Enhancing the Quality of Simulated Networking Data using Generative Adversarial Networks

Harshavardhan Battula[1] , **Aman Gaurav**[2] , **Ajit Patankar**[3]
[1,2,3]CTO and Pathfinding Team, Juniper Networks Inc.
[hbattula, agaurav, apatankar]@juniper.net

Abstract

Correctly estimating the network traffic data to build robust networking devices is crucial. Expected traffic in the network can be modeled as counting the number of occurrences of an event in a unit of time, and mathematical methods like the Poisson process can solve this problem. These processes are constrained due to the limited set of parameters used for modeling. In recent years, machine learning techniques have been extensively used for network traffic prediction due to their ability to learn complex patterns from historical data to predict a good forecast. However, the performance of these models trained for these tasks depends on the size and quality of the dataset. In the case of 5G network devices, collecting the data is expensive, and we are often left with insufficient and sensitive data. Therefore, we need a solution that correctly solves the problems of data augmentation and network traffic prediction. Generative Adversarial Networks(GANs) have seen exponential growth in recent years and are primarily being used for style transfer and data augmentation. This paper focuses on a GAN-based approach for creating synthetic data for dataset augmentation, and improving the quality of mathematically simulated 5G data using Poisson process for network traffic prediction. We also discuss how synthetic and simulated data can be used for different downstream applications.

1 Introduction

AI/ML models have become an integral part of the computer network and cloud management systems. For example, traffic throughput by switches displays seasonality characteristics related to demand changes at different times during the day. Traditional rule based or statistical techniques are powerful enough to model such multi-variate temporal behavior. In this case, we can rely on machine learning models to capture complex patterns and assist with monitoring and control functionality.

However, the advent of AI/ML models is predicated on the availability and quality of data. This problem is particularly exacerbated in network and cloud systems due to factors such as dynamic configuration, tail events, and frequent introduction of new technologies. We consider this problem of more importance than developing a model that meets specific metrics. Thus, this paper focuses on the data acquisition problem, and in particular, we espouse the use of Generative Adversarial Networks (GANs) to augment telemetry data collected from field deployments.

The contributions of this work are as follows:

- Modelling a simulated rule-based dataset that matches the expected distribution using the Poisson process.
- Train a robust Generative Adversarial Network on limited real data to capture the underlying pattern in the time series. Thus, providing a generator which is capable of generating synthetic dataset with high fidelity.
- Generating an interpolated dataset which suits different downstream applications.

2 Related Works

Auto regressive models were used for time series generation due to their stability, quick inference to the new predictions as there is no truncated back propagation through time. By taking the input data from the previous time steps, they can predict the current/next time step [2]. Oord et al. [13] used CNNs and Autoregressive RNNs to generate pixels sequentially for images. Autoregressive models are quite expensive as they are complex, and the time elapsed to generate for one time step is quite high.

Generative Adversarial Networks became popular really quickly since their introduction by Goodfellow et al. [6] but there is relatively less work in literature on time series generation compared to image generation. Some of the early works were by Mogren et al. [10], they proposed C-RNN-GAN and used RNN based architecture to generate music data. Esteban. C, 2017 [3] used similar RNN based architecture with conditional inputs to generate synthetic time series data for sensitive medical information, in particular, they used Recurrent Conditional Generative Adversarial Network (RCGAN),

to capture the distributions of attributes like oxygen saturation, heart rate monitoring, etc. Gaurav et al. [4][9][5] have used similar dense neural networks for times series data modelling on PPG and sensor data. COT-GAN proposed by Xu et al [15] has a modified loss function formulated using ideas from Causal Optimal Transport (COT). TimeGAN proposed by Yoon et al. [16], attempted to solve time series generation by projecting the sequences into latent space using an auto encoder, the encoder and generator are trained using an extra supervised loss term for class conditionals. DoppleGANger proposed by Lin et al. [8] generates meta data along with the associated time series records with an auxilary Generator and Discriminator. Smith et al.[11], proposed Conditional GAN for time series generation where WGANs were used similar to TimeGAN based architecture, in which the first network is trained to map the latent dimension to spectrogram images, and the second network maps the spectrograms to time series. Smith et al.[12] proposed a WGAN based architecture on a 1-dimensional time series data, this method is scaled version of WGAN from 3D/2D data to 1D.

