A Review On Breaking the Limits of Time Series Forecasting Algorithms

Ammar Raneez
Computer Science and Engineering
University of Westminster
London, UK
ammarraneez@gmail.com

Torin Wirasingha
Department of Computing
Informatics Institute of Technology
Colombo, Sri Lanka
torin.w@iit.ac.lk

***Abstract*—Time Series (TS) forecasting has stagnated owing to algorithm restrictions, therefore systems developed using these methods can only perform so well. TS remains a challenge despite recent advances in Deep Learning (DL) in Natural Language Processing (NLP) and Reinforcement Learning (RL). This paper reviews the literature on these algorithms, highlights studies using them, and shows their limits. Neural Ordinary Differential Equations (NODEs) with continuous-time and continuous-depth tackle TS forecasting issues. Liquid Time-Constant (LTC) networks, a more advanced and reliable implementation of these NODEs, provides fluidity. We propose a new design that uses the LTC's liquid adaptability and is more adaptable to manage immediate changes. These algorithms are more steady, adaptive, and versatile than DL, which may help overcome its TS forecasting shortcomings.**

Keywords—Time Series (TS) Forecasting, Neural Ordinary Differential Equations (NODEs), Liquid Time-constant Networks (LTCs), Ordinary Differential Equations (ODEs), Stochastic Differential Equations (SDEs)

I. INTRODUCTION

In this ever-changing world, the need for accurate forecasting has become more and more demanding. Time series (TS) forecasting has been studied for several decades, even before Artificial Intelligence (AI) became popular. Various approaches have been proposed throughout the years to produce effective forecasts; however, most have become obsolete.

Until recently, upon introducing neural ordinary differential equations (NODEs) [21], systems have all been utilizing these traditional and obsolete approaches for their forecasting requirements. This new family of Deep Learning (DL) algorithms has demonstrated promising results in their application in TS forecasting, piquing many academic researchers' interest in deviating from the traditional Deep Neural Network (DNN) architectures to implement more flexible, adaptable, and efficient models.

Although these models have produced great accomplishments, it has been noticed that they could be more stable and fail in modeling uncertainties [22]. Attempts were made to solve stability issues, the most influential and groundbreaking being the Liquid Time-Constant Network (LTC) [23]. However, the issue of modeling uncertainty still lingers.

In this paper, we initially delve into the vast ocean of TS forecasting algorithms and conduct a summarized review of the most popular ones. We then propose a novel architecture inspired by the LTC that could solve the issue of modeling uncertainty while also being robust and stable.

II. BACKGROUND

A. *Time Series Forecasting*

Many academics have been drawn to TS forecasting to provide a reliable, scalable, and practical solution. It may be extremely useful in solving various real-world issues, such as forecasting the weather, traffic, or patient EEGs in medical monitoring [1]. Any problem with a temporal component can be considered a potential domain for applying TS forecasting, which is one of many reasons for so much demand and significant impact on business.

TS forecasting is a significant business issue and an area where Machine Learning (ML) could create an impact. It is the foundation for contemporary business practices, including pivotal domains like customer management, inventory control, marketing, and finance. As a result, it has a comprehensive financial impact, with millions of dollars for subtle improvements in forecasting accuracy [2].

Until recent advancements in ML, traditional statistical models have been used with the help of domain expertise. However, with increasingly available data and computing power, ML has become vital in creating forecasting models for the next generation [3]. ML provides a means of learning temporal dynamics in a data-driven manner compared to their traditional counterparts [4].

In particular, DL has gained tremendous popularity in recent times for its remarkable accomplishments in image classification, Natural Language Processing (NLP), and Reinforcement Learning (RL), as it can learn representations from complex data without needing explicit engineering [3]. However, DL has hit a plateau in performance upon application in TS forecasting.

B. *Statistical-based Forecasting Techniques*

Statistical forecasting is an application of statistics to historical data to forecast what could happen in the future.

