# Gen-TWIN: Generative-AI-Enabled Digital Twin for Open Radio Access Networks

Osman Tugay Basaran*, Davide Villa†, Pedram Johari†, Michele Polese†,
Claudio Fiandrino‡, Falko Dressler*, Tommaso Melodia†
*School of Electrical Engineering and Computer Science, TU Berlin, Germany
†Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, U.S.A
‡IMDEA Networks Institute, Madrid, Spain
{basaran, dressler}@ccs-labs.org, {villa.d, p.johari, m.polese, melodia}@northeastern.edu, {claudio.fiandrino}@imdea.org

***Abstract*—The realization of efficient Artificial Intelligence (AI) solutions for the optimization of next-generation Radio Access Network (RAN) relies on the availability of expansive, high-quality datasets that accurately capture nuanced, site-specific conditions. However, obtaining such abundant, domain-specific measurements poses a significant challenge, especially as network complexity and energy efficiency demand surge toward 6G. In response, we introduce GenerativeAI-enabled Digital Twin (Gen-TWIN), a synthetic data generation framework underpinned by a soft-attention LSTM-based generative adversarial network (soft-GAN). Our model augments realistic transmitter and receiver-focused RF datasets by supplementing scarce empirical samples and providing the synthetic data volumes essential for training advanced AI models on RAN. Accuracy results show that soft-GAN provided 19% performance improvement compared to baseline models.**

***Index Terms*—Digital Twin, Generative AI, O-RAN, Generative Adversarial Network, 5G/6G.**

## I. Introduction

The evolution of communication networks has been marked by continuous technological advancements and a growing complexity in network management. Among the significant innovations in this landscape is the concept of Digital Twin (DT), which has its roots in the industrial and manufacturing sectors [1] but has recently found extensive applications in communication networks [2]. Initially, DTs were leveraged to create virtual replicas of physical assets to optimize operations, predict failures, and enhance the overall system performance. In the context of Fifth Generation (5G) networks, DTs have facilitated enhanced performance monitoring, predictive maintenance, and improved service quality through the creation of highly accurate virtual models of network elements and their behaviors. These virtual models have enabled network operators to simulate and analyze various scenarios, leading to better-informed decisions and more efficient resource allocation. However, as we transition towards Sixth Generation (6G) networks [3], the complexity and scale of network operations are expected to surge exponentially. This growing complexity makes it increasingly difficult to create and manage accurate DTs at scale, necessitating the integration of Artificial Intelligence (AI) to enhance scalability [4].

Generative Artificial Intelligence (GenAI), particularly models such as Generative Adversarial Networks (GANs) [6] and Variational Autoencoders (VAEs), offer a robust framework for the creation and enhancement of DTs. Leveraging these capabilities, GenAI can also be employed to generate network data and enhance DT constructions [7]. The recent research report on DTs from the O-RAN ALLIANCE next Generation Research Group (nGRG) also mentions Synthetic Data Generation and Data Augmentation topics [8]. Accordingly, training robust AI/ML models (e.g., AI/ML models deployed in RAN Intelligent Controllers (RICs) that power Digital Twin for Radio Access Network (DT-RAN) deployments are often constrained by the difficulty of acquiring sufficient site-specific data [9]. According to the use case based on the report, a beamforming optimization model trained exclusively with dense urban data from San Francisco is unlikely to generalize to the nuanced traffic patterns, propagation environments, and interference conditions of a rural cell site in California. Data augmentation serves as a solution to existing limitations of DT-RAN. Although time series augmentation has been explored in the literature [10], studies on data augmentation specifically for DT-RAN remain scarce [11]. Furthermore, existing DT-RAN research does not specifically address Open Radio Access Networks (O-RAN).

To address these challenges, we propose the novel GenerativeAI-enabled Digital Twin (Gen-TWIN) platform (see Fig. 1), which leverages our innovative *soft*-GAN model. This a sophisticated soft-attention GAN based on Long Short-Term Memory (LSTM) layers meticulously designed to generate synthetic DT-RAN Radio Frequency (RF) data for both transmitter and receiver ends. Our model can be used to augment the limited datasets collected through field measurements by synthetically replicating key features of RF signals. The core results of our study are summarized as new contributions ("C") as follows:

- **C1.** We propose the Gen-TWIN platform, integrating Digital Twin and Generative Artificial Intelligence layers.
- **C2.** We design a novel soft-attention LSTM-based time-series GAN model, named *soft*-GAN, specifically for DT-RANs.
- **C3.** To evaluate the performance of *soft*-GAN, we implement and compare it against various baseline generative models.

## II. Gen-TWIN Platform

Our proposed platform comprises two integral components, DT and GenAI layers, as can be seen in Fig. 1.

