Evaluating Tabular and Time-Series Data Augmentation for 6G-Relevant Network-Performance Regression

Quinton den Haan[1]

Supervisors: Yuandou Wang[1]**, Rihan Hai**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Quinton den Haan
Final project course: CSE3000 Research Project
Thesis committee: Yuandou Wang, Rihan Hai, Julian Urbano

Abstract

Data-driven methods are expected to play an important role in future sixth-generation (6G) wireless systems, where network data can support performance prediction, simulation, and network optimization. However, collecting large and representative network-performance datasets can be difficult, which motivates the use of data augmentation. This study evaluates how different tabular and time-series augmentation techniques compare when addressing data scarcity in datasets relevant to future 6G systems. Two regression tasks are studied: a tabular AMF performance task using XGBoost and a time-series Python web-server performance task using an LSTM. Four tabular augmentation methods are evaluated: Gaussian Noise, SMOGN, CTGAN, and TVAE. Four time-series augmentation methods are evaluated: Jittering, Time Warping, TS-Mixup, and Frequency-domain augmentation. The methods are compared using downstream regression performance, statistical realism metrics, and diagnostic analysis of augmented data and test-set residuals. The results show that augmentation does not consistently improve regression performance. In the tabular task, all augmentation methods reduced performance compared with the XGBoost baseline. In the time-series task, Frequency-domain augmentation was the only method that improved the LSTM baseline, substantially reducing RMSE and MAE, although the final test-set R^2 remained negative. The diagnostics suggest that useful augmentation depends not only on preserving marginal distributions or value ranges, but also on preserving task-relevant feature-target relationships and temporal structure. Overall, the findings show that augmentation effectiveness is method- and data-type dependent, and that predictive performance should be evaluated together with statistical fidelity diagnostics.

1 Introduction

Reliable and low-latency communication for data-heavy applications is set to be enabled by the rapid evolution toward sixth-generation (6G) wireless systems[1]. Future networks are expected to support increasingly data-driven tasks such as model training, simulation, and network optimization[2]. However, the effectiveness of these methods is constrained strongly by the availability of representative data. In real life, collecting large, diverse and high-quality 6G datasets, originating from end devices, network infrastructure, or experimental testbeds is difficult since real network measurements can be very expensive to obtain, limited in coverage, noisy or imbalanced[3; 4; 5; 6]. This essentially becomes a bottleneck for downstream 6G applications, and thus calls for the necessity of robust data enhancement techniques.

Recent research[7; 6] shows that data augmentation[8][9] is gradually becoming one of the reliable approaches to address the above mentioned issues It is the process of generating synthetic data from already existing samples and it has been explored extensively by broader machine learning literature.

In tabular data, this can be done using simple perturbation methods such as feature perturbation[8], oversampling methods such as SMOGN[10], or the more recent deep generative models such as CTGAN[11] and VTAE[11]. In time-series data, common approaches include methods that add simple noise such as jittering[12], temporal alignment modifying methods such as time warping[13; 9], sequence-mixing methods such as MixUp[14; 15], or frequency representation perturbing methods such as Frequency-Domain augmentations[12; 16]. These methods can potentially improve generalization by giving a model a wider range of training data. However, despite the availability of such a variety of augmentation techniques, these methods have not been systematically compared on datasets relevant to future 6G systems.

This work addresses this gap by answering the following main research question: *How do different tabular and time-series augmentation techniques compare when addressing data scarcity in datasets relevant to future 6G systems?* To answer this, two sub-questions are investigated:

- How does each augmentation technique affect downstream regression performance?
- How well do augmented datasets preserve statistical realism with respect to the original training data?

To arrive at a solution, the main contribution is a systematic experimental framework in which we compare four tabular augmentation methods: feature perturbation, SOMGN, CTGAN and TVAE; and four time-series augmentation methods: jittering, time warping, TS-MixUp and frequency-domain augmentation. It is conducted in a controlled and reproducible experimental environment.

2 Related Work

2.1 Data augmentation in the 6G context

Data augmentation has become an important technique for improving ML performance when available data are limited, imbalanced, or insufficiently representative. By generating synthetic samples from existing data observations, augmentation can increase diversity and improve model generalization. In the context of future 6G systems, data augmentation is relevant because data-driven network management will require large and diverse datasets, while collecting real network measurements can be expensive and incomplete[3; 4; 5].

The need for augmentation is particularly important in wireless and network-performance settings. Wireless data can have complex structures, including temporal dependencies, channel effects, user mobility, and heterogeneous network conditions. Therefore, simply increasing the number of samples is not enough. Augmented data should also preserve the statistical and task-relevant structure of the original data. If augmentation changes important relationships in the data, it may reduce downstream model performance instead of improving it[17; 16].

2.2 Related Work

Several recent studies have explored data augmentation in wireless communication contexts.