3 Procedure

3.1 Generative Adversarial Networks

A standard Generative Adversarial Network (GAN) for time series will have two components, the generator (G), and a discriminator (D). The objective of the generator is to generate a window of synthetic samples, $\tilde{x} \in \mathbb{R}^{w \times d}$ using a random latent vector $z \in \mathbb{R}^z$ as an input. The objective of the discriminator is distinguish the synthetic series, $\tilde{x} = G_\theta(z) \sim p_G(z)$ from real series, $x \sim p_{data}(x)$. The generator, G_θ is then trained along with the discriminator (D) until they reach an equilibrium. The parameters, (θ) of the generator are updated using the gradient of the output of the discriminator. A regular GAN is prone to problems like vanshing gradient, and mode collapse due to the stability of discriminator during the training process. To address this problem, Wasserstein GANs were introduced by Arjovsky et al.[1] WGANs replace standard binary cross entropy objective function with earth mover's distance.

An improved version of WGAN (WGAN-GP) is introduced by Gulrajani et al. [7] which adds a gradient penalty term as a regularizer to improve the stability of the discriminator. A simple flowchart of WGAN-GP is shown in Fig 1. The objective functions of the Discriminator and Generator are as follows. The equation 1 represents the loss of the generator which is the expected value of predictions from the discriminator on the synthetic samples. The equation 2 represents the loss of the discriminator, and it is Wasserstein distance between the expected values of discriminator from synthetic samples and real samples along with the gradient penalty term. The equation 3 represents the computation of gradient penalty term maintaining the 1-L continuity by constraining the maximum norm of the gradient to 1.

$$\mathbb{L}_G = -\mathbb{E}_{\tilde{x} \sim p_g}[D(\tilde{x})] \tag{1}$$

$$\mathbb{L}_{\mathbb{D}} = \mathbb{E}_{\tilde{x} \sim p_g}[D(\tilde{x})] - \mathbb{E}_{x \sim p_d}[D(x)] + \text{GP} \tag{2}$$

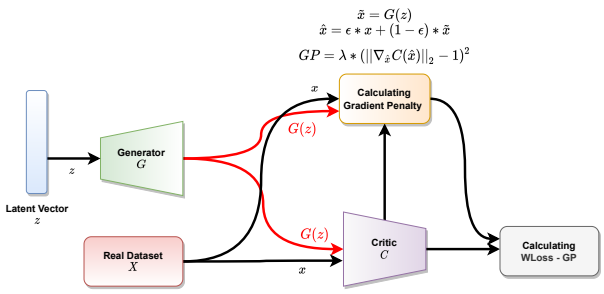

Figure 1: Flowchart of WGAN-GP(when computing W-Loss with gradient penalty).

$$\text{GP} = \lambda \mathbb{E}_{\tilde{x} \sim p_g}[(||\nabla D(\alpha x + (1 - \alpha \tilde{x}))||_2 - 1)^2] \tag{3}$$

The gradient penalty is calculated on the interpolated series, which is a weighted combination of real series, x and synthetic series $\tilde{x}$.

3.2 Synthetic Dataset with WGAN-GP

For the time series generation, we used RCGAN based architecture using Wasserstein Loss and Gradient Penalty (WRGAN-GP). Using an auxillary discriminator or encoder based architectures as proposed in [16], [8], [11] is computationally expensive. Our choice of using Wasserstein Loss with Gradient Penalty as loss function is to avoid problems like mode collapse. We noticed that the fidelity, diversity and usefulness of the generated data using WRGAN-GP is comparable to the approaches that uses auxiliary models.

We start off with a real world 5G network traffic data generated from a lab deployed 5G router on premise. We used the pre-processing and post processing methods suggested by Wiese et al. [14]. A standard LSTM cell controls the flow of the data using gates, namely the input gate(i), an output gate(o) and a forget gate(f). Both the generator (G), and discriminator, (D) are LSTM-based to process the sequences. The generator returns a window of synthetic sequence, (f) given a random noise vector, (z) and the hidden states, (h, c) as an input. At time (t), define the series from the real data $x(t) \in \mathbb{R}^{w \times d}$ based on a sliding window of size (w) over the historical time series data for training. The LSTM cell inside the generator uses latent vector, (z) as the sequence along with (h, c) to predict the series window (f_t). The size of z is a hyper parameter, as the value increases, we will be leading towards more deterministic mapping.

