GAMVT: A Generative Algorithm for MultiVariate Timeseries Data

Jamie Thorpe[*] Srideep Musuvathy[*] Stephen Verzi[*] Eric Vugrin[*]
Matthew Dykstra[*] Meghan Galiardi Sahakian[*]

Abstract
Intrusion detection and response algorithms are key technologies for improving network resilience against cyber-attacks. A key challenge to the development of these technologies is the sparsity of robust and realistic data sets that describe attack features. Data generation techniques have been previously developed to augment data sets when representative data sets are lacking, but algorithms for common applications (e.g., image and text) are frequently not applicable to the analysis of cyber systems. Cyber analyses often leverage timeseries data from a variety of system sensors, and, when viewed together, they form multivariate timeseries. The range of techniques for generating multivariate timeseries data is limited. This paper introduces GAMVT, a new method for generating multivariate timeseries data. GAMVT needs relatively few training samples compared to machine learning-based methods. GAMVT statistically characterizes each class of samples and generates new samples that are both distinct from and reasonably similar to the training samples. This paper demonstrates GAMVT for a space-cyber use case and compares results to those from generative adversarial networks and variational autoencoders.

Key Words: data generation, timeseries, cyber

1. Introduction

As cyber-attacks continue to evolve, tools for intrusion detection and response are becoming even more important to the security and resilience of critical cyber systems. However, the robust development and testing of these tools requires data that is not always readily available. Often, representative datasets for cyber systems, particularly data for anomalous or compromised conditions on the system, are either non-existent or extremely limited. Such datasets must be augmented before they can be used for tool development.

Typically, in a cyber system, data will come from a variety of system sensors. For example, if the cyber system in question is a car, timeseries might be collected from the speedometer, odometer, temperature and fuel gauges, etc. When viewed collectively, these sensor timeseries create multivariate timeseries samples which represent the state of the car over periods of time. Samples may be partitioned into classes based on different events which occurred in the system. One class may show the car's state during a normal drive, while another may include the car braking quickly to avoid a collision. For cyber systems in general, it is critical to identify key events that occur in the multivariate timeseries, as these events signal real-life conditions that may need to be addressed (detected, mitigated, etc.).

A variety of data augmentation tools currently exist, especially for text and image data. Unfortunately, these tools are not always applicable to multivariate timeseries data. Many existing techniques were not initially designed to leverage the unique relationships present in multivariate timeseries data, and applying such techniques to multivariate timeseries data could generate invalid samples. For instance, a simple way to augment an image dataset is to perturb the images by zooming, flipping, moving the subject, etc. However, this approach applied to multivariate timeseries data could disrupt the underlying temporal structure or truncate certain variables from the samples. In addition to the difficulty of

[*]Sandia National Laboratories, 1515 Eubank Blvd Albuquerque NM 87123

modifying models to use multivariate timeseries data, many existing techniques have large training data requirements, which is often infeasible for the limited datasets that exist for cyber systems. Tools specifically catered to multivariate timeseries data with small data requirements are preferable, but research in this area is very limited. Therefore, a new algorithm is needed to generate multivariate timeseries samples in order to augment the limited initial dataset and make it more robust for tool development and testing.

This paper introduces GAMVT: a Generative Algorithm for MultiVariate Timeseries data. GAMVT (pronounced "gamut") is a statistics-based approach that aims to mirror the process a human might use to identify patterns. Using these patterns, GAMVT can generate new multivariate timeseries samples which are representative of the system but still distinct from existing samples. Section 2 will discuss some related work in data generation. Section 3 will give a deep dive on the different stages of GAMVT. Section 4 explains timeseries clustering, a class of algorithms on which an early stage of GAMVT relies. Section 5 will give an overview of the use case for this work and present results comparing GAMVT to two existing data generation methods. Finally, Section 6 will discuss some areas for future work and extensions to GAMVT.

2. Related Work

Many techniques exist for data generation, but this paper will focus on data-driven approaches. Such approaches start with an existing set of data, train a mathematical model, and use that model to generate additional data (Sarkar, 2018). The most popular use cases for data-driven approaches are image and text generation. Much less focus has been given to data-driven approaches for generating multivariate timeseries data.

2.1 Image/Text Generation

Approaches for image/text generation can be loosely grouped into two categories: data manipulation and deep learning approaches (Shorten and Khoshgoftaar, 2019). Data manipulation such as simple perturbations (flip, rotate, zoom, etc.) work well for certain image datasets. However, applying these methods to multivariate timeseries will often generate invalid timeseries by disrupting the strict underlying temporal structure. Time runs in a single direction for all samples, and different variables or sensors are represented along the non-temporal dimension. Perturbing these samples disrupts this structure.

On the deep learning front, several approaches have been shown to be highly effective for image or text data. Generative Adversarial Networks (GANs) leverage competition between two sub-models to generate realistic data samples (Goodfellow et al., 2014). GANs have been successful on image datasets such as handwritten digits, faces, and cartoons, as well as text-to-image translations. Variational Auto-Encoders (VAEs) are another deep learning approach that learn how to encode data into a reduced representation and then to decode the data back into the full representation (Doersch, 2016; Kingma and Welling, 2019). Once trained, the decoding part of the VAE can be used for generation. VAEs have been successful on both image and text generation.

Despite the major successes of GANs and VAEs, such deep learning approaches require large datasets in order to train the models well. Thus, when the purpose of data generation is to augment a small dataset, deep learning models are often not the optimal approach because the existing dataset is too limited to sufficiently train the models. Additionally, any models designed to work well with image and text need to be modified to process multivariate timeseries due to the differences between spatial and temporal aspects of the data.