Wen et al.[6] studied generative AI for data augmentation in wireless networks and discussed the potential of generative models for mitigating data scarcity across physical-, network-, and application-layer wireless tasks. Their case study used a diffusion-based approach to generate channel state information for Wi-Fi gesture recognition and showed that augmented CSI data can improve recognition performance. This work demonstrates the potential of generative augmentation in wireless systems, but it mainly focuses on generative methods and signal-like wireless data rather than comparing classical and generative methods on network-performance regression data.

Serbetci et al.[18] proposed wireless channel-aware augmentation methods for deep-learning-based indoor localization. Instead of applying generic transformations directly, their work used wireless channel and system knowledge to create augmentations that better match channel behavior. Their results showed that domain-aware augmentation can reduce the burden of data collection and improve localization accuracy. This highlights the importance of preserving domain structure when augmenting wireless data.

Patel et al.[19] investigated conditional GAN-based data augmentation for automatic modulation classification. Their work used CGAN-generated signal samples to improve CNN-based modulation classification when labeled wireless data are limited. This supports the idea that generative augmentation can be useful in wireless learning tasks, but the task is classification-based and signal-oriented rather than regression-based network-performance prediction.

Gajjar et al.[7] proposed LLM-AUG, a framework that uses large language models for wireless data augmentation in low-shot RF learning tasks such as modulation and interference classification. Their results show that augmentation can improve performance under limited labeled data and distribution shift. However, the focus is again on classification and RF signal representations, not tabular or time-series network-performance regression.

The gap. Even though existing wireless augmentation studies show how useful data augmentation can be in wireless communication settings, there has not been systematic comparison that show how classical perturbation methods, regression-oriented oversampling methods, deep tabular generative models, and time-series transformations compare under the same controlled regression framework. This study addresses that gap by systematically comparing four tabular augmentation methods and four time-series augmentation methods on 5G/core-network and cloud-native performance datasets used as experimentally available proxies for future 6G network-performance data. The methods are evaluated using both downstream regression performance and statistical realism metrics.

3 Preliminaries

This section defines the experimental problem setting and introduces the augmentation methods used in this study.

3.1 Problem setting

The experiment consists of two supervised regression settings: one tabular and one time-series setting. In both cases, the goal is to evaluate whether augmenting the training data improves downstream prediction performance while preserving statistical realism.

For the tabular setting, each observation is represented as a feature vector $x_i \in \mathbb{R}^d$ with a continuous target $y_i \in \mathbb{R}$. The training set is

$$D_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{n}$$

A tabular augmentation method creates additional samples

$$\tilde{D}_{\text{train}} = \{(\tilde{x}_j, \tilde{y}_j)\}_{j=1}^{m}$$

which are combined with the original training set. The downstream model is trained on

$$D_{\text{aug}} = D_{\text{train}} \cup \tilde{D}_{\text{train}}$$

For the time-series setting, each input sample is a sliding window

$$\tilde{D}_{\text{train}}^{\text{TS}} = \left\{(\tilde{X}_j, \tilde{y}_j)\right\}_{j=1}^{m}, \qquad \tilde{X}_j \in \mathbb{R}^{T \times d}$$

where T is the window length and d is the number of features. The target y_i is the next-step latency value. A time-series augmentation method transforms the training windows into additional windows $\tilde{X}_i$, while validation and test windows remain unchanged. This ensures that performance is measured only on real unseen data.

In this study, the tabular task predicts lat100 from resource and workload features. The time-series task predicts next-step lat100 from sliding windows of resource, workload, and usage features.

3.2 Tabular Data Augmentation

Feature Perturbation (Gaussian Noise)

Gaussian Noise[8] is used as a simple feature-perturbation baseline. For a feature vector x_i, the augmented sample is created as

$$\tilde{x}_i = x_i + \epsilon,$$

where $\epsilon_j \sim \mathcal{N}(0, \sigma^2 s_j^2)$, and s_j is the standard deviation of feature j in the training set. The target is kept unchanged:

$$\tilde{y}_i = y_i.$$

This method is implemented manually. It is simple, fast, and easy to reproduce. It can simulate small measurement fluctuations.

SMOGN

SMOGN[10] is a regression-oriented oversampling method designed for imbalanced regression problems. It focuses on rare or extreme target regions and generates additional samples using interpolation and Gaussian noise. A simplified intuition is that a synthetic sample is generated between neighboring observations:

$$\tilde{x} = x_i + \lambda(x_j - x_i),$$

where x_j is a neighboring sample and $\lambda \in [0, 1]$. Gaussian noise may then be added depending on the neighbourhood structure. In this study, SMOGN is implemented using the Python smogn package[20].

It targets underrepresented regions of the regression target, which can be useful when rare high-latency cases are important.