1. ARIMA

The Autoregressive Integrated Moving Average (ARIMA) model is a combination of the Autoregressive (AR) and Moving Average (MA) models. The ARIMA model predicts future values based on the past [5]. This specific model is the most popular in TS forecasting due to its considerably good performance; however, specific vital points that cause these models to produce inaccuracies need to be considered: it bases its future values on its past, therefore, can be inaccurate when used in open systems; it is pretty complicated and hence lacks expressivity; and, it cannot be used for seasonal data.

2. SARIMA

Seasonal Autoregressive Integrated Moving Average (SARIMA) is an extension to ARIMA that was introduced to handle seasonality in data. Research performed by [6] has proven that SARIMA produced better performance than ARIMA; however, its parameters are more sensitive to changes where even a slight change could result in poor performance.

3. GARCH

Generalized Autoregressive Conditional Heteroskedasticity (GARCH) is an approach to estimating market volatility while being more contextual than others when predicting prices or rates [7]. Research conducted by [8] identified that the forecasts attempted by GARCH were more accurate than ARIMA's as ARIMA cannot capture volatility in the dataset.

4. PROPHET

Facebook introduced an approach to solving the challenge of forecasting at scale via adjusting parameters [9]. It was designed to forecast daily data consisting of weekly and yearly seasonality, considering the effects of holidays [10]. Therefore, it performs best for highly seasonal data.

Is there a clear better choice? Upon understanding current statistical algorithms, research suggests that there is no transparent superior model as the differences in errors and accuracy are low, which signifies that the better-performing model will change according to the domain, the problem, and the dataset.

C. Deep Learning-based Forecasting Techniques

Studies have shown that certain TS forecasting domains perform better on statistical models (ex: [11]). At the same time, other studies have shown that DL models have outperformed statistical models (ex: [12]). Therefore, it is also expected to be worth evaluating DL models.

1. LSTM

Long Short-Term Memory (LSTMs) was introduced by [13] to solve the issues in previous RNNs of the inability to store information. They paved the path for more performant future RNNs as they remember short-term patterns, which made modeling temporal dependencies for larger horizons possible [1]. Studies conducted to identify the impact of LSTMs on TS proved multiple advantages, the most prominent being the ability to extract meaningful information from TS data.

2. GRU

Gated Recurrent Units (GRUs) are, in effect, a simplification of an LSTM where the ‘forget’ and ‘input’ gates are combined. GRUs achieve similar results to LSTMs while taking less computational time [1]. There is no clear distinction between GRUs and LSTMs on which one performs better; therefore, generally, both are attempted.

GRUs and LSTMs are similar in performance. Another stalemate arises as there is a dispute in the works of [14] and [15]: [14] demonstrated that the GRU performance was superior to ARIMA and LSTM; meanwhile, [15] observed better performance in LSTMs compared to ARIMA and GRU. Therefore, it is evident that there is no superior between GRU and LSTM. However, these non-linear models perform better than statistical models, in this case, ARIMA, which bestows more credibility on the study conducted by [16].

3. N-BEATS

Neural Basis Expansion for Interpretable Time Series (N-BEATS) is a relatively new state-of-the-art forecasting algorithm. The proposed architecture aimed at solving univariate point forecasting that carries multiple properties, the most impactful being interpretable and generalizable to multiple domains [17]. N-BEATS overcame the M4 competition’s previous winner, a hybrid statistical and neural net model, while being a pure DL architecture that is simple, generic, and expressive. This finding inspired the authors to challenge the claim of [18]: the future of TS forecasting is hybrid architectures of statistical and neural net models.

4. TFT

Temporal Fusion Transformer (TFT) is a novel attention-based architecture proposed by Google to solve the challenge of multi-horizon forecasting optimized explicitly for outstanding performance and interpretability – as existing solutions are typically a ‘black box.’ The architecture has specialized features that make it possible to choose required features and remove unnecessary ones [19]. In experimentation, it was identified that there were significant performance improvements compared to existing benchmarks. The proposed architecture was influenced upon reviewing existing transformer-based architectures (ex: [20]) that had yet to consider different types of inputs present in multi-horizon forecasting or assumed the required exogenous features are always available.