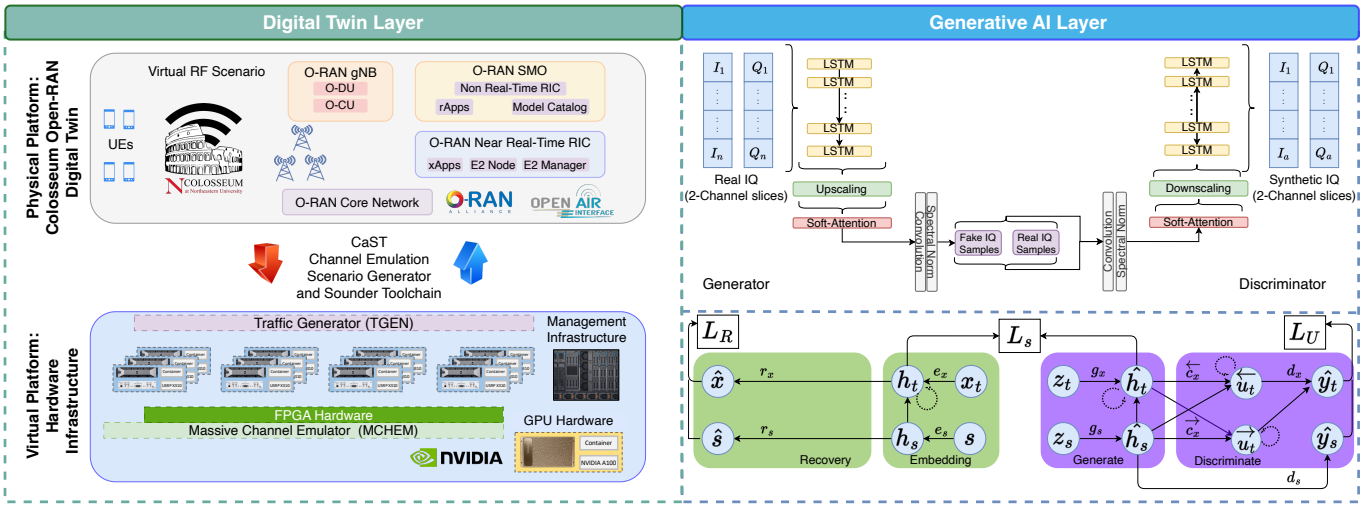


Fig. 1. **Left:** O-RAN Digital Twin Layer; Colosseum—the world's largest wireless network emulator with hardware in the loop [5] to create a comprehensive twin of the O-RAN network. **Right:** Generative AI Layer; Proposed *soft*-GAN Model — soft-attention-LSTM based Generative Adversarial Network.

**Digital Twin Layer:** Colosseum has 128 pairs of general-purpose compute servers and Software-Defined Radios (SDRs), referred to as Standard Radio Node (SRN); an advanced channel emulation system, known as Massive Channel Emulator (MCHEM); a cutting-edge AI/ML infrastructure; a Channel emulation generator and Sounder Toolchain (*CaST*) [12]; and a digital twin O-RAN system. Each SDR is connected to a Dell server via a National Instruments/Ettus USRP X310 SDR using a dedicated 10 Gbps link. Users can reserve SRNs and deploy custom or pre-configured open-source cellular protocol stacks using Linux Container (LXC). MCHEM is responsible for creating digital representations of real-world RF propagation scenarios, accurately reflecting the state of the RF channel at specific time instances. At its core, MCHEM consists of four quadrants containing 64 Field Programmable Gate Arrays (FPGAs) and 128 SDRs that handle the conversion between RF and baseband signals. For advanced AI/ML deployments, two NVIDIA DGX A100 stations with 8 GPUs each, providing 10 petaFLOPS of compute power.

**Generative AI Layer:** Our GAN framework can be described by the following minmax game:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1-D(G(\mathbf{z})))], \tag{1}$$

where:

- $\mathbf{x}$ represents real In-phase and Quadrature (IQ) data samples from DT layer.
- $p_{\text{data}}(\mathbf{x})$ denotes the distribution of real time series data.
- $\mathbf{z}$ represents random noise vectors sampled from a prior distribution $p_{\mathbf{z}}(\mathbf{z})$ (e.g., a Gaussian distribution).
- $G(\mathbf{z})$ generates synthetic IQ data samples from the noise vector $\mathbf{z}$.
- $D(\mathbf{x})$ represents the probability that $\mathbf{x}$ originates from the real data distribution rather than from $G$.

In the context of IQ data augmentation, the generator $G$ learns to produce realistic IQ data sequences, while the discriminator $D$ tries to distinguish between real and synthetic IQ data.