CTGAN

CTGAN[11] is a deep generative augmentation method designed for tabular datasets containing mixed numerical and categorical features. It uses a generator-discriminator framework in which the generator learns to create synthetic tabular samples, whereas the discriminator will try to distinguish synthetic samples from real data samples. In this study, CTGAN is implemented using SDV's CTGANSynthesizer[21]. It can generate new combinations of feature and target values rather than only perturbing existing rows, which can be useful for complex tabular distributions.

VTAE

TVAE[11] is a tabular variational autoencoder. It is a probabilistic and, similar to CTGAN, deep generative model for synthetic tabular data generation. It first compresses input data into a lower-dimensional latent representation, then reconstructs synthetic samples from the learned latent space. Unlike GAN-based (Generative Adversarial Network[22]) methods, it utilizes probabilistic latent-space sampling instead of adversarial training, making model training more stable. In this study, TVAE is implemented using SDV's TVAESynthesizer[21].

3.3 Time-Series Data Augmentation

Jittering

Jittering adds random Gaussian noise to each timestep of a sequence:

$$\tilde{X}t, j = Xt, j + \epsilon_{t,j},$$

where $\epsilon_{t,j} \sim \mathcal{N}(0, \sigma^2)$. This method is implemented manually based on standard definitions in time-series augmentation literature[9].

It is simple and can make a model more robust to small measurement fluctuations.

Time Warping

Time Warping[9] changes the temporal alignment of a sequence by applying a smooth random time transformation. Conceptually, a warped sequence can be written as

$$\tilde{X}t, j = X\tau(t), j,$$

where $\tau(t)$ is a smooth warped time index and interpolation is used when $\tau(t)$ is non-integer. It was implemented as a generalized version of the time_warp function from the Uchida Lab time_series_augmentation repository[9]. It was adapted to handle multivariate windows with shape $(\text{samples}, \text{timesteps}, \text{features})$.

It can simulate variation in the speed or alignment of temporal patterns

TS-Mixup

TS-Mixup is a time-series adaptation of the Mixup[14; 15] principle. Two training windows and their targets are linearly combined:

$$\tilde{X} = \lambda X_i + (1 - \lambda)X_j,$$

$$\tilde{y} = \lambda y_i + (1 - \lambda)y_j,$$

where $\lambda \sim \text{Beta}(\alpha, \alpha)$. This method is implemented manually for windowed time-series regression.

It creates smooth intermediate samples and often keeps values within observed numerical ranges.

Frequency-Domain Augmentation

Frequency-domain augmentation[12; 16] modifies each feature sequence after transforming it into the frequency domain. For a feature sequence $X_{\cdot,j}$, the method can be written as

$$\tilde{X}_{\cdot,j} = \mathcal{F}^{-1}(\mathcal{F}(X_{\cdot,j}) + \eta),$$

where $\mathcal{F}$ is the real Fourier transform and η is small complex Gaussian noise added to the Fourier coefficients. This study implements a simplified FFT coefficient perturbation baseline inspired by the general frequency-domain augmentation principle described by Wen et al[16].

It perturbs the global spectral structure of the sequence rather than adding independent pointwise noise.

4 Methodology

4.1 General Experimental Pipeline

The study consists of two experimental tracks: one for tabular data and one for time-series data, both tracks follow the same procedure.

First, preprocessing is applied to remove timestamp columns from the inputs, convert memory-limits to numerical values, and remove latency percentile columns that would leak information about the prediction target. The data are split into training, validation where needed, and test sets.

Then, baseline regression models are trained on the datasets without augmentation. These baseline models establish a reference level of predictive performance. The regression model used for tabular and time-series augmentation are respectively XGBRegressor from XGBoost[23] and LSTM from Tensorflow[24].

Each augmentation method will then be applied to training data to generate additional synthetic training samples and obtain a new augmented dataset, the validation and test sets remain unchanged.

Finally, after augmentation, the same downstream model is trained using the augmented dataset.

4.2 Evaluation Metrics

Predictive performance is evaluated on the untouched test set using **RMSE**, **MAE**, and test-set $\mathbf{R^2}$. RMSE and MAE are the primary metrics because they directly measure prediction error in the original latency unit. R^2 is reported as an additional normalized measure of predictive performance.

Statistical fidelity is evaluated by comparing the augmented training set with the original training set. In this study, we check if the augmented training data preserve marginal feature distributions, pairwise numerical relationships, and observed value ranges.

The **KS (Kolmogorov-Smirnov)** statistic[25] is used to compare marginal distributions. For a feature column, it is defined as

$$D = \sup_z |F_{\text{real}}(z) - F_{\text{aug}}(z)|,$$

where F_{real} and F_{aug} are the empirical cumulative distribution functions of the real and augmented data. Lower KS values indicate more similar distributions.

