PC406: BTech Mini Project Report
Synthetic Scattering Cross-Section Data Generation using GANs

Project guide: Prof. Bhaskar Chaudhury

1st Vishwa Shah
201801036@daiict.ac.in

2nd Riddhi Tanna
201801427@daiict.ac.in

***Abstract*—This project aims to generate synthetic data that can be used to train ML models that are used for the inverse swarm-detection problem using two different GAN models, `TimeGAN` and `TGAN`. We cluster the data into four clusters and then generate synthetic data for each cluster separately. Post that, we filter out the noise and analyze the utility of the obtained data. We observe that the distribution of synthetic data is similar to that of the original data and the utility of the synthetic data obtained using both models is comparable and marginally better than previously used, geometric averaging.**

***Index Terms*—Generative Adversarial Networks, Deep Learning, Plasma Science, TGAN, TimeGAN, t-SNE, DTW**

I. Introduction

Electron-neutral scattering cross-sections are fundamental quantities in simulations of low-temperature plasmas, used for many technological applications today. Several macroscopic quantities (known as 'swarm parameters') can be calculated using these microscopic cross-sections. However, the inverse swarm problem of calculating cross-sections from swarm parameters is challenging because a unique solution does not exist. In 2021, researchers investigated the use of an artificial neural network (ANN), convolutional neural network (CNN) and densely connected convolutional network (DenseNet) to solve the inverse swarm problem [1]. However, training neural networks requires a large amount of data. Since the nature of this problem is such that only a limited amount of real data is available, it is necessary to produce synthetic data that can be fed as input to the neural networks. Researchers in [1] have used geometric averaging to create synthetic data. While their DenseNet model can solve the inverse swarm problem, the researchers believe that better accuracy may be achieved if the quality of synthetic data produced can be improved. Hence, we explore the use of Generative Adversarial Networks (GANs) to generate synthetic data for this problem. Previously, researchers have used geometric averaging to generate synthetic data for this [1]. We generate data using `TimeGAN` and `TGAN` and compare their utility with previously generated data using geometric averaging. For the analysis, we use t-SNE and Dynamic Time Warping.

II. Previous efforts

Multiple methods exist for the production of synthetic data. Some of them are statistical methods such as classification and regression trees [2] and geometric averaging [1]. These involve fitting a joint multivariate probability distribution on real-world data and drawing samples from the obtained distribution. Others are generative algorithms that apply neural networks such as variational autoencoders and GANs to generate synthetic data. Bayesian networks can also be used to generate synthetic data [3] [4].

All of the above-described methods work well for non-sequential data. However, cross-section data is sequential. For generating synthetic cross-section data, researchers previously clustered the data into three clusters manually and then generated synthetic physical cross-sections for their models by calculating the weighted geometric average of two actual cross-sections belonging to the same cluster [1]. We explore GANs that can effectively generate sequential synthetic data to improve the data quality. GANs are a class of deep learning models used to generate data by approximating the distribution of the original data. Ian Goodfellow first introduced them in 2014 [5]. A GAN consists of a generator, *G*, and a discriminator, *D*. The generator aims to generate fake samples to fool the discriminator, whereas the discriminator tries to discriminate between fake and real samples. The network gets trained when the discriminator cannot guess whether the sample is real or fake.

III. Our Approach

Since we are dealing with sequential data, it is advantageous to consider it as time-series data. Hence, we can also apply some methods used for time-series data for our data. We have used two different GAN models, `TimeGAN` and `TGAN` to produce synthetic data.

A. Time-Series Generative Adversarial Network (`TimeGAN`)

`TimeGAN` proposed by Yoon et al. [6] maintains the temporal dynamics while preserving the feature correlations, which is essential to produce sequential data. Unlike other GANs, the `TimeGAN` architecture introduces the concept of supervised loss, where the model captures temporal dynamics with the supervision of original data. The `TimeGAN` model is a framework of four networks, the Generator, the Discriminator and the recovery and embedding network [6]. The model's training is done in three steps: First, the embedding network is trained, followed by the training of the supervised network (Generator) and then finally, the joint model consisting of all four networks is trained.