The training process iteratively updates the parameters of $G$ and $D$ using gradient-based optimization. Specifically, the discriminator is trained to maximize the likelihood of correctly distinguishing between real and synthetic samples:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] - \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]. \tag{2}$$

The generator is trained to minimize the likelihood of the discriminator correctly distinguishing synthetic from real samples:

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z(z)}[\log D(G(z))]. \tag{3}$$

These loss functions are alternately optimized in each iteration, leading to the gradual improvement of both networks. Once trained, the generator can produce synthetic univariate time series that augment the original dataset. Given a noise vector $z$, the generator produces a synthetic time series $\hat{x} = G(z)$. These synthetic samples can be used to increase the size of the training dataset, thereby improving the performance and generalization of machine learning models. The augmented dataset $\mathbf{X}'$ can be represented as $\mathbf{X}' = \mathbf{X} \cup \{G(z_i)\}_{i=1}^{N}$ where $\mathbf{X}$ is the original dataset and $N$ is the number of synthetic generated samples.

## III. Data Augmentation with *soft*-GAN

### A. Fundamentals of soft-GAN Model

The values of the time series data are denoted as $m_{i,t} \in \mathbb{R}$, where $i \in \{1,2,3,\ldots,N\}$ represents the index of individual samples and $t \in \{1,2,3,\ldots,T\}$ indicates the time points. The associated matrix of time feature vectors is $X_{1,T} = (x_1,\ldots,x_T)$ in $\mathbb{R}^{D\times T}$, where $D$ represents the number of time features. Consequently, we take the time series to have a fixed length $\zeta$, denoted as $\hat{M}_{i,t,\zeta} = (\hat{m}_{i,t},\ldots,\hat{m}_{i,t,\zeta})$. Then, using a generator function $G$ and a fixed time sequence $t$, we can model $\hat{M}_{i,t,\zeta} = G(n,\phi(i),X_{t:t+\zeta})$, where $n$ is a noise vector and $\phi$

is an embedding function that maps the index of the time series to a vector representation. In the generator $G$ and discriminator $D$, we use a soft attention mechanism along with convolutional spectral normalization $SN$ as described in [13].

For a sequence of length $l$, we can write:

$$\begin{aligned} p &: R^{n_f x l} \rightarrow R^{n_f^{'} x l} \\ x &\longmapsto \gamma softA(f(x)+f(x)) \end{aligned} \tag{4}$$

$$f(x) = SN(LeakyReLU(c(x))), \tag{5}$$

where $n_f^{'}$ is the number of output features, $c$ is the convolution operator, and *softA* is the soft-attention mechanism. We use *LeakyReLU* [14] to avoid the well-known dying ReLU problem where neurons can become inactive and always output zero for any input. On the generator side, we add a spectral normalization layer to further stabilize the training process and prevent sudden escalation of the gradient magnitude [15]. $\phi$ is a learnable parameter that, when indexed to 0, influences the network's ability to learn local features through the function $p$.

**soft-GAN Generator:** The first layer of the generator, $G1$, uses the main function block $p$:

$$\begin{aligned} G1 &: R^{n_f^{'} x 2^3} \rightarrow R^{n_f x 2^3} \\ \hat{M}_0 &\longmapsto \hat{M}_1 = p(\hat{M}_0), \end{aligned} \tag{6}$$

where $i \in [2,L]$. $G1$ maps two channel-separated IQ sequences $\hat{M}_{i-1}$ to an output $\hat{M}_i$, then applies an upscale to the function block $p$:

$$\begin{aligned} G_i &: R^{n_f x 2^{i+2}} \rightarrow R^{n_f x 2^{i+3}} \\ \hat{M}_{i-1} &\longmapsto \hat{M}_i = p(UP(\hat{M}_{i-1})). \end{aligned} \tag{7}$$

In the last layer of the generator, a multivariate sequence is obtained univariately as $\hat{M}_{i,t,\zeta}$ through a one-dimensional convolution operation.

**soft-GAN Discriminator:** The discriminator design is the same as the generator. The discriminator maps the generator's output $\hat{M}_{i,t,\zeta}$ and $X_{t:t+\zeta}$ to a discriminator score $d$. The *LeakyReLU* activation function is then used as follows:

$$d_{L+1} : R^{1+D,\zeta} \rightarrow R^{n_f,\zeta} \tag{8}$$

$$(\hat{M}_{L+1}, X_{t:t+\zeta}) \longmapsto \hat{M}_L = LR(c_1(\hat{M}_{L+1}, X_{t:t+\zeta})), \tag{9}$$

where $i \in [2, L]$. Downscaling is then applied to the main function block:

$$\begin{aligned} d_i &: R^{n_f x 2^{i+3}} \rightarrow R^{n_f x 2^{i+2}} \\ Y_i &\longmapsto Y_{i-1} = DOWN(m(Y_i)). \end{aligned} \tag{10}$$