Correlation Similarity[26] is used to compare pairwise numerical relationships. For two columns A and B, SDMetrics computes a correlation coefficient for the real data and for the augmented data, then converts the difference into a similarity score. A score close to 1 means that the pairwise correlation is similar in the real and augmented data.

Boundary Adherence[26] measures whether augmented values remain within the minimum and maximum values observed in the real training data. For a numerical feature (j), this can be interpreted as the proportion of augmented values satisfying

$$\min(x_j^{\text{real}}) \leq \tilde{x}_j \leq \max(x_j^{\text{real}}).$$

Higher Boundary Adherence indicates stronger preservation of observed value ranges.

For the time-series experiment, realism metrics are computed on a row-level representation derived from all timesteps in the augmented training windows. This allows the same realism metrics to be applied consistently to both tabular and time-series augmented training data.

5 Experimental Setup

This section describes the datasets, preprocessing, software environment, and model settings used in the experiments.

5.1 Software Environment

All experiments were implemented in Python 3.11.0 using Jupyter Notebook. The main libraries used are the following:

- pandas and numpy for data processing
- scikit-learn for preprocessing and regression metrics
- xgboost.XGBRegressor (3.2.0) for tabular regression
- tensorflow.keras.layers.LSTM (2.17.0) for time-series regression
- scipy for KS (ks_2samp) and interpolation (CubicSpline)
- matplotlib for plotting
- smogn for SMOGN
- sdv (1.36.2) for CTGAN and TVAE
- sdmetrics for Correlation Similarity and Boundary Adherence

For reproducibility, the random seed was fixed to 42 using NumPy, Python's random module, TensorFlow/Keras, and PyTorch where relevant. TensorFlow deterministic operations were enabled in the notebook.

5.2 Datasets

This research uses 5G/core-network and cloud-native performance datasets as experimentally available proxies for future 6G network-performance data.

AMF Performance Dataset

The AMF Performance dataset[27] is publicly available through Zenodo. The dataset contains performance measurements collected from the Access and Mobility Management Function, a core component of the 5G core network architecture.

The dataset contains measurements describing both resource allocation and runtime behaviour, including CPU limits, memory limits, CPU usage, memory usage, and the number of parallel registration requests. In addition, several latency percentiles are provided.

Although originally a time-series dataset, it is used for evaluating tabular data augmentation techniques because it can be treated as containing independent observations represented by numerical feature vectors and continuous regression targets.

Python Web Server Performance Dataset

The Python Web Server Performance dataset originates from the same benchmarking framework[27]. It contains measurements collected from containerized Python web servers under varying workloads and resource allocations. During testing, ApacheBench was used to generate requests with different request counts and concurrency levels.

The dataset includes resource-allocation parameters, runtime resource utilization metrics, workload characteristics, and response-time measurements.

Although the dataset consists of timestamped observations, sliding-window segmentation is applied to construct sequences of observations. This transforms the dataset into a time-series prediction problem suitable for evaluating time-series augmentation techniques.

5.3 Pre-processing

For the AMF dataset, 8000 rows were randomly sampled using random_state. The selected input features were ram_limit, cpu_limit, ram_usage, cpu_usage, and n, and the target variable was lat100. The timestamp column, the mean column, and all remaining latency percentile columns were removed to avoid target leakage due to high correlation to the target.

For the Python web-server dataset, the first 13000 chronologically ordered rows were used after sorting by timestamp. The selected input features were ram_limit, cpu_limit, ram_usage, cpu_usage, n, and c, and the target variable was again lat100. The remaining latency percentile columns were again removed because they are highly correlated with the target feature and would therefore leak target information into the model. The timestamp column was used only to preserve chronological ordering and was not used as a model feature.

For the tabular dataset, a random 80/20 train/test split was used. For the time-series dataset, the raw rows were split chronologically before window creation. The final 20% of rows were used as the test portion. From the remaining 80%, 15% was used as a real validation portion for LSTM early stopping. Sliding windows of length 10 were then created separately for the training, validation, and test portions. This avoids overlap between train, validation, and test windows.

5.4 Hyperparameters

For the augmentation techniques, CTGAN and TVAE specifically, the number of generated synthetic samples was set to 50% of the original training-set size. Both synthesizers train on 500 epochs.

For the downstream regression model XGBRegressor, the same model configuration was used for the baseline and all tabular augmentation methods:

- n_estimators = 200
- learning_rate = 0.05
- max_depth = 6
- subsample = 0.9
- colsample_bytree = 0.9
- objective = "reg:squarederror"
- n_jobs = -1

For the downstream model LSTM, the same model configuration was used for the baseline and all time-series augmentation methods:

- LSTM units = 32
- Dropout = 0.10
- Dense hidden layer = 16 units, ReLU activation
- Output layer = 1 unit
- Optimizer = Adam
- Learning rate = 0.001
- Loss = mse
- Epochs = 100
- Batch size = 64
- Early stopping patience = 5
- Validation split = fixed real validation set

The LSTM target values were scaled during training and inverse-transformed before computing final regression metrics, so that RMSE and MAE were reported in the original latency unit.

