Enhancing Time Series Forecasting with Variational Autoencoder-based Data Augmentation and Temporal Pattern Analysis using Neural Networks*

Mukenze Ngalamulume Junior /s per 1st, Doohee Chung /s per 2nd
Handong Global University, Department of Advanced Convergence,
Handong Global University, School of Global Entrepreneurship and Information Communication Technology,
Pohang-si, South Korea
Email: fbj.mkz@gmail.com, profchung@handong.edu

Abstract

Time series forecasting is vital for various domains such as finance, energy, and economics, where accurate predictions are essential. However, forecasting accuracy is often hindered by insufficient or imbalanced datasets. This study focuses on improving time series forecasting through the use of data augmentation techniques, specifically leveraging Variational Autoencoders (VAE) to generate synthetic data that captures key temporal patterns. The synthetic data enriches the training dataset, helping predictive models generalize better.

We apply the VAE-based augmentation to Long Short-Term Memory (LSTM) networks and Temporal Fusion Transformer (TFT) models, chosen for their proven efficacy in time series forecasting. To further enhance the analysis of temporal dynamics, Fast Fourier Transformation Analysis is employed to decompose time series into frequency components, allowing for the assessment of the quality of synthetic data against the original, based on the index of frequency similarity; compares the frequency similarity between an original time series (Original) and several synthetic series generated by different VAE models (TimeVAE, BiLSTM-VAE, Conv1D-VAE, Dense-VAE), using the Fast Fourier Transform (FFT) and cosine and pattern similarity. Gaussian Mixture Models (GMM) allow clustering to be applied on an enriched dataset combining real and synthetic data (from a BILSTM-VAE model) to identify hidden profiles in financial time series (here, copper trading data).

Our experiments demonstrate that VAE-driven data augmentation with BiLSTM-VAE and Gaussian Mixture Model significantly improves the performance of LSTM and TFT models, offering more representative and diverse training samples.

Additionally, we explored several architectures and approaches that could lead to the injection of synthetic data into the network of a predictive model. This work contributes a focused and scalable framework for enhancing time series forecasting through advanced data augmentation and temporal pattern analysis, with direct applications in industries such as global raw material markets and energy forecasting, where precise predictive modeling is crucial.

Keywords

Variational Autoencoder (**VAE**), Long Short-Term Memory (**LSTM)**, Bidirectional Long Short-Term Memory (**BiLSTM),** Temporal Fusion Transformer (**TFT**), Fast Fourier Transform (**FFT**), Gaussian Mixture Models (**GMM)**.

1. INTRODUCTİON

The field of time series prediction is vast and involves many facets of life and society. This broad range of application areas has allowed for numerous experiments over time, and multiple experts and researchers have come to the same conclusion: that when data are few and, more importantly, have little variability, it may affect the series' prediction. This has given rise to several academic and scientific studies aimed at improving the performance resulting from this process of predicting a time series, whether in the fields of finance and commerce or electricity and energy.

One of the numerous approaches being investigated and used by the scientific community is the use of data augmentation to target various issues that may affect performance concerning the forecasting of a time series. The augmentation of data may be used as a regularization strategy to address the overfitting problems in neural networks. Taking this issue into consideration while acknowledging that we are well-positioned in the case of chronological series, this gap can be filled by addressing the optimization problem at two levels connected by a two-step process: initial training of a model that is not expanded for a limited number of times, followed by an iterative division procedure. (Nochumsohn, L., & Azencot, O., 2025)

It is possible to divide data augmentation techniques into two groups: traditional approaches that modify input data by transformations such as noise injection, return, and reframing. Furthermore, advanced data augmentation techniques consistently use generative models, such as the Auto-Encoder family, which includes Auto-Encoder Variational, and Generative Adversarial Networks (GANs), such as the Deep Convolutional Generative Adversarial Network DC-GAN, to generate new data instances. These examples are brand-new products that were created entirely using the distributions of known data. (Islam, T., Hafiz, M. S., Jim, J. R., Kabir, M. M., & Mridha, M. F., 2024)

Considering this, our work involves using data augmentation based on a VAE architecture to enhance Time Series forecasting using neural networks. The following is a summary of the main contributions of our work:

1. We developed a novel approach to data augmentation using a Variational Autoencoder (VAE) equipped with BiLSTM layers. The VAE captures a compact representation of the data within a low-dimensional latent space and subsequently reconstructs the input data from this condensed form. This compressed representation is optimized to generate new data points. This method enables subtle data creation, yielding notable benefits in terms of enhancing the diversity of datasets.

2. The Fast Fourier Transform is introduced as a guide for selecting the quality of synthetic data against the original one.

3. The synthetic data generated through this process is then combined with the original data, passed through a Gaussian mixture model for capturing hidden profiles via clustering, and fed into a recurrent network with LSTM layers predicting each cluster, capturing hidden profiles and temporal dependencies. This network processes two inputs: one for the synthetic data and another for the original data, ultimately predicting the target variable.

4. Additionally, we have implemented a Transformer-based network that utilizes the same methodology, specifically employing the Time Fusion Transformer (TFT), to check the effectiveness of data augmentation on transformer architecture.

5. Our approach has been tested using data related to the evolution of commodity prices, such as copper. However, this methodology can be readily extended to forecast other commodities, including natural gas, uranium, steel, and oil, by incorporating technical indicators such as the Moving Average Convergence-Divergence (MACD), Exponential Moving Average (EMA), and Bollinger Bands for future research.

Although individual components of this framework exist in the literature, their integration and adaptation for time series forecasting, particularly in the context of commodity price prediction, had not been explored in this manner before.

2. RELATED WORK

The scientific community in machine learning acknowledges that time series forecasting is a fundamental challenge across various domains, including finance, retail, energy consumption, meteorology, predictive maintenance, and inventory management. State-of-the-art machine learning algorithms require a large volume of high-quality data. However, limited sample sizes and imbalanced data distributions across different class labels often hinder model performance. Such limitations can significantly degrade the predictive capability of machine learning models. Data augmentation has emerged as a promising solution to enhance data quality and improve model generalization (Szlobodnyik, G., & Farkas, L. , 2021).

Historically, time series forecasting (TSF) has been performed using econometric models, such as the Auto-Regressive Integrated Moving Average (ARIMA), due to their strong theoretical foundation and efficiency. However, these classical approaches present several limitations, including data integrity issues, assumptions of linearity, fixed temporal dependence, limited forecasting horizons, and univariate modeling constraints. In contrast, machine learning (ML) techniques, particularly deep learning (DL) models, offer a compelling alternative. They can effectively address these challenges, particularly with their ability to process multiple input variables, capture nonlinear dependencies, and handle missing data. These characteristics make them particularly valuable for multi-step forecasting and scenarios involving nonlinear relationships between input and output variables, highlighting the advantages of machine learning over traditional econometric models in forecasting (Ouyang, Z., 2023).

Recently, there has been a growing interest in exploring data augmentation techniques for time series and deep learning applications. This interest is underscored by the significant findings of (Wen, 2020), who note that data augmentation in time series forecasting is evolving across multiple directions. One significant approach involves transformations in the time-frequency domain, where spectral analysis techniques, such as the Fourier and wavelet transforms, are used to modify and enhance input data. Another essential focus is the augmentation of imbalanced classes, addressing challenges posed by skewed data distributions that often hinder model performance. Researchers have also emphasized the importance of selecting and combining different augmentation strategies, ensuring that generated samples contribute meaningfully to model generalization rather than introducing noise or redundancy.
Additionally, Gaussian process-based augmentation methods have been explored to generate synthetic data while maintaining statistical consistency with the original dataset. Lastly, deep generative models, such as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), have gained traction as powerful tools for creating realistic synthetic time series data, enabling robust and scalable learning models. These advancements collectively improve predictive accuracy and reliability in time series forecasting and deep learning applications (Wen, 2020).

One of the most straightforward techniques involves time-domain transformations, which directly manipulate the original time series data. Methods such as Gaussian noise injection, noise pattern generation, and window cropping modify the temporal structure of the data, helping models generalize better. In addition, some studies have explored augmentation techniques in the frequency domain, where amplitude and phase spectrum perturbations introduce variations that enhance the model's ability to detect anomalies. As recent studies show, this approach has been efficient in time series anomaly detection (Wen, 2020). Beyond time-domain transformations, time-frequency analysis has emerged as a valuable augmentation technique, particularly for deep neural networks. Researchers can use the Fourier Transform to generate time-frequency features from sensor-based time series data. These transformed representations can then be augmented, improving the ability of models to recognize human activity patterns and other complex temporal behaviors (Steven Eyobu, O., & Han, D. S. , 2018).

One more adopted method is decomposition-based augmentation, which has proven particularly useful in time series forecasting and anomaly detection. This approach breaks down time series data into distinct components, allowing models to learn patterns more effectively. Techniques such as Seasonal-Trend decomposition using Loss (STL) (Cleveland, 1990) and Robust-STL decomposition (Wen, Q., Gao, J., Song, X., & al., 2019) extract fundamental structures from the data, helping to separate trend, seasonality, and residual noise. Gaussian mixture models (Cao, H., Tan, V. Y., & Pang, J. Z. , 2014) have enhanced classification performance, particularly in scenarios with imbalanced class distributions. By decomposing time series into trend (Tt), seasonality (St), and residual components (rt), these methods facilitate better model generalization and improve performance in classification tasks where minority classes are underrepresented (Qingsong Wen et al., 2022). These advancements in data augmentation continue to drive improvements in deep learning-based time series forecasting, making models more robust and adaptable to real-world applications.

