Journal of Computer Science and Artificial Intelligence
ISSN: 3078-8242 (Print), ISSN: 3078-2309 (Online) | Vol. 9, No. 2, 2026

Time Series Forecasting in Data Science: From Classical Models to Deep Learning

Mingjia Ran

College of Liberal Arts, University of Minnesota Twin Cities, Minneapolis, MN 55455, USA

Abstract: Time series forecasting has gone through the evolution of models that were fitted independently, to deep neural architectures pretrained using billions of time steps of time series across various domains. This review summarizes this tran sition, from Box-Jenkins autoregressive integrated moving average (ARIMA) models to ensemble machine learning, recurrent neural networks, attention-based Transformer models and recently developed zero-shot foundation models. The results of the large-scale empirical study of the M4 forecasting competition show that the ranking of statistical, machine learning and deep learning methods is highly series dependent and greatly affected by the length of forecast, the frequency of the series and the number of training data. This literature has three multi-objective challenges that arise: computational latency involving quadratic attention complexity and low accuracy and interpretability when operating in multi-horizon settings, and poor performance in non-stationary, high-frequency streaming settings. A basic mathematical formulation, a chronological synthesis of model classes since the 1950s, explicit equations for ARIMA, LSTM gating and scaled dot-product attention, a comparative benchmark analysis of 5 representative architectures, and an analysis of open research gaps regarding concept drift, foundation model zero-shot transfer, spatiotemporal chaos, and explainability. The review is based on primary sources, covering statistics, machine learning and d eep learning venues from 1957 to 2026. It includes the following topics: Time series forecasting, ARIMA, Recurrent neural networks, Attention mechanisms, Temporal Fusion Transformer, Foundation models, Concept drift.

Keywords: Time series forecasting, ARIMA, Recurrent neural networks, Attention mechanisms, Temporal Fusion Transformer, Foundation models, Concept drift.

1. Introduction

Forecasting systems created in the present day have to balance the two competing demands: statistical accuracy at long time horizons, and inference latency at the scale of millions of concurrent series. ARIMA, formalized by Box and Jenkins [1] is a linear, stationary generative process, which assumes that the process is linear and stationary, and that it fits independently for each series: assumptions and workflow that remain challenging to execute within contemporary data streams. This was empirically evaluated by the M4 Competition which tested 61 forecasting methods on 100,000 real series, and that, when it came to the post-sample performance, the methods that could be classified as pure machine learning were outperformed by the statistical and hybrid methods but used much more computing [2]. To clarify this structural relationship, an engineering analogy is provided to help understand why it is not only the accuracy that affects architectural choices. The square cube law is said to be in effect in structural engineering when the volume of an object grows at a faster rate than its surface area and the state of the model expand disproportionately compared to the rate at which the model parameters are updated compared to the rate at which the model is updated. For example, a Transformer's self-attention mechanism has $O(L^2)$ cost in terms of time and memory, where L is the length of the lookback window [3]. Therefore, doubling the context length can quadruple the time and memory requirements. When a pipeline's structure works well with high-frequency sensor or transaction data, it is unlikely to perform well in the latency dimension as well, so this is a structural cap for distributed pipelines. This trade-off is again redefined by more than 100-billion-time series data used to train the pretrained time series foundation models [4] at deployment time and the zero-shot inference. It will cover the whole path from classical linear model to ensemble model and deep learning, with each paradigm explored from the multi-objective perspective of operational accuracy, computational latency, and systemic scale.

2. Narrative Review Methodology

The most recent deep learning and foundation model research on time series forecasting (TSF) from 1957-2026 was found in the statistics journals (International Journal of Forecasting, Management Science), the machine learning and data mining journals (NeurIPS, ICLR, AAAI, ACM SIGKDD), and in the open preprint repository arXiv, which contains most of the latest foundation model and deep learning research. In addition to model-class terms (ARIMA, exponential smoothing, gradient boosting, LSTM, Transformer, foundation model), task-related terms (forecasting, multi-horizon, multivariate) were also searched. The authors preferred to cite original architecture papers rather than secondary commentary, and competition reports that provided a comparison of several method classes to similar evaluations were preferred if they had been conducted recently (such as the M4 [2] and M5 [5] competitions), as were survey articles that were published recently and provided a general overview of the deep learning forecasting community [6]. For reported empirical performance, this review does not reproduce reported performance metrics in isolation but directs the reader to the original study in which the number was reported. When no empirical information on the performance of a particular architecture class can be found in one location, then instead of creating one, the limitation is explicitly stated.

