MANTECH Publications
Journal of Artificial Intelligence, Machine Learning and Soft Computing
Volume 8, Issue 2, July-December 2023

Machine Learning for Time-Series and Sequential Data

Abhimanu Seth[1], Jakir Khan[2]
Associate Professor[1], Students[2]
Department of Artificial Intelligence
Silver lake University, India
Email: *Abhimanu15seth@gmail.com, 25_kjajir@yahoo.com*

Abstract

Time-series and sequential data are ubiquitous in fields ranging from finance and healthcare to weather forecasting and IoT sensor networks. Analyzing such data presents unique challenges due to temporal dependencies, non-stationarity, and complex patterns that evolve over time. Machine learning (ML) methods, including traditional statistical approaches, deep learning architectures, and hybrid models, have demonstrated significant promise in handling these challenges. This paper provides a comprehensive review of the current state-of-the-art ML techniques for time-series and sequential data, focusing on their applications, strengths, and limitations. It also explores recent advances in sequence modeling, including recurrent neural networks (RNNs), long short-term memory networks (LSTMs), gated recurrent units (GRUs), and transformer-based models. Finally, we highlight key challenges, future research directions, and practical considerations for deploying ML solutions in time-series forecasting and sequential data analysis.

Keywords: *Time-Series Analy, Sequential Data, Machine Learning, Recurrent Neural Networks, LSTM, Transformer Models, Forecasting, Temporal Dependencies*

INTRODUCTION

Time-series and sequential data are datasets where observations are collected over time or in a specific sequence. Examples include stock prices, climate measurements, patient vital signs, web traffic logs, and sensor data from IoT devices. The temporal or sequential order in these

datasets introduces unique challenges that differentiate them from typical tabular datasets. Traditional machine learning models, designed for independent and identically distributed (i.i.d.) data, often fail to capture the dependencies and patterns inherent in sequential data.

Machine learning has emerged as a powerful approach to tackle these challenges. By learning from historical patterns, ML models can perform forecasting, anomaly detection, classification, and pattern recognition. This paper reviews recent advances in ML techniques for time-series and sequential data, comparing classical methods with modern deep learning approaches.

2. BACKGROUND

Understanding time-series and sequential data is fundamental for selecting appropriate machine learning methods. Both types of data share similarities but also present distinct challenges due to the importance of temporal or sequential ordering. This section provides a detailed overview of their characteristics and implications for modeling.

2.1 Time-Series Characteristics

Time-series data consists of observations recorded sequentially over time. These datasets are ubiquitous across domains such as finance, healthcare, energy, climatology, and industrial monitoring. Successful modeling of time-series data requires a clear understanding of their key characteristics, which directly influence preprocessing, feature engineering, and model selection.

1. **Trend**
    - The trend represents the long-term progression in the data, indicating systematic upward or downward movements over time.
    - For example, the average monthly temperature may show a gradual increase due to climate change, or stock market indices may exhibit an upward trend over years.
    - Trends can be linear or nonlinear and can be extracted using methods like moving averages, polynomial regression, or decomposition techniques (e.g., STL decomposition).

2. **Seasonality**
    - Seasonality refers to periodic fluctuations in data that repeat at regular intervals, such as daily, weekly, monthly, or yearly cycles.
    - Examples include electricity demand peaking in summer months, increased retail sales

during holidays, or higher website traffic during weekdays.

- Seasonality can significantly affect predictive modeling and often requires specialized approaches like SARIMA or seasonal feature engineering to capture cyclical patterns effectively.

3. **Noise**
   - Noise represents random or unexplained variations in the data that do not follow any discernible trend or pattern.
   - Sources of noise include sensor errors, data entry mistakes, environmental factors, or market volatility.
   - Noise reduction techniques, such as smoothing filters, wavelet transforms, or robust regression, are often applied to improve model performance.

4. **Non-stationarity**
   - A stationary time-series has statistical properties, such as mean, variance, and autocorrelation, that remain constant over time. Many classical time-series models, including ARIMA, assume stationarity.
   - Real-world data is often non-stationary, meaning its properties change over time due to external influences or structural shifts. For example, a sudden policy change can alter economic indicators, or a disease outbreak can abruptly change hospital admissions.
   - Detecting non-stationarity using statistical tests (e.g., Augmented Dickey-Fuller test) and applying transformations such as differencing, detrending, or seasonal adjustment is crucial before modeling.