2.2 Multivariate Timeseries Generation

A few very recent efforts specifically focused on multivariate timeseries data have found some early success generating data using Long Short-Term Memory (LSTM)-based GAN models. Luo successfully trained LSTM-based GANs on multivariate timeseries data, but it was with the goal of imputing missing values in existing samples, not of generating new samples (Luo et al., 2018). Leznik demonstrated successful application of an LSTM-based GAN working to continuous network resource use data (Leznik et al., 2021). However, this work was built around continuous datastreams, where generated data was analyzed on a continuous basis, and an existing dataset was leveraged. Finally, Chen used an LSTM-based GAN to generate discrete data samples to augment the rare event samples in a rare event-prediction task (Chen et al., 2021). Several verification methods showed that their augmented dataset significantly outperformed the original unbalanced dataset. However, their use case was not constrained by limited amounts of data. In addition, the generalizability of Chen's approach has not yet been shown.

In all three works described above, there was sufficient data available to leverage the power of a deep learning data augmentation approach. Sahakian and colleagues discuss approaches taken in attempt to overcome the additional challenges of limited initial data. Progress was made in applying deep learning techniques to limited sets of multivariate timeseries data (Sahakian et al., 2021), but this work also highlighted the need for a new approach to multivariate timeseries data generation which does not rely on deep learning models. GAMVT is the proposed approach.

3. GAMVT

This section will discuss the design of GAMVT, walk through the full process of the algorithm, and give an illustrative example.

3.1 Design Considerations

The design of GAMVT was motivated by the hypothesis that the process of manual data sample classification could be simplified and automated, and that the information gathered from this process could be used to generate new samples. Manual classification of multivariate timeseries data usually involves identifying patterns, or events, in the timeseries. A human analyst might be able to recognize specific events in the data and pick out which samples contain similar sequences of events. If the analyst has some prior knowledge, this task can be further informed by the relationships between variables as well as the causal relationship between physical conditions in the system being analyzed and the events visible in the timeseries. GAMVT attempts to emulate this process by quantifying the patterns that define each class of samples. GAMVT then uses this information to generate new samples that fit those patterns. GAMVT was designed to:

- Honor the flow of time in the timeseries. Time moves forward from left to right along the temporal axis of the samples.
- Maintain separation between the outputs of different sensors. Although some sensors may be strongly correlated, each individual sensor is represented as a single variable in the data.
- Allow for control over which class of sample is generated. GAMVT maintains separate information for each class, allowing sample generation to be informed by the desired class label.

Although GAMVT is not a machine learning algorithm, similar language is used to facilitate more direct comparison with existing machine learning data generation approaches. The initial dataset is referred to as the “training set”, and the final data output from GAMVT is the “generated set”. Samples are further subdivided into unique “classes” and labeled.

3.2 The GAMVT Algorithm

The GAMVT algorithm consists of four major stages shown in Figure 1. The data from the training set is preprocessed. Then, each class is individually characterized. These characteristics are used to generate new samples. Finally, any desired postprocessing is applied to the data.

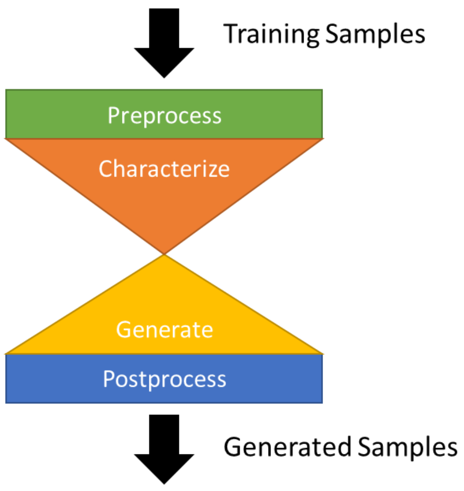

Figure 1: The GAMVT Algorithm

The resulting generated dataset contains samples that are unique from the training samples but still representative of the system and class. The generated samples should be indistinguishable from the training data in terms of variable types and temporal patterns. Samples can be generated specifically to address under-represented classes. All of this can be accomplished with limited training data, making GAMVT a valuable tool for augmenting multivariate timeseries datasets. The following subsections detail these stages of GAMVT.

3.2.1 Preprocessing

The Preprocessing stage serves two purposes. This first is to capture certain artifacts of the training data that must be maintained in the generated data, such as data types and sample shape. The data type of each variable should be consistent between the training and generated sets. For example, if the output of a sensor in the real system is a timeseries of integers, then the corresponding variable in generated data should also be a timeseries of integers.

The second purpose of the Preprocessing stage is largely driven by the performance of Characterization stage of GAMVT. The first step of Characterization is Timestep Clustering, which will be further explained in subsection 3.2.2. During development of GAMVT, Timestep Clustering often produced more intuitive results if the input data was simplified to capture the defining features of the events of interest in the timeseries. For example, if one variable represents the current position of a vehicle, but a particular event is defined by the velocity of the vehicle, then a good preprocessing step would be to calculate the rate of change of the position data.

Any simplifying function can be applied at the Preprocessing stage so long as it is invertible during the Postprocessing stage. Examples might include calculating rate of change over time or deviation from expected data for a particular variable.

Applying multiple transformations to a single variable may also be valuable. Transformations may be applied in combination or separately. Transformed timeseries may either replace the original timeseries for that variable or be appended to the data as an additional variable. Appending the transformed data gives GAMVT more timeseries to use to find patterns during the Characterization stage. This additional input could result in more diverse samples with a wider variety of distinguishable patterns, or it may make identifying the optimal relationships and patterns more difficult due to the presence of unnecessary data. Preprocessing function should be chosen and parametrized based on the dataset at hand.

3.2.2 Characterization

Once the training data is preprocessed, the next stage of GAMVT is Characterization. The purpose of this stage is to reduce each class of samples to a set of defining characteristics. This occurs in several steps, as detailed below.