6 Results

In this section, we present the results of the data augmentation experiments. We first compare the regression performance and realism of the tabular augmentation methods along with its baseline, then we do the same for the time-series augmentation experiment.

Method	RMSE	MAE	R^2
Baseline	2.523619e+06	9.479656e+05	0.983792
GN	2.728919e+06	9.734254e+05	0.981048
SMOGN	2.866206e+06	1.072793e+06	0.979093
CTGAN	2.972432e+06	1.313823e+06	0.977514
TVAE	2.871573e+06	1.246597e+06	0.979014

Table 1: Benchmark results over the tabular augmentation techniques, namely no augmentation (baseline), feature perturbation with Gaussian noise (GN), SMOGN, CTGAN and TVAE. We report the RMSE, MAE and R^2 for their regression task.

6.1 Tabular Augmentation Evaluation

The effect of augmentations on the AMF performance dataset are illustrated in Table 1. The baseline XGBoost model achieved the best result, with RMSE 2.52×10^6, MAE 9.48×10^5, and $R^2 = 0.9838$. All tabular augmentation methods reduced performance compared with the baseline. Gaussian Noise was the least harmful method, increasing RMSE by about 8.1% and reducing R^2 to 0.9810. SMOGN increased RMSE by about 13.6% and reduced R^2 to 0.9791. CTGAN and TVAE caused larger performance drops (17.8% and 13.8% increase in RMSE), with CTGAN giving the weakest tabular result: RMSE 2.97×10^6, MAE 1.31×10^6, and $R^2 = 0.9775$.

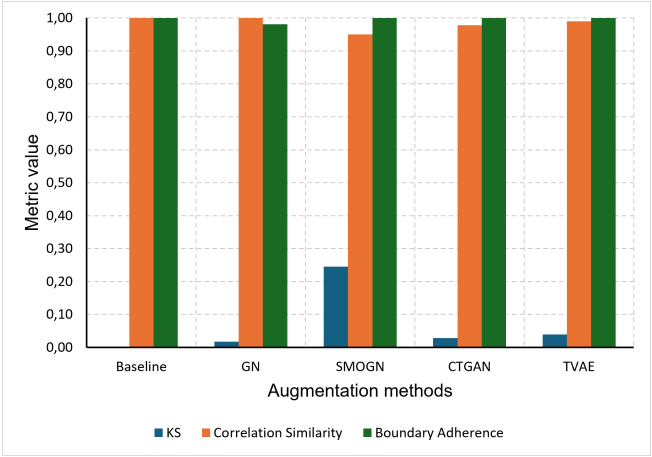

Figure 1: Tabular realism comparison across the baseline and augmentation methods. KS measures marginal distribution difference, where lower is better. Correlation Similarity and Boundary Adherence measure preservation of pairwise relationships and observed value ranges, where higher is better.

Figure 1 indicates the recorded tabular realism metrics. Gaussian Noise produced the most statistically similar augmented training set, with the lowest KS value, 0.0173, and the highest Correlation Similarity among the augmented methods, 0.9998. CTGAN and TVAE also had low KS values, 0.0281 and 0.0392 respectively, and both achieved perfect Boundary Adherence. However, their lower predictive performance shows that staying within observed value ranges does not necessarily preserve the feature-target relationship. SMOGN had the largest distributional shift, with

Method	RMSE	MAE	R^2
Baseline	6.310240e+07	3.453243e+07	−21.997886
Jittering	7.595157e+07	4.380219e+07	−32.317287
TW	7.335879e+07	4.495415e+07	−30.081398
TSMix	1.535801e+08	1.094081e+08	−135.227833
Freq-D	3.158871e+07	1.951103e+07	−4.763150

Table 2: Benchmark results over the time-series augmentation techniques, namely no augmentation (baseline), Jittering, Time Warping (TW), TS-Mixup (TSMix) and Frequency-Domain augmentation (Freq-D). We report the RMSE, MAE and R^2 for their regression task.

$KS = 0.2447$, and the lowest Correlation Similarity, 0.9505, although it maintained high Boundary Adherence.

6.2 Times-series Augmentation Evaluation

The effect of augmentations on the Python web-server dataset are illustrated in Table 2. The baseline LSTM performed poorly on the chronological test set, with RMSE 6.31×10^7, MAE 3.45×10^7, and $R^2 = -22.00$. Jittering and Time Warping worsened the baseline, increasing RMSE by approximately 20.4% and 16.3% respectively. TS-Mixup performed worst, increasing RMSE to 1.54×10^8 and reducing R^2 to -135.23. Frequency-domain augmentation was the only time-series method that improved performance, reducing RMSE by about 49.9%, reducing MAE by about 43.5%, and improving R^2 to -4.76.