5. **Temporal Dependencies**
   - Time-series data points are not independent; the value at time *t* often depends on previous values. Capturing these dependencies is critical for accurate forecasting.
   - Autocorrelation and partial autocorrelation functions are tools used to quantify dependencies at different time lags, guiding model selection and lag feature engineering.

By understanding these characteristics, data scientists can choose models that handle trends, seasonal effects, noise, and non-stationarity, ensuring more accurate and reliable forecasts.

2.2 Sequential Data

Sequential data refers to datasets where the **order of observations is critical** and can contain complex dependencies that are not strictly temporal but still ordered. Sequential data appears in numerous domains beyond time-series, including natural language, biological sequences, clickstreams, and system logs. Unlike traditional tabular data, the sequence itself conveys significant information, and models must preserve this ordering to capture patterns effectively.

1. **Nature of Dependencies**
    - Dependencies in sequential data can be **short-term**, where current observations primarily depend on a few immediate predecessors, or **long-term**, where patterns from far back in the sequence influence the current state.
    - For example, in language modeling, the meaning of a word may depend on the context several sentences earlier. In user behavior modeling, a purchase decision may be influenced by browsing activity from weeks prior.
2. **Applications and Examples**
    - **Natural Language Processing (NLP):** Sentences and documents where word order affects meaning.
    - **Clickstream Analysis:** Tracking sequences of user actions on websites to predict behavior or recommend content.
    - **Bioinformatics:** DNA, RNA, or protein sequences where nucleotide or amino acid order is critical for function.
    - **Industrial Processes:** Sensor readings from machines where current readings depend on past operational states.
3. **Challenges in Sequential Modeling**
    - **Variable sequence length:** Sequences may differ in length, requiring padding, truncation, or dynamic architectures.
    - **Long-range dependencies:** Capturing relationships across distant elements in the sequence is non-trivial.
    - **Noise and irregular sampling:** Missing values or inconsistent intervals can degrade model performance.
4. **Modeling Implications**
    - Models for sequential data must capture both **local patterns** (short-term dependencies) and **global context** (long-term dependencies).

- Traditional time-series models can handle some sequential data but may fail with complex patterns, motivating the use of **recurrent neural networks (RNNs), LSTMs, GRUs, and transformer-based architectures**.

Understanding sequential data characteristics ensures that appropriate machine learning strategies are applied, leading to better predictive performance and more interpretable insights.

3. TRADITIONAL MACHINE LEARNING APPROACHES

Traditional machine learning and statistical methods have long been the foundation for modeling time-series and sequential data. These approaches rely on mathematical formulations and assumptions about the data structure, such as linearity and stationarity. While they may not capture complex nonlinear relationships as effectively as modern deep learning models, they remain highly interpretable, computationally efficient, and widely used in practical forecasting tasks.

3.1 Statistical Models

Statistical models are often the first choice for time-series analysis due to their simplicity and interpretability. They are particularly effective for univariate time-series with well-defined trends and seasonality.

1. ARIMA (AutoRegressive Integrated Moving Average)

- **Overview:**

ARIMA models combine three components to capture patterns in stationary time-series:

- **Autoregressive (AR):** Models the relationship between the current observation and a number of lagged observations.
- **Integrated (I):** Differencing applied to make non-stationary data stationary.
- **Moving Average (MA):** Models the relationship between the observation and past forecast errors.

- **Mathematical Formulation:**

  An ARIMA(p, d, q) model is defined as:

  yt′=c+∑i=1pϕiyt−i′+∑j=1qθjϵt−j+ϵty'_t = c + \sum_{i=1}^{p} \phi_i y'_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_tyt′=c+i=1∑pϕiyt−i′+j=1∑qθjϵt−j+ϵt

Where:

- yt′y'_tyt′ is the differenced series (after d differences),
- ppp is the AR order,
- ddd is the differencing order,
- qqq is the MA order,
- ϕi\phi_iϕi are autoregressive coefficients,
- θj\theta_jθj are moving average coefficients,
- ϵt\epsilon_tϵt is the white noise error term.

- **Applications:**
  - Stock price and financial forecasting
  - Economic indicators prediction
  - Energy demand modeling
- **Limitations:**
  - Assumes linear relationships in the data.
  - Sensitive to outliers and non-stationary behavior not captured by differencing.
  - Requires manual selection of parameters (p, d, q), often guided by autocorrelation (ACF) and partial autocorrelation (PACF) analysis.