Another powerful technique in time series augmentation is Dynamic Time Warping (DTW), a widely used method for measuring similarity between time series by aligning sequences non-linearly. A particularly effective variation of DTW, Guided Warping, has been developed to generate realistic synthetic time series by aligning a "student" time series with a "teacher" reference sequence. Unlike purely random transformations, this method ensures that the original feature values are preserved while introducing meaningful variations. By maintaining the structural integrity of the data while augmenting its diversity, Guided Warping has proven to be a valuable approach for enhancing dataset variability and improving predictive model generalization, making it especially useful for time series forecasting tasks. In deep learning, time series data augmentation is crucial in improving model generalization and data diversity (Iwana, B. K., & Uchida, S., 2021). Brian Kenji and Seiichi Uchida emphasize that most time series augmentation techniques rely on random transformations applied directly to the data. These transformations include jittering, which introduces random noise to perturb the original signal, and rotation, which flips univariate time series or rotates multivariate data. Slicing, another common technique, involves cropping time series segments, while permutation rearranges data slices to introduce variability. Additionally, scaling modifies the magnitude of patterns within the series, whereas time warping deforms time steps non-linearly to create variations in the sequence's temporal structure.

Beyond general applications in deep learning, data augmentation has proven particularly valuable in sensor-based applications, where models rely on time series data for health monitoring and activity recognition. For instance, Um et al. (2017) applied a combination of jittering, slicing, permutation, magnitude warping, and time warping to enhance Parkinson's disease monitoring using deep temporal convolutional neural networks (CNNs). These augmentation techniques helped improve model performance by providing more diverse and representative training data (Um et al., Data Augmentation of Wearable Sensor Data for Parkinson's Disease Monitoring Using CNNs, 2017). Similarly, Rachid and Louis (2019) utilized time-warping augmentation in conjunction with long short-term memory (LSTM) networks to recognize construction equipment activities, demonstrating the effectiveness of augmentation in real-world industrial applications (Rashid, K. M., & Louis, J. , 2019).

Among the most adopted generative approaches are Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), both of which have demonstrated remarkable success in creating synthetic yet highly representative datasets. One notable advancement in this field is the Recurrent Generative Adversarial Network (RCGAN), introduced by Esteban et al. (2017). These models integrate recurrent neural networks (RNNs) into the generator and discriminator, producing realistic real-valued time series data. By leveraging the sequential nature of RNNs, RCGAN effectively captures temporal dependencies, making it particularly valuable for applications such as medical time series generation (Esteban, C., Hyland, S. L., & Rätsch, G. , 2017).

Variational Autoencoders (VAEs) have been widely explored for synthetic time series generation. VAEs operate by learning a latent distribution (g(z|x)), where the encoder maps input data into a probabilistic latent space, and the decoder reconstructs the original sequence from this compressed representation. Unlike traditional autoencoders, VAEs introduce stochasticity in their encoding process, allowing the generation of diverse and novel data samples. Maxim Goubeaud et al. (2021) demonstrated that VAEs can effectively generate synthetic time series data using fully connected layers with ReLU activations, further highlighting their potential in augmenting datasets for deep learning models (Goubeaud, 2021).

These advancements in time series augmentation techniques continue to push the boundaries of deep learning-based forecasting and classification, enabling models to achieve higher accuracy, improved generalization, and better adaptability to real-world scenarios. By leveraging these deep generative approaches, researchers and practitioners can enhance training datasets, mitigate data scarcity issues, and improve machine learning models' robustness, particularly in domains where high-quality labeled data is limited.

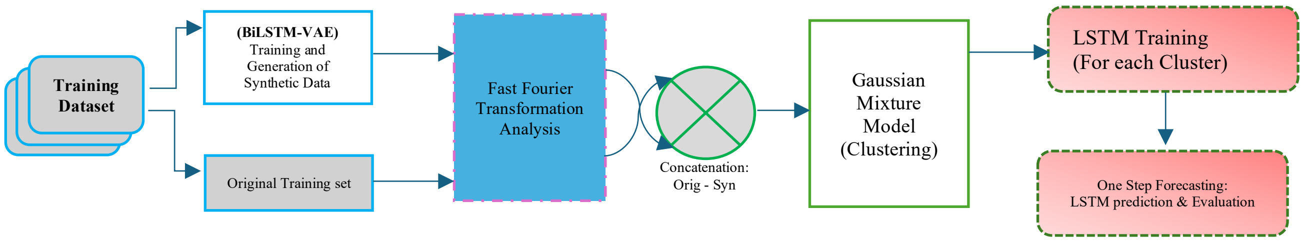
Let us now address another key aspect: demand forecasting. As outlined in the scientific work titled “Methodology for Improving the Performance of Demand Forecasting Through Machine Learning”, accurate demand forecasting is a critical strategic lever for companies seeking to enhance decision-making and strengthen their competitive advantage. In this context, the authors, including Doohee Chung et al., proposed an innovative approach based on a hybrid model that combines the strengths of K-means clustering, LASSO-based feature selection, and sequential temporal prediction using LSTM neural networks. This integrated method addresses key limitations such as uncertainty, complexity, non-linearity in demand behavior, and the lack of or bias in historical data. It enables intelligent data segmentation (via K-means), selection of the most relevant features (LASSO), and capture of complex temporal dependencies (LSTM) (Chung, D, Park, S., Song, Y., Chae, J., &. Yoon, D. , 2023).

This methodology directly inspires us in exploring how to inject synthetic data, generated by a dedicated data generation component, into a strengthened forecasting pipeline.

3. METHODOLOGIES

3.1. The general frameworks

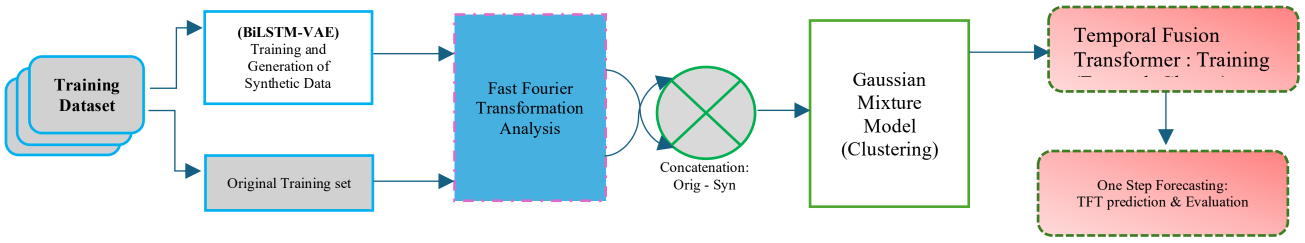
Approach 1: Proposed



Training Dataset
(BiLSTM-VAE) Training and Generation of Synthetic Data
Original Training set
Fast Fourier Transformation Analysis
Concatenation: Orig - Syn
Gaussian Mixture Model (Clustering)
LSTM Training (For each Cluster)
One Step Forecasting: LSTM prediction & Evaluation

Figure 1: The general framework proposed with four phases: Data generation, Assessment of synthetic data quality, data clustering, Predictive phase (Approach 1: Recurrent Network)

Approach 2: Experimented



Training Dataset
(BiLSTM-VAE) Training and Generation of Synthetic Data
Original Training set
Fast Fourier Transformation Analysis
Concatenation: Orig - Syn
Gaussian Mixture Model (Clustering)
Temporal Fusion Transformer : Training
One Step Forecasting: TFT prediction & Evaluation

Figure 2: The general framework proposed with four phases: Data generation, Assessment of synthetic data quality, data clustering, Predictive phase (Approach 2: Transformer Network)

In the proposed framework illustrated in Figure 1, the hybrid model is structured into four key phases. The first phase involves the generation of synthetic data using a Variational Autoencoder based on BiLSTM (VAE-BiLSTM), where the encoder consists of bidirectional LSTM layers and the decoder employs standard LSTM layers. The second phase focuses on the evaluation of the quality of the generated synthetic data to ensure its validity and representativeness (FFT). In the third phase, the original and synthetic data are merged and subjected to clustering, using a Gaussian Mixture Model (GMM) to identify latent structures within the dataset. Finally, the fourth phase consists of forecasting within each identified cluster, allowing for more targeted and accurate predictions.

In the second approach (Figure 2), the same sequential logic is applied; however, the principal distinction lies in the substitution of the prediction model. Instead of classical LSTM architectures, this approach leverages a Transformer-based network, specifically the Temporal Fusion Transformer (TFT), which offers enhanced capabilities in modeling complex temporal dependencies and variable importance for multivariate time series forecasting.

3.2. TimeVAE to BiLSTM-VAE Architecture

It is important to recall that the primary objective of a Variational Autoencoder (VAE), as illustrated in Figure 3, is to generate new, realistic data by learning an approximation of the underlying probability distribution of the input data p(x). This process consists of two main stages. In the first stage, a latent vector is generated by sampling from a predefined prior distribution, typically a standard Gaussian distribution z~p(z). In the second stage, a new data point is generated conditionally from this latent representation, following the conditional distribution x~p(x|z).

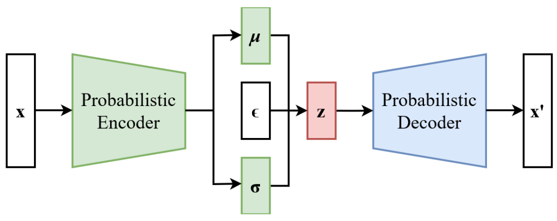


Figure 3: VAE Architecture, Source: https://commons.wikimedia.org/wiki/File:Reparameterized_Variational_Autoencoder.png

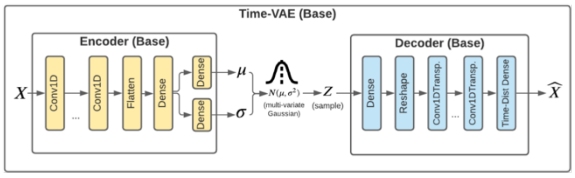


Figure 4: TimeVAE Architecture, Abhyuday Desai & al. , 2021,"TIMEVAE: A Variational Auto-Encoder For Multivariate Time Series Generation"

Unlike traditional autoencoders, which learn a deterministic latent representation as a fixed point, in Figure 4 we can see how the Variational Autoencoder (VAE) produces a probabilistic latent distribution q(z|x) that approximates the true posterior p(z|x). In this context, the Time-VAE architecture, introduced by Abhyuday Desai in "TimeVAE: A Variational Auto-Encoder for Multivariate Time Series Generation", is specifically designed for generating multivariate time series data. The encoder processes an input sequence X through a 1D convolutional layer (Conv1D) to extract local temporal patterns. This is followed by a Flatten operation and a Dense layer that encodes more abstract features (Desai, A., Freeman, C., Wang, Z., & Beaver, I. , 2021).