Are DL models superior to statistical models? Based on the literature, the belief that statistical models are superior at all times is only partially justifiable. It even cannot be deduced that non-linear DL models are superior. Therefore, it is worth investigating both schools of thought when solving TS problems and choosing the model that demonstrates better performance for that problem.

Please refer to the supplementary material S1 for a more detailed breakdown of these algorithms.

III. Concerns on Existing Used Techniques

A. Issues in statistical techniques

Based on the literature, the following drawbacks can be identified with statistical-based models.

- Linearity
- Suitability for mostly seasonal data
- Lack of interpretability and expressivity
- Altering model parameters requires more in-depth domain knowledge

B. Issues in Deep Learning techniques

As identified by [18], in contrast to other domains of NLP and computer vision, DL models still struggle in TS forecasting. Additionally, it is worth noting that although the non-linearity introduced by neural networks improves performance, they lack expressivity.

A general issue with the above models is that they are static and lack adaptability. TS data is volatile, ever-changing, and unpredictable; they can get unexpected characteristic changes to their inputs; therefore, a fixed statistical model or neural network has the possibility of struggling, which could be a reason for the identification of [19].

Given the open-ended conclusion on the comparison between statistical and DL models, the natural question is **how to proceed**.

IV. Neural Ordinary Differential Equations

NODEs are a new family of DL models that are continuous-depth, have constant memory cost, implicitly trade numerical precision for speed, and, most importantly, can adapt the evaluation strategy based on inputs [21].

Paper [21] claimed that RNNs with continuous-time hidden states determined by ODEs are effective algorithms for modeling TS data, proven by their ability to adapt computation depending on the input data. They proposed the following change to the classical equation followed by RNNs and residual networks.

Equation followed by traditional residual networks

$$h_{t+1} = h_t + f(h_t, \theta_t) \tag{1}$$

Equation proposed

$$\mathrm{d}h(\mathrm{t})/\mathrm{dt} = f(h_t, \mathrm{t}, \theta_t) \tag{2}$$

Where $h_t \in \mathbb{R}^D$, $t \in \{0 \ldots T\}$

The equation proposed utilizes an ODE specified by a neural network to "*parameterize the derivative of the hidden units*" instead of specifying a sequence of hidden layers.

The ability to adapt to incoming data streams could be a promising application in the domain of TS forecasting. However, these ODEs fail to identify uncertainties in predictions and are not robust [22].

A. *Liquid Time-Constant Networks*

Research [23] proposed a solution to improve the performance of NODEs, as experiments performed on other NODE architectures produced underwhelming performance compared to traditional LSTM architectures. The proposed architecture exhibits stable and bounded behavior alongside being highly expressive. The alternate formulation demonstrates 'liquid' features which give rise to better performance for TS predictions.

It is also speculated that the solution can learn during training and while in use by continuously changing the underlying formulations to adapt to the changes in incoming inputs [24]. What is impactful is that these networks can adapt to the changes in real-world systems while also being robust, stable, and more interpretable.

The architecture is richer in information while being scaled down than other neural networks, which adds another advantage of it being easy to glance into the 'black box' of the underlying network to understand the reasoning behind certain computations – as realized, is another drawback of standard neural networks.

The formula is an alternative to what [25] proposed. Instead of using a neural network to define the derivative, an additional term assists the system in reaching equilibrium [21].

V. Neural Stochastic Differential Equations

ODEs can be used to abstract deterministic models, as utilized by [21], to propose neural ODEs. Hence, the natural question arises: what **could be used if a state of randomness exists?**

SDEs are ODEs with additional noise, which can be used to model uncertainty. Neural SDEs are similar to NODEs in that an ODE can be used as the solver; however, the additional focus is the 'stochastic evolution' instead of the 'deterministic evolution' [26]. Considering NODEs and their expressive power, SDEs can enhance the expressive power even further, as proposed by [27].

The issue is that data prone to tiny and rapid changes cannot be modeled well by NODEs, as these types of data are bound to have noise. SDEs are suitable for fitting such data as they can model instantaneous changes via random noise.