2. SARIMA (Seasonal ARIMA)

- **Overview:**

  SARIMA extends ARIMA by incorporating seasonal components to model periodic fluctuations. Seasonal ARIMA is represented as ARIMA(p, d, q)(P, D, Q)s, where:
  - P,D,QP, D, QP,D,Q are the seasonal AR, differencing, and MA orders
  - sss is the length of the seasonal cycle (e.g., 12 for monthly data with yearly seasonality)

- **Mathematical Representation:**

  ΦP(Bs)ϕp(B)(1−B)d(1−Bs)Dyt=ΘQ(Bs)θq(B)ϵt\Phi_P(B^s) \phi_p(B) (1-B)^d (1-B^s)^D y_t = \Theta_Q(B^s) \theta_q(B) \epsilon_tΦP(Bs)ϕp(B)(1−B)d(1−Bs)Dyt=ΘQ(Bs)θq(B)ϵt

Where:

- BBB is the backshift operator (Byt=yt−1B y_t = y_{t-1}Byt=yt−1)
- ϕp(B)\phi_p(B)ϕp(B) and θq(B)\theta_q(B)θq(B) are non-seasonal AR and MA polynomials
- ΦP(Bs)\Phi_P(B^s)ΦP(Bs) and ΘQ(Bs)\Theta_Q(B^s)ΘQ(Bs) are seasonal AR and MA polynomials

- **Applications:**
  - Retail sales forecasting during holidays
  - Temperature and weather prediction
  - Tourism and traffic volume forecasting
- **Limitations:**
  - Increased complexity due to multiple parameters.
  - Difficult to capture sudden shocks or nonlinear changes.\

Model	**Strengths**	**Limitations**
ARIMA	Simple, interpretable	Assumes linearity, stationary data
SARIMA	Handles seasonality	Complex to tune, linear assumption
Exponential Smoothing	Robust to noise	Limited for long-term dependencies

3.2 Machine Learning Regression Models

While statistical models like ARIMA and Exponential Smoothing work well for linear or moderately complex time-series, many real-world datasets exhibit **nonlinear dependencies**, multivariate interactions, and complex temporal patterns. Traditional machine learning (ML) regression models provide an alternative by treating time-series forecasting as a supervised learning problem. These models can capture nonlinear relationships and interactions among features, making them suitable for more challenging datasets.

1. Transforming Time-Series for Supervised Learning

Most ML algorithms require **input features (X)** and a **target variable (y)**. To apply ML models to time-series data, we must convert sequences into a supervised learning format, often using **lagged features**, **rolling statistics**, and **time-based encodings**.

a. Lag Features:

- Lag features are previous observations included as predictors for the current or future time step.
- Example: To predict yty_tyt using the past 3 time steps:

Xt=[yt−1,yt−2,yt−3],yt=ytX_t = [y_{t-1}, y_{t-2}, y_{t-3}], \quad y_t = y_tXt=[yt−1,yt−2,yt−3],yt=yt

- Lag selection depends on the autocorrelation structure of the series.

b. Rolling Statistics:

- Compute statistical summaries (mean, standard deviation, min, max) over a sliding window of past observations.
- Example: Rolling mean over past 5 time steps:

RollingMeant=yt−1+yt−2+yt−3+yt−4+yt−55\text{RollingMean}_t = \frac{y_{t-1} + y_{t-2} + y_{t-3} + y_{t-4} + y_{t-5}}{5}RollingMeant=5yt−1+yt−2+yt−3+yt−4+yt−5

- Rolling features help capture short-term trends and smooth out noise.

c. Time-Based Encoding:

- Encoding temporal information as features can improve model performance. Common encodings include:
  - **Hour, day, month, weekday, quarter** for seasonal patterns
  - **Cyclic encoding** for periodic features:

Hour_sin=sin⁡(2π·hour24),Hour_cos=cos⁡(2π·hour24)\text{Hour_sin} = \sin\left(\frac{2\pi \cdot \text{hour}}{24}\right), \quad \text{Hour_cos} = \cos\left(\frac{2\pi \cdot \text{hour}}{24}\right)Hour_sin=sin(242π·hour),Hour_cos=cos(242π·hour)

- Cyclic encoding prevents artificial discontinuities (e.g., hour 23 → hour 0).