3. Fundamental Concepts: The Mathematical Formulation

A univariate or multivariate time series is denoted $X_t \in R^m$, a vector of m observed variables at discrete time index t. Given a lookback window of length P, the general multi-step-ahead forecasting objective is

$$\widehat{Y_{t+1:t+H}} = f(X_{t-P:t}; \Theta) \quad (1)$$

Where H is the forecasting horizon, $X_{t-P:t}$ is the conditioning window of P past observations, $f(\cdot)$ is the forecasting function (statistical, tree-based, or neural), and Θ represents the complete set of parameters or network weights estimated from training data. The function f may produce point forecasts, full predictive distributions, or quantile estimates, depending on architecture. Every model family surveyed below from ARIMA's linear difference equations to a Transformer's stacked attention layers is an instantiation of this single objective, differing chiefly in the functional form of f, the structure imposed on Θ, and the assumptions made about stationarity and inter-series dependence.

4. Historical Evolution of Forecasting Paradigms

There have been eight distinct eras in the history of forecasting methodology, not so much characterized by a single landmark paper, as by a change in the predominant mathematical primitive employed to link past observations to future ones. The first approach, which did not require any complete distributional assumptions, and was still adaptive and recursive, was exponentially smoothing, proposed by Holt [7] for the situation of a seasonally varying series, and extended by Winters [8]. "Box and Jenkins formalized the statistically rigorous alternative, ARIMA [1], 10 years later, which remained the dominant operational paradigm until the 1990s. With the beginning of the 2000s, feature-engineered machine learning emerged, based on ensemble trees [9] and kernel methods [10] of lagged representations of the series. This approach was extended by gradient boosting [11] and modular additive regression [12] which were more scalable and more interpretable, respectively. The introduction of recurrent neural networks (RNNs) was done, along with the use of LSTMs [13] and its simplified GRU [14] counterpart, introducing learned and nonlinear temporal memory. However, the sequential bottleneck of recurrence had to be reconsidered through attention-based Transformer models [15-17] which had been able to be as accurate as Transformer models on several benchmarks and even surpassed it on others. The contemporary era, originating in 2023, prioritizes pretrained foundation models, with pretrained foundation models instead of training per-deployment models, with zero-shot inference capabilities. The forecasting models have been developed over the years as indicated in Table 1.

Table 1. Historical Progression of Forecasting Models

Era	Model Class	Dominant Mathematical Framework	Structural Bottlenecks
1957–1960s	Exponential smoothing (Holt-Winters) [7, 8]	Recursive weighted averaging of level, trend, and seasonal components	No native covariate support; fixed seasonal period; manual smoothing parameter tuning
1970–1990s	Box-Jenkins ARIMA/SARIMA [1]	Linear stochastic difference equations via the backshift operator	Requires stationarity and linearity; fit independently per series; brittle under regime shifts
2000s	Kernel and ensemble machine learning [9, 10]	Regression on hand engineered lagged feature windows	Manual feature engineering; no native temporal memory; dimensional expansion with lag depth
Era	Model Class	Dominant Mathematical Framework	Structural Bottlenecks
2010–2018	Gradient boosting and additive regression [11, 12]	Additive decision trees; piecewise trend plus seasonal terms	Static covariate handling; no learned sequence representation; weak extrapolation beyond training range
2014–2017	Recurrent neural networks [13, 14]	Gated recurrent state-space recursion	Sequential computation prevents parallelization; vanishing gradients over long horizons
2019–2021	Attention-based Transformer models [15-17]	Scaled dot-product, sparse, or autocorrelation-based attention	$O(L^2)$ or near quadratic memory cost; large parameter counts; high training cost
2022–2023	Efficient linear and patch-based re-evaluation [18-20]	Channel independent linear projection; patch tokenization	Limited explicit cross-channel modeling: performance varies sharply across datasets
2023–2026	Pretrained foundation models [4, 21-23]	Decoder-only or encoder-based generative pretraining at scale	High pretraining compute cost; domain shift in zero-shot transfer; limited internal transparency