B. Tabular Generative Adversarial Network (`TGAN`)

`TGAN`s are a class of GANs used for the generation of tabular data using recurrent neural networks [7]. The columns of a table can either contain numerical values or categorical

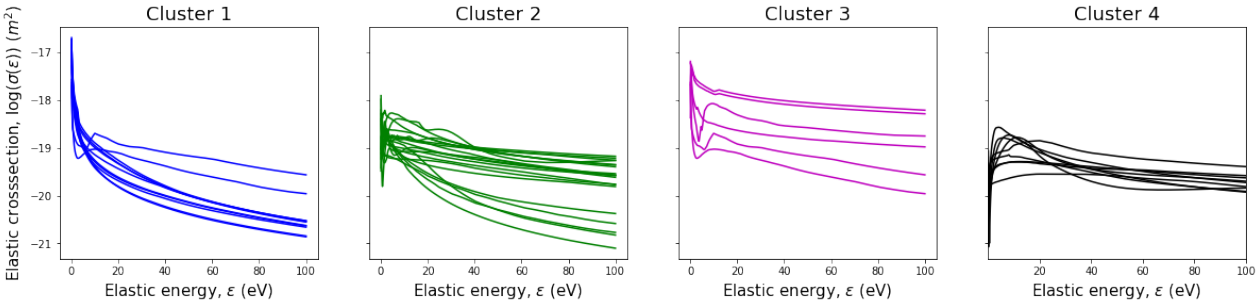


Figure 1: Clusters obtained after clustering the original data using *tslearn.clustering.kmeans*

values. Numerical values can be represented using continuous variables, and categorical values can be represented using discrete variables. `TGAN`s can generate discrete as well as continuous variables. A `TGAN` uses Long Short-Term Memory (LSTM) network in the generator and a Multi-Layer Perceptron (MLP) as the discriminator. The input to the `TGAN` is a table of data in which the columns can either be numerical (continuous) or categorical (discrete). The generator G is optimized by minimizing the Kullback-Leibler (KL) divergence. The discriminator is optimized using the conventional cross-entropy loss. In this way, it learns the marginal distribution of each column.

IV. Dataset

We obtain real data from the LXCat website. The dataset contains the previous determinations of elastic momentum transfer, ionization and excitation cross sections for 46 different gases. Our project concentrates on generating synthetic data for elastic momentum transfer cross-sections (MTCS). Time series data can be represented as a table with time in the first column and the corresponding values of a feature in the second column. In our data, we can consider the elastic energy as a proxy for time and the values of elastic MTCS as a feature. Since we have 46 gases, we would have 46 features. Our dataset is thus tabular and sequential.

V. Methodology

A. Data preprocessing and clustering

We first perform basic null checks and clean the data to obtain a table containing 100 columns and 46 rows of data for the 46 gases. Researchers in [1] cluster the data into three clusters manually. However, a more sophisticated clustering algorithm can be applied to obtain better clusters. We cluster our data into four distinct clusters using a library called *tslearn* [8]. *tslearn*'s *clustering.kmeans* module performs k-means clustering on time series by aligning two time series using Dynamic Time Warping (DTW) as a similarity metric since Euclidean distance does not accurately measure similarity between time series data [9]. Algorithm 1 explains the calculation of DTW path between two time series. *tslearn* uses the DTW Barycenter Averaging Algorithm (DBA). DBA minimizes the DTW distance between the barycenter (centroid) and the cluster members. The barycenters have an average shape similar to the shape of the cluster members. Hence, it ignores temporal shifts between the members of a particular cluster [9]. Fig. 1 shows the four clusters obtained using *tslearn*. After clustering the data, we preprocess the data for `TimeGAN` and `TGAN` separately.