2. Common Machine Learning Models

a. Random Forest Regression (RF):

- Ensemble of decision trees trained on bootstrap samples with feature randomness.
- Captures nonlinear dependencies and interactions.
- Robust to overfitting and outliers.
- **Limitations:** Does not inherently handle temporal order; relies on feature engineering.

b. Gradient Boosting Machines (GBM, XGBoost, LightGBM):

- Sequentially builds trees, each correcting the errors of the previous one.
- Highly effective for structured time-series data with nonlinear patterns.
- Supports regularization to prevent overfitting.
- Limitations**:** Computationally expensive for very large datasets; sensitive to noisy lag features.

c. Support Vector Regression (SVR):

- Learns a regression function that minimizes error within a specified margin (ϵ\epsilonϵ).
- Effective for small to medium datasets with moderate nonlinearity.
- Can use kernel functions (e.g., RBF) to model nonlinear patterns.
- **Limitations:** Scaling issues with large datasets; careful hyperparameter tuning is required.

d. k-Nearest Neighbors Regression (kNN):

- Predicts values based on the average of k most similar past observations.
- Simple and interpretable.
- **Limitations:** Poor performance for long-term dependencies or high-dimensional sequences.

3. Advantages of ML Regression Models

- Capable of modeling **nonlinear relationships** and multivariate dependencies.
- Flexibility in incorporating **external variables** (e.g., weather, promotions, macroeconomic indicators).
- Can be combined with feature engineering for short-term forecasting.

4. Limitations and Challenges

a) **Long-Range Dependencies:**
   - ML regression models rely on input features and cannot inherently capture temporal dependencies beyond the lag window.

b) **High-Dimensional Sequences:**
   - Large numbers of lag features or rolling statistics can lead to overfitting and

increased computation.

c) **Feature Engineering Requirement:**
   - Performance heavily depends on the quality of lag, rolling, and time-based features.

d) **Non-Stationarity:**
   - Sudden changes or trends not captured by engineered features can reduce accuracy.

4. DEEP LEARNING FOR SEQUENTIAL DATA

Deep learning has revolutionized the analysis of time-series and sequential data by enabling models to automatically learn complex nonlinear relationships and temporal dependencies. Unlike traditional machine learning approaches, deep learning architectures can model long-term dependencies, high-dimensional sequences, and multivariate interactions without extensive feature engineering. This section explores the most widely used deep learning models for sequential data.

4.1 Recurrent Neural Networks (RNNs)

Overview:

Recurrent Neural Networks (RNNs) are designed specifically for sequential data by introducing **recurrent connections** that allow information to persist across time steps. Unlike feedforward neural networks, RNNs maintain a hidden state that captures information from previous observations.

Mathematical Formulation:

At time step ttt, the hidden state hth_tht and output ytyty_tyt are computed as:

ht=f(Wxhxt+Whhht−1+bh)h_t = f(W_{xh} x_t + W_{hh} h_{t-1} + b_h)ht=f(Wxhxt+Whhht−1+bh) yt=g(Whyht+by)y_t = g(W_{hy} h_t + b_y)yt=g(Whyht+by)

Where:

- xtx_txt is the input at time ttt,
- hth_tht is the hidden state,
- yty_tyt is the output,
- Wxh,Whh,WhyW_{xh}, W_{hh}, W_{hy}Wxh,Whh,Why are weight matrices,
- bh,byb_h, b_ybh,by are bias terms,
- fff and ggg are activation functions (e.g., tanh, ReLU).

Advantages:

- Can model sequential dependencies of arbitrary length.
- Shares weights across time steps, reducing the number of parameters.

Limitations:

- **Vanishing Gradient Problem:** Gradients diminish exponentially during backpropagation through many time steps, making it hard to learn long-term dependencies.
- **Exploding Gradient Problem:** Gradients can grow uncontrollably, causing unstable training.
- Ineffective for very long sequences without architectural modifications.

Applications:

- Short-term forecasting
- Sensor data analysis
- Simple NLP tasks

4.2 Long Short-Term Memory Networks (LSTMs)

Overview:

LSTMs were introduced to address RNN limitations, particularly vanishing gradients. LSTMs use a **memory cell** along with **gates** to control the flow of information:

- **Forget Gate (ftf_tft)** – decides which information to discard from the cell state.
- **Input Gate (iti_tit)** – determines which new information to add to the cell state.
- **Output Gate (oto_tot)** – decides which information to output from the cell.