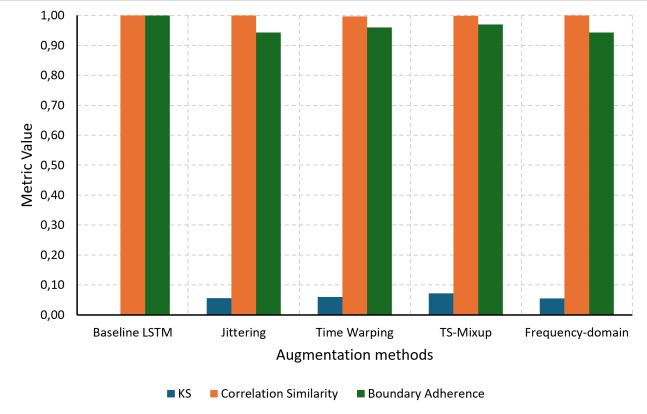

Figure 2: Time-series realism comparison across the baseline and augmentation methods. KS measures marginal distribution difference, where lower is better. Correlation Similarity and Boundary Adherence measure preservation of pairwise relationships and observed value ranges, where higher is better.

Figure 2 presents the recorded time-series realism metrics. All time-series methods preserved pairwise correlations very well, with Correlation Similarity values above 0.997. Frequency-domain augmentation had the lowest KS value, 0.0553, and the highest Correlation Similarity, 0.99997, indicating the strongest preservation of marginal distributions and pairwise numerical relationships among the time-series methods. Jittering produced similar KS and Correlation Similarity values, but had lower Boundary Adherence, 0.9427. Time Warping had KS = 0.0601, Correlation Similarity = 0.9973, and Boundary Adherence = 0.9604. TS-Mixup achieved the highest Boundary Adherence, 0.9703, but also had the highest KS value, 0.0719, and the weakest predictive performance.

Overall, the results show that data augmentation did not consistently improve regression performance. In fact, for the tabular regression task, all augmentation methods degraded performance compared with the original training data. For the time-series regression task, only Frequency-domain augmentation improved the LSTM performance and R^2 while preserving the statistical fidelity relatively well, although the resulting R^2 remained negative. The realism metrics provide useful additional information, but they do not perfectly predict downstream performance.

7 Discussion

This section provides a more detailed analysis of the collected results and discusses the limitations within this experiment

7.1 Interpretation of the tabular results

The tabular task predicts lat100 from resource and workload-related features. The baseline XGBoost model achieved the best performance, suggesting that the original training data already contained enough information for this prediction task. This is supported by the diagnostics: the real training and test sets were highly similar, with an average KS distance of only 0.0173. Therefore, augmentation had limited opportunity to improve generalization.

Gaussian Noise was the least disruptive tabular method. It had a low KS distance from the original training data and almost no feature-target correlation shift, with an average shift of only 0.0002. This explains why it caused the smallest performance drop. However, because the baseline was already strong, the noisy copies did not add useful new information and slightly reduced performance.

SMOGN changed the training distribution much more strongly. Its average KS distance was 0.2447, and the mean lat100 increased from approximately 1.17×10^7 to 2.90×10^7. This confirms that SMOGN strongly emphasized high-latency regions. Although this matches the purpose of regression oversampling, the residual diagnostics show that it increased error in the broader high-latency region and increased overall MAE. Therefore, the oversampled target distribution did not align well enough with the test set.

CTGAN and TVAE generated values within the observed training boundaries, but both reduced regression performance. Their diagnostics show that the generated data preserved value ranges better than feature-target relationships. CTGAN had a larger feature-target correlation shift than TVAE, which is consistent with its larger performance degradation. This suggests that the generated rows were numerically plausible but less useful for predicting lat100.

Overall, the tabular results show that preserving boundaries or marginal distributions is not sufficient. For this task, augmentation was only useful if it preserved the feature-target relationship, and none of the tabular methods improved on the already strong baseline.

7.2 Interpretation of the time-series results

The time-series task was more difficult because it predicts next-step lat100 from chronological sliding windows. The diagnostics show a much stronger train-test distribution shift than in the tabular task: the average KS distance between real training and real test data was 0.3787. This supports the poor baseline LSTM performance and explains why the negative R^2 should be interpreted as poor generalization rather than a calculation error.

The baseline LSTM strongly over-predicted on the test set, with a mean residual of approximately -3.04×10^7, and it over-predicted in about test set,73.7% of test cases. Its error also increased over time, with MAE rising from approximately 1.54×10^7 in the first chronological test segment to 5.29×10^7 in the final segment. This indicates that chronological distribution shift was a major difficulty.

Jittering did not address this problem. Although it preserved feature-target correlations relatively well, it introduced boundary violations, with an average violation rate of 5.7%. It also increased errors in high-latency regions and later test segments. This suggests that independent pointwise noise did not create useful temporal variations for next-step latency prediction.