1. Timestep Clustering -
   Timestep clustering assigns a label to each timestep of the multivariate timeseries by determining which timesteps are most similar to each other. This should subdivide the timeseries into sections defined by different "events" and help to identify where certain events reoccur in the samples. Timestep clustering can be achieved in many different ways. For more information on the specific timestep clustering techniques tested in this work, see the sections on Timeseries Clustering in Section 4.

   The output of Timestep Clustering should be a series of labels, one for each timestep in each sample. These labels are referred to as "cluster assignments", and a series of consecutive timesteps with the same cluster assignment is referred to as a "section". The cluster assignments will be used to separate out different events in the data, identify patterns in the order and timing of events, and collect aggregate statistics about the raw values of the data under different events.

2. Pattern Inference -
   It is assumed that training samples are subdivided into classes based on different events that occurred in the system during sample collection and that these events are identifiable during Timestep Clustering described above. The goal of Pattern Inference is to use these cluster assignments to infer a pattern that is unique to the given class.

   Pattern Inference is performed once for each class of training samples. For all samples in a given class, the cluster assignments are first collapsed, removing repeated assignments. The collapsed cluster assignments represent the order (but not the length) of identified sections. The set of collapsed assignments is then generalized to a single class pattern. This is similar to defining a regular expression that would match to the collapsed cluster assignments of each sample.

   In addition to the general pattern, statistics about section length are collected for each class. These section statistics record how many consecutive timesteps tended to belong to the same cluster, including the maximum and minimum length of the section and a distribution of the section lengths seen in the training set. If there are few samples in the class, the distribution will be represented as a finite set of lengths. If there is a larger variety of training samples, a random normal distribution can be parametrized to represent viable section lengths. In addition to being defined on a per-class basis, section statistics are further subdivided by section label and by the number of times a section with that same label has appeared earlier in the sample. This specificity allows for more complex patterns to be captured, improving the realism of generated samples.

3. Aggregate Value Statistics -
   Aggregate statistics are collected about the values in the preprocessed training data on a per-class, per-cluster, per-variable basis. The collected statistics include the mean, minimum, and maximum value across all the samples of a given class. As with Pattern Inference, subdividing the data in such a specific way further constrains the data generation process, creating more realistic samples. In addition, the underlying meaning of each cluster label must be captured. Each cluster label should be consistently defined across the full dataset. Therefore, all timesteps with the given cluster label, regardless of originating sample or class, are used to calculate the covariance between variables when a timestep falls into that cluster.

3.2.3 Generation

The next stage of GAMVT is sample Generation. The class characteristics collected in the Characterization stage allow GAMVT to generate new samples that are within the parameters of each class. The only input that is needed is the desired class label for the new sample. The desired sample shape is assumed to be the same as the training samples. The process described below is depicted in Figure 2.

1. Gather Relevant Characterization Data -
   Using the input class label, GAMVT selects the relevant class pattern, section statistics, and aggregate value statistics collected during the characterization step.

2. Select Specific Sample Pattern -
   Recall that a “section” is a series of consecutive timesteps in a sample with the same cluster label, and that the class pattern is a general definition which encodes variations in order of sections for samples within the class. There could be several specific patterns of sections encoded in this definition. A specific pattern for this sample is randomly chosen from the general class pattern.

3. Expand Sample Pattern to Timeseries of Labels -
   Given the specific sample pattern, the class section statistics are used to expand this pattern from the collapsed cluster assignment representation into a full timeseries of cluster assignment labels. Recall that these section statistics include minimum and maximum bounds on valid section length as well as a distribution of section lengths seen in the training data. This distribution is used to randomly select a length for each section, with the following constraints: 1) the minimum and maximum statistics ensure that sections of a given cluster label are not shorter or longer than sections of that same label in the training data, and 2) the randomly chosen section lengths must sum exactly to the full length of the desired sample size.

   The result of this process is a full timeseries of cluster assignment labels.

4. Expand Labels into Multivariate Timeseries Sample -
   GAMVT iterates through the timeseries of cluster assignment labels and expands each timestep from a univariate label to a multivariate timestep. This part of the generation process uses the class value statistics and the covariance matrices collected during characterization.

   The timeseries of cluster assignments is considered on a per-section basis. GAMVT parametrizes a random normal multivariate value generator with the covariance matrix relevant to the section's cluster label. Using this generator, GAMVT generates enough multivariate timesteps to fill the section. Then, the relevant class value statistics are used to ensure that each variable in each timestep falls within an expected range given sample class and section cluster label. If this is not true for any one variable in a given timestep, the timestep is discarded and a new one is generated. This cycle continues until there are enough valid multivariate timesteps to fill the current section. GAMVT then moves on to the next section and repeats the process through the end of the sample.

   This "generate-then-test" method of timestep generation is inefficient. In addition, due to the random value generator component, output can be noisy, depending on the tightness of the constraints from characterization. These issues are both left to future work (Section 6).