VI. The Ultimatum - LTCs with SDE Flexibility

Based on the above analysis, it is unclear what exactly would be the next step or even the end proposition. The LTC architecture proposed by [23] utilizes the more obsolete ODE, which cannot model randomness and instantaneous changes. However, the usage of SDEs is more flexible as it handles randomness.

An intuitive next step that a researcher could investigate is obtaining inspiration from the LTC architecture, replacing the underlying ODE with an SDE instead, which manifests a novel and more flexible implementation capable of handling random noise.

A. *Formulation*

Considering the formulation proposed by the original LTC, the structure can be tweaked by utilizing a linear system of SDEs to declare the flow of the network instead of ODEs. The proposed linear ODE system by [29] can be improved by adding uncertainty to produce the following: $\mathrm{dx(t)/dt} = -\mathrm{x(t)}/\tau + \mathrm{S(t)} + \sigma(\mathrm{x(t)})\mathrm{B}$.

This can then be plugged into the LTC formulation instead of the ODE system to manifest the following novel formula. *The supplementary material S2 provide more in-depth proof.*

$$\frac{dx(t)}{dt} = -\left[\frac{1}{\tau} + f\big(x(t), I(t), t, \theta\big) - \sigma B\right] x(t) + f\big(x(t), I(t), t, \theta\big)A \tag{3}$$

The sole difference from the original LTC is the additional "$-\sigma B$" term which can model uncertainty.

B. *Forward propagation with implicit SDE solvers*

Paper [23] determined that their LTC architecture that uses a linear system of ODEs was 'stiff equations.' They also found that regular Runge-Kutta was not suitable for solving LTCs; therefore, they designed a custom ODE solver by combining both implicit and explicit Euler methods.

As this system uses SDEs, SDE solvers must be used. As [23] determined, the architecture is a system of stiff equations. Therefore, as [29] decided, researchers must use an implicit solver to ensure stability. Additionally, researchers can combine an explicit solver to achieve further stability. For this research, the authors will use an implicit SDE solver – creating a custom SDE solver by fusing an explicit solver within is something researchers should explore in the future.

To solve the state of these SDEs, we propose using implicit Euler-Maruyama solver, as it can handle all forms of noise and it is immensely popular. Having mentioned this, evaluating other implicit SDE solvers is recommended to determine whether there are more suitable options.

Note: L = number of steps

TABLE I. COMPLEXITIES OF BPTT AND ADJOINT SENSITIVITY

	BPTT	**Adjoint sensitivity**
Time	O(L)	**O(LlogL)**
Memory	O(L)	**O(1)**
Forward accuracy	High	High
Backward accuracy	**High**	Low

C. Training the network with BPTT

Training these networks has a trade-off between accuracy and memory. [21] promoted the use of the adjoint sensitivity method to perform reverse-mode Automatic Differentiation (AD), which is more memory efficient. Paper [23] mentioned that this method introduced more numerical errors and opted to use the traditional Back Propagation Through Time (BPTT) approach, which is more accurate but consumes more memory.

Although a technique of adjoints exists specifically for SDEs, they cannot be used, as determined by [26], and hence requires a custom-built backpropagation rule.

The authors suggest the BPTT approach to maintain high accuracy with the sacrifice of lower memory consumption, as it can be determined whether said architecture is more accurate. Researchers must investigate reverse-mode AD in the future as it is the recommended approach when memory efficiency is more important.

It is worth noting that using the BPTT approach carries added benefits, such as being able to be used as a Recurrent Neural Network (RNN) layer alongside popular optimization algorithms such as stochastic gradient descent (SGD) and Adam [30].

Table 1 summarizes the time and memory complexities of the vanilla BPTT and adjoint sensitivity approaches.

VII. CONCLUSION AND FUTURE WORK

We aimed to summarize existing TS forecasting algorithms and their respective applications, highlighting the issues of forecasting systems utilizing these algorithms. We then dive into a new family of DL algorithms that attempt to fix certain issues by providing continuous-time architectures while introducing other issues and not providing the most effective solution. We further dive into a more cutting-edge algorithm that attempts to fix these issues in NODEs; however, the drawback is that the internal system uses obsolete technology. We finally propose a new architecture heavily inspired by the LTC but by modifying the underlying linear system and proposing respective forward and backward propagation algorithms while also mentioning some compromises that require future research.