Mathematical Formulation:

ft=σ(Wf·[ht−1,xt]+bf)f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)ft=σ(Wf·[ht−1,xt]+bf) it=σ(Wi·[ht−1,xt]+bi)i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)it=σ(Wi·[ht−1,xt]+bi) C~t=tanh⁡(WC·[ht−1,xt]+bC)\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)C~t =tanh(WC·[ht−1,xt]+bC) Ct=ft∗Ct−1+it∗C~tC_t = f_t * C_{t-1} + i_t * \tilde{C}_tCt=ft∗Ct−1 +it∗C~t ot=σ(Wo·[ht−1,xt]+bo)o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)ot=σ(Wo·[ht−1 ,xt]+bo) ht=ot∗tanh⁡(Ct)h_t = o_t * \tanh(C_t)ht=ot∗tanh(Ct)

Where CtC_tCt is the cell state and ∗*∗ denotes element-wise multiplication.

Advantages:

- Captures long-term dependencies efficiently.
- Handles sequences with complex temporal patterns.
- Widely adopted for time-series forecasting, NLP, and speech recognition.

Limitations:

- Computationally intensive due to multiple gates.
- Training can be slow for very long sequences.

Applications:

- Energy load forecasting
- Stock price prediction
- Patient vital sign prediction in healthcare

4.3 Gated Recurrent Units (GRUs)

Overview:

GRUs are a simplified variant of LSTMs that combine the forget and input gates into a **single update gate**. GRUs have fewer parameters and are computationally more efficient while often achieving performance comparable to LSTMs.

Mathematical Formulation:

zt=σ(Wz·[ht−1,xt])z_t = \sigma(W_z \cdot [h_{t-1}, x_t])zt=σ(Wz·[ht−1,xt]) rt=σ(Wr·[ht−1,xt])r_t = \sigma(W_r \cdot [h_{t-1}, x_t])rt=σ(Wr·[ht−1,xt]) h~t=tanh⁡(Wh·[rt∗ht−1,xt])\tilde{h}_t = \tanh(W_h \cdot [r_t * h_{t-1}, x_t])h~t=tanh(Wh·[rt∗ht−1,xt]) ht=(1−zt)∗ht−1+zt∗h~th_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_tht=(1−zt)∗ht−1+zt∗h~t

Where:

- ztz_tzt is the update gate
- rtr_trt is the reset gate
- h~t\tilde{h}_th~t is the candidate hidden state

Advantages:

- Fewer parameters → faster training
- Comparable performance to LSTMs
- Handles long sequences reasonably well

Limitations:

- Slightly less flexible than LSTMs in modeling very complex dependencies

Applications:

- Real-time anomaly detection
- Sequential recommendation systems
- Financial series prediction

4.4 Convolutional Neural Networks (CNNs) for Time-Series

Overview:

Although CNNs are primarily designed for image processing, they are effective for **capturing local temporal patterns** in time-series data. Temporal Convolutional Networks (TCNs) apply **causal convolutions** to ensure predictions at time ttt depend only on past observations.

Key Characteristics:

- **Causal Convolution:** Ensures no information leakage from future time steps.
- **Dilated Convolution:** Expands the receptive field without increasing network depth, allowing the model to capture long-term dependencies efficiently.

Advantages:

- Parallelizable → faster training than RNNs
- Captures local temporal patterns well
- Suitable for high-frequency data

Limitations:

- May struggle with very long-range dependencies if dilation is insufficient
- Requires careful selection of kernel size and dilation factor

Applications:

- High-frequency financial forecasting
- Sensor signal classification
- Speech and audio processing

4.5 Transformer-Based Models

Overview:

Transformers, initially developed for NLP, leverage **self-attention mechanisms** to model dependencies across the entire sequence simultaneously, enabling **long-range dependency modeling** and **parallel computation**.

Key Concepts:

- **Self-Attention:** Calculates attention scores between all pairs of positions in a sequence, allowing the model to focus on relevant past observations.
- **Positional Encoding:** Adds information about the order of sequence elements, since transformers are permutation-invariant.