5. Classical Statistical Models: MA, AR, ARIMA, and Holt-Winters

The autoregressive integrated moving average model, $ARIMA(p, d, q)$, expresses a time series through its backshift-operator governing equation:

$$\phi(B)(1-B)^d X_t = \theta(B) Z_t \quad (2)$$

Here B is the backshift operator, defined by $BX_t = X_{t-1}$; $\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$ is the autoregressive polynomial of order p, capturing the linear dependence of X_t on its own past values; $\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$ is the moving average polynomial of order q, capturing the linear dependence of X_t on past forecast errors; $(1-B)^d$ is the differencing operator applied d times to induce stationarity in a non-stationary series; and Z_t is a white noise error term with zero mean and constant variance.

It is a precise and closed-form formulation, but it's limited. Unlike other models, the ARIMA model is only structurally linear, and the architecture is incapable of explicitly modeling nonlinearities or time-varying interactions between covariates. The forecast will always approach a stationary mean, and this is what is assumed. However, one or more of these is physically impossible: the variance is not constant, the linear autocorrelation is not valid, or there is a random event in the sensor network. Holt-Winters exponential smoothing [7, 8] loosens some of these requirements by using adaptive recursive weighting but is still linear and still needs to fit

parameters separately for each series, and thus cannot be scaled to the deployment of millions of concurrent models.

6. Machine Learning and Covariate-Based Forecasting

In tree-based ensembles, forecasting is formulated as a supervised regression task: the values of the target variable and exogenous covariates are collected into a feature table and a tree ensemble like Random Forest [9] or XGBoost [11] is trained to predict the next value in the target variable from this engineered feature table. Support Vector Regression is an extension of the same supervised approach to model nonlinear relationships in transformed features space, via kernel methods [10]. Two practical benefits are offered by this as compared to classic statistical models: a) the use of arbitrary exogenous covariates is supported natively, and b) the training of a model is done across many series at once, instead of one model per series.

This reformulation also introduces structural cost. The inclusion of supplementary rolling windows of historical lags, adding calendar indicators, and adding exogenous regressors increases the feature space's dimensionality combinatorially with lags and regressors. It is more expensive to train due to the dimensional expansion and also has a higher risk of overfitting than the information in each additional lag because tree and kernel methods assume that any feature can replace any other (but not necessarily order), whereas the observations have an inductive bias that the features must be ordered.

The direct impact of the evaluation of the machine learning submissions in the M4 Competition is illustrated: when evaluating in a post-sample fashion, statistical and hybrid combinations dominated machine learning methods in terms of accuracy, and the latter required significantly more computational resources [2]. The construction of the XGBoost gradient-boosted trees is an efficient process, and the scaling of the tree finding the splits is done efficiently by a histogram, with an efficiency of almost O (n log n) per tree [11] but the manual lag-feature engineering at scale is costly.

7. Deep Learning Architectures: RNNs, LSTMs, and GRUs

Recurrent neural networks process a sequence one time step at a time, maintaining a hidden state h_t updated as a function of the current input and the previous hidden state. Standard RNNs suffer from vanishing or exploding gradients over long sequences, motivating the Long Short-Term Memory (LSTM) cell [13], which introduces a separate cell state C_t regulated by three multiplicative gates. The forget gate determines what fraction of the previous cell state to retain:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3}$$

The input gate determines how much new information to write into the cell state, computed alongside a candidate update:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{5}$$

The cell state is then updated by combining the retained past state with the gated candidate:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{6}$$

Finally, the output gate controls how much of the cell state is exposed as the new hidden state:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{7}$$

$$h_t = o_t \odot \tanh(C_t) \tag{8}$$

In each equation, σ denotes the sigmoid activation, $\odot$ denotes elementwise multiplication, W and b are learned weight matrices and bias vectors, and $[h_{t-1}, x_t]$ denotes concatenation of the previous hidden state and current input.