The final operation is handled by a fully connected layer:

$$\begin{aligned} d_1 &: R^{n_f} \rightarrow R \\ Y_i &\longmapsto Y_0 = SN(FC(LeakyReLU(softA(c(Y_1))))) \end{aligned} \tag{11}$$

**Algorithm 1** *soft*-GAN Generator

**Input:** Noise vector $z \in \mathbb{R}^{(batch\_size, seq\_len, input\_dim)}$

**Input:** Real IQ Samples
**First LSTM Layer:** Process IQ and $z$ with an LSTM layer having 256 units, return sequences, dropout of 0.3, and recurrent dropout of 0.3.
$h_1 = \text{LSTM}_{256}(z)$
**Batch Normalization:** Normalize the output.
$h_1 = \text{BatchNorm}(h_1)$
**Second LSTM Layer:** Process the output with another LSTM layer having 256 units, return sequences, dropout of 0.3, and recurrent dropout of 0.3.
$h_2 = \text{LSTM}_{256}(h_1)$
**Batch Normalization:** Normalize the output.
$h_2 = \text{BatchNorm}(h_2)$
**Third LSTM Layer:** Process the output with another LSTM layer having 128 units, return sequences, dropout of 0.3, and recurrent dropout of 0.3.
$h_3 = \text{LSTM}_{128}(h_2)$
**Spectral Normalization:** Normalize the output.
$h_3 = \text{SpectNorm}(h_3)$
**Attention Mechanism:** Apply soft-attention to the output and concatenate it with the LSTM output.
attention_out = Attention($h_3$,$h_3$)
context_vector = Concat($h_3$,attention_out)
**Fully Connected Layer:** Apply a dense layer with 128 units and LeakyReLU activation.
$d_1 = \text{Dense}_{128}$(context_vector,activation='LeakyReLU')
**Dropout:** Apply dropout with a rate of 0.3.
$d_1 = \text{Dropout}_{0.3}(d_1)$
**Output:** Apply a final dense layer with 2 units to produce the synthetic IQ data.
output = $\text{Dense}_2(d_1)$
**Return:** Samples

### *B. Data Collection*

The data collection operation aims to collect IQ samples that can serve as a baseline for the data augmentation step. We collect IQ samples through two different testbeds: (i) POWDER for real-world data, and (ii) Colosseum for emulated data. A summary of the parameters for the data collection is shown in Table I. Additionally, Fig. 2 depicts the data collection scenarios for the real world (POWDER), and the digital world (Colosseum). The POWDER testbed [16] is a city-scale facility located at the University of Utah in Salt Lake City, UT. The platform supports a wide range of hardware and software configurations, enabling experimentation with real-world Over-The-Air (OTA) scenarios and the development of new wireless technologies. It provides various types of end-to-end software-defined nodes with different capabilities, deployed across the university campus.

In our real-world OTA data collection, we leverage two of its rooftop base station nodes, as shown by the red circles in Fig. 2: Honors and USTAR. For the emulation data, we utilize

Table I
DATA COLLECTION PARAMETERS SUMMARY BETWEEN THE TWO TESTBEDS: POWDER FOR OTA DATA, AND COLOSSEUM FOR EMULATED DATA.

| Parameter | POWDER | Colosseum |
|---|---|---|
| SDR | USRP X310 | USRP X310 |
| Nodes | Honors and USTAR Rooftop BS | Digitized Honors and USTAR Nodes |
| Frequency | 3.46-3.465 GHz | 1 GHz |
| Bandwidth | 1-10 MHz | 80 MHz |
| Time Capture | 30 seconds | 30 seconds |
| Capture Size | 2,57 GB | 2,96 GB |
| Sample Time | 10 MS/s | 10 MS/s |
| Tx gain | 25 dB | 15-25 dB |
| Rx gain | 25 dB | 15 dB |

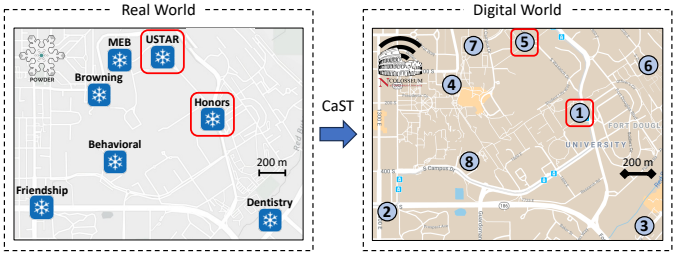


Fig. 2. Data collection scenarios for the real world (POWDER, left picture) and digital world (Colosseum, right picture). The red circles indicate the actual nodes used for data collection. Colosseum node 6 is a remnant of the now-decommissioned POWDER Medical Tower Rooftop node.

Colosseum's capabilities for digitization of the real-world [17]. We use a DT representation of the POWDER platform to recreate a similar real-world environment in an emulated one. Thus, we use the POWDER scenario created and described in [18] to collect the data between the same digitized nodes for Honors and USTAR rooftop base stations, digitized as nodes 1 and 5.