Advantages:

- Efficiently models long-range dependencies
- Highly parallelizable → faster training on long sequences
- Scalable to multivariate and high-dimensional time-series

Limitations:

- High computational cost for extremely long sequences
- Requires large datasets to avoid overfitting

Applications:

- Multivariate time-series forecasting (e.g., energy grids, traffic)
- NLP tasks like language modeling and machine translation
- High-frequency trading

5. APPLICATIONS

5.1 Financial Forecasting

ML models, particularly LSTMs and transformers, have been widely used for stock price prediction, risk modeling, and algorithmic trading. By learning temporal dependencies and incorporating exogenous variables, these models outperform classical statistical methods in capturing market dynamics.

5.2 Healthcare and Medical Time-Series

Time-series ML methods are crucial for analyzing patient vitals, ECG signals, and wearable sensor data. Predictive models help in early detection of anomalies, disease progression modeling, and personalized healthcare recommendations.

5.3 Energy Load and Environmental Forecasting

Forecasting electricity demand, solar radiation, or air pollution levels requires capturing both seasonal trends and sudden changes. Hybrid models combining LSTMs with traditional statistical approaches often yield high accuracy.

5.4 Natural Language Processing (NLP)

Sequential ML models like RNNs, LSTMs, and transformers are foundational in NLP for tasks

such as sentiment analysis, machine translation, and speech recognition.

CHALLENGES

1. **Non-Stationarity:** Many real-world time-series are non-stationary, making model generalization difficult.
2. **Data Scarcity:** Long sequences with limited labeled data hinder deep learning performance.
3. **Multivariate Dependencies:** Modeling interactions between multiple time-series is computationally complex.
4. **Interpretability:** Deep models are often black-boxes, limiting adoption in critical domains.
5. **Scalability:** High-frequency data can result in huge sequences, requiring efficient computation.

RECENT ADVANCES

1. **Hybrid Models**

   Combining classical statistical methods with deep learning, e.g., ARIMA-LSTM, leverages linear and nonlinear patterns simultaneously, improving forecasting accuracy.
2. **Attention Mechanisms**

   Attention layers in transformers or hybrid LSTM-attention models allow models to focus on relevant parts of the sequence, enhancing long-range dependency learning.
3. **Self-Supervised Learning**

   Pretraining on large unlabeled sequences helps models learn general temporal features, which can then be fine-tuned for downstream tasks.
4. **Probabilistic Forecasting**

Probabilistic models provide uncertainty estimates along with predictions. Methods like **DeepAR** and **Temporal Fusion Transformers** model the entire predictive distribution, crucial for decision-making under uncertainty.

TOOLS AND FRAMEWORKS

- **TensorFlow & PyTorch:** For building RNN, LSTM, GRU, and transformer models.
- **scikit-learn & XGBoost:** For traditional ML approaches and preprocessing.
- **Prophet:** Facebook's library for automated time-series forecasting.
- **tslearn & darts:** Python packages dedicated to time-series machine learning.

CONCLUSION

Machine learning for time-series and sequential data has evolved significantly from traditional statistical models to sophisticated deep learning and transformer-based architectures. Each approach offers distinct advantages depending on data characteristics, task requirements, and computational resources. While deep learning methods, particularly LSTMs and transformers, excel at capturing complex nonlinear and long-range dependencies, challenges remain in interpretability, scalability, and handling non-stationary data. Future research is likely to focus on hybrid architectures, self-supervised learning, probabilistic modeling, and explainable models, enabling more robust and trustworthy time-series analysis across diverse domains.

REFERENCES

1. Box, G. E., Jenkins, G. M., & Reinsel, G. C. (2015). *Time Series Analysis: Forecasting and Control*. Wiley.
2. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.
3. Vaswani, A., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.
4. Lai, G., et al. (2018). Modeling long- and short-term temporal patterns with deep neural networks. *ACM SIGIR*.
5. Zhang, G., Eddy Patuwo, B., & Hu, M. Y. (1998). Forecasting with artificial neural networks. *Journal of Forecasting*, 17(5-6), 489–495.
6. Lim, B., et al. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4), 1748–1764.
7. Brownlee, J. (2018). *Deep Learning for Time Series Forecasting*. Machine Learning Mastery.
8. Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1), 37–45.
9. Lai, K., et al. (2020). Probabilistic forecasting for time series with Bayesian recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(7), 2174–2185.
10. Oord, A. V. D., Dieleman, S., & Schrauwen, B. (2016). WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.