. Test 10 had a result of 0.997 for 'Simulation Time', indicating that the real and synthetic scores for feature importances are nearly identical.

B. *Composite Results*

The key metrics chosen to create the composite score were SDV Pair Trends, Overall Quality, Simulation Time KS Complement, Correlation of Feature Importances (Simulation Time), Correlation Similarity Score. These metrics were chosen to assess the statistical similarity between the synthetic and real data.

To derive an overall quality assessment, we used a weighted composite score to rank each model, the top five performing models are listed in Table V. A weighting was applied to each metric as follows: Data Quality 20%, Pair Trends 30%, Simulation Time KS Comp 20%, and Feature Importances Correlation 30%. By combining different evaluation metrics, the composite score provides a balanced view of each model's quality.

TABLE V. COMPOSITE METRIC SCORE

Test	*Epochs*	*Batch Size*	*Composite Score*
17	**2000**	**50**	**0.9485**
10	1000	100	0.9473
18	2000	100	0.9429
7	500	250	0.9402
19	**2000**	**250**	**0.9394**

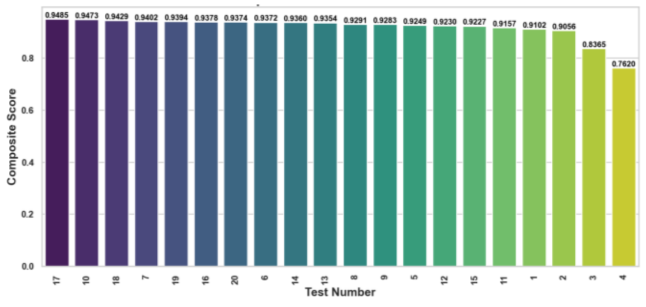

Figure 4. Composite score for 20 CTGAN models.

Model 17 (2000 Epochs, Batch Size 50) produced the highest composite score of 0.9485, this indicates a good similarity between the two datasets. The results suggest that using longer training times with low to moderate Batch Sizes are best for learning the distribution of the data. The poorest performers were models trained with only 100 Epochs, the composite score decreased as the Batch Size increased from 50 to 500. A larger batch size can speed up the training process but it results in fewer updates to the Generator's weights per epoch. Smaller batch sizes allow for more updates per epoch but they also increase training times. The composite scores are visualized in Figure 4, the bar chart shows the results of all 20 models for each of the key metrics. The visualization ranks the models from best to worst, with the best performing model ranked first.

V. CONCLUSION

In this paper, we discussed integrating a data generation pipeline into the MAPE-K loop, with the goal of alleviating data scarcity in autonomous space missions. Training a CTGAN often involves trial and error, making the integration of an evaluation component vital. Training parameters can greatly influence the quality of the synthetic data produced so it is important to evaluate the quality of the synthetic data produced. The results demonstrated that synthetic data generated by a CTGAN can closely mimic the real data in terms of feature relationships and distributions. Training the CTGAN for a high number of Epochs combined with a low Batch Size (2000 Epochs, Batch Size 50) produced the highest quality synthetic data. Future work will focus on the data interpolation component to generate new configurations of the swarm not present in the original dataset.

While synthetic data can offer a solution to the problem of data scarcity, it is vital that the practical utility of the data is evaluated. This presents a new challenge that requires a suite of metrics to give a comprehensive evaluation of quality. In addition to post-training evaluation metrics, it is important to pre-process the training dataset to remove errors and biases that could be propagated into the synthetic data.

This paper demonstrates the utility of using synthetic data to increase the size of an existing dataset. However, it does not address the practical constraints associated with equipping craft with hardware capable of performing the data generation. Generating synthetic data requires significant computational resources which may not be practical when operating in a constrained environment. Future work will look at solutions such as distributing computational load across the swarm or extending processing times to simulate hardware limitations, and increase system resilience by distributing the reliance of the Swarm away from one AM in case of damage or malfunction.

REFERENCES

[1] NASA. *Autonomous Nano Technology Swarm ANTS*, 2010 [Online]. Available from:
https://attic.gsfc.nasa.gov/ants/ArchandAI.html 2025.01.23

[2] S. A. Curtis, W. Truszkowski, M. L. Rilee, and P. E. Clark, "ANTS for Human Exploration and Development of Space," IEEE Aerosp. Conf. Proc., vol. 1, pp.255–261, June 2003, doi:10.1109/AERO.2003.1235057.

[3] P. A. Oche, G. A. Ewa, and N. Ibekwe, "Applications and Challenges of Artificial Intelligence in Space Missions," IEEE Access, vol. 12, pp. 44481–44509, Mar. 2024, doi: 10.1109/ACCESS.2021.3132500.

[4] IBM, *An Architectural Blueprint for Autonomic Computing*, IBM, 2003. [Online]. Available from: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf &doi=0e99837d9b1e70bb35d516e32ecfc345cd30e795 2025.01.23

[5] R. Sterritt, "Autonomic Computing," Innov. Syst. Softw. Eng., vol. 1, pp. 79–88, Apr. 2005, doi: 10.1007/s11334-005-0001-5.

[6] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," vol. 36, no. 1, pp. 41-50, Jan. 2003, doi: 10.1109/MC.2003.1160055.

[7] D. Werner, *Big data, advanced algorithms and new approaches for space missions.* [Online]. Available from: https://spacenews.com/big-data-advanced-algorithms-and-new-approaches-for-space-missions/ 2025.01.23

[8] I. Goodfellow et al., "Generative adversarial networks," Commun. ACM, vol. 63, no. 11, pp. 139–144, Oct. 2020, doi: 10.1145/3422622.

[9] L. Alzubaidi et al., "A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications," J. Big Data, vol. 10, no. 1, Apr. 2023, doi: 10.1186/s40537-023-00727-2.

[10] M. S. Hedayati, A. Barzegar, and A. Rahimi, "Mitigating Data Scarcity for Satellite Reaction Wheel Fault Diagnosis with Wasserstein Generative Adversarial Networks," 2024 IEEE Int. Conf. Progn. Heal. Manag. ICPHM 2024, pp. 367–376, June 2024, doi: 10.1109/ICPHM61352.2024.10627589.

[11] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," Adv. Neural Inf. Process. Syst. NeurIPS, Dec. 2019.

[12] C. Saunders, R. Sterritt, and G. Wilkie, "Collective communication strategies for space exploration," JBIS - J. Br. Interplanet. Soc., vol. 72, no. 12, pp. 416–430, 2019.

[13] A. Figueira and B. Vaz, "Survey on Synthetic Data Generation, Evaluation Methods and GANs," Mathematics, vol. 10, no. 15, pp. 1–41, 2022, doi:10.3390/math10152733.

[14] J. Jordon et al., *Synthetic Data -- what, why and how?*, [Online]. Available from: https://arxiv.org/pdf/2205.03257 2025.01.23

[15] A. Goncalves, P. Ray, B. Soper, J. Stevens, L. Coyle, and A. P. Sales, "Generation and evaluation of synthetic patient data," BMC Med. Res. Methodol., vol. 20, no. 1, pp. 1–40, May 2020, doi:10.1186/s12874-020-00977-1.

[16] DataCebo. *The Synthetic Data Vault*, March 2023. [Online]. Available from: https://sdv.dev/SDV/index.html 2025.01.23

[17] A. Paszke, "Pytorch: An imperative style, high-performance deep learning library," Adv. Neural Inf. Process. Syst., vol. 32, Dec. 2019. doi:10.48550/arXiv.1912.01703