As shown in Table I, in both cases, we leverage USRP X310 as SDR hardware. The frequencies used for POWDER fall in the CBRS band between 3.46 and 3.465 GHz with bandwidths of 1 MHz and 10 MHz, respectively. In contrast, Colosseum uses its typical emulation of the baseband signal with a frequency of 1 GHz and a bandwidth of 80 MHz. The data capture duration for both platforms is 30 seconds per use case, with a sampling frequency of 10 megasamples per second. The transmission and reception gains of each SDR vary between 15 and 25 dB to compensate for the long distance of about 800 meters between the two nodes.

In each scenario, we leverage CaST [12], which is based on the GNU Radio software development toolkit, to send a Binary Phase-Shift Keying (BPSK)-modulated Galois Linear Feedback Shift Register (GLFSR) code sequence from the sender node (Honors) and to store the raw received IQ samples without any processing or equalization on the receiver side (USTAR).

## IV. EXPERIMENT EVALUATIONS

### A. Hyper-parameter Tuning and Training

In our training stage, the initial step in handling the dataset involves splitting it into three distinct parts: 80% training set, 10% validation set, and 10% test set. This division ensures that the model has adequate data for training, tuning, and evaluation, ultimately leading to better performance and generalization. Preprocessing is a critical step in preparing the data for training. The main goal is to normalize the IQ data to ensure consistency and stability during the training process. Normalization involves scaling the data so that its values fall within a specific range, -1 to 1. We achieve this by computing the mean ($\mu$) and standard deviation ($\sigma$) of the training data and then normalizing each data point accordingly.

Table II
HYPER-PARAMETER TUNING STAGE OF *soft*-GAN

| Hyper-param. | Search Space | Selected Parameter |
|---|---|---|
| Input size | [2, 4, 8, 16, 32] | 8 |
| Hidden layers | [1, 2, 3] | 3 |
| Hidden layers | [0.5, 1.5, 2.0, 3.0, 5.0] | 2.0 $x$ Input Size |
| Learning rate | [0.0001, 0.0005, 0.001, 0.005, 0.01] | 0.005 |
| Batch size | [64, 128, 256, 512, 1024] | 256 |
| Loss Function | [MAE, MSE, sMAPE] | MAE |
| Training steps | [1000, 5000, 10,000, 20,000, 40,000] | 20,000 |
| Optimizer | [RMSprop, SGD, Adam] | Adam |

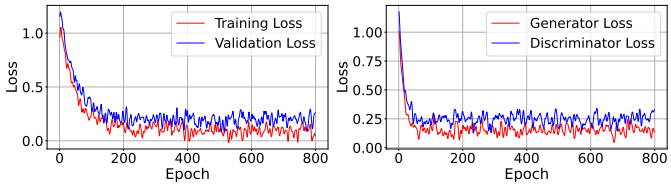


Fig. 3. *soft*-GAN training and validation loss over epochs (left picture). Generator and discriminator loss over epochs (right picture).

Table II shows our parameter space. During the hyperparameter tuning stage, an input size of 8 achieves a balance between model complexity and performance. Larger input sizes increase the model's complexity without significantly improving performance. Having three hidden layers allows the model to capture more complex patterns in the data. Setting the hidden layer size to $2 \times InputSize$ ensures that the network has sufficient neurons to learn intricate features of IQ samples without overcomplicating the model. A learning rate of 0.005 is a moderate choice that facilitates faster convergence while avoiding the pitfalls of overshooting the optimal point, which can occur with a higher learning rate. A batch size of 256 provides a good compromise between training stability and computational efficiency. We found that larger batch sizes require more memory and longer computation times per update, while smaller batch sizes lead to noisy gradient estimates.

As shown in Fig. 3 (left), both training and validation losses start at a relatively high value, around 1.2. This is expected as the model begins learning the data patterns from scratch. As training progresses, both losses decrease significantly, stabilizing at lower values, around 0.2. The training loss exhibits a smoother decline compared to the validation loss, which shows more fluctuations. However, these fluctuations are within a narrow range, and both losses remain close to each other throughout the training process. This behavior suggests that the model is learning effectively without overfitting. If the model were overfitting, we would observe the validation loss diverging from the training loss, with the validation loss increasing while the training loss continues to decrease. Conversely, underfitting would be indicated by both losses remaining high and not