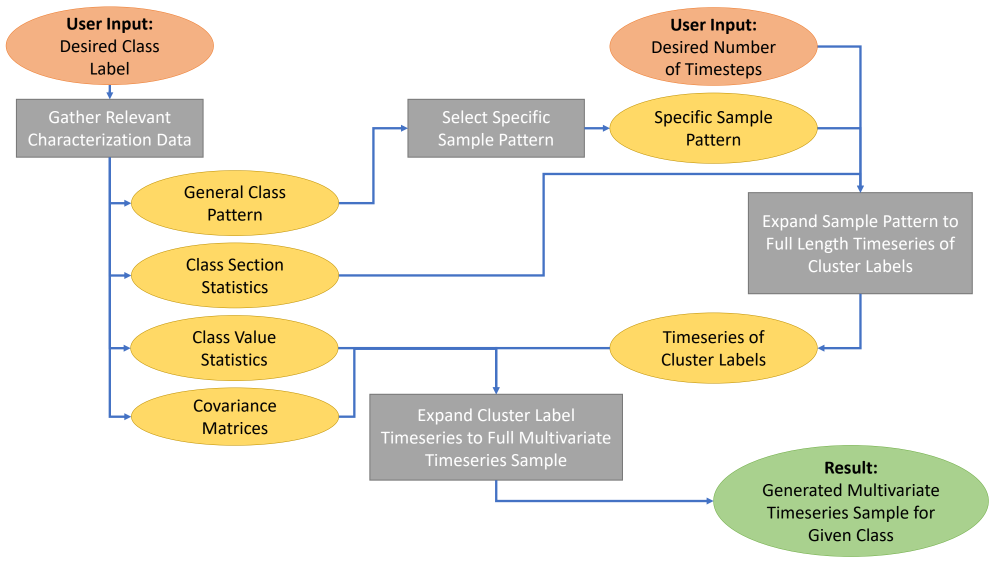
User Input:
Desired Class Label
Gather Relevant Characterization Data
General Class Pattern
Class Section Statistics
Class Value Statistics
Covariance Matrices
Select Specific Sample Pattern
User Input:
Desired Number of Timesteps
Specific Sample Pattern
Expand Sample Pattern to Full Length Timeseries of Cluster Labels
Timeseries of Cluster Labels
Expand Cluster Label Timeseries to Full Multivariate Timeseries Sample
Result:
Generated Multivariate Timeseries Sample for Given Class

Figure 2: The GAMVT Generation Stage Flow Chart

3.2.4 Postprocessing

The output of the Generation stage is a set of samples which adhere to all patterns and statistics collected during the Characterization stage. However, these statistics were collected on preprocessed data. The Postprocessing stage reinforces the data types collected for each variable, inverts any functions applied during preprocessing, and applies any additional smoothing or other postprocessing functions.

Postprocessing functions can require parametrization, and the best parameters may depend on how the relevant preprocessing function (if any) was parametrized or on the dataset itself. If a preprocessing function required certain input data (e.g. to subtract expected values from actual values), then this same data may be needed to successfully invert that function during postprocessing. As another example, one preprocessing function may calculate change over time in a particular variable. Inverting this function requires an initial seed timestep, which could reasonably be drawn directly from one of the training samples or generated by taking some function over a subset of the training data.

Postprocessing may include applying functions unrelated to the Preprocessing stage, with the purpose of making the generated data less distinguishable from the training data. For instance, sinusoidal training data tends to result in highly oscillatory generated data due to the inability to sufficiently constrain the random value generator during the Generation stage. A good postprocessing step might attempt to fit a sine wave to the oscillatory generated data samples.

GAMVT concludes after all desired postprocessing functions have been applied.

3.3 Illustrative Example of GAMVT Application

Figure 3 shows a notional example of the GAMVT algorithm. For simplicity, the example skips preprocessing and postprocessing. This example starts with three training samples of the same class of data. During the Characterization stage of GAMVT, the three previously described steps (Timestep Clustering, Pattern Inference, Aggregate Statistics Collection) are followed. For this example, the class pattern $A[BA]^{1,2}$ means that the class can be characterized by sequences that consist of section of “A” followed by one or two sections consisting of a subsection of “B” followed immediately by a subsection of “A”. Recall that the actual number of timesteps in these sections can vary, and this variation is captured by section statistics. Section statistics and aggregate value statistics are not represented in the figure.

The Generation stage is used to generate one sample in this example. A specific sample pattern, in this case $ABABA$, is chosen from the general class pattern. Using the section statistics to determine the length of each section, the collapsed pattern is expanded into a full timeseries of cluster assignments, such as $ABBBAABBA$. The covariance matrices and aggregate value statistics are then used as previously described to generate a multivariate timeseries sample. This sample will then go through postprocessing (not illustrated here) before being added to the generated dataset.

4. A Closer Look at Timeseries Clustering

Timeseries clustering is the first step of the Characterization stage for GAMVT, and it lays the foundation for the subsequent Characterization steps. The goal of this process is to divide the timeseries samples into sections where similar events are occurring. This helps to identify and differentiate events in the timeseries and to determine how long they last. This section will discuss the algorithms considered for timeseries clustering.

4.1 TICC: Toeplitz Inverse Covariance-Based Clustering

Toeplitz Inverse Covariance-Based Clustering (TICC) is an algorithm designed to cluster multivariate timeseries data into sections of similar timesteps (Hallac et al., 2017). TICC uses correlation networks (also called Markov Random Fields) to capture the relationships between different variables at a single timestep and across consecutive timesteps. This information is used to segment the series into clusters.