The domain of TS forecasting has been stagnant and hindered by the abovementioned more traditional statistical and DL techniques. However, recent advancements in DL have introduced NODEs – a new family of DL architectures that have attempted to surpass these limitations – some being extremely close. As this domain is relatively new, it is important to focus on this research area as the next big thing could be soon.

REFERENCES

[1] P. Lara-Benítez, M. Carranza-García, and J. C. Riquelme, "An experimental review on deep learning architectures for time series forecasting," *International Journal of Neural Systems*, vol. 31, iss. 3, pp. 2130001, Mar 2021.

[2] L. J. Chaman, "Answers to your forecasting questions," *Journal of Business Forecasting*, vol. 36, iss. 2, Summer 2017.

[3] B. Lim and S. Zohren, "Time series forecasting with deep learning: a survey," *Phil. Trans. R. Soc.*, vol. 379, iss. 2194, Feb 2021.

[4] N. K. Ahmed, A. F. Atiya, N. E. Gayar and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econometric Reviews,* vol. 29, iss. 5-6, pp. 594–621, Sep 2010.

[5] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*, 5th ed. John Wiley & Sons, 2015.

[6] M. Valipour, "Long-term runoff study using SARIMA and ARIMA models in the United States: runoff forecasting using SARIMA," *Meteorological Applications*, vol. 22, iss. 3, pp. 592–598, Jul 2015.

[7] R. F. Engle, "Autoregressive Conditional Heteroscedasticity with estimates of the variance of United Kingdom inflation," *Econometrica*, vol. 50 iss. 4, pp. 987, Jul 1982.

[8] S. P. Bhardwaj, R. K. Paul, D. R. Singh and K. N. Singh, "An empirical investigation of Arima and Garch models in agricultural price forecasting," *Economic Affairs*, vol. 59, iss. 3, pp. 415, Jan 2014.

[9] S. J. Taylor and B. Letham, "Forecasting at scale," PeerJ Preprints. Rep. 5, 2017.

[10] R. J. Hyndman and G. Athanasopoulos. (2021, May 31). *Forecasting: principles and practice* (3rd ed)[Online]. Available: https://otexts.com/fpp3/.

[11] R. Zhang, et al., "Comparison of ARIMA and LSTM for prediction of hemorrhagic fever at different time scales in China," *PLOS ONE*, vol. 17, iss. 1, pp. e0262009, Jan 2022.

[12] S. Siami-Namini, N. Tavakoli and A. Siami Namin, "A comparison of ARIMA and LSTM in forecasting time series," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Orlando, FL, 2018, pp. 1394-1401.

[13] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9 iss. 8, pp. 1735–1780, Nov 1997.

[14] L. Kuan, et al., "Short-term electricity load forecasting method based on multilayered self-normalizing GRU network," in *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, Beijing, 2017, pp. 1-5.

[15] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep LSTM recurrent networks," *Neurocomputing*, vol. 323, pp. 203–213, Jan 2019.

[16] M. Maiti, Y. Vyklyuk and D. Vuković, "Cryptocurrencies chaotic co - movement forecasting with neural networks," *Internet Technology Letters*, vol. 3 iss. 3, May 2020.

[17] B. N. Oreshkin, D. Carpov, N. Chapados and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," *arXiv* [Online], Feb 20 2020. Available: https://arxiv.org/abs/1905.10437.

[18] S. Makridakis, E. Spiliotis and V. Assimakopoulos, "The M4 competition: results, findings, conclusion and way forward," *International Journal of Forecasting*, vol. 34 iss. 4, pp. 802–808, Oct 2018.

[19] B. Lim, S. O. Arik, N. Loeff and T. Pfister, "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, iss. 4, pp. 1748-1764, Jun 2021.