Table III
NRMSE AND CONTEXT FID-SCORES (LOWER VALUES IN THESE METRICS INDICATE BETTER MODEL PERFORMANCE)

| | NRMSE | | Context FID-scores | |
|---|---|---|---|---|
| Models | 128 | 256 | 128 | 256 |
| TimeGAN | 0.27±0.06 | 0.26±0.02 | 0.23±0.04 | 0.63±0.10 |
| Autoencoder | 0.45±0.07 | 0.43±0.01 | 0.58±0.03 | 0.42±0.10 |
| BiLSTM | 1.03±0.055 | 0.97±0.003 | 1.07±0.20 | 0.92±0.07 |
| BiLSTMGAN | 0.95±0.06 | 1.13±0.04 | 0.71±0.01 | 0.23±0.05 |
| CnnGAN | 1.87±0.07 | 2.73±0.02 | 1.87±0.09 | 1.63±0.06 |
| **soft-GAN (ours)** | 0.23±0.02 | 0.21±0.07 | 0.15±0.01 | 0.09±0.04 |

Table IV
DISCRIMINATIVE AND PREDICTIVE SCORE RESULTS (LOWER VALUES IN THESE METRICS INDICATE BETTER MODEL PERFORMANCE)

| | Discriminative Score | | Predictive Score | |
|---|---|---|---|---|
| Models | 128 | 256 | 128 | 256 |
| TimeGAN | 0.028±0.003 | 0.013±0.01 | 0.019±0.08 | 0.17±0.03 |
| Autoencoder | 0.15±0.03 | 0.17±0.08 | 0.24±0.09 | 0.23±0.01 |
| BiLSTM | 0.04±0.03 | 0.053±0.002 | 0.18±0.04 | 0.13±0.04 |
| BiLSTMGAN | 0.113±0.09 | 0.088±0.02 | 0.27±0.017 | 0.09±0.03 |
| CnnGAN | 0.43±0.08 | 0.39±0.01 | 0.36±0.008 | 0.33±0.004 |
| **soft-GAN (ours)** | 0.017±0.02 | 0.013±0.06 | 0.016±0.004 | 0.009±0.006 |

Table V
COMPARISON OF CLASSIFICATION PERFORMANCE OF BASELINE MODELS

| Models | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| TimeGAN | 0.83 | 0.68 | 0.84 | 0.87 |
| Autoencoder | 0.88 | 0.63 | 0.82 | 0.85 |
| BiLSTM | 0.78 | 0.70 | 0.80 | 0.81 |
| BiLSTMGAN | 0.83 | 0.73 | 0.81 | 0.83 |
| CnnGAN | 0.77 | 0.61 | 0.79 | 0.73 |
| **soft-GAN (ours)** | **0.93** | **0.78** | **0.91** | **0.89** |

decreasing significantly. The close alignment of the training and validation losses in our model demonstrates a good balance, indicating that the *soft*-GAN is generalizing well to unseen data.

Figure 3 (right) plot shows the generator and discriminator losses over the epochs. Initially, both losses start high, around 1.0, which is typical at the beginning of GAN training. As training proceeds, the generator loss decreases steadily, stabilizing around 0.1 to 0.2. The discriminator loss also decreases but at a slower rate and stabilizes around 0.2 to 0.3. This dynamic is crucial in GAN training. The generator aims to produce data that the discriminator cannot distinguish from real data, while the discriminator aims to correctly identify real and generated data. The observed balance between these losses indicates that the training process is stable. If the discriminator loss were to decrease too rapidly, it would suggest that the discriminator is too powerful, making it difficult for the generator to improve. On the other hand, if the generator loss were too low compared to the discriminator loss, it might indicate mode collapse, where the generator produces limited diversity in the generated data. The balance observed here suggests that both the generator and discriminator are improving in tandem, contributing to the overall performance of the model. The stability and convergence of these loss curves provide a numerical testament to the model's performance. The training and validation losses stabilizing around 0.2, and the generator and discriminator losses around 0.1 to 0.3, indicate that the model has learned the underlying data distribution effectively.

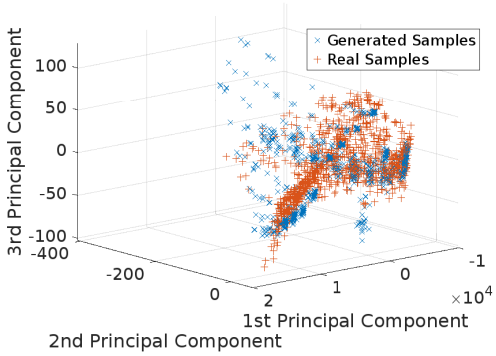


Fig. 4. Components of real and generated samples in the PCA subspace; Generated samples (marked as "×") exhibit a substantial overlap with the real samples (marked as "+").