Subsequently, two separate Dense layers output the parameters of the latent distribution: the mean μ and the log-variance log(σ2). The latent vector zzz is sampled from a normal distribution N(μ,σ^2) using the Reparameterization Trick, which preserves model differentiability during training. The sampling is defined as:

$$z \;=\; \mu + \sigma\text{^}2.\,\varepsilon, ou\ \varepsilon \sim N(0, I) \qquad \textit{Eq (1)}$$

In the decoding phase, zzz is first passed through a Dense + Reshape layer, then processed by a Conv1DTranspose layer to reconstruct the temporal structure. The output sequence is finally generated using a TimeDistributed Dense layer. The entire architecture is trained using the VAE loss function, which combines a reconstruction loss (e.g., MSE) with a Kullback-Leibler (KL) divergence term as regularization.

$$\mathcal{L}_{\theta,\phi}\ VAE(X, X\_hat) = -\ E_{q,\phi(z|x)}[logp_\theta(x \mid z)]\ +\ D_{KL}(q_\phi(z \mid x)\ ||\ p_\theta(z)) \quad \textit{Eq. (2)}$$

The term « $-\log_{p_\theta}(x|z)$ » quantifies how accurately the reconstructed output x-hat matches the original input X. Its objective is to ensure that the decoder's output is as close as possible to the true input. The second term, $D_{KL}(\mathrm{q}_\emptyset(z|x)\ ||\ \mathrm{p}_\theta(z))$, measures the Kullback-Leibler divergence between the learned latent distribution $\mathrm{q}_\emptyset(z|x)$ and the prior distribution $\mathrm{p}_\theta(z)$.. This term encourages the latent space to be continuous and well-regularized, typically aligning it with a standard normal distribution.

It is also important to note that the input data X must be represented as 3-dimensional tensors, i.e., 3D : X $\epsilon\ \mathbb{R}^{N \times T \times D}$, where: N is the batch size, T is the number of time steps, D is the number of features or observed variables.

Building upon the original TimeVAE architecture, we retained its overall structure while introducing key modifications to develop our own variant, named VariationalAutoencoderBiLSTM. This model is a type of Variational Autoencoder (VAE) specifically designed to handle multivariate time series data, with a particular emphasis on modeling temporal dependencies through the integration of Bidirectional LSTM (BiLSTM) layers within the encoder.

The architecture is composed of two main components in Figure 5. First, the probabilistic encoder receives an input sequence of shape (seq_len,feat_dim) and processes it through one or more BiLSTM layers, enabling the model to capture both forward and backward temporal dependencies. The encoded output is then projected into a latent space using two parallel dense layers that compute the mean and log-variance (z_log_var) of the latent distribution. A sampling layer is employed to draw latent vectors from the resulting Gaussian distribution, following the standard reparameterization trick, which ensures differentiability during training.

Second, the decoder reconstructs the original sequence from the sampled latent vector z. It begins by replicating the latent vector across the temporal dimension using a RepeatVector layer. The repeated sequence is then processed by a stack of LSTM layers (mirroring the encoder but in reverse order), producing a sequential output. A final TimeDistributed(Dense) layer maps each timestep back to the original feature space, generating a prediction at each point in the sequence.

The model inherits from a base class, BaseVariationalAutoencoder, providing a modular and extensible structure that supports training, saving, and inference functionalities in a consistent manner.

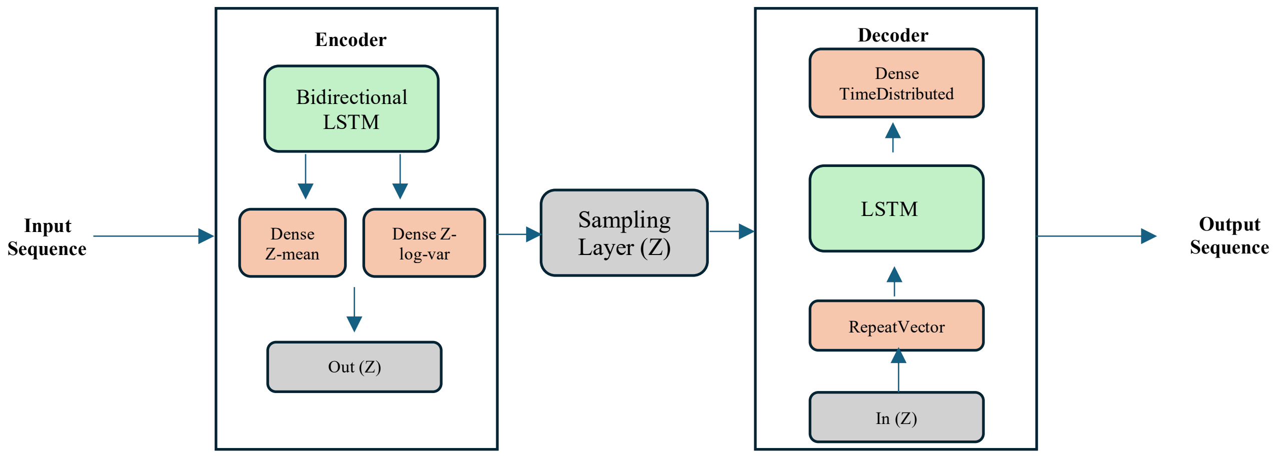


Figure 5: Architecture BiLSTM-VAE

An advanced version of the VariationalAutoencoderBiLSTM, similar to TimeVAE, can be extended to include time series-specific layers that capture polynomial trends and seasonal patterns. Such architectural enhancements would contribute to making the time series generation process more interpretable. Finally, the BiLSTM-VAE model is trained using the Evidence Lower Bound (ELBO) loss function, where a weighting factor λ is applied to the reconstruction loss to balance it against the regularization term:

$$L_{biLTSM\_VAE} = \lambda . Reconstruction\ Loss + KL\ Divergence \quad \textit{Eq. (3)}$$

3.3. Fast Fourier Transformation (FFT)

A promising deep learning paradigm for time series analysis has been demonstrated by the quick exploration and exploitation of Fourier transforms for time series analysis, such as efficiency and global view. The Fourier transform makes it easier to obtain frequency spectrums with a global view of time series, which is useful for collecting global aspects of time series, and plentiful periodic information for time series analysis, such as seasonal patterns. The Fourier Transform's benefits include efficiency, sparse representation, global perspective, and decomposition. The original time series can be broken down into several frequency components using the Fourier transform, which can reveal important details about the time series, like trends and seasonal patterns.

Specifically, DWT can produce multi-scale representations by breaking down a time series into a collection of sub-series with frequencies arranged from high to low. While a long-term analysis concentrates more on the low-frequency components, a short-term time series analysis typically relies on high-frequency patterns. Finding and obtaining useful information for time series analysis is naturally aided by the Fourier transform, which breaks down time series into multi-frequency components (Yi, 2023).

However, as mentioned in their article titled "*An Algorithm for the Machine Calculation of Complex Fourier Series", Cooley, J. W., & Tukey, J. W. (1965)* stated that the objective of the Fast Fourier Transform (FFT) is to efficiently compute the frequency components X(j) of a sequence of length N, by leveraging the signal coefficients in the time domain and the N-th roots of unity. The algorithm exploits the symmetry and periodicity properties of these roots to reduce computational complexity (Cooley, J. W., & Tukey, J. W. (). , 1965). First, if N = r1·r2, the FFT algorithm reorganizes the indices into two components (e.g., rows and columns) and decomposes the calculation into two smaller Fourier transforms—this is the factorization step of N. Second, by recursively repeating this decomposition m times such that N = r1·r2·····r_m, the total number of operations becomes significantly reduced:

$$T(N) = N\cdot(r1+r2+\cdots+rm) \approx O(NlogN) \quad \textit{Eq. (4)}$$

This leads to a total computational complexity of approximately O(NlogN), which is a major improvement over the O(N2) operations required by the naive Discrete Fourier Transform (DFT).

Hence,

$$X(j) = \sum_{k=0}^{N-1} A(k)\ .W^{jk},\ with\ W = e^{-2\pi i/N} \quad \textit{Eq. (5): Frequency components}$$

where:

A(k) are the signal coefficients in the time domain
X(j) are the frequency components
W is the N-th root of unity

3.4. Gaussian Mixture Models (GMM)

The Gaussian Mixture Model (GMM) is a probabilistic model that assumes data are generated from a combination (or "mixture") of multiple normal (Gaussian) distributions. Each Gaussian component represents a latent subgroup within the data. The probability density function of a GMM is defined as (Reynolds, D. , 2015):

$$p(x) = \sum_{k=1}^{K} w_k \, N(x|\mu_k, \Sigma_k) \quad \textit{Eq. (6)}$$

Where:

- K is the number of components (clusters),
- wk are the weights (which sum to 1),
- μk is the mean of the kth component,
- Σk is the covariance matrix of the kth component,
- N(x|μk, Σk) is the multivariate Gaussian (normal) density function.