1) `TimeGAN`: The input to our model is a 3D dataset of shape $(c-s+1) \times s \times n$, where c is the number of gases in the cluster, s is the sequence length and n is the number of features. Our initial tabular data has dimensions $t \times n$, where t is the number of timesteps. We reshape the data such that each of the $c-s+1$ elements contains features for a window of size s. Hence, the windows would contain features for 0 to $s-1$, 1 to s, ..., $c-2s+1$ to $c-s+1$. Fig. 2 is representative of the same transformation.
However, `TimeGAN` produces new time steps while keeping the features constant. For our purpose, we need to keep the time steps constant, but we need to produce sequences for different gases. Hence, we consider time steps as features instead of the gases. As `TimeGAN` maintains both temporal and feature correlations, this transformation is justified. The data generated through this model is also a 3D dataset of shape $N \times s \times n$, where N is the sample size. As the model requires the input 3D data to have at least 5 timestep tables for learning, the sequence length should be provided accordingly based on the number of entries in the cluster. Cluster 3 and cluster 4 have only 6 and 9 sequences respectively, which are small datasets for this model. Hence, we merge the third and the fourth cluster. The results obtained are discussed in Section VI.

2) `TGAN`: As `TGAN`s are not made explicitly for sequential data, we can consider each time step as a continuous column with a specific distribution. This way, `TGAN` can learn the distribution of each column [7]. In our case, we have 100 time steps, and hence, we can feed 100 continuous columns as input to the `TGAN`. This table is given as the input to train the model. Tabular data of shape 10000×100 is generated as the model's output for 10,000 samples.

B. Using different GAN models

We use `TimeGAN` and `TGAN` to generate synthetic elastic MTCS data for each of four clusters. Fig. 3 shows the results obtained for cluster 1. While `TimeGAN` and `TGAN` have been able to learn the trends and the range of the data well, there

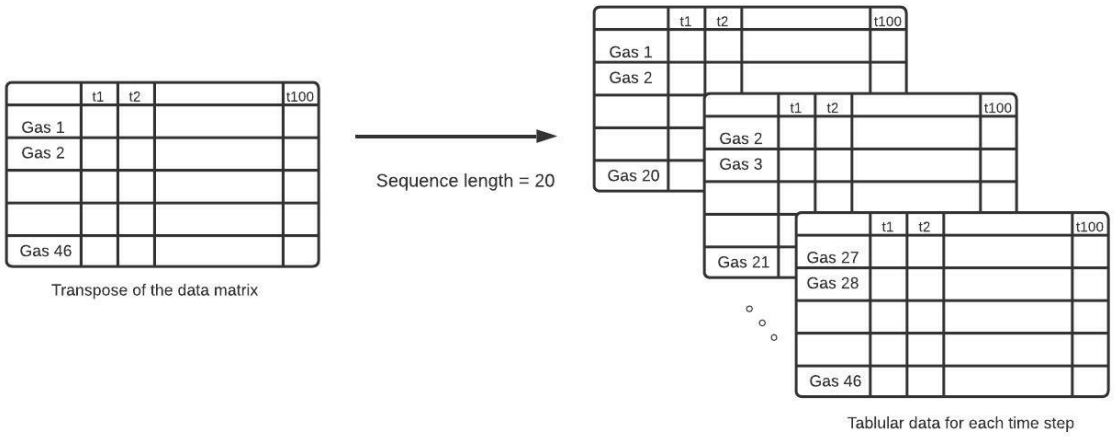


Figure 2: Input 3D data to the `TimeGAN` model, time steps considered as features

appears to be a lot of noise. Hence, we use a filter to keep the required frequencies and smoothen the trend.

C. Filtering data

1) `TimeGAN`: The smoothing of data obtained from `TimeGAN` is done with the help of the Savitzky-Golay filter (`savgol` filter) [10]. We use the `scipy` implementation of the `savgol` filter.

2) `TGAN`: A low pass filter such as a Butterworth filter can be used to filter out high-frequency components from the sequential data obtained from `TGAN`.

The results obtained before and after filtering the data for both the models are discussed in Section VI.

D. Data Utility Analysis

The synthetic data generated is used to train different models, and it is crucial to ensure that the distributions of the original and synthetic datasets are similar. We perform general and specific utility tests for our generated datasets to answer how close our data is to the real data. A specific utility check is used to compare the synthetic data generated from two of the models used here. We use two methods for our analysis: t-SNE and Dynamic Time Warping.