TICC has many tunable parameters, but this research focused on three:

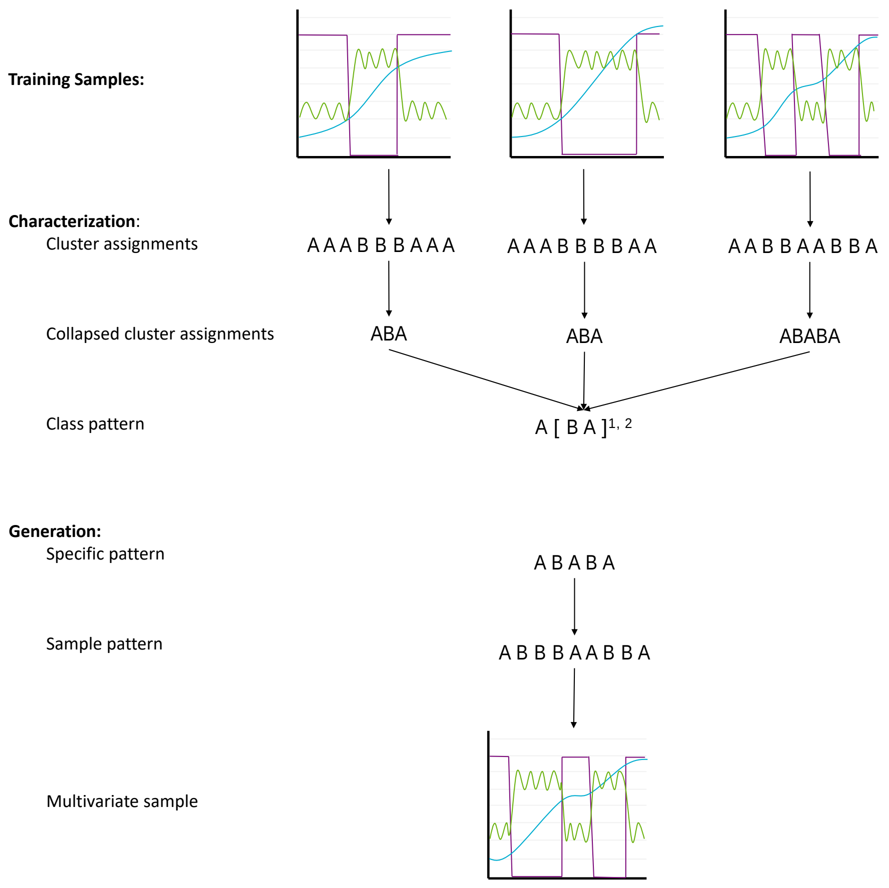

Figure 3: Notional Example of the GAMVT Algorithm

- Number of Clusters - TICC will create exactly this number of clusters, even if it is not optimal. This parameter should be carefully tuned, especially if the desired number of clusters is not clear from manually inspecting the dataset.

- Window Size - This parameter refers to the number of timesteps which will be considered when assigning each cluster label. For example, if the window size is set to five, then TICC will consider the current timestep being clustered as well as the next four timesteps in order to determine a cluster label. A window size that is too large may miss quick changes in events, while a window size that is too small may be hypersensitive to noise.

- Beta - This parameter determines how influential the additional timesteps in a window are when clustering the current timestep.

Some limitations were discovered while investigating the application of TICC to the use case described in Section 5. During experimentation, TICC was not able to capture relationships that were defined by a trend over time (i.e. linear rate of change). This is because TICC uses the raw values at each timestep to calculate correlation networks, not the change in data over time. This issue motivated GAMVT's Preprocessing stage. By preprocessing and simplifying the data to reflect key trends, TICC was better able to identify individual events in the data.

Another limitation to overcome was the fact that TICC operates on a single multivariate timeseries sample at a time, and that the correlation networks are not easily shared between different runs of the algorithm. Therefore, in order for the same correlation networks to be applied across all training samples, the samples needed to be carefully concatenated. This ensured that the definition of each cluster label was consistent and that all unique events were captured across the full set. Concatenating the samples as-is would cause the first timesteps of one sample to influence the clusters at the end of the previous sample, which is not desired behavior. Buffer timesteps were inserted to overcome this problem.

After careful parameter tuning and some processing to address the aforementioned challenges, TICC worked very well overall on the use case dataset (see Section 5). However, TICC had a lot of trouble on noisy, highly oscillatory, or wave-like data, and would often produce errors or never converge on such samples.

4.2 Distance-Based Methods

Given both the successes and challenges seen with TICC, a simpler distance-based method was investigated. Each timestep was treated as an n-dimensional point, where n is the number of variables in the multivariate timeseries sample. Then, all timesteps were fed individually to a simple K-Means clustering algorithm (Lloyd, 1982). The only parameter this method required was the desired number of clusters, which serves the same function for K-Means clustering as it does for TICC.

As with TICC, all samples were concatenated before applying the model. Note that, because K-Means does not use future timestep information, there was no need to add a buffer to any of the concatenated data as with TICC. In addition to cluster consistency, sample concatenation facilitates normalization of the entire dataset on a per-variable basis. This normalization step was not needed with TICC, but for a distance-based clustering method, normalization keeps the calculated distances from being heavily influenced by variables with large magnitude values.

On the use case dataset (Section 5), K-Means was able to find intuitive cluster labels, performing about as well as TICC. On some of the noisier datasets where TICC would not converge, K-Means was able to return cluster labels, although they often reflected the noisiness of the data and were not very informative. K-Means is certainly a simpler approach, which was viable for our use case. However, as TICC is a more sophisticated algorithm, it may perform better than K-Means on more complex datasets.

5. Results

5.1 Use Case

This use case was motivated by the development an automated response tool for a cyber system. This tool would be informed by a dataset of historical attacks to the system. Then, when a threat is identified on the system, recent sensor data would be forwarded to the automated response tool. The response tool would use this data to identify the current threat based on its similarity to historical threats, typically by applying a machine learning classifier. The automated response tool relies heavily on the historical threat dataset, which is a dataset of multivariate timeseries samples representing sensor output under different threat conditions. Because a large dataset of this type is often not available, a small initial dataset is generated and then augmented with synthetic samples. This is where GAMVT, or other data generation approaches, can come into play.