Time Warping also reduced performance. It caused the largest lag-1 autocorrelation shift among the time-series methods, approximately 0.0489. Since the target depends on the original chronological window, changing temporal alignment while keeping the same target likely weakened the relationship between input windows and next-step latency.

TS-Mixup performed worst. Although it often stayed within observed value ranges, it had the highest time-series KS shift and produced severe over-prediction, with a mean residual of approximately -1.09×10^8. This indicates that linearly mixing windows and targets created samples that were numerically valid but not representative of realistic latency behaviour.

Frequency-domain augmentation was the only method that improved the LSTM baseline. It reduced MAE in all target ranges and all chronological test segments. It also had the smallest feature-target correlation shift and the smallest lag-1 autocorrelation shift, approximately 0.00015. This suggests that it introduced useful variation while preserving temporal structure better than Jittering, Time Warping, and TS-Mixup.

However, its R^2 remained negative, so the result should be interpreted as a relative improvement over a weak baseline rather than a fully successful prediction model. The main remaining challenge is the strong chronological distribution shift between training and test data.

7.3 Limitations

This controlled experiment has several limitations. Firstly, only one tabular dataset and one time-series dataset were used. Both are network-performance datasets relevant to future 6G network-management scenarios, but they are not complete representations of all possible 6G data types.

Secondly, the results depend on the chosen downstream models. Both chosen models in this experiment were known to be good standards from literature. XGBoost performed very strongly on the tabular task, leaving little room for augmentation to improve performance. The LSTM, in contrast, performed poorly on the chronological time-series test set. Different model architectures and hyperparameters, could lead to different conclusions.

Thirdly, the realism metrics measure statistical fidelity, not complete physical validity. KS, Correlation Similarity, and Boundary Adherence capture marginal distributions, pairwise relationships, and observed value ranges, but they do not guarantee that augmented samples obey all real network constraints or preserve long-range temporal dependencies.

Finally, the augmentation methods were implemented as practical baseline variants with fixed parameters. More extensive hyperparameter tuning or more advanced domain-specific augmentation methods may produce different results.

8 Conclusions and Future Work

This research explored how different tabular and time-series augmentation techniques compare when addressing data scarcity in datasets relevant to future 6G systems. Two experimental tracks were evaluated: a tabular AMF performance task using XGBoost and a time-series Python web-server performance task using an LSTM. The study compared Gaussian Noise, SMOGN, CTGAN, TVAE, Jittering, Time Warping, TS-Mixup, and Frequency-domain augmentation.

The results show that augmentation did not consistently improve regression performance. In the tabular experiment, the baseline XGBoost model achieved the best predictive performance, and all augmentation methods reduced performance. Gaussian Noise caused the smallest degradation, while CTGAN caused the largest degradation. The diagnostic results suggest that this was because the tabular training and test distributions were already highly similar, leaving limited room for augmentation to improve generalization. SMOGN shifted the target distribution strongly toward higher latency values, while CTGAN and TVAE preserved observed value ranges but changed feature-target relationships.

In the time-series experiment, the baseline LSTM generalized poorly to the chronological test set, indicating that the task was substantially more difficult than the tabular setting. Jittering, Time Warping, and TS-Mixup further reduced performance. Frequency-domain augmentation was the only method that improved the LSTM baseline, reducing RMSE and MAE substantially. Residual and temporal diagnostics suggest that this method was more successful because it introduced variation while preserving feature-target relationships and temporal structure better than the other time-series methods. However, the final R^2 remained negative, so the improvement should be interpreted as relative improvement over a weak baseline rather than as a fully successful prediction model.

These findings answer the research question by showing that augmentation effectiveness depends on the data type, downstream model, prediction task, and augmentation mechanism. The results also show that statistical realism metrics are useful for interpreting augmented data, but they should be considered together with predictive performance and residual diagnostics. Preserving marginal distributions or value

boundaries alone is not sufficient if the augmented data do not preserve task-relevant feature-target or temporal relationships.

The main contribution of this work is a controlled comparison of practical augmentation methods for tabular and time-series network-performance regression tasks relevant to the 6G context. The study provides evidence that augmentation should not be applied automatically, but should instead be evaluated using both downstream predictive utility and diagnostic measures of data realism.

Future work should evaluate additional datasets, including real 6G testbed data when available. It should also test different downstream models, such as tree-based models on flattened time-series windows, temporal convolutional networks, or transformer-based time-series models. Further research could include multiple random seeds, synthetic-only realism analysis, feature-target fidelity metrics, and more detailed temporal-dependency diagnostics. Finally, future work should investigate domain-specific augmentation methods that explicitly preserve network constraints and temporal relationships.