1) t-Distributed Stochastic Neighbor Embedding (t-SNE): t-SNE [13] is an unsupervised non-linear technique used to visualize high dimensional data. It reduces the dimensions of the data while preserving the clustering of the data in higher dimensions. The algorithm finds similarity between instances in the higher dimensional space using a Gaussian distribution and the lower dimensional space using a t-distribution. Later, the gradient descent approach minimizes the Kullback-Liebler (KL) cost function (distance between higher and lower dimension probabilities) to get the optimized values. Section VI-C1 compares t-SNE plots for data obtained using both models.

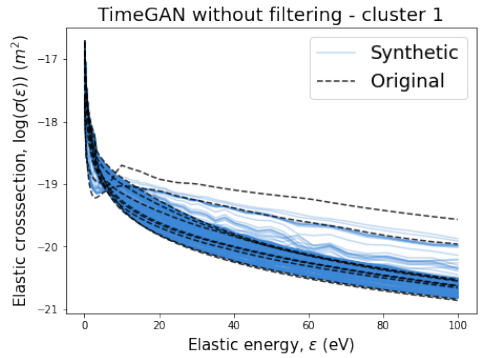
2) Dynamic Time Warping: Dynamic Time Warping (DTW) is an algorithm used to measure similarity between two temporal sequences varying in speed. Algorithm 1 describes the algorithm used to calculate the dtw cost between two time series. The dtw cost is the minimum cumulative distance between two time series.

Algorithm 1: Dynamic Time Warping

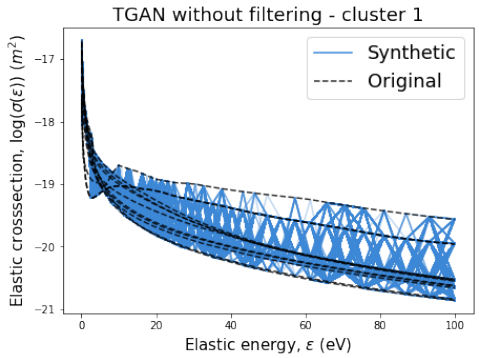
input : Time series $x_{1,N}$ and $y_{1,M}$
output: DTW cost matrix $D_{N+1,M+1}$

Cost matrix: $D_{N+1,M+1}$
for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** M **do**
    $D_{i,j} = \infty$
$D_{0,0} = 0$
for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** M **do**
    $cost \leftarrow d(x[i], y[j])$
    $D[i,j] \leftarrow cost + min(D[i-1,j], D[i,j-1], D[i-1,j-1])$
$dtw \leftarrow D[N,M]$
return D, dtw

We use the `dtw-python` module [11] [12] to implement this algorithm for our data. We randomly take one synthetic and one real sample from each cluster and compute the dtw distance between them. After obtaining dtw, we can normalize it to obtain normalized distance, $dtw_{norm} = \frac{dtw}{N+M-1}$. Here, N is the length of the first sequence, and M is the length of the second sequence. In our case, $N = M = 100$. The normalized metric gives us the average distance each point in the sequence would have to move to align the two sequences perfectly. Hence, the smaller the distance, the better is the alignment. The results obtained are discussed in Section VI-C2.