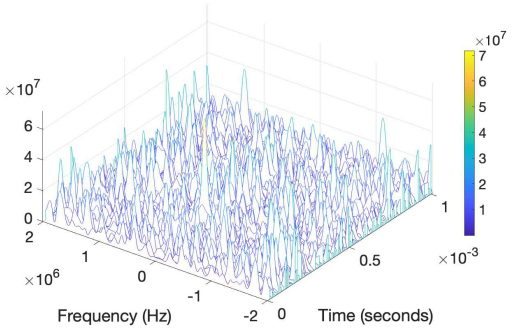


Fig. 5. Spectrogram of *soft*-GAN generated synthetic RF (25 dB gain) Signals

## B. Performance Comparison with Baseline Models

In this section, we provide a comprehensive evaluation of the proposed *soft*-GAN architecture against baseline models for time-series data augmentation, all trained on our dataset, including TimeGAN [10], Autoencoder [19], BiLSTM [20], BiLSTMGAN [21], and CNNGAN [22]. The performance metrics presented in Table III include Normalized Root Mean Squared Error (NRMSE) and Context Fréchet Inception Distance (FID)-scores, which are widely used in computer vision to evaluate the quality of synthetic data [23]. Table IV includes Discriminative and Predictive Scores, derived as evaluation criterion from baseline work [10], where a post-hoc time-series classification model (2-Layer LSTM) is trained to distinguish between original and generated sequences, labeled as "real" and "generated," respectively. The classification error on a held-out test set is reported as a quantitative metric for similarity assessment. In Table V, the results are presented using well-known classification metrics, including accuracy, precision, recall, and F1-score.

Performance results show that the use of a soft-attention mechanism allows the model to focus on the most relevant parts of the input sequence, enhancing its ability to capture long-range dependencies and intricate patterns in the time-series data. Table III shows how *soft*-GAN achieves an NRMSE of approximately 0.23 and 0.21, outperforming the best-performing baseline, TimeGAN, whose NRMSE ranges from 0.26 to 0.27. Statistically, this equates to about an 11–19% reduction in the

NRMSE relative to strong baselines. Lower NRMSE signifies a tighter fit of the generated time series to the real data distribution, indicating that *soft*-GAN's internal soft-attention and recurrent mechanisms effectively preserve key temporal dependencies. Decreasing the FID-score relative to leading baselines suggests that *soft*-GAN maintains more accurate local and global temporal structures. Table IV shows that for 128-length sequences, *soft*-GAN achieves a Discriminative Score approximately 39% lower than TimeGAN (0.017 vs. 0.028) and significantly lower than all other baselines. This improvement at the 256-length scale is also notable, matching TimeGAN at approximately 0.013 but with a more stable variance. Such a low Discriminative Score suggests that the synthetic sequences from *soft*-GAN are statistically indistinguishable from real data, reinforcing that our generative model is not merely capturing coarse trends but rather the nuanced statistical behavior observed in authentic RAN measurements.

To further validate the utility of generated data, Table V presents results from a classification task trained on synthetic samples. *soft*-GAN achieves an accuracy of 0.93, outperforming all baselines by at least 5 percentage points. Its F1-score of 0.89 represents a $\sim 9\%$ improvement over the closest competitor. This higher classification metric confirms that *soft*-GAN does not just produce visually or statistically similar data, but also preserves class-discriminative features. Higher precision results of our model indicate that *soft*-GAN has a lower false positive rate. *Soft*-GAN improves accuracy by 5-11% over TimeGAN and 7-19% over BiLSTM.

The Principal Component Analysis (PCA) projection in Fig. 4 offers a reduced-dimensionality view of both real and *soft*-GAN generated samples. If we consider the distributions of points along each principal component axis, the generated samples do not form a distinctly separate cluster; rather, they integrate into the same regions occupied by the real samples. This suggests that the generative model has learned not only the mean and covariance structure of the underlying data but also the higher-order moments that define the geometry of the data manifold in a lower-dimensional space. The close spatial proximity and pattern similarity between real and synthetic points implies that *soft*-GAN captured complex temporal correlations, amplitude variations, and other spectral signatures pertinent to RAN signals. The spectrogram in Fig. 5 provides a joint time-frequency representation of a *soft*-GAN generated RF signal at 25 dB signal power level. By applying a time-frequency decomposition—commonly via Short-Time Fourier Transform (STFT)—we visualize how signal energy is distributed across frequencies as a function of time. The generated spectrogram exhibits key spectral and temporal patterns that mirror realistic RF conditions.

## V. Conclusion

In this work, we introduced Gen-TWIN, a synthetic data generation DT platform, that fulfills the data-intensive demands of advanced DT-RAN operations. By producing high-fidelity RF signals, Gen-TWIN empowers AI-driven network management with richer, more representative training sets. The resulting improvements in model accuracy and robustness underscore the platform's potential to catalyze the evolution toward highly efficient, next-generation DT-RANs. Subsequent research may explore incorporating multi-modal data sources into Gen-TWIN's generative framework, further diversifying training sets and model robustness.