Unlike methods such as K-means, which assume spherical clusters of equal size, the Gaussian Mixture Model (GMM) offers several advantages: It captures the variance and correlation between dimensions through the covariance matrix Σk; It is probabilistic; assigning a probability of membership to each cluster for every data point; It is more flexible in modeling clusters with elongated, elliptical shapes or varying sizes.

3.5. Long Short-Term Memory (LSTM)

In the context of time series forecasting, Recurrent Neural Networks (RNNs) have been widely adopted due to their ability to model temporal dependencies. However, traditional RNNs face significant limitations in capturing long-term dependencies, as they suffer from vanishing or exploding gradient problems. To address this issue, Hochreiter and Schmidhuber (1997) introduced the Long Short-Term Memory (LSTM) architecture, which incorporates gating mechanisms that regulate the flow of information and preserve memory over longer sequences. (Hochreiter, S., & Schmidhuber, J. , 1997)

The LSTM architecture is composed of three primary gates: the forget gate, the input gate, and the output gate. The forget gate determines the extent to which the previous cell state ct−1 is retained, calculated by applying a sigmoid activation to a weighted sum of the current input x_t and the previous hidden state ht−1. The output value of this gate ranges between 0 and 1, thereby controlling how much information is discarded or preserved. The input gate decides which components of the new information should be added to the cell state. It combines a sigmoid gate and a hyperbolic tangent function to regulate the update. The cell state is then updated using both the retained information from the previous state and the new candidate values. Finally, the output gate produces the hidden state ht, which is passed to the next time step and is also used as the model output at that time step. This is achieved by applying a sigmoid function to the current input and previous hidden state, and then modulating the updated cell state passed through a tanh activation function.
The mathematical formulation of the LSTM unit is given by:

Eq. (7)

$$\begin{aligned}
i_t &= \sigma\,(W_{ih}\, h_{t-1} + W_{ix}\, xt + b_i) \\
c\hat{}_t &= \tanh\,(Wc_h\, h_{t-1} + W_{cx}\, x_t + b_c) \\
c_t &= c_{t-1} + i_t \cdot c_t \\
o_t &= \sigma\,(W_{ho}\, ht-1 + W_{ox}\, x_t + b_o) \\
h_t &= o_t \cdot \tanh\,(c_t)
\end{aligned}$$

In this study, we apply the LSTM architecture to a cluster-based time series forecasting task. The goal is to model and predict the Close variable for each subgroup identified through a Gaussian Mixture Model (GMM) clustering process. For each cluster, a separate LSTM model is trained using sequences of historical values. The training loop includes a validation split and tracks the learning history for each cluster, enabling performance monitoring and model comparison across different subgroups of the dataset. This cluster-wise training strategy allows the LSTM models to better specialize and capture the intra-cluster temporal dynamics, ultimately enhancing the forecasting accuracy for heterogeneous time series data. Such an approach demonstrates the practical effectiveness of LSTM networks when combined with unsupervised learning techniques like GMM for time series segmentation.

3.6. Temporal Fusion Transformer (TFT)

The Temporal Fusion Transformer (TFT) is a Transformer-based model that leverages self-attention to capture the complex temporal dynamics of multiple time sequences. It supports multiple time series in a set of multivariate time series. Moreover, TFT allows a model to output multi-step predictions of one or more target variables including prediction intervals. It can be accountable for many types of features and the predictions can be interpreted in terms of variable importance and seasonality (Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. , 2021).

The Temporal Fusion Transformer (TFT) is designed to forecast future values by incorporating a rich set of inputs that capture both temporal dynamics and static context. Specifically, it leverages three categories of input features: (1) past target values y observed within a fixed look-back window of length k (2) time-varying exogenous variables, which include both known inputs x (such as calendar information or planned events) and a priori unknown inputs z (such as future covariates or control variables), and (3) static covariates s, which encode contextual metadata about the observed entities and remain invariant over time. By integrating these heterogeneous inputs, TFT is able to model complex temporal relationships and dependencies. Those above four components can be combined into the model as (Huy, 2022) :

$$yt = T_t + S_t + C_t + I_t \qquad Eq. (8)$$

where:

- y_t : is the observed value at time t.
- T_t : is the trend component at time t.
- S_t : is the seasonal component at time t.
- C_t : is the period component at time t.
- I_t : random component at time t.

The implementation of the Temporal Fusion Transformer (TFT) requires input data to be structured in the Extended Time-Series format, which is specifically designed to accommodate a wide range of temporal and static features. The model distinguishes between various variable types, including time-varying known variables (e.g., weather forecasts or calendar events), time-varying unknown variables (e.g., sales or stock prices), static covariates (e.g., product category or location), and the target variable to be predicted. At the core of the TFT architecture are four principal components that together support both model complexity and interpretability. First, the model employs Gated Residual Networks (GRNs) in conjunction with Gated Linear Units (GLUs), forming a gating mechanism that allows for adaptive feature learning.

This mechanism enables the model to dynamically bypass or activate layers based on the complexity of the data, thus improving computational efficiency and generalization. Second, the Variable Selection Network (VSN) is used to automatically identify the most relevant features at each time step. Each variable is processed through its own GRN and assigned a selection weight, which allows the model to capture context-aware feature importance in a transparent manner. The third component, the Static Covariate Encoder, integrates time-invariant features using dedicated GRNs that generate context vectors. These vectors are subsequently used to enhance temporal feature representations, guide variable selection, and initialize the sequential decoder. The fourth component is the decoder itself, which combines several layers to capture temporal patterns. It includes a sequence-to-sequence layer for modeling local dynamics, a static enrichment layer for injecting static feature context into each step, a multi-head attention mechanism for long-term dependency modeling, and a feedforward network with gated residuals to balance model expressiveness and interpretability (Lim, 2021).

Unlike traditional point prediction models, TFT produces probabilistic forecasts by estimating prediction intervals through quantile regression. This approach enables the model to express uncertainty in its predictions by providing upper and lower bounds for each forecast step. The total training loss is computed as the aggregated quantile loss across all time steps and quantiles, optimizing the model for accurate and robust multi-horizon forecasting.

$$\mathcal{L}(y, y\_hat) = \frac{1}{N} \sum_{i=1}^{N} \sum_{q \in Q} \sum_{t=1}^{T} L_q(y_{i,t}, y\_hat_{i,t}^{(q)}) \qquad Eq. (9)$$

$$\mathcal{L}_q(y, y\_hat) = max\,(q(y, y\_hat), (q-1)(y - h\_hat)) \qquad Eq. (10)$$

Where:

- L is the total loss,
- N is the number of samples (e.g., batch size),
- Q is the set of quantiles (e.g., {0.1, 0.5, 0.9}),
- T is the prediction horizon (number of time steps),
- $y_{i,t}$ is the true value for sample i at time step t,
- $y\_hat_{i,t}^{(q)}$ is the predicted value at quantile q,
- L_q is the quantile loss function

3.7. Dataset

We selected secondary data from the website https://tradingnomics.com, specifically focusing on the data for Copper. Trading Economics provides accurate information for 196 countries, including historical data and forecasts for over 20 million economic indicators, exchange rates, stock indices, government bond yields, and commodity prices. The data for economic indicators are based on official sources, not third-party data providers, and the facts are regularly checked for inconsistencies. So far, the suggested approach can be applied to data that has the same scope as NaturalGas, Uranium, Cooking Coal, and Crude Oil.

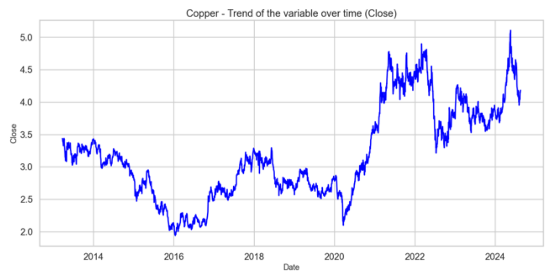


Figure 6: Copper – Close value trading Trend

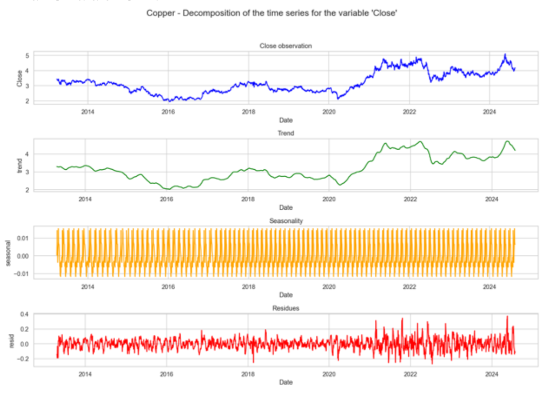


Figure 7: Copper – Close value Decomposition of Series

In the selected dataset on Copper, we chose to do our experiments on the prediction of the closing values of the Stock Market as the target feature (Close). Figure 6 shows the closing values of the market from 2010 to 2024. In Figure 7, the decomposition of the series into Trend and Seasonality reveals that the closing values of the Copper market have evolved with an upward trend. The data is separated into two parts, with a total of 2834 observations. We allocate 80% (2267 observations) to the Train Dataset and 20% (567 observations) to the Test Dataset.

3.8. Hyperparameters Settings

The configuration parameters provided define the architectural and training settings for four different Variational Autoencoder (VAE) variants: `timeVAE`, `vae_dense`, `vae_conv`, and `vae_bilstm`. All models share common hyperparameters, including a latent dimension size of 8 (`latent_dim: 8`), a three-layer architecture with hidden layer sizes of 50, 100, and 200 units respectively (`hidden_layer_sizes: [50, 100, 200]`), and a reconstruction loss weighting factor of 3.0 (`reconstruction_wt: 3.0`), which emphasizes the importance of accurate input reconstruction during training. The training batch size is uniformly set to 16 across all models.