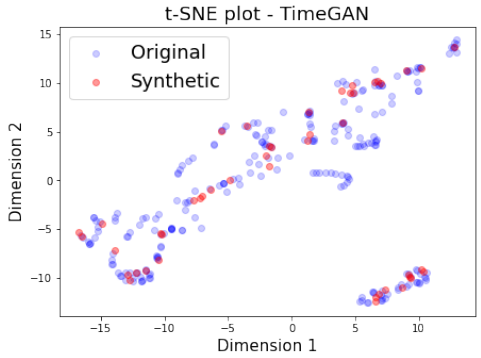


(a) Synthetic samples generated by TimeGAN without filtering

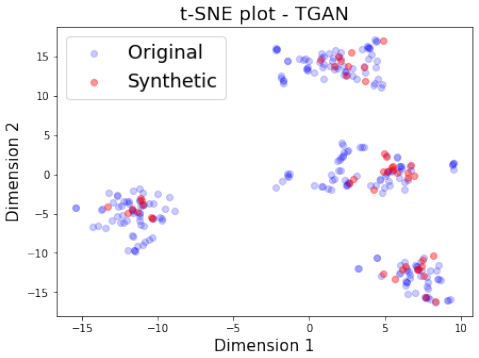


(b) Synthetic samples generated by TGAN without filtering

Figure 3: Samples generated for cluster 1 using TimeGAN and TGAN - without filtering, it is evident that there is noise in the data although the models have learnt the patterns well



(a) TimeGAN: 246 random synthetic samples and 46 real samples considered



(b) TGAN: 246 random synthetic samples and 46 real samples considered

Figure 4: t-SNE results for TimeGAN and TGAN model considering time sequences as the features

VI. Results

A. TimeGAN

Fig. 3a shows some synthetic samples obtained by using TimeGAN. The model can learn the pattern of the different series in the clusters. Fig. 5 shows the synthetic samples generated using the TimeGAN after filtering. Fig. 5d compares the original and synthetic data generated for all four clusters. The synthetic data significantly improves after filtering using the Savitzky-Golay filter (savgol filter) [10].

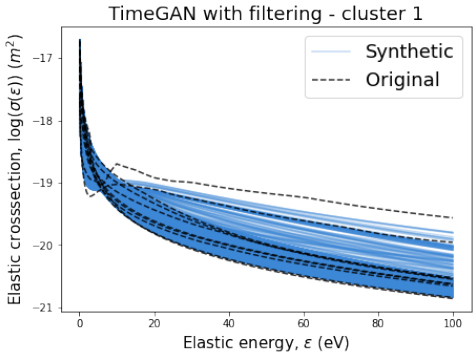
B. TGAN

Fig. 3b shows 10,000 synthetic samples obtained by using TGAN. After filtering all four clusters, the data significantly improves since we remove noise. This can be seen in Fig. 6.

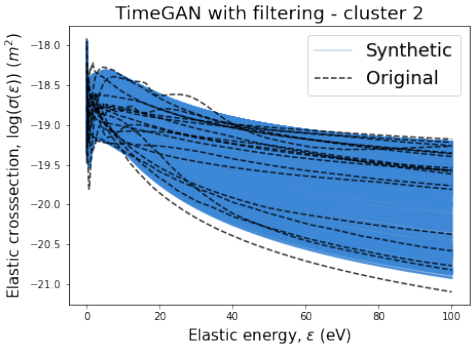
C. Data Utility Analysis

1) t-Distributed Stochastic Neighbor Embedding (t-SNE): 246 synthetic samples were appended with the original dataset for both TimeGAN and TGAN models. t-SNE was performed with the help *sklearn's TSNE module* on the concatenated dataset. Fig. 4 shows the t-SNE results of original and synthetic data of both models. The plots tell us that they follow similar distributions.

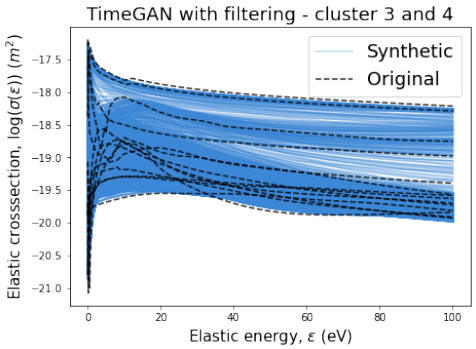
2) Dynamic Time Warping: We consider one random real and one random synthetic sample from each cluster for both TimeGAN and TGAN. We then run 20,000 Monte-Carlo simulations and obtain the average value of dtw_{norm}. We compare the dtw_{norm} of TGAN and TimeGAN for cluster 1 and cluster 2. For cluster 3 and 4, TGAN and TimeGAN cannot be compared because they were trained using different training sets, as mentioned in Section V-A1. Table I shows the comparison of dtw_{norm} for both the models. For cluster 1 and cluster 2, TimeGAN and TGAN have similar scores. However, for clusters 3 and 4, the scores of TGAN are better than that of TimeGAN. This may be because we merged clusters 3 and 4, so there are more patterns to learn. Table II compares the results of the generative models with the previously used geometric averaging method of synthetic cross-section data generation for elastic MTCS [1]. dtw_{norm} for TimeGAN and TGAN are marginally less than that of geometric averaging.