$$\tilde{y} = \text{LSTM}(z, (h, c)) \tag{4}$$

$$f_t = ReLU(\tilde{y}) \tag{5}$$

The discriminator takes real, x or synthetic, $\tilde{x}$ sequences as an input to make a distinctions. We then trained the network as shown in Figure 1 for 3000 steps using mini-batch gradient descent. In each epoch, the discriminator is trained

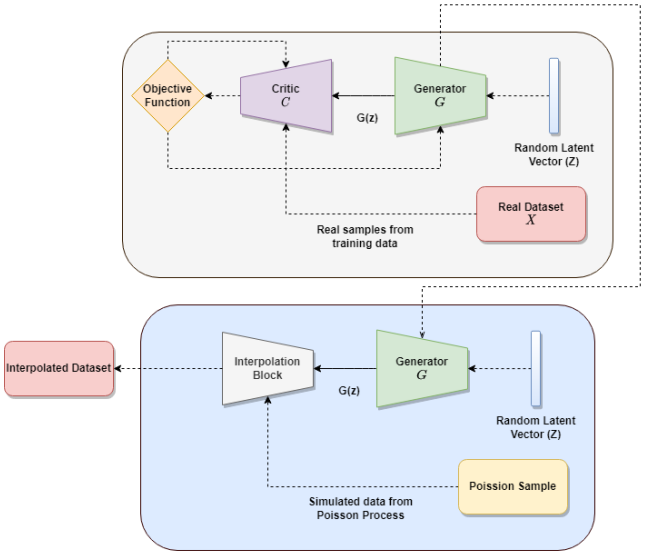

Figure 2: Flowchart for generating interpolated dataset, $I(\lambda, z)$ using synthetic , $(\tilde{X})$ and simulated data, (X^p).

5 times, and the generator is trained once. As per equations (4 & 5), $\tilde{y}$ is the output of the LSTM cell with latent vector, z as an input. From Fig 1, the generator's update is solely dependent on feedback from the discriminator, and the discriminator's learning is a crucial. The generator's loss, $(\mathbb{L}_G)$ as per the equation 1 is the expected value of synthetic series being real, and the discriminator's loss, $(\mathbb{L}_D)$ is wasserstein distance between X, and $\tilde{X}$ with a gradient penalty term as per equation 2.

3.3 Simulated Dataset with Poission Process

Our simulated dataset is application based generated from a poisson like mathematical process, and is a network traffic trace data at 1 *(sample/sec)* frequency. Network traffic throughput by switches displays seasonality characteristics that is related to demand changes at different times during the day. The synthetic data is generated from a poisson process defining packet arrivals per second, where the arrival rate of packets during the daytime (8 am to 10 pm) is defined by poisson process variable $\lambda_{day} = 10$. To accommodate lowered network traffic at night as generally observed historically, the Poisson variable(λ) is multiplied by a deactivation rate factor (ω_D). As a result, for the nighttime (10 pm till 8 am) network traffic is generated through a Poisson process defined by $\lambda_{night} = 2$. The resultant simulated data, (X^p) generated is superposition of two poisson distributions. Fig 3 shows simulated data for a single attribute.

3.4 Refining the Quality of Simulated Dataset

The synthetic dataset is post processed and the sequences are reconstructed from the synthetic sequences using the methods suggested by Wiese et al. [14]. The synthetic dataset, $(\tilde{X})$ can be used along with the simulated dataset, (X^p) to

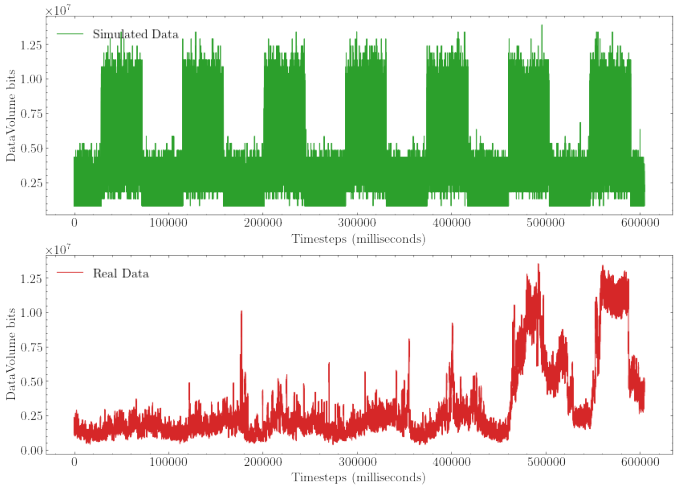

Figure 3: Top to bottom, simulated data set using poission process in green, and real data set in red.