The LSTM model is configured to predict a target time series value using a sliding window of 30 time steps (`time_step=30`). It consists of two successive LSTM layers, the first with `64 units` and the second with `32 units`, followed by a final Dense layer with a single output. The model is trained using the `Adam optimizer` for `50 epochs`, with the mean squared error (`MSE`) as the loss function. The dataset is split into `80%` for training and `20%` for testing using the `train_test_split` method, with `random_state=42` to ensure reproducibility. The model is trained independently for each cluster identified in the data (`Cluster_Label`), allowing predictions to be specialized according to subgroups. Input data is reshaped into a 3D tensor format (samples, time steps, features), with only one input variable (`Average_Close`). The batch size defaults to Keras's standard value, typically set to 32.

In the Fast Fourier Transform (FFT) implementation, the primary hyperparameters are minimal and relate mainly to the signal configuration and sampling. The length of the time series, denoted as N, defines the number of data points used in the FFT. The sampling interval T is set to 1, assuming a daily frequency in the original signal. The frequency components are calculated using fftfreq (N, T), which produces the corresponding frequency bins up to the Nyquist frequency (N/2). The spectral amplitudes are obtained by applying the FFT to each time series (original and synthetic) and retaining only the first half of the spectrum via np.abs(fft(signal))[:N // 2], reflecting the positive frequencies.

In the implementation of Gaussian Mixture Model (GMM) employed to identify the optimal number of latent clusters within the combine dataset, the model is evaluated for a range of components $n \in [1,9]$, where each value of n represents a potential number of Gaussian distributions used to model the data. For each configuration, the GMM is initialized with a fixed random_state=42 to ensure reproducibility. The performance of each model is quantitatively assessed using two model selection criteria: the Bayesian Information Criterion (BIC) and the Akaike Information Criterion (AIC). These criteria measure the goodness-of-fit while penalizing model complexity.

In implementation of the Temporal Fusion Transformer (TFT), the model is trained for multivariate time series forecasting on a dataset structured according to the extended time series format. The batch size is set to 64, with a model hidden size of 160 units and 4 attention heads to capture temporal dependencies via multi-head self-attention. A learning rate of 0.001 is employed alongside a gradient clipping value of 0.1 to ensure stable training dynamics and mitigate exploding gradients.

The model is trained on sequences with a maximum encoder length of 90 and a prediction horizon of 30 time steps. The training and validation datasets are created using the TimeSeriesDataSet API from PyTorch Forecasting, including support for static categorical features (Cluster_Label), time-varying known reals (year, month, day, weekday), and time-varying unknown reals (Open, High, Low). The target variable Close is normalized using a robust group normalizer.

The training is conducted on a single GPU for a maximum of 45 epochs using the Ranger optimizer. The loss function is the Root Mean Square Error (RMSE), and the training process includes callbacks for early stopping (patience = 5) and learning rate monitoring. Logging is handled by TensorBoard for experiment tracking and visualization.

4. RESULTS AND DISCUSSION

4.1. Results

Data Generation with VAE

The data generation trained are from four different architectures of the Variational Autoencoder (VAE) model, namely: TimeVAE, VAE_BiLSTM, VAE_Conv, and VAE_Dense. Most of these models exhibited a rapid reduction in total loss at the beginning of training, indicating that they quickly learned from the data. All models demonstrated good performance in reconstructing the input data.

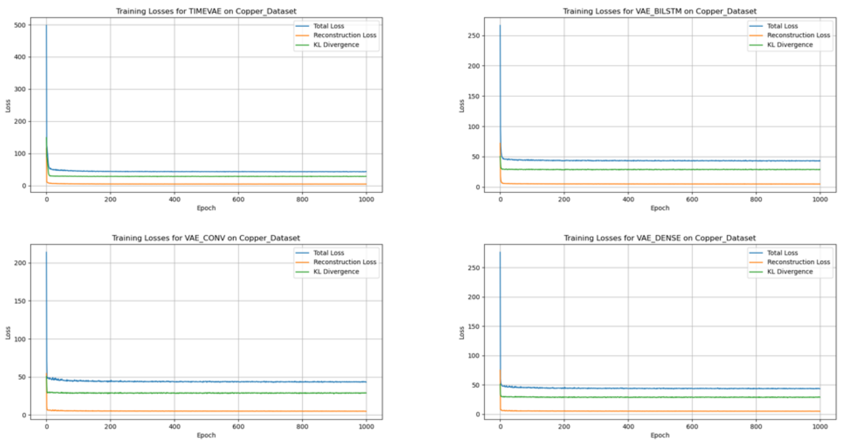


Figure 8: Training Loss TimeVAE, VAE_BiLSTM, VAE_Conv, and VAE_Dense

Although the KL divergence in Figure 8 decreases during training, it tends to stabilize after a few epochs. This behavior suggests that the models progressively learn to optimize the latent distribution but no longer significantly alter their internal representations as training progresses.

The following figures display 2D projections of the latent representations learned by each VAE model after training in Figure 9, using the t-SNE (t-Distributed Stochastic Neighbor Embedding) dimensionality reduction technique. In each plot, red points correspond to the original input data, while blue points represent the generated samples (also referred to as the “prior”), which are either produced by the model before training or after applying regularization.

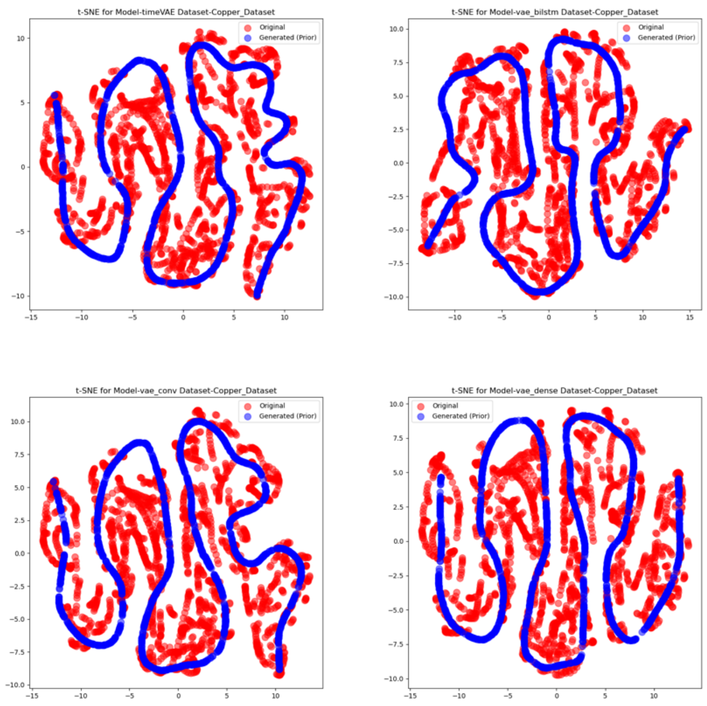


Figure 9: t-SNE visualization TimeVAE, VAE_BiLSTM, VAE_Conv, and VAE_Dense

The t-SNE visualization of the TimeVAE in Figure 9 reveals a clear separation between original data points and generated samples (priors). The blue points (generated samples) are tightly clustered around the red points (original data), indicating well-structured latent representations. This suggests that TimeVAE is capable of generating latent representations that are closely aligned with real data in the latent space—an indication of strong generalization ability. In contrast, while the BiLSTM-based VAE also shows a good separation between generated and original points, the clusters are less dense compared to TimeVAE. Although the generated samples remain relatively close to the originals, their latent representations appear less coherent. This observation prompted a deeper investigation through Fourier Transform analysis.

Furthermore, the t-SNE plot of the convolutional VAE (VAE_CONV) exhibits a clear separation between original and generated data points; however, the generated samples are more dispersed relative to the originals. This dispersion may imply that VAE_CONV struggles to capture the full complexity of the real data with the same precision as TimeVAE or VAE_BiLSTM. The broader spread of the generated samples could also indicate that the latent representations are more diverse or less regular. Similarly, the dense VAE (VAE_DENSE) shows a relatively clear separation between the two data types, but the generated points are slightly farther from the originals when compared to other models. The blue points in this case appear more widely distributed in latent space, suggesting that VAE_DENSE may face greater difficulty in capturing the true underlying structure of the real data.

From these observations, we conclude that evaluating the quality of synthetic data requires a more comprehensive approach. This approach should go beyond the visual analysis possible with the visualization of synthetic data generated by different VAE architectures, as presented in Figure 10, and assess the spectral similarity between the generated and original time series. This provides the rationale for the application of Fourier Transform Analysis in the subsequent section.

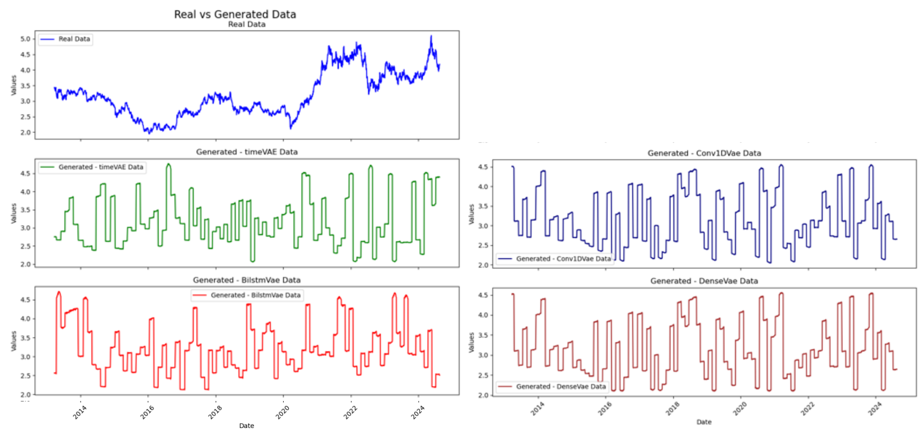


Figure 10: Visualization synthetic data

Frequency similarity between original and synthetic signals

The frequency similarity, as visualized in Figure 11 with a Spectral Analysis by Fourier Transform between the original time series and various VAE-based synthetic signals, was assessed using cosine similarity on their respective Fourier amplitude spectra. The results indicate a high degree of spectral alignment between the original and synthetic signals across all models. Specifically, in Table 1, we can see how the BiLSTM-VAE achieved the highest frequency similarity with the original data (cosine similarity = 0.9900), suggesting superior preservation of temporal frequency characteristics. The Conv1D-VAE, Dense-VAE, and TimeVAE models also exhibited strong alignment with cosine similarity values of 0.9850, 0.9849, and 0.9848, respectively. These findings highlight the effectiveness of VAE architectures in maintaining the frequency structure of time series data, with the BiLSTM-VAE demonstrating slightly better fidelity in spectral preservation.