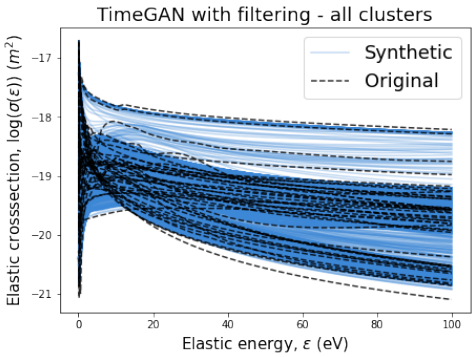
(a) Synthetic data generated using `TimeGAN` for cluster 1



(b) Synthetic data generated using `TimeGAN` for cluster 2

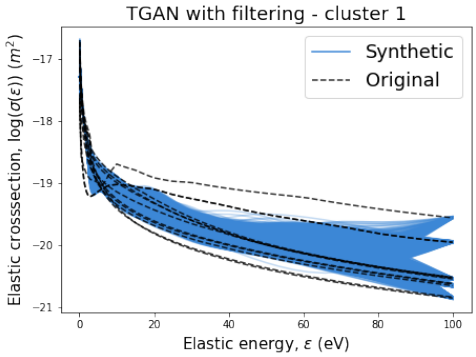


(c) Synthetic data generated from `TimeGAN` for cluster 3 and 4

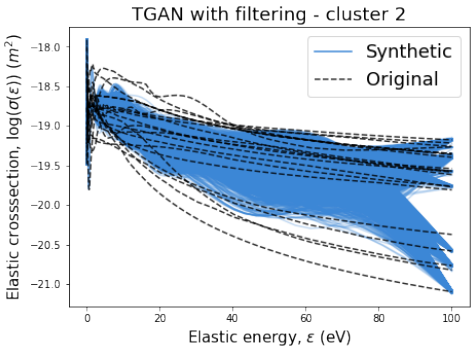


(d) Synthetic data generated using `TimeGAN`: combined for all clusters

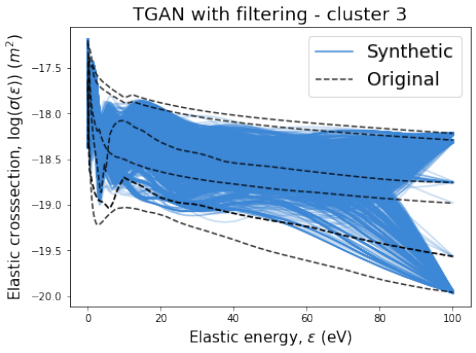
Figure 5: 10,000 samples generated for each cluster using `TimeGAN`, smoothed using a `savgol` filter



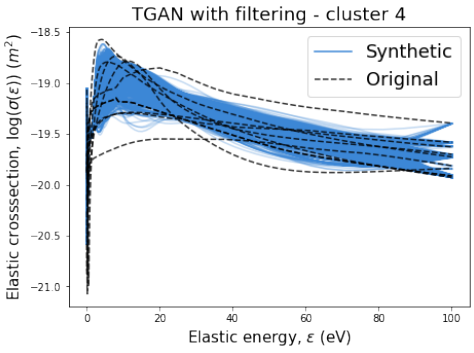
(a) Synthetic data generated using `TGAN` for cluster 1



(b) Synthetic data generated using `TGAN` for cluster 2



(c) Synthetic data generated from `TGAN` for cluster 3



(d) Synthetic data generated using `TGAN` for cluster 4

Figure 6: 10,000 samples generated for each cluster using `TGAN`, filtered using a Butterworth filter