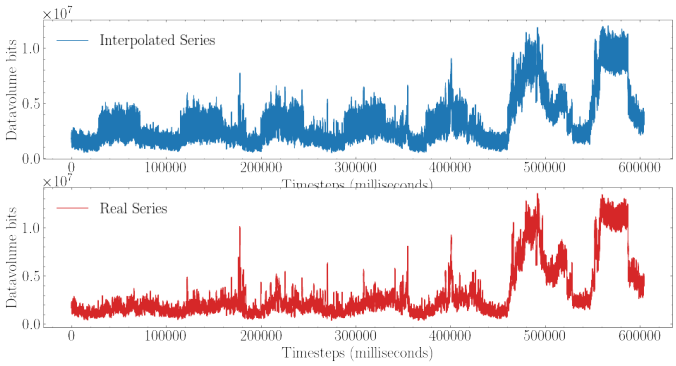

Figure 4: Interpolated Dataset by merging poission and synthetic data sets.

generate a dataset to generate an interpolated dataset, $I(\lambda, z)$ which suits the need for approximating the network traffic on routers. This interpolated dataset, $I(\lambda, z)$ as shown in the Fig 4 is a weighted sum of simulated and synthetic datasets. The poisson process can be used to model the user activity, by changing the variable λ. The generator can model the time series by learning the underlying pattern using a random latent vector. The hyperparameter β controls the weights of simulated and synthetic datasets. A typical interpolated dataset which resembles the real dataset relatively will have high value of β. Equation 6 below shows the weighted combination of synthetic, $\tilde{X}$ and simulated datasets, X^p.

$$I(\lambda, z) = \beta X^p + (1 - \beta)\tilde{X} \tag{6}$$

4 Observations

4.1 Quality of Synthetic Dataset

We tested the performance of the RCGAN, WRGAN, TimeGAN using all the evaluation metrics mentioned above using the publicly available **Stocks** dataset. The dataset is multi-variate, so it is very good for benchmarking. The

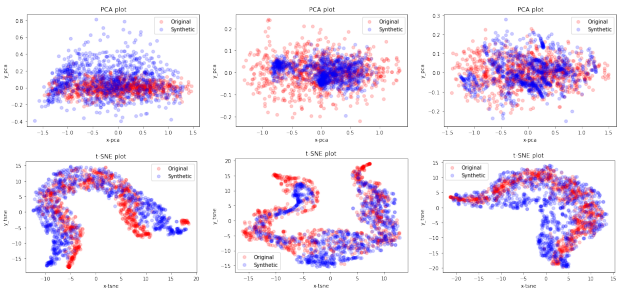

Figure 5: From left to right, we have the PCA plots of the real and synthetic datasets using RCGAN, TimeGAN and WRGAN-GP on top. t-SNE plots of the real and synthetic datasets of the same models in the bottom.

historical Google stocks data from 2004 to 2019, includes features such as volume, high, low, opening, closing, and adjusted closing price for the stock.

We used Adam optimizer with default values of learning rate (α = 0.001), (β_1, β_2) = (0.9, 0.999). In order to reduce gains through network changes, we used GRU cells in all experiments and trained the models for same steps.

Suppose for synthetic dataset ($\tilde{X}$), say x^t corresponds to the value of the feature *traffic_device: x*. A significant deviation in the value of x^t at timestep(t) should be treated as an anomaly. For the time series generation problem, the range of values a synthetic attribute can take, and the trend of the attribute should be similar to real time series. We used the same metrics used by Yoon et al. [16] for evaluation. PCA and t-SNE plots for visual inspection of diversity, Discriminative Score for Fidelity and Predictive Score for Usefulness. Fidelity is measured quantitatively using the Discriminative score.