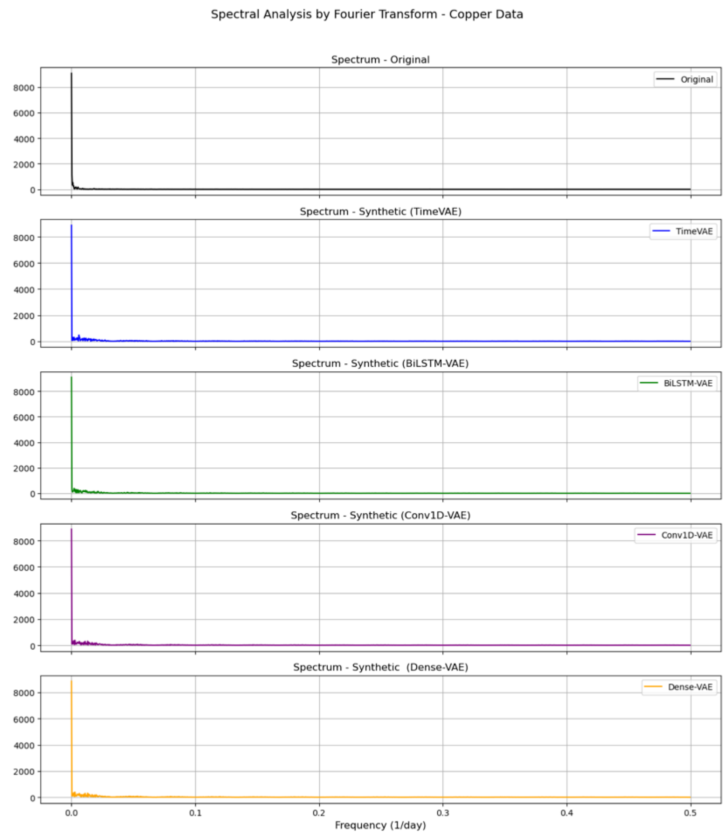


Figure 11: Spectral Analysis by Fourier Transformation

Table 1: Cosine Similarity (frequency domain - FFT)

	TimeVAE	BiLSTM-VAE	Conv1D-VAE	Dense-VAE.
Cosine Similarity (Frequency Domain)	0.9848	0.9900	0.9850	0.9849

GMM – Optimal number of Clusters

The figure 12 presents the Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC) scores across varying numbers of components in a Gaussian Mixture Model (GMM), aimed at determining the optimal number of clusters for a given augmented data. Both BIC and AIC are model selection criteria that balance model fit with model complexity, where lower values indicate a better trade-off between these two aspects.

In the plot, the scores decrease rapidly from 1 to 4 components, suggesting that adding clusters significantly improves model fit up to this point. After four components, the BIC and AIC values begin to plateau and eventually increase slightly beyond eight components, indicating diminishing returns in model performance with added complexity. The minimum values of both BIC and AIC occur at 8 components, suggesting this is the optimal number of clusters for modeling the underlying data augmented distribution. Importantly, BIC is slightly more conservative due to its heavier penalty on model complexity, yet both criteria converge on a similar conclusion. This coherence reinforces the robustness of the chosen model configuration.

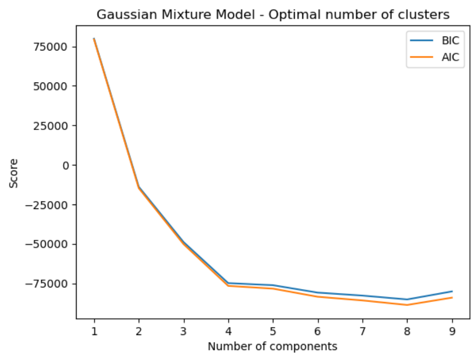


Figure 12: Gaussian Mixture Model - Optimal number of clusters

Prediction with Long Short-Term Memory

The following figures presents the training phase of each Clusters with the LSTM Model (figure 13).

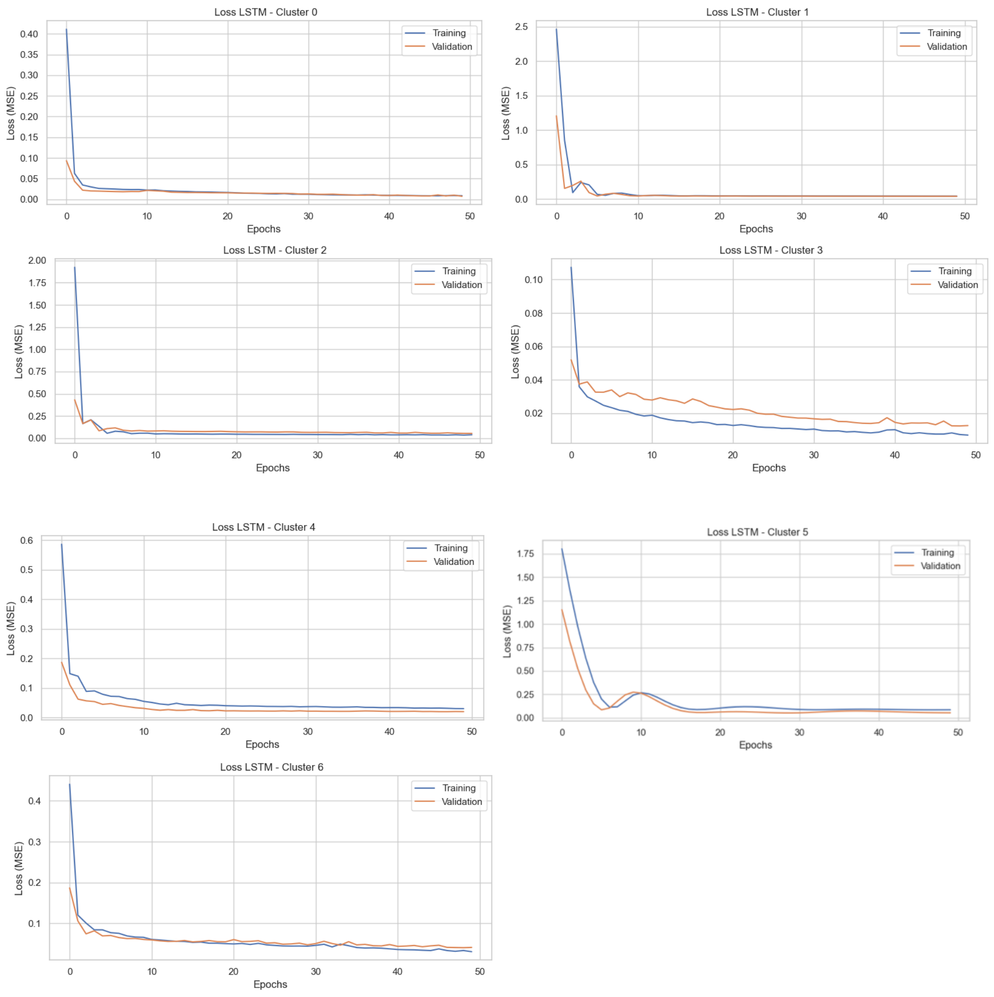


Figure 13: Training phase of each cluster (8 clusters)

Prediction on Test Set: the following figure represents the prediction on the cluster 4 and 6. These two clusters have the same pattern in the prediction over time. Based on this observation, we decided to train the model with 3 clusters, observing that on the results, LSTM model is predicting following 3 clusters in global. The evaluation of LSTM models across both augmented and original datasets—clustered via Gaussian Mixture Models (GMM) into 8 and 3 clusters—reveals the notable impact of data augmentation on prediction performance. In the 8-cluster configuration, the LSTM trained on augmented data (using BiLSTM-VAE) achieved a mean R^2 score of 0.545, outperforming the same model trained on original data, which attained a lower mean R^2 of 0.329.

Although the Mean Absolute Error (MAE) was slightly lower for the original data (0.243 vs. 0.212), the overall fit to the variance in the target variable improved substantially with augmentation. However, the modified MAPE (mMAPE) remained substantially higher in the augmented version (92.16%) compared to the original (24.66%), likely due to extreme errors in underrepresented or noisy clusters such as Cluster 7. This suggests that while augmentation improves generalization in some regions, it may introduce artifacts in others.In the 3-cluster configuration, the benefits of augmentation were even more pronounced. The LSTM on the augmented test set achieved a mean R² score of 0.927, significantly higher than the 0.786 obtained from the model trained on the original data. The MSE, RMSE, and MAE were also considerably lower for the augmented model (e.g., RMSE: 0.217 vs. 0.135), though the augmented data incurred a higher mMAPE (34.11%) compared to the original (14.77%). This again suggests improved precision in absolute terms but greater relative error on smaller values, of which we can visualize the prediction results against the actual values, for cluster 4 in Figure 14.