The system of interest for this use case was a space-cyber system, which was modeled using the NASA-developed emulation platform NOS^3 (Geletko et al., 2019). NOS^3

Attack Class	Description	Experiments Run
Class 0	No Attack/Baseline. Contains two planned 3-minute experiments, one at 14 minutes and one at 21 minutes.	5
Class 1	One or both experiments from Baseline are deleted.	3
Class 2	The start times for both experiments from Baseline are changed.	20
Class 3	Baseline experiments are replaced with multiple experiments at random start times, filling camera memory with useless data.	20

Table 1: Attack Classes

emulates a small satellite containing multiple hardware payloads. Emulation allows us to model the satellite in orbit using virtual hardware running real software. NOS^3 was configured to execute an imaging-based mission where the camera would take images over two locations of interest. Using the NOS^3 platform, a total of 48 experiments were run for 30 minutes each, producing multivariate timeseries samples representing the scenarios in Table 1. Because these experiments run in real time, only a small dataset was collected.

For each experiment described in Table 1, data was collected from the GPS and camera payload. Figure 4 visualizes one sample of the baseline data from Attack Class 0. The format of the image is as follows:

- Row 1 - Position data for the satellite, with X-, Y-, and Z-coordinate data in the red, green, and blue channels respectively.
- Row 2 - Velocity data for the satellite, with X-, Y-, and Z-coordinate data in the red, green, and blue channels respectively.
- Row 3 - Camera on/off state, indicating periods of time where the camera is actively collecting an image. Data is recorded in the red channel.
- Row 4 - Camera memory usage; note this value increases whenever the camera is collecting an image. Data is recorded in the red channel.
- Row 5 - Placeholder for data downlink status; nothing of interest in current dataset. Data is recorded in the red channel.
- Row 6 - Timestamp for the given column of data, recorded in the green channel only.

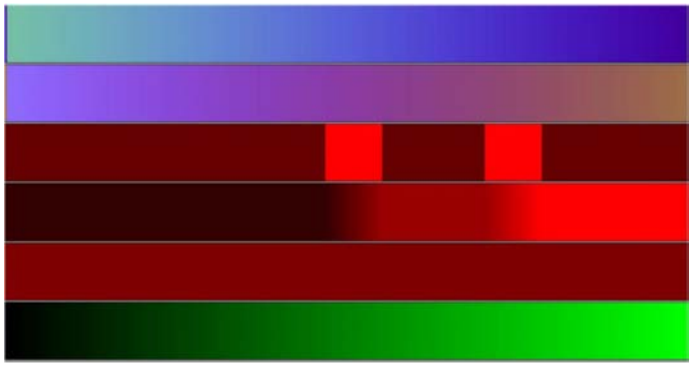

Figure 4: Visualization of baseline experiment (Attack Class 0).

The goal is to use a data generation algorithm to augment the small dataset collected from NOS^3 without having to deal with the real-time constraints of emulation. The augmented dataset should be a more robust representation of the potential threat space, which supports improved performance of the downstream machine learning task in the automated response tool described earlier.

5.2 GAMVT Parametrization

GAMVT's Preprocessing must be parametrized to allow for optimal identification of events in the data. For this use case, the identifiable events were the camera experiments, which involved a change of camera state and an increase in stored camera data. The following preprocessing steps were applied to the data:

1. Expected position and velocity was subtracted from actual position and velocity. Because the satellite orbit is preprogrammed in NOS^3, the expected position and velocity values over time are known. Due to the nature of the NOS^3 emulation, there is no noise in the emulated orbit, so subtracting expected values actually sets all position and velocity data to zero. In a more realistic emulation, remaining noise may need to be further addressed.

2. Rows 4 and 6 were transformed to reflect change over time. For Row 4, this transformation converts the raw memory usage data into the rate at which camera data was collected and stored, which better captures the relationship of interest between the camera on/off state and the memory usage. For Row 6, the transformation simplifies the timestamp data so that it is less likely to obscure the GAMVT Characterization process.

The Postprocessing stage of GAMVT only inverted these functions and did not apply any additional processing. Reversing the "rate of change" preprocessing transformation required an initial timestamp seed, for which the first timestep in one of the training samples was used.

5.3 Other Methods Tested

Two deep learning data generation methods were applied to this use case in addition to GAMVT. Both methods are commonly used for image- and text-based data, but they were modified for multivariate timeseries data to compare to GAMVT.

5.3.1 Generative Adversarial Network

A Generative Adversarial Network (GAN) is a deep learning method which leverages two "competing" submodels, a Generator and a Discriminator. The goal of the Discriminator is to differentiate between training data and data created by the Generator model. The goal of the Generator is to create samples so similar to the training data that the Discriminator's accuracy is no better than random guessing. This type of model has been shown to work particularly well with image data. The Generator and Discriminator can have a variety of underlying model structures, but the structures of the two submodels should mirror each other. In addition, the GAN can be developed to take some sort of additional condition as input, which allows for learning specific to each class of training samples. Previous work investigated different input formats, conditions, and submodel structures. In the end, a Multilayer Perceptron (MLP) -based Conditional GAN was the best-performing GAN model. See this paper's companion paper for further details (Sahakian et al., 2021).

5.3.2 Variational Autoencoder

A Variational Autoencoder (VAE), also a deep learning method, has been used especially with image-based data. A VAE can be split into two phases: and Encoder and a Decoder. The Encoder uses various deep learning layers to reduce the input training data to a smaller latent layer, which could be thought of as an "encoding" of the training data. Then the Decoder phase builds a new sample from this encoding and some random input. The structure of the Encoding layers and the Decoding layers should mirror each other. Also, like GAN's, VAE's can be formulated to be Conditional, allowing for specification of class during encoding and decoding. Previous work investigated the conditioning used in the model as well as the type of convolution applied in the model layers. A Conditional VAE with convolutions which acted only along single rows and "telescoped" in size, changing length as the size of the layer changed, was the best-performing VAE model. See this paper's companion paper for further details (Sahakian et al., 2021).