Table 1 shows the Discriminative and Predictive Scores of different models with **stocks** dataset. If a classifier cannot distinguish the synthetic data from the real data, it means the quality of generated data is good and has high fidelity. The lower the discriminative score, the better. In order to be useful, the sampled data should inherit the predictive characteristics of the original. To measure usefulness, a post-hoc sequence-prediction model in a Train on Synthetic Test on Real(TSTR) setting is trained with synthetic data to predict next-step temporal vectors over each input sequence. Then, we evaluate the trained model on the real dataset. Performance is measured in terms of the mean absolute error (MAE). From table 1, WRGAN-GP gives better predictive and discriminative score than RGAN but falls behind when compared to TimeGAN.

4.2 Usage of Synthetic Dataset

A synthetic time series dataset has the underlying pattern and the trend of the actual dataset. This property of GANs to generate synthetic samples with such high fidelity can be used to add some anonymity to the feature in the dataset, which means a sensitive feature can be protected. Given that the generator(G_θ) captures the underlying distribution of the real dataset, we can combine it with different combinations

Table 1: Results of different GAN architectures on **stocks** dataset

Scores		
Metric	Method	Stocks
Discriminative Score (Lower the Better)	RGAN	0.196 ± 0.21
	WRGAN-GP	$\mathbf{0.13 \pm 0.01}$
	TimeGAN	0.102 ± 0.021
Predictive Score (Lower the Better)	RGAN	0.056 ± 0.013
	WRGAN-GP	0.038 ± 0.005
	TimeGAN	0.038 ± 0.001

of simulated data to simulate different datasets for various use-cases with- out compromising the privacy of sensitive attributes. As the interpolated dataset, $I(\lambda, z)$ is the weighted combination of the simulated dataset, and synthetic dataset, we can use it for training robust anomaly detection models as per the change in user behavior by controlling the parameter (λ). Meta learning can be applied by performing initial training on the larger synthetic or interpolated dataset, and then fine-tuning can be done with the real dataset for downstream applications. If an attribute has few missing values or the feature is sparse, we can use the interpolated dataset to impute the attribute values.

5 Conclusion

We have described, trained, and evaluated a WRGAN-GP on a 5G network traffic time series dataset. We then showed how a synthetic dataset is identical to the real dataset, and it can be used to preserve the privacy of sensitive attributes without destroying the underlying time series. This synthetic dataset is then combined with the simulated dataset to create an interpolated dataset. Finally, we argued that this interpolated dataset is a more robust dataset that can be fed into training the machine learning forecasting and anomaly detection models for future self-correcting network routers and switches in networking applications. The ML models can be trained on this interpolated data for data traffic routing algorithms on devices as well as direct deployment of ML models on SOCs. These meta models trained on such datasets followed by fine-tuning with real datasets, would be robust enough for deployment on future 5G devices.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[2] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neu-*

ral Information Processing Systems, volume 28. Curran Associates, Inc., 2015.

[3] C. Esteban, S. L. Hyland, and G. Rätsch. Real-valued (medical) time series generation with recurrent conditional gans, 2017.

[4] A. Gaurav, M. Maheedhar, V. N. Tiwari, and R. Narayanan. Cuff-less ppg based continuous blood pressure monitoring—a smartphone based approach. In *2016 38th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pages 607–610. IEEE, 2016.

[5] A. Gaurav, M. Maheedhar, V. N. Tiwari, and R. Narayanan. Method and electronic device for cuff-less blood pressure (bp) measurement, July 7 2020. US Patent 10,702,169.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[7] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans, 2017.

[8] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar. Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 464–483, New York, NY, USA, 2020. Association for Computing Machinery.

[9] M. Maheedhar, A. Gaurav, V. Jilla, V. N. Tiwari, and R. Narayanan. Stayfit: A wearable application for gym based power training. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6290–6293. IEEE, 2016.

[10] O. Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training, 2016.

[11] K. E. Smith and A. O. Smith. Conditional gan for time-series generation, 2020.

[12] S. A. Smith, K.E. Time series generation using a one dimensional wasserstein gan. spain: Universdad de granada., 2019.

[13] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *CoRR*, abs/1601.06759, 2016.

[14] M. Wiese, R. Knobloch, R. Korn, and P. Kretschmer. Quant GANs: deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440, apr 2020.

[15] T. Xu, L. K. Wenliang, M. Munn, and B. Acciaio. Cot-gan: Generating sequential data via causal optimal transport. *Advances in Neural Information Processing Systems*, 33:8798–8809, 2020.

[16] J. Yoon, D. Jarrett, and M. van der Schaar. Time-series generative adversarial networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.