9 Responsible Research

9.1 Reproducibility

All experiments were implemented in Python using a Jupyter Notebook. The preprocessing steps, augmentation methods, model training procedures, evaluation metrics, and diagnostic exports were kept in a single executable workflow to make the experimental pipeline easier to inspect and rerun. The same train-test split strategy, feature selection, target variable, and downstream model configuration were used for the baseline and augmentation methods within each experimental track. The experiment can be found in our repository[1].

To improve reproducibility, random seeds were fixed where possible for NumPy, Python's random module, TensorFlow/Keras, and PyTorch. The main package versions were also recorded.

9.2 Generative AI usage

Supportive AI tools were used to enhance the quality and efficiency of this work. They were used to:

- debug code
- assist with grammar, spelling and structural improvement of this paper.

All experimental implementation, analysis of results and conclusion are the author's own.

References

[1] W. Saad, M. Bennis, and M. Chen, "A vision of 6g wireless systems: Applications, trends, technologies, and open research problems," 2019.

[2] A. Patil, S. Iyer, and R. J. Pandya, "A survey of machine learning algorithms for 6g wireless networks," 2022.

[3] M. Herlich and S. Farthofer, "Wireless communication data sets for machine learning," 01 2020.

[4] M. Günes, "On the scientific value of large-scale testbeds for wireless multi-hop networks," 2017.

[5] R. Singh, H. Kumar, and S. R.K, "Sampling based approaches to handle imbalances in network traffic dataset for machine learning techniques," in *Computer Science amp; Information Technology (CS amp; IT)*, WimoN 2013, p. 37–48, Academy Industry Research Collaboration Center (AIRCC), 2013.

[6] J. Wen, J. Kang, D. Niyato, Y. Zhang, J. Wang, B. Sikdar, and P. Zhang, "Generative ai for data augmentation in wireless networks: Analysis, applications, and case study," 11 2024.

[7] P. Gajjar, M. Tiwari, S. Seth, and V. K. Shah, "Llm-aug: Robust wireless data augmentation with in-context learning in large language models," 2026.

[8] M. H. L. d. Boni, I. Gervasio Sene Junior, and R. Martins da Costa, "Tabular data augmentation using artificial intelligence: A systematic review and taxonomic framework," *IEEE Access*, vol. 13, pp. 138950–138969, 2025.

[9] B. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *PLOS ONE*, vol. 16, p. e0254841, 07 2021.

[10] P. Branco, L. Torgo, and R. P. Ribeiro, "SMOGN: a pre-processing approach for imbalanced regression," in *Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications* (P. B. Luís Torgo and N. Moniz, eds.), vol. 74 of *Proceedings of Machine Learning Research*, pp. 36–50, PMLR, 22 Sep 2017.

[11] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," 06 2019.

[12] G. Iglesias, E. Talavera, González-Prieto, A. Mozo, and S. Gómez-Canaval, "Data augmentation techniques in time series domain: a survey and taxonomy," *Neural Computing and Applications*, vol. 35, pp. 1–23, 03 2023.

[13] K. Kamycki, T. Kapuscinski, and M. Oszust, "Data augmentation with suboptimal warping for time-series classification," *Sensors*, vol. 20, p. 98, 12 2019.

[14] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," 2018.

[15] K. Aggarwal and J. Srivastava, "Embarrassingly simple mixup for time-series," 2023.

[16] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, IJCAI-2021, p. 4653–4660, International Joint Conferences on Artificial Intelligence Organization, Aug. 2021.

[1] https://github.com/Qton42/AugmentationFor6GDatasets

[17] S. M. Xie*, A. Raghunathan*, F. Yang, J. C. Duchi, and P. Liang, "When covariate-shifted data augmentation increases test error and how to fix it," 2020.

[18] O. G. Serbetci, D. Burghal, and A. F. Molisch, "Wireless channel aware data augmentation methods for deep learning-based indoor localization," 2024.

[19] M. Patel, X. Wang, and S. Mao, "Data augmentation with conditional gan for automatic modulation classification," pp. 31–36, 07 2020.

[20] N. Kunz, "SMOGN: Synthetic minority over-sampling technique for regression with gaussian noise," 2020.

[21] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," in *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 399–410, Oct 2016.

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

[23] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, M. Li, J. Xie, M. Lin, Y. Geng, Y. Li, J. Yuan, and D. Cortes, *xgboost: Extreme Gradient Boosting*, 2026. R package version 3.3.0.0.

[24] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[25] F. J. Massey, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[26] DataCebo, Inc., *Synthetic Data Metrics*, 2023.

[27] M. Mekki, N. Toumi, and A. Ksentini, "Microservices configurations and the impact on the performance in cloud native environments," in *LCN 2022, 47th Annual IEEE Conference on Local Computer Networks, 26-29 September 2022, Edmonton, Canada* (IEEE, ed.), (Edmonton), 2022. 2022 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.