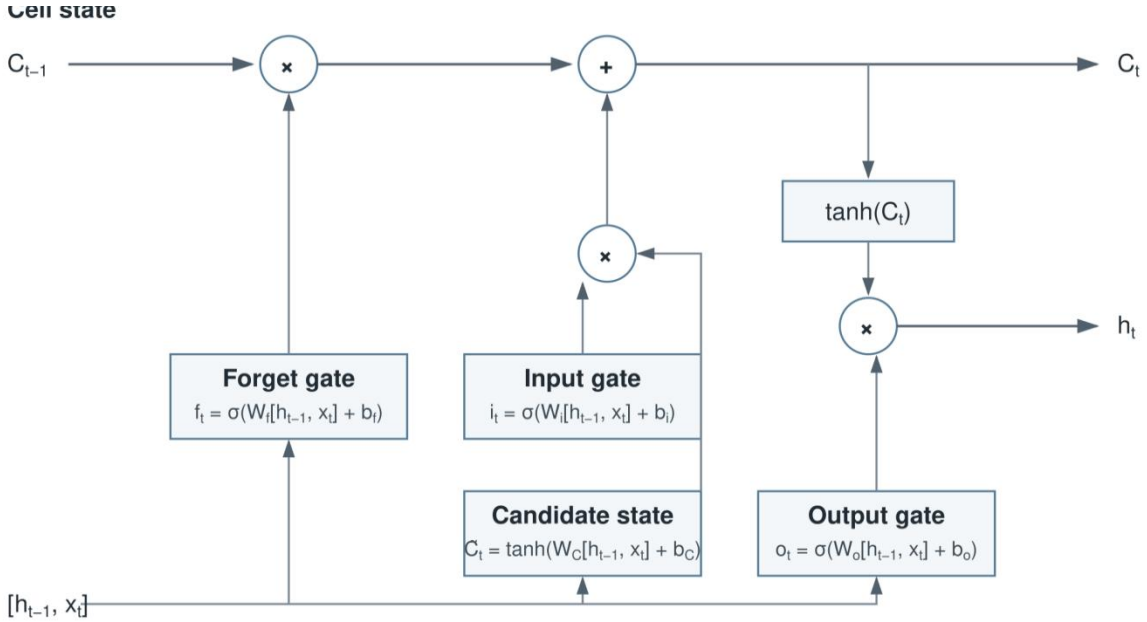

Figure 1. Schematic block diagram illustrating the forget, input, and output gating pathways that regulate cell-state updates within an LSTM forecasting unit [13].

Gated Recurrent Unit (GRU) [14] is introduced in this design by merging forget and input gates into a single update gate and eliminating the separate cell state, thereby decreasing the numbers of parameters, while retaining the primary functional advantages of the gating mechanism. while keeping most of the gating effect. DeepAR [24] and N-

BEATS [25] leverage recurrent and feedforward residual architectures to probabilistic and interpretable multi-horizon forecasting, respectively, with N-BEATS showing an 11% accuracy improvement in comparison to a simple statistical baseline and a 3% improvement compared to the M4 winning hybrid approach using a pure deep learning architecture [25], which highlights that even without explicit time series structure, well-designed deep networks can match or even outperform classical and hybrid methods.

8. The Transformer Paradigm: Attention and Temporal Fusion

The Transformer architecture [3] replaces recurrence with self-attention, allowing every position in a sequence to attend directly to every other position in a single layer. Scaled dot-product attention is computed as

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V \tag{9}$$

Where Q, K, and V are the query, key, and value matrices derived from the input sequence via learned linear projections, and d_k is the dimensionality of the key vectors, used as a scaling factor to prevent the dot products from growing too large for stable softmax gradients.