5.4 Metrics

Two metrics, Quality and Diversity, were developed to quantitatively evaluate the results of the three generation methods under test. Each metric was used to evaluate the goodness of the entire dataset generated by each method. The purpose and interpretation of these metrics will be described here. For further discussion and formulation, see (Sahakian et al., 2021).

Quality measures how similar the generated data is to original training data of the same class using a classification model trained on the original data. The ideal Quality score for this metric is 1.0. A Quality score lower than 1.0 would indicate that the generated data is likely dissimilar to the training data. A Quality score higher than 1.0, while rare, would indicate that the model performed better on the generated data than the training data. In this case, the choice of model used to evaluate Quality should be reconsidered.

Diversity measures how unique the generated samples are compared to training samples of the same class and compared to other generated samples. The ideal Diversity score is 1.0, which would indicate the same amount of data diversity in the training, generated, and combined datasets for a given class. A Diversity score lower than 1.0 would signal a generator that regenerates existing samples. A Diversity score higher than 1.0 could be indicative of noise in the generated samples or of generated samples that are not representative of their intended class.

Both metrics are required to fully evaluate the generated data. A generator that simply returns the training dataset would get a perfect Quality score, but an extremely low Diversity score. A generator that returns random values would likely score fairly high Diversity, but very low Quality. A balance must be struck between the Quality and Diversity metrics. Although both metrics have an ideal score of 1.0, some deviation from this score on either or both metrics does not necessarily indicate a poor generator. It would be better to have both scores be close to but not exactly 1.0 than to have a perfect score for one metric and a poor score for the other.

5.5 Discussion

The three methods under evaluation are GAMVT, GANs, and VAEs. See selected result images from Attack Class 3 in Figure 5. Both deep learning captured certain aspects of the system data well, but there were still some fundamental issues. Both models produced somewhat noisy results, particularly the VAE. In addition, the relationship between the camera on/off state (Row 3) and the camera memory usage (Row 4) was not well-captured by either model. In fact, the periods of time where the camera is on in the GAN-generated sample are short and sporadic, which is unrealistic given the underlying physical process.

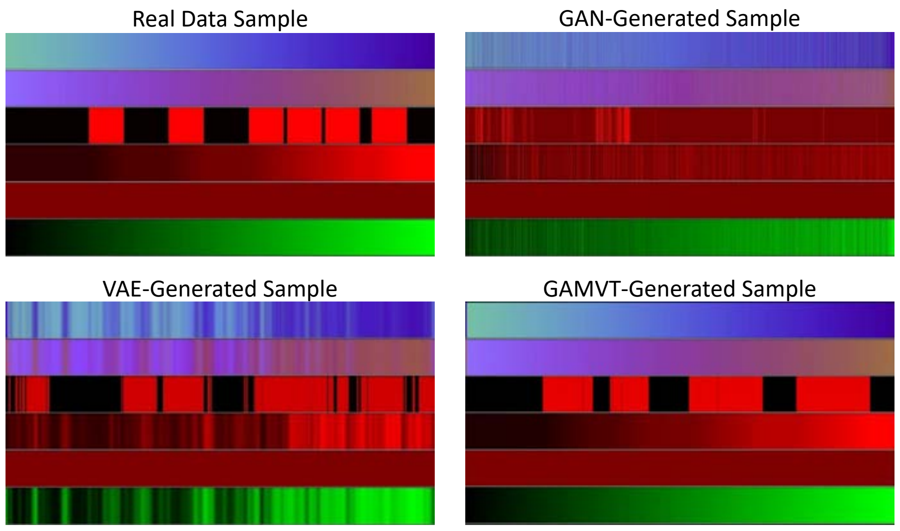

Figure 5: Selected Sample for Class 3 Using Each Generation Technique

Now consider the output of GAMVT, also shown in Figure 5. The gradients in Rows 1, 2, and 6 are very smooth, similar to the training data. There's a clear relationship between Rows 3 and 4, as desired. Further, Row 3 camera experiments are of realistic duration. Qualitatively, from these selected images for Class 3, GAMVT is generating more realistic data.

Now the generated datasets are evaluated as a whole. Figure 6 shows a selected sample generated for each class by each generation method. Table 2 shows the Quality and Diversity metrics calculated on the entirety of the datasets generated by each method.

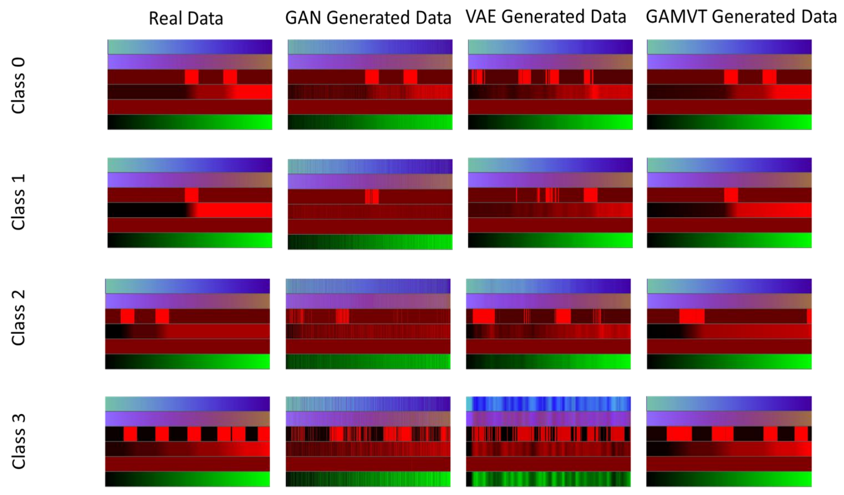

Figure 6: Selected Sample for Each Class and Each Generation Technique

Consider the metrics shown in Table 2. The GAN model generated the highest Quality score, but GAMVT's Quality score was very similar. The VAE model produced data with

	GAN	**VAE**	**GAMVT**
Quality	0.7875	0.4775	0.7513
Diversity	23.2840	24.1815	1.4388

Table 2: Quality and Diversity Score for Each Generated Dataset

a very low Quality score. However, GAMVT is the only method with a Diversity score close to 1.0. The GAN and VAE models both generate datasets with much higher Diversity scores, which is reflective of the noise in the generated samples from these models. Thus, GAMVT is determined to be quantitatively superior based on these metrics.