[20] S. Li, et al., "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *2019 33rd Conference on Neural Information Processing Systems (NeurIPS)*, Vancouver, 2019.

[21] R. T. K. Chen, Y. Rubanova, J. Bettencourt and D. Duvenaud, "Neural Ordinary Differential Equations," *arXiv* [Online], Dec 14 2019. Available: https://arxiv.org/abs/1806.07366.

[22] S. Anumasa and P. K. Srijith, "Latent Time Neural Ordinary Differential Equations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36 iss. 6 pp. 6010–6018, Jun 2022.

[23] R. Hasani, M. Lechner, A. Amini, D. Rus and R. Grosu, “Liquid Time-constant networks,” *arXiv* [Online], Dec 14 2020. Available: https://arxiv.org/abs/2006.04439.

[24] MIT News | Massachusetts Institute of Technology. (2021, Jan. 28). *“Liquid” machine-learning system adapts to changing conditions* [Online]. Available: https://news.mit.edu/2021/machine-learning-adapts-0128.

[25] K. Funahashi and Y. Nakamura, “Approximation of dynamical systems by continuous time recurrent neural networks,” *Neural Networks*, vol. 6, iss. 6, pp. 801–806, Jan 1993.

[26] B. Tzen and M. Raginsky, “Neural Stochastic Differential Equations: deep latent Gaussian models in the diffusion limit,” *arXiv* [Online], Oct 28 2019. Available: https://arxiv.org/abs/1905.09883.

[27] S. Peluchetti and S. Favaro, “Infinitely deep neural networks as diffusion processes,” *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, vol. 108, pp. 1126-1136, 2020.

[28] L. Lapicque, “Recherches quantitatives sur l’excitation electrique des nerfs traitee comme une polarization,” *Journal de Physiologie et de Pathologie Generalej* vol. 9, pp. 620–635, Oct 1907.

[29] W. H. Press, *Numerical recipes: the art of scientific computing*, 3rd ed. Cambridge, UK; New York: Cambridge University Press, 2007.

[30] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, *arXiv* [Online], Jan 30 2017. Available: https://arxiv.org/abs/1412.6980.

[31] K. Cho, et al., Learning phrase representations using RNN encoder-decoder for statistical machine translation, Sep 2014.

[32] S. Bouktif, A. Fiaz, A. Ouni and M. Serhani, “Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: comparison with machine learning approaches †,” *Energies*, vol. 11 iss. 7, pp. 1636, Jun 2018.

[33] Y. Wang, W. Liao and Y. Chang, “Gated Recurrent Unit network based short-term photovoltaic forecasting,” *Energies*, vol. 11, iss. 8, pp. 2163, Aug 2018.

[34] I. Yenidogan, A. Cayir, O. Kozan, T. Dag and C. Arslan, “Bitcoin forecasting using ARIMA and PROPHET,” in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, Sarajevo, 2018, pp. 621–624.

[35] YouTube (2021, Jun. 2). *Directions in ML: Latent Stochastic Differential Equations: An Unexplored Model Class* [Online]. Available: https://www.youtube.com/watch?v=6iEjF08xgBg.

[36] U. Ugurlu, I. Oksuz and O. Tas, “Electricity price forecasting using recurrent neural networks,” *Energies*, vol. 11, iss. 5, pp. 1255, May 2018.

[37] C. Pan, J. Tan, D. Feng and Y. Li, “Very shortterm solar generation forecasting based on LSTM with temporal attention mechanism,” in *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, Chengdu, China, 2019, pp. 267-271.

[38] T. Fischer and C. Krauss, “Deep learning with long short-term memory networks for financial market predictions,” *European Journal of Operational Research* vol. 270, iss. 2, pp. 654-669, Oct 2018.