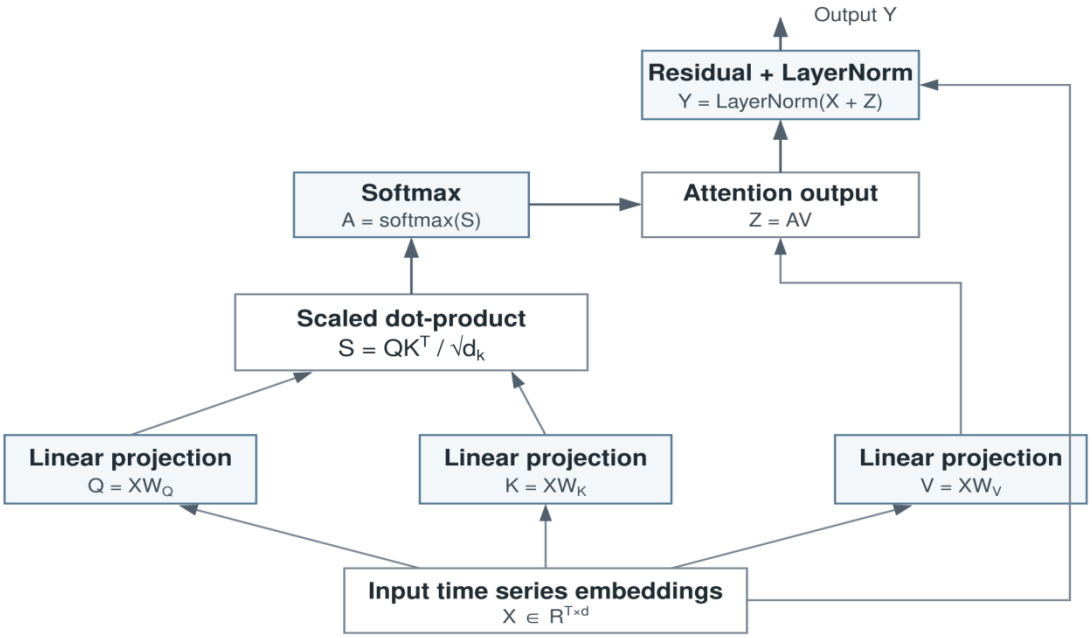

Figure 2. Comprehensive technical block diagram illustrating the flow of query, key, and value matrices within a temporal self-attention forecasting block [3].

This 'recurrence' constraint is removed in the case of forecasting where training can be done in parallel across the lookback window. In the particular case of multi-horizon forecasting, the Temporal Fusion Transformer [15] combines the interpretability of the self-attention mechanism with the recurrence architecture for local processing and achieves significant improvements over the state of the art on several real-world datasets, giving knowledge of the subset of input features and time steps that shape each output.

The main limitation of dense attention is its quadratic computational cost overhead of dense pairwise sequence evaluation scales quadratically. is the time and memory cost of the lookback window L: When L is the size of the lookback window, all queries are attentive to all keys, thus all queries take $O(L^2)$ time and memory. The most important challenge is the same for long-horizon forecasting as in energy grid or financial tick level data, where thousands of time steps are employed, which results in this complexity. This is directly addressed by Informer [16] with a new module, ProbSparse self-attention, which takes advantage of the empirical observation that the vast majority of query-key attention distribution comes from a small number of query-key pairs, resulting in O(L log L) complexity without degrading prediction accuracy. Instead of using a dot-product attention that introduces periodicity explicitly in the series decomposition, Autoformer [17] suggests using an autocorrelation mechanism. It was found, however, that even with a simple one-layer linear model that split the data into trend and seasonality (DLinear), there was no improvement that could be attributed to the attention mechanism, but rather a different way to perform the direct multi-step forecasting which was not based on an autoregressive model [18]. To regain the competitiveness of Transformer models in the long-horizon benchmarks, PatchTST [19] partially addressed this by breaking the series into patches before applying channel-independent attention, thereby regaining competitiveness.

9. Advanced Benchmark Characterization

No literature was found that directly compares all five architectures on the same set of data for each model class, each of which has undergone a careful study in its own "parent study" for this set of data. The synthesis above therefore reports evidence within each row and does not suggest a general ranking of head-to-head evidence, but rather indicates a series of independently reported pieces of evidence. Table 2 shows the comparative analysis of the Metric and Complexity of the Forecasting Architectures.