In addition to better qualitative and quantitative performance, GAMVT has other desirable qualities. Since GAMVT was developed for multivariate timeseries data, it is able to leverage the information embedded in the underlying structure of this type of data. In addition, GAMVT is not a deep learning method, and thus does not struggle with insufficient training data or model explainability as many deep learning methods do. Due to all of these factors, GAMVT was chosen as the best algorithm for this use case.

6. Future Work

First, a great test of GAMVT will be applying the GAMVT-augmented dataset to a downstream machine learning process. It is anticipated that the model trained on the augmented dataset will outperform the model trained on the small dataset generated through system emulation alone.

Much of the future development of GAMVT should be focused on supporting more complex datasets.

- Generator Distributions - Currently, the core of GAMVT's Generation stage is a random normal value generator, but this distribution may not be optimal for generating realistic output for all system sensors. It could be beneficial to explore generating samples using a variety of distributions.

- Efficient Constrained Timestep Generation - Section 3.2.3 described an inefficient "generate-then-test" method for generating timesteps for new samples which conformed to collected class characteristics. Developing a better way to approach generation would be preferable.

- Oscillatory Data - Currently, GAMVT relies on postprocessing to adequately handle and generate wave-like or highly oscillatory data. Although this is a reasonable approach, it is perhaps not ideal. There may be better ways to handle this type of data during the Characterization and Generation stages rather than leaving it all to postprocessing.

- Complex Patterns - GAMVT has never been tested on the case where multiple known events occur in a single sample, either independently or at the same time. If classes of samples are not easily separable, the collection of a unique pattern for each class of samples may become more difficult or require additional development.

- Sensor Relationships - The initial use case did not involve many different relationships between sensors. There is a physical relationship between position and velocity data, but this relationship wasn't leveraged in the initial use case. The only inter-sensor relationship used was the relationship between camera on/off state and the camera memory usage, and this relationship was relatively simple. It is necessary to test GAMVT with more complex or higher-order relationships between two or more sensors to see if these relationships can be adequately captured by GAMVT.

In addition, an established methodology for finding the best set of preprocessing and postprocessing functions for a given dataset would alleviate the burden on the analyst making these configurations. The use case dataset from Section 5.1 was simple enough that it was easy to make ad hoc decisions for the best set of functions and adjust as needed. This approach will likely be insufficient as datasets become larger and more complex, and it will be useful to have a general method for determining which processing functions will allow GAMVT to best capture and reproduce trends and relationships in the data.

7. Conclusion

Tools for intrusion detection and response on cyber systems are key to improving the resilience of these systems to an ever-evolving array of attacks. The robust development and testing of these tools requires threat datasets which often do not exist for cyber systems. Even if a small dataset does exist or can be generated, it must be augmented with some data generation technique. However, such techniques for augmenting small multivariate timeseries datasets have not been available historically. GAMVT helps to fill this gap. Both qualitatively and quantitatively, this work showed GAMVT outperforms modified versions of existing data generation techniques which have had great success generating other types of data. GAMVT is specifically designed to interpret the relationships embedded in multivariate timeseries data without requiring a large training dataset, which are major benefits over existing data generation methods.

Acknowledgement

This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2021-11188 C.

References

Chen, Y., Kempton, D. J., Ahmadzadeh, A., & Angryk, R. A. (2021). Towards synthetic multivariate time series generation for flare forecasting. *CoRR*, *abs/2105.07532*. https://arxiv.org/abs/2105.07532

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Geletko, D. M., Grubb, M. D., Lucas, J. P., Morris, J. R., Spolaor, M., Suder, M. D., Yokum, S. C., & Zemerick, S. A. (2019). Nasa operational simulator for small satellites (nos3): The stf-1 cubesat case study.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

Hallac, D., Vare, S., Boyd, S. P., & Leskovec, J. (2017). Toeplitz inverse covariance-based clustering of multivariate time series data. *CoRR*, *abs/1706.03161*. http://arxiv.org/abs/1706.03161

Kingma, D. P., & Welling, M. (2019). An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*.

Leznik, M., Michalsky, P., Willis, P., Schanzel, B., Östberg, P.-O., & Domanschka, J. (2021). Multivariate time series synthesis using generative adversarial networks. *ICPE '21: Proceedings of the ACM/SPEC International Conference on Performance Engineering*. https://dl.acm.org/doi/10.1145/3427921.3450257

Lloyd, S. P. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, *28*, 129–137.

Luo, Y., Cai, X., ZHANG, Y., Xu, J., & xiaojie, Y. (2018). Multivariate time series imputation with generative adversarial networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/96b9bff013acedfb1d140579e2fbeb63-Paper.pdf

Sahakian, M. G., Musuvathy, S., Thorpe, J., Verzi, S., Vugrin, E., & Dykstra, M. (2021). Threat data generation for space systems. *Proc. of 2021 IEEE Space Computing Conference*.

Sarkar, T. (2018). Synthetic data generation—a must-have skill for new data scientists. *Towards Data Science, Dec*, *19*.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*. https://doi.org/10.1186/s40537-019-0197-0