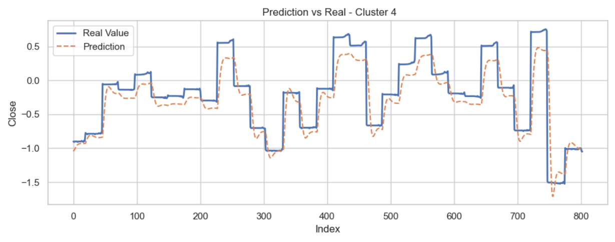


Figure 14: Prediction vs Real - Cluster 4 (LSTM+BiLSTM_VAE+GMM)

Overall, the results as presented in Table 2 demonstrate that data augmentation using BiLSTM-VAE combined with GMM clustering can improve model generalization and reduce absolute prediction errors, especially when the number of clusters is reduced (3-cluster scenario), as we can visualize the training phases of each cluster in Figure 15, and results of the prediction against real values for cluster 2 in Figure 16. These findings support the use of generative augmentation techniques in time series forecasting tasks, particularly when applied with appropriate clustering strategies.

Table 2: Model Performance Comparison (lstm benchmark vs lstm proposed)

LSTM on Augmented Test Set with 8 cluster (GMM)					
Clusters	MSE	RMSE	MAE	mMAPE	R2 Score
6	0,005525	0,074331	0,054507	21,028684	0,995123
4	0,03011	0,173523	0,050198	18,90884	0,926126
1	0,027644	0,166264	0,104537	6,866675	0,345782
2	0,365298	0,604399	0,397617	116,70687	0,438837
7	0,272726	0,522232	0,451008	297,304033	0,021311
Mean	**0,1402606**	**0,3081498**	**0,2115734**	**92,1630204**	**0,5454358**
LSTM on Original Test Set with 8 cluster (GMM)					
1	0,102195	0,31968	0,287858	33,899202	-0,282514
2	0,076275	0,276179	0,237131	22,618533	0,637947
7	0,014147	0,11894	0,090682	17,284567	0,683529
5	0,208207	0,456297	0,357737	24,836063	0,279486
Mean	**0,100206**	**0,292774**	**0,243352**	**24,6595913**	**0,329612**
LSTM on Augmented Test Set with 3 cluster (GMM)					
0	0,004841	0,069575	0,050144	21,146108	0,995729
2	0,031313	0,176956	0,069474	30,422039	0,923174
1	0,163855	0,40479	0,19005	50,773811	0,862367
Mean	**0,06666967**	**0,217107**	**0,10322267**	**34,113986**	**0,92709**
LSTM on Original Test Set with 3 cluster (GMM)					
0	0,022026	0,148411	0,121546	14,710982	0,902795
1	0,01476	0,121491	0,09651	14,823284	0,669808
Mean	**0,018393**	**0,134951**	**0,109028**	**14,767133**	**0,7863015**

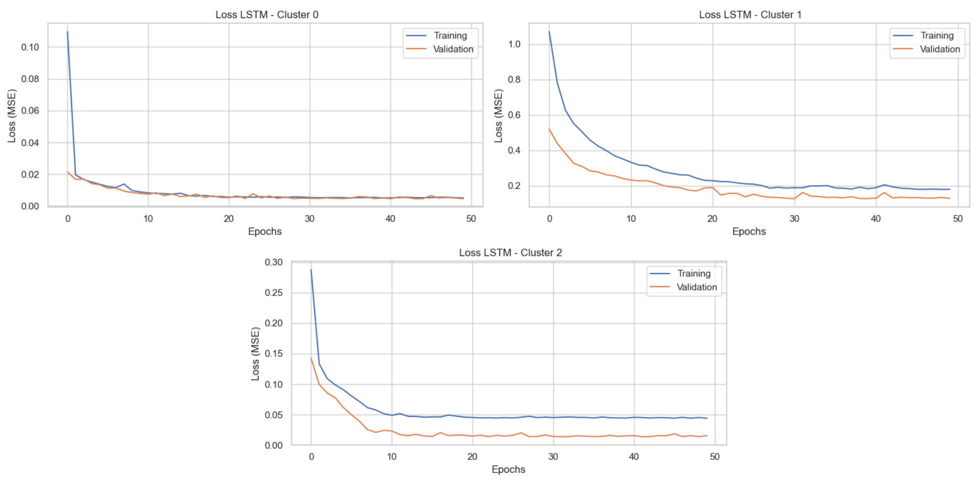


Figure 15: Training phase of each cluster (3 clusters)

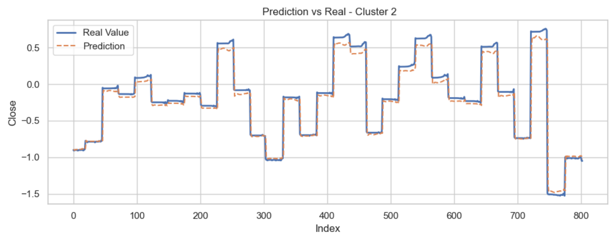


Figure 16: Prediction vs Real - Cluster 2 (LSTM+BiLSTM_VAE+GMM)

Prediction with Time Fusion transformer

The evaluation results of the Temporal Fusion Transformer (TFT) model trained on the augmented copper dataset—clustered using Gaussian Mixture Models (GMM)—demonstrate highly effective predictive performance. Quantitatively, Table 3 shows the model achieves a Mean Squared Error (MSE) of 0.0017 and a Root Mean Squared Error (RMSE) of 0.0415, indicating minimal deviation between predicted and actual values. The Mean Absolute Error (MAE) is 0.0300, suggesting that, on average, the predictions are very close to the true values. Furthermore, the Modified Mean Absolute Percentage Error (mMAPE) of 3.69% reflects strong relative accuracy, even in the presence of small values, and confirms that the model generalizes well to unseen data. Most notably, the R^2 score of 0.9979 reveals that the model explains over 99.7% of the variance in the target variable (Close), which is a strong indicator of an excellent fit.

Visually, in Figure 17, the prediction plot further confirms the model's reliability. The blue curve, representing the observed Close prices, closely aligns with the orange curve denoting the predicted median (quantile 0.5). The narrow shaded confidence interval—representing uncertainty between quantiles 0.1 and 0.9—demonstrates the model's confidence in its predictions. Attention weights, plotted on the secondary axis, highlight which parts of the input sequence the model focuses on during forecasting, adding an interpretability layer to the model's output. Overall, the combination of low error metrics, high R^2, and visually coherent predictions suggests that the TFT model, trained on both real and synthetic clustered data, offers a robust and interpretable solution for time series forecasting in copper trading scenarios.

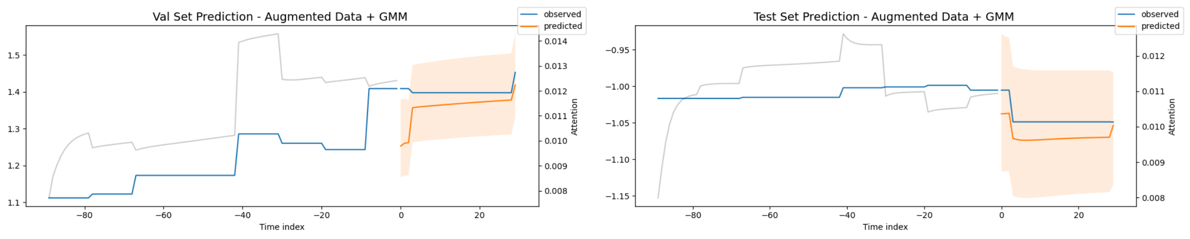


Figure 17: Copper Close value Prediction Validation vs Test

Table 3: TFT + BilstmVAE + GMM performance on Validation and Test

MSE	RMSE	MAE	mMAPE	R2	Obs
0,0017	0,0415	0,0300	3,69%	**0,9979**	val set
0,0006	0,0238	0,0234	2,24%	**-2,3281**	Test set

Figure 18 provides an overview of the encoder variable analysis and reveals that the variable *Low* (lowest price) is by far the most influential, accounting for more than 50% of the total importance. This suggests that the model extracts essential information from this variable to understand the past dynamics of the time series. Other temporal features such as *month*, *day*, and *weekday* also contribute meaningfully, though to a lesser extent, while variables like *Open*, *year*, or *relative_time_idx* play a more modest role.

Regarding the decoder variables, a similar trend is observed: the *Low* variable retains the highest importance, followed by *Open*, which becomes particularly relevant in the prediction context. Variables such as *day*, *weekday*, and *High* also exhibit moderate influence, whereas *relative_time_idx* is nearly negligible, indicating that the temporal position alone is not a key factor in future forecasting.

The analysis of static variables shows that preprocessing parameters—especially *Close_scale* and *Close_center*—are highly influential for model performance. This highlights the effectiveness of normalization techniques applied to the target variable. Additionally, *Cluster_Label*, which reflects a prior segmentation of the dataset (likely via a Gaussian Mixture Model), contributes valuable information, confirming the relevance of a clustering-based approach. The variable *encoder_length* plays a more marginal, yet non-negligible, role.