Table 2. Comparative Metric and Complexity Analysis of Forecasting Architectures

Model	Accuracy	Complexity	Latency	Covariates
ARIMA	Competitive, struggles with nonlinearity	Low	Low	Limited
Prophet	Reliable for seasonal/holiday trends	Moderate	Low	Native
XGBoost	High; requires manual lag engineering	High	Low-Mod	Native
LSTM	Strong; manages long-term dependencies	Very High	Mod-High	Native
TFT	State-of-the-art; interpretable multi-horizon	Extreme	Moderate	Native

10. Research Gaps and Future Directions

A persistent discrepancy separates the theoretical breakthroughs published at machine learning venues from the operational constraints of applied enterprise forecasting what is demonstrated at machine learning conferences and what's utilized in the production enterprise forecasting. There are four gaps that reappear and are significant.

The production of the data-generating process is typically not stationary in most commercial applications, and the production forecasting systems are typically trained once and then deployed, making this assumption. The concept drift adaptation has been studied widely in the streamed machine learning literature [26] but drift-aware retraining triggers are not often used in deployed deep forecasting pipelines, leaving models to degrade parameter accuracy continuously until a scheduled retraining cycle occurs.

Pretrained foundation models are able to make zero-shot predictions on held-out test sets of data drawn from the model's own training distribution, but the performance of these models is reported primarily in held-out test sets of data sampled from the model's training distribution; it is unclear whether zero-shot transfer holds true for out-of-distribution industrial data which is structurally different from publicly available benchmark sets.

While the literature does have examples of regular (and periodic) systems, like daily commuting networks [27, 28], the existing literature provides limited analysis regarding chaotic or rapidly shifting spatiotemporal systems, where the structure of the graph used in the model is changing as fast as the signal it carries, like cascading failure events on the grid or extreme weather events. Finally, there is a known gap in tools to communicate the outcomes of forecasting (compared to classification) results. Although explainability and interpretability are often treated as synonymous in practice, they are actually quite different, and most of the explainable AI techniques have been designed for classification problems and not regression over time problems that rely on attention weights that can be easily misunderstood as causal explanations [29].

Table 3. Identified Gaps and Required Innovation Pathways

Gap Area	Current Limitation	Evidence/Driver	Required Innovation
Concept Drift	Pipelines assume a static process between scheduled retrains	Well-studied in streaming machine learning, rarely integrated into deep forecasting deployment	Drift-aware retraining tied to residual monitoring
Zero-Shot Foundation Models	Strong zero-shot results skew toward each model's own pretraining distribution	TimeGPT, Lag-Llama, Moirai, Chronos, TimesFM report peak performance on aligned benchmarks	Leakage-controlled zero-shot benchmarks across industrial domains
Spatiotemporal Chaos	Graph models validated mainly on regular, recurring patterns	GNN traffic forecasting literature concentrates on stable, periodic networks	Chaos-aware regularization with adaptive graph structure learning
Explainability (XAI)	Attention weights misread as causal; XAI methods originate in classification	2025 survey finds explainability/interpretability conflated in financial forecasting	Forecast-specific XAI with calibrated temporal attribution

11. Conclusion

The historical trajectory does not demonstrate the complete replacement of ancestral frameworks; rather, successive iterations optimize specific computational bottlenecks while inheriting localized structural assumptions whilst keeping most of the other parameters. A linear, per-series formulation of ARIMA was replaced by approaches that were able to deal with covariates natively and that ignored the temporal ordering as an inductive bias. Recurrent networks removed that ordering, learning gating and not being able to be trained in parallel. The sequential bottleneck was removed, while the quadratic growth in memory was introduced, and is partially addressed by sparse and linear re-evaluations. Foundation architectures successfully transition computational overhead from local deployment phases into centralized offline pretraining and into one-time pretraining but empirical evidence of the actual level of zero-shot ability to go beyond each model's training distribution remains less substantiated than the theoretical capabilities of the underlying networks.