Supplementary Materials

S1. Summarized analysis

TABLE II. ANALYSIS OF FORECASTING ALGORITHMS

Ref.	Brief	Improvements/Contribution	Limitations/Future work
Statistical-based forecasting algorithms			
[5]	**ARIMA**. A statistical analysis model for understanding the dataset or predicting future trends. This model depends on past values to predict the future and uses lagged moving averages to smoothen the data.	Improved performance for TS forecasting data that correlate with values ahead of time.	Does not handle well with nonlinear data and long-term forecasting. Furthermore, it performs best on univariate analysis and cannot capture data volatility.
[7]	**GARCH**. A modeling technique that specializes in predicting volatility in data.	Captures volatility in datasets and boasts significant performance improvements in the family of statistical forecasting algorithms.	Needs to improve interpretability and adaptability.
[9]	**Prophet**. A modular regression model with interpretable parameters. These parameters can be adjusted according to the problem by domain experts, similar to ARIMA.	Solves forecasting at scale, where scale refers to three types. 1) A large number of people forecasting. 2) A large variety of problems. 3) A large number of forecasts being created.	It uses simple and weak assumptions and produces much poorer performance than ARIMA. And it does not model relationships between the past and future.
DL-based forecasting algorithms			
[13]	**LSTM**. An algorithm that learns to bridge minimal time lags by enforcing constant error flows. It learns much faster, creates more successful runs, and can solve complex tasks that have not been solved before.	Improved performance for short-sequence predictions. Overcame error back-flow problems present in conventional BPTT, where they tended to blow up or vanish.	Prediction capacity limits long sequence performance, where the MSE and RMSE rise unacceptably. Therefore, there are better solutions for predictions of the distant future. They are also prone to overfitting.
[31]	**GRU**. Similar architecture to that of LSTMs but combine the ‘forget’ and ‘input’ gates to create two gates, ‘reset’ and ‘update,’ instead of the three found in LSTMs.	Solve the vanishing gradient problem in RNNs as LSTMs, but also consume less memory and run faster.	Suitable for problems with smaller datasets and tend to be less accurate for datasets with larger sequences.
[17]	**N-BEATS**. An architecture that solves the univariate time series point	Outperformed the M4 competition winner of the previous year and	Tailored specifically for univariate TS analysis, therefore, would

	forecasting problem. It carries some benefits, some of which are being understandable, easily applicable to multiple other fields, and being fast to train.	improved the statistical benchmark forecast.	perform poorly on multivariate analysis. Additionally, Meta-learning is speculated to be a reason for the performance and must be investigated.
[19]	**TFT**. An attention-based architecture that solves multi-horizon forecasting with interpretability of the used inputs.	Demonstrate significant performance improvements over set benchmarks for a variety of datasets.	Training and inference times are expensive and require moderately extensive resources. Hardware optimizations can reduce these.
[23]	**LTC**. A novel formulation of the NODE architecture. Boasts superior expressivity that is capable of adapting to unforeseen changes.	Surpassed traditional DL and statistical models and overcame the underwhelming performance of other NODE architectures.	It cannot model uncertainty and is computationally intensive.

TABLE III. FEW STUDIES ASSOCIATED WITH THESE ALGORITHMS

Ref.	**Technology**	**Outperforms**	**Findings**
Statistical-based forecasting algorithms			
[11]	ARIMA	LSTM	ARIMA outperformed the LSTM model for monthly and weekly forecasts, while LSTM performed better for daily forecasts. Additionally, they mentioned that there is no clear superior.
[34]	Prophet	ARIMA	Prophet is strong to outliers and missing data. It is also optimized for business forecasts with trends and seasonality within and nonlinear data growths, which ARIMA cannot handle.
DL-based forecasting algorithms			
[14]	GRU	LSTM, traditional GRU	Scaled exponential linear units were proposed to deal with vanishing gradients. These architectures significantly outperformed LSTM and traditional GRU models.
[36]	GRU	MLP, LSTM	Performed much better than LSTM and trained faster as it is simply a simpler form of LSTM.
[33]	GRU	LSTM, ARIMA	K-Means clustering and Pearson coefficient were used to cluster groups and extract the most important features, respectively, which were then used to train the model.
[15]	LSTM	ARIMA, GRU	Genetic algorithms were used to create an optimized LSTM architecture.
[32]	LSTM	MLP, Linear regression	The most optimal time lags and number of layers for an LSTM was found using genetic algorithms.
[38]	LSTM	MLP, Logistic regression	Peeked into the internals of the LSTM to find common stock patterns in noisy data.
[37]	LSTM	MLP, traditional LSTM	Utilized a novel attention-based LSTM architecture to improve the performance of traditional LSTMs.
[17]	N-BEATS	Competition winner	Surpassed the previous winner of the M4 competition with a significant difference in performance. Concluded that hybrid models are only sometimes the best-performing.
[23]	LTC	LSTM, GRU, (Continuous time) CT-RNN, CT-GRU	A more stable implementation of the NODE can be built by using a linear system of ODEs to declare the network's flow.