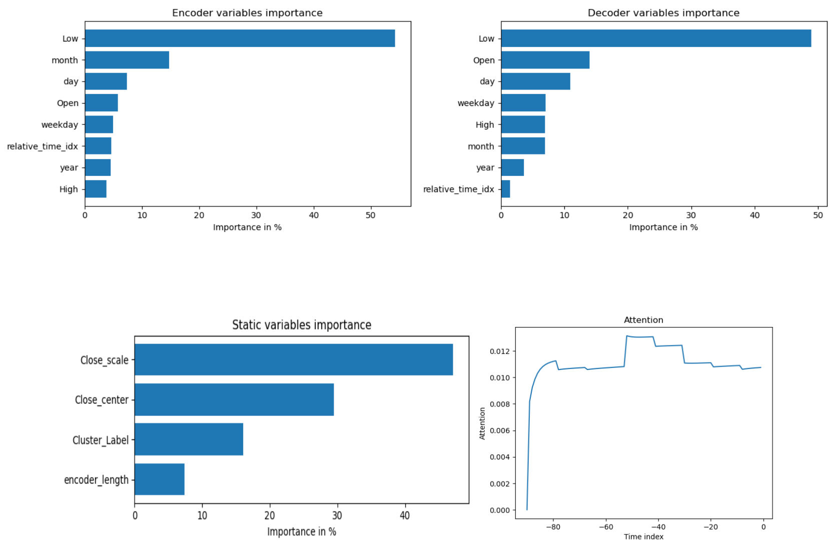


Figure 18: Interpretable TFT (Encoder, Decoder Static variavles & Attention)

Finally, the attention weight visualization shows that the model assigns greater importance to recent observations, which aligns with the predictive logic of time series analysis. However, a degree of stability in attention weights across a broader historical window suggests that the model does not completely disregard earlier observations.

Overall, these results demonstrate the TFT model's strong capability to capture temporal dynamics while leveraging exogenous variables, static features, and attention mechanisms to generate robust predictions. The central role of price-related variables (*Low*, *Open*, *Close*), along with the impact of data augmentation and clustering with GMM, emerge as key performance drivers in this context.

4.2. Discussion

Table 4 presents a comparative evaluation of several forecasting models applied to time series data, including combinations of LSTM, BiLSTM-based Variational Autoencoders (BiLstmVAE), and the Temporal Fusion Transformer (TFT), all integrated within a clustering framework based on Gaussian Mixture Models (GMM). The baseline model, LSTM + GMM, demonstrates relatively satisfactory performance on the test set, with an MSE of 0.0184, RMSE of 0.1350, and MAE of 0.1090. The mMAPE of 14.77% and R^2 score of 0.7863 indicate a moderate level of predictive accuracy and goodness of fit.

Integrating synthetic data via BiLstmVAE (i.e., LSTM + BiLstmVAE + GMM) paradoxically leads to an increase in MSE (0.0667) and RMSE (0.2171), while the MAE slightly decreases (0.1032). However, a sharp rise in mMAPE (34.11%) is observed, suggesting reduced reliability in relative error terms. Interestingly, the R^2 score increases to 0.9271, indicating that the model captures overall trends well. This highlights the inherent trade-off between pointwise accuracy and trend learning when synthetic data is used for augmentation, whether for general time series forecasting or demand forecasting applications within the VAE-based data augmentation paradigm.

Table 4: Copper – Comparison Predictive Models (Original and Synthetic Data)

Model	MSE	RMSE	MAE	mMAPE	R2	Observation
Lstm + GMM (Benchmark)	0,018393	0,134951	0,109028	14,767133	**0,7863015**	Result on Test set
Lstm + BiLstmVAE + GMM	0,06666967	0,217107	0,10322267	34,113986	**0,92709**	Result on Test set
TFT + BiLstmVAE + GMM	0,0017	0,0415	0,0300	3,69%	**0,9979**	Result on val set
TFT + BiLstmVAE + GMM	0,0006	0,0238	0,0234	2,24%	**-2,3281**	Result on Test set

A key challenge after generating synthetic data lies in its injection into the original dataset to leverage its potential in improving predictive performance. In our experiments, we explored various integration approaches involving model architecture modifications (e.g., parallel dual branches, transfer learning-based fine-tuning, residual connections with synthetic data) and different synthetic data injection strategies (e.g., concatenation, tensor addition, multiplication). Our findings indicate that for daily-level data, model performance remained relatively stable even with augmented data. However, in the case of periodic trends, the impact was more pronounced.

When integrating TFT with BiLstmVAE and GMM, we observed significant performance improvements on the validation set, achieving an MSE of 0.0017, RMSE of 0.0415, MAE of 0.0300, mMAPE of 3.69%, and R^2 of 0.9979. These results confirm the superiority of TFT in capturing complex temporal dynamics through its combined use of historical context and explanatory variables. The inclusion of synthetic data generated by BiLstmVAE and clustered via GMM further enhances the temporal representations learned by TFT.

However, on the test set, the same configuration (TFT + BiLstmVAE + GMM) yields even lower absolute errors (MSE = 0.0006, RMSE = 0.0238, MAE = 0.0234, mMAPE = 2.24%), but paradoxically results in a negative R^2 score (-2.3281). This typically indicates that the model underperforms compared to a naive predictor based on the mean. Such anomalies may stem from a distributional shift between training/validation and test sets, an issue exacerbated by the use of synthetic data without adequate domain adaptation. It may also reflect overfitting, particularly when the test set exhibits low variance or contains outliers that heavily influence the R^2 computation. This points to an open research avenue around enhancing TFT's adaptability in the presence of synthetic data integration and better understanding its limitations under such conditions.

Conclusion

In the specific context of commodity trading, such as copper, where accurate demand and market trend forecasting is strategically crucial for inventory, procurement, and logistics planning, the proposed hybrid framework (LSTM + BiLstmVAE + GMM) offers promising opportunities. This architecture combines the strengths of temporal clustering (GMM), synthetic data generation (VAE), and advanced sequential modeling (LSTM). Experimental results show that this combination significantly enhances the model's ability to capture complex time series dynamics while enriching its learning capacity via data augmentation.

However, the use of TFT + BiLstmVAE + GMM, although experimentally compelling, presents challenges. The integration of synthetic data remains a critical issue—particularly in low-variance environments or when data distribution shifts occur between training and test phases. As observed, strong performance in terms of absolute errors may be coupled with a negative R^2, indicating a loss of global coherence in aggregate predictions. This phenomenon likely results from overfitting or poor generalization of injected synthetic data, underlining the need for a more robust adaptation framework within TFT when incorporating such data.

Operationally, in commodity markets, this framework could prove particularly useful for modeling periodic demand patterns, detecting consumption anomalies, and anticipating seasonal volatility typical of copper, oil, or agricultural product markets. Nonetheless, real-world deployment should involve robust validation protocols, continuous monitoring of data distributions, and potentially, the exploration of alternative predictive models beyond LSTM, tailored to each cluster or data type.

Ultimately, this study opens the door for applied research on the adaptability of deep learning models enhanced by synthetic data in highly volatile domains such as commodities. It also highlights the need for adaptive mechanisms that can effectively balance learning from enriched datasets with reliable generalization to evolving real-world data streams.

REFERENCES

Nochumsohn, L., & Azencot, O. (2025). Data augmentation policy search for long term forecasting. Transactions on Machine Learning Research.

Islam, T., Hafiz, M. S., Jim, J. R., Kabir, M. M., & Mridha, M. F. (2024). A systematic review of deep learning data augmentation in medical imaging: Recent advances and future research directions. *Healthcare Analytics*, 5.

Szlobodnyik, G., & Farkas, L. . (2021). Data augmentation by guided deep interpolation. *Applied Soft Computing*, 111.

Ouyang, Z. (2023). Time series forecasting: From econometrics to deep learning . *University of Orléans, INSA CVL, PRISME.*

Wen, Q. S. (2020). Time series data augmentation for deep learning: a survey. *In Proceedings of the 30th International Joint Conference on Artificial Intelligence*, 4653–46.

Steven Eyobu, O., & Han, D. S. . (2018). Feature representation and data augmentation for human activity classification based on wearable IMU sensor data using a deep LSTM neural network.

Cleveland, W. S. (1990). STL: A seasonal trend decomposition procedure based on loess. Journal of Official Statistic.

Wen, Q., Gao, J., Song, X., & al. (2019). RobustSTL: A robust seasonal trend decomposition algorithm for long time series. In Proceedings of the Thirty Third AAAI Conference on Artificial Intelligence. 5409–5416.

Cao, H., Tan, V. Y., & Pang, J. Z. . (2014). A parsimonious mixture of Gaussian trees model for oversampling in imbalanced and multimodal time-series classification. IEEE transactions on neural networks and learning systems. 2226-2239.

Iwana, B. K., & Uchida, S. (2021). Time series data augmentation for neural networks by time warping with a discriminative teacher. *25th International Conference on Pattern Recognition (ICPR), IEEE*, 3558-3565.

Rashid, K. M., & Louis, J. . (2019). Window-warping: A time series data augmentation of IMU data for construction equipment activity identification. In ISARC. *Proceedings of the international symposium on automation and robotics in construction.*, 651-657.

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

Goubeaud, M. J. (2021). Using variational autoencoder to augment sparse time series datasets . *7th international conference on optimization and applications (ICOA), IEEE*, 1-6.

Chung, D, Park, S., Song, Y., Chae, J., &. Yoon, D. . (2023). Methodology for improving the performance of demand forecasting through machine learning.

Desai, A., Freeman, C., Wang, Z., & Beaver, I. . (2021). Timevae: A variational auto-encoder for multivariate time series generation.

Yi, K. Z. (2023). A survey on deep learning based time series analysis with frequency transformation.

Cooley, J. W., & Tukey, J. W. (). . (1965). An algorithm for the machine calculation of complex Fourier series. . *Mathematics of computation*, 297-301.

Reynolds, D. . (2015). Gaussian mixture models. In Encyclopedia of biometrics. *Springer, Boston, MA*, 827-832.

Hochreiter, S., & Schmidhuber, J. . (1997). Long short-term memory. . *Neural computation*, 1735-1780.

Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. . (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International journal of forecasting*, 1748-1764.

Huy, P. C. (2022). Short-term electricity load forecasting based on temporal fusion transformer model. *Ieee Access*, 10, 106296-106304.

Tables

Figures