## Acknowledgments

This work has been funded by the German Federal Ministry of Education and Research (BMBF, Germany) as part of the 6G Platform under Grant 16KISK050, as well as 6G Research and Innovation Cluster 6G-RIC under Grant 16KISK020K. It has also been partially funded by the U.S. National Telecommunications and Information Administration (NTIA)'s Public Wireless Supply Chain Innovation Fund (PWSCIF) under Award No. 25-60-IF011 and by the U.S. NSF under Award CNS-1925601. C. Fiandrino is a Ramón y Cajal awardee (RYC2022-036375-I), funded by MCIU/AEI/10.13039/501100011033 and the ESF+.

## References

[1] S. Mihai, M. Yaqoob, D. V. Hung, et al., "Digital Twins: A Survey on Enabling Technologies, Challenges, Trends and Future Prospects," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2255–2291, 2022.

[2] N. P. Kuruvatti, M. A. Habibi, S. Partani, B. Han, A. Fellan, and H. D. Schotten, "Empowering 6G Communication Systems With Digital Twin Technology: A Comprehensive Survey," *IEEE Access*, vol. 10, pp. 112 158–112 186, Nov. 2022.

[3] X. Lin, L. Kundu, C. Dick, E. Obiodu, T. Mostak, and M. Flaxman, "6G Digital Twin Networks: From Theory to Practice," *IEEE Communications Magazine*, vol. 61, no. 11, pp. 72–78, Nov. 2023.

[4] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, "Digital-Twin-Enabled 6G: Vision, Architectural Trends, and Future Directions," *IEEE Communications Magazine*, vol. 60, no. 1, pp. 74–80, Jan. 2022.

[5] L. Bonati, P. Johari, M. Polese, et al., "Colosseum: Large-Scale Wireless Experimentation Through Hardware-in-the-Loop Network Emulation," in *IEEE DySPAN 2021*, Los Angeles, CA: IEEE, Dec. 2021.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al., "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, Oct. 2020.

[7] H. Chai, H. Wang, T. Li, and Z. Wang, "Generative AI-Driven Digital Twin for Mobile Networks," *IEEE Network*, vol. 38, no. 5, pp. 84–92, Sep. 2024.

[8] O-RAN next Generation Research Group (nGRG), "Digital Twin RAN: Key Enablers," O-RAN Alliance (O-RAN), Research Report, Oct. 2024.

[9] M. Polese, M. Dohler, F. Dressler, et al., "Empowering the 6G Cellular Architecture with Open RAN," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 245–262, Feb. 2024.

[10] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *NeurIPS 2019*, Vancouver, Canada: Curran Associates Inc., Dec. 2019.

[11] J. Zhang, Y. Zhang, J. Wang, et al., "Towards 6G Digital Twin Channel Using Radio Environment Knowledge Pool," arXiv, cs.NI 2312.10287, Mar. 2023.

[12] D. Villa, M. Tehrani-Moayyed, P. Johari, S. Basagni, and T. Melodia, "CaST: a toolchain for creating and characterizing realistic wireless network emulation scenarios," in *ACM MobiCom 2022, WINTECH Workshop*, Sydney, Australia: ACM, Oct. 2022, pp. 45–52.

[13] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," arXiv, cs.NI 1802.05957, Feb. 2018.

[14] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical Evaluation of Rectified Activations in Convolutional Network," arXiv, cs.NI 1505.00853, Nov. 2015.

[15] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-Attention Generative Adversarial Networks," arXiv, cs.NI 1805.08318, Jun. 2019.

[16] J. Breen, A. Buffmire, J. Duerig, et al., "POWDER: Platform for Open Wireless Data-driven Experimental Research," in *ACM MobiCom 2020, WINTECH Workshop*, Virtual Conference: ACM, Sep. 2020, pp. 17–24.

[17] D. Villa, M. Tehrani-Moayyed, C. P. Robinson, et al., "Colosseum as a Digital Twin: Bridging Real-World Experimentation and Wireless Network Emulation," *IEEE Transactions on Mobile Computing*, vol. 23, no. 10, pp. 9150–9166, Oct. 2024.

[18] L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "SCOPE: an open and softwarized prototyping platform for NextG systems," in *ACM MobiSys 2021*, Virtual Conference: ACM, Jun. 2021, pp. 415–426.

[19] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," arXiv, cs.NI 2003.05991, Apr. 2021.

[20] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF Models for Sequence Tagging," arXiv, cs.NI 1508.01991, Aug. 2015.

[21] A. Benchama and K. Zebbara, "Novel Approach to Intrusion Detection: Introducing GAN-MSCNN-BILSTM with LIME Predictions," arXiv, cs.NI 2406.05443, Jun. 2024.

[22] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," arXiv, cs.NI 1511.06434, Jan. 2016.

[23] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," arXiv, cs.NI 1706.08500, Jan. 2018.