S2. Proof of Proposed Formula

1. Transitioning from an ODE to an SDE

In simple terms, an SDE is an ODE with additional noise added at each step, which the model can use to model uncertainty.

$$\text{Assume an ODE is: } \frac{dx}{dt} = f(x); \textit{ which obtains the expected slope of } x(t) \quad (4)$$

The above ODE can be used to calculate the 'expected' slope, whereas the 'realized' slope differs from the 'expected' due to random noise, also called random Gaussian perturbations or Gaussian white noise. With that in consideration, the following can be derived:

$$\text{An SDE is: } \frac{dx}{dt} = f(x) + \mathcal{E}_{t+dt};\ \text{where } \mathcal{E}_{t+dt} \text{ is} \sim N(0,1) \tag{5}$$

Where $N(0,1)$ is a Gaussian 0,1 random variable

However, noise can be of varying intensities (some could be high, some could be low). Considering this varying intensity, the SDE can be further expressed as follows:

$$\frac{dx}{dt} = f(x) + g(x) * \mathcal{E}_{t+dt};\ \text{where g(x) is the intensity} \tag{6}$$

As implied, the missing factor in the existing architecture that consists of ODEs is the absent stochastic transition dynamics (i.e., a noise for each timestep – which is vital to model the tiny unobserved interactions). The above equation considers the small unobserved interactions and uncertainties that could occur; this is further important in the context of TS data, as the initial state of data is unlikely to be certain.

2. Adding neural networks into SDE dynamics

Based on the findings of [35], the noise mentioned in the previous step can be considered as Brownian motion, a generalized form of the Gaussian noise. Researchers can produce the following by plugging Brownian motion into the equation determined in the previous step.

$$dx = f\big(x(t)\big)dt + \sigma\big(x(t)\big)dB(t) \tag{7}$$

A neural network can be integrated into the above equation to solve the system, resulting in the following equation:

$$dx = f_\theta(x(t))dt + \sigma_\theta\big(x(t)\big)dB(t) \tag{8}$$

where f is usually a tiny neural network and θ are its parameters

3. Integrating the above equation into the LTC architecture

Moving back to the main problem at hand, the author can now construct a new formula by using the equation determined in the previous step.

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + S(t) \tag{9}$$

As the above equation is a linear system of ODEs initially proposed by [28], the author could add the uncertainty noise to the equation to produce the following:

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + S(t) + \sigma\big(x(t)\big)B \tag{10}$$

The above equation now defines a stochastic process instead of deterministic evolution. Therefore, researchers can model any tiny unobserved interactions. Finally, the following could be derived by applying this to the LTC formula:

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + S(t) + \sigma\big(x(t)\big)B \tag{11}$$

Replace S(t) with the nonlinearity proposed,

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + f(x(t), I(t), t, \theta)(A - x(t)) + \sigma\big(x(t)\big)B \tag{12}$$

Expand out the equation,

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} - f(x(t), I(t), t, \theta)x(t) + \sigma\big(x(t)\big)B + f(x(t), I(t), t, \theta)A \tag{13}$$

Lastly, refactor the equation into the format of the original LTC

$$\boxed{\frac{dx(t)}{dt} = -\left[\frac{1}{\tau} + f(x(t), I(t), t, \theta) - \sigma B\right]x(t) + f(x(t), I(t), t, \theta)A} \tag{14}$$