Cluster	dtw_{norm} (TimeGAN)	dtw_{norm} (TGAN)
1	**0.040**	0.047
2	0.132	**0.098**
3/(3 + 4)	**0.140**	0.640
4/(3 + 4)	**0.084**	0.640

Table I: Comparison of Normalized DTW cost for a random real sample and a random synthetic - average over 50,000 iterations

Method	dtw_{norm}
Geometric Averaging	0.46
TimeGAN	**0.43**
TGAN	0.45

Table II: Comparison of Normalized DTW cost for a random real sample and a random synthetic - average over 50,000 iterations for three different methods after merging all clusters

VII. Conclusion

We have used two generative models to obtain synthetic cross section data for elastic momentum transfer. This data can be used to train neural network models built to solve the inverse swarm problem [1]. Since, our data is tabular and sequential, it is highly beneficial to consider it as time-series and apply methods used for time-series analysis on our data as well. We compare the performance of two GAN models: TimeGAN and TGAN. TimeGAN is primarily used to generate synthetic time-series data and we are successfully able to manipulate our data to fit this model since it preserves temporal and feature correlations. TGAN on the other hand, is primarily suitable for tabular data. In this case, as well, we are able to manipulate our data to fit the model by considering the time steps as continuous columns. The synthetic data generated using both models are able to learn the distribution of the original dataset even though we have relatively less data to train generative models. The data generated by TimeGAN and TGAN have similar utility measures. We compare the data obtained with that obtained by geometric averaging in [1]. We see that TimeGAN and TGAN can generate marginally better synthetic data. This comparison is made on the basis of dtw_{norm}, which is the normalized minimum cumulative distance between two time-series data (sequential data, in our case).

Further, these methods can be applied to generate synthetic ionization and excitation cross section data. All of this data combined can then be fed as input to DenseNet to see if it affects the accuracy of the model.

References

[1] Vishrut Jetly, Bhaskar Chaudhury, "Extracting electron scattering cross-sections from swarm data using deep neural networks", *IOPscience journal*, June 2021.

[2] Jerome P Reiter. "Using cart to generate partially synthetic public use microdata", *Journal of Official Statistics*, 21(3):441, 2005.

[3] Isabel Valera, Melanie F Pradier, Maria Lomeli, and Zoubin Ghahramani. "General latent feature models for heterogeneous datasets." *arXiv:1706.03779*, 2017

[4] Dhamanpreet Kaur *et al*. "Application of Bayesian networks to generate synthetic health data". *Journal of the American Medical Informatics Association, Volume 28, Issue 4*, April 2021

[5] Ian J. Goodfellow et al., "Generative Adversarial Nets", June 2014.

[6] Yoon, Jinsung Jarrett, Daniel Schaar, Mihaela. "Time-series Generative Adversarial Networks" (2019).

[7] Lei Xu, Kalyan Veeramachaneni, "Synthesizing Tabular Data using Generative Adversarial Networks", 2018

[8] Romain Tavenard et al., "Tslearn, A Machine Learning Toolkit for Time Series Data", *Journal of Machine Learning Research, Volume 21, Number 118*, 2020

[9] Francois Petitjean, Alain Ketterlin, Pierre Ganc-arski, "A global averaging method for dynamic time warping, with applications to clustering", *Pattern Recognition Volume 44, Issue 3*, March 2011

[10] Abraham. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures." *Analytical Chemistry* 1964 36 (8), 1627-1639 DOI: 10.1021/ac60214a047

[11] Giorgino. "Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package. J. Stat. Soft., 31" (2009), doi:10.18637/jss.v031.i07

[12] Tormene, T. Giorgino, S. Quaglini, M. Stefanelli (2008). "Matching Incomplete Time Series with Dynamic Time Warping: An Algorithm and an Application to Post-Stroke Rehabilitation." *Artificial Intelligence in Medicine, 45(1), 11-34.* doi:10.1016/j.artmed.2008.11.007

[13] L.J.P. van der Maaten and G.E. Hinton. "Visualizing High-Dimensional Data Using t-SNE". Journal of Machine Learning Research 9(Nov):2579-2605, 2008.