The following are the three findings of this review that are directly relevant to practice. First, the final conclusion of the M4 Competition which confers that hybrid statistical and machine learning systems are better than either approach alone but has not been contradicted by more recent deep learning work; indeed, the hybrid systems have been rephrased, with N-BEATS-style pure deep learning systems being close to comparable to well-designed hybrids and frequently failing to yield statistically significant improvements over hybrid configurations. Second, on some long-horizon benchmarks, simple linear models perform similarly to the complex Transformer models , which suggests that there is no direct correlation between model complexity and forecasting accuracy, and that the selection of multi-step training method is as crucial as the selection of attention mechanism. Third, each of the claims from the various classes of architecture this review covered is based on different originating studies and datasets, demonstrating that there is no universally agreed-upon leakage-controlled evaluation

protocol, yet broad enough to enable comparing the performance of statistical, machine learning, deep learning and foundation model forecasters. The next major stage in the field might not be a time of introducing another model architecture, but when that assessment gap is closed.

References

[1] Box, G. E. P., & Jenkins, G. M. (1970). Time series analysis: Forecasting and control. Holden-Day.

[2] Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 competition: 100,000 time series and 61 forecasting methods. International Journal of Forecasting, 36(1), 54–74. https://doi.org/10.1016/j.ijforecast.2019.04.014

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems (Vol. 30, pp. 5998–6008). Curran Associates.

[4] Garza, A., Challu, C., & Mergenthaler-Canseco, M. (2023). TimeGPT-1. arXiv:2310.03589.

[5] Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). The M5 accuracy competition: Results, findings, and conclusions. International Journal of Forecasting, 38(4), 1346–1364. https://doi.org/10.1016/j.ijforecast.2021.11.013

[6] Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: A survey. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 379(2194), 20200209. https://doi.org/10.1098/rsta.2020.0209

[7] Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted averages. International Journal of Forecasting, 20(1), 5–10. https://doi.org/10.1016/j.ijforecast.2003.09.013 (Original work published 1957)

[8] Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. Management Science, 6(3), 324–342. https://doi.org/10.1287/mnsc.6.3.324

[9] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

[10] Vapnik, V. N. (1995). The nature of statistical learning theory. Springer.

[11] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). Association for Computing Machinery. https://doi.org/10.1145/2939672.2939785

[12] Taylor, S. J., & Letham, B. (2018). Forecasting at scale. The American Statistician, 72(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

[13] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[14] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (pp. 1724–1734). Association for Computational Linguistics.

[15] Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. International Journal of Forecasting, 37(4), 1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

[16] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, pp. 11106–11115).

[17] Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Advances in Neural Information Processing Systems (Vol. 34, pp. 22419–22430). Curran Associates.

[18] Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2023). Are transformers effective for time series forecasting? In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 37, No. 9, pp. 11121–11128).

[19] Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers. In Proceedings of the International Conference on Learning Representations.

[20] Challu, C., Olivares, K. G., Oreshkin, B. N., Dubey, R., Mergenthaler-Canseco, M., & Garza, A. (2023). NHITS: Neural hierarchical interpolation for time series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 37, pp. 6989–6997).

[21] Rasul, K., Wu, C., Schneider, A., & Moor, T. (2024). Lag-Llama: Towards foundation models for probabilistic time series forecasting. arXiv:2310.08278.

[22] Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., & Sahoo, D. (2024). Unified training of universal time series forecasting transformers. In Proceedings of the 41st International Conference on Machine Learning (Vol. 235, pp. 53140–53164). Proceedings of Machine Learning Research.

[23] Ansari, A. F., et al. (2024). Chronos: Learning the language of time series. arXiv:2403.07815.

[24] Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting, 36(3), 1181–1191. https://doi.org/10.1016/j.ijforecast.2019.07.001

[25] Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In Proceedings of the International Conference on Learning Representations.

[26] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4), 1–37. https://doi.org/10.1145/2523813

[27] Wu, Z., Pan, S., Long, G., Jiang, J., & Zhang, C. (2019). Graph WaveNet for deep spatial-temporal graph modeling. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (pp. 1907–1913).

[28] Jiang, W., & Luo, J. (2022). Graph neural network for traffic forecasting: A survey. Expert Systems with Applications, 207, 117921. https://doi.org/10.1016/j.eswa.2022.117921

[29] Arsenault, P.-D., Wang, S., & Patenaude, J.-M. (2025). A survey of explainable artificial intelligence (XAI) in financial time series forecasting. ACM Computing Surveys, 58(2), 23. https://doi.org/10.1145/3733875