

Comparing Forecasting Models for Univariate Time-Series Analysis

Jasper Dong
University of Washington
chjdong@uw.edu

Abstract

Time-series Forecasting (TSF) is the problem of predicting future values given past measurements, that is highly applicable in the fields of economics, atmospheric sciences, computer science, and many others. Traditional forecasting models lay predominantly in the field of statistics, where models such as the Auto-Regressive Integrated Moving Average (ARIMA) model have been popular in their use due to their ability to accurately predict future lags of time series. Recently, with the rise in popularity of machine learning (ML) and artificial intelligence (AI) methods, deep learning (DL) models have been shown to be able to sufficiently tackle the TSF problem as well. Recurrent neural networks (RNNs) and Long-Short Term Memory (LSTM) models have been empirically shown to be able to outperform traditional methods in processing and predicting time-series data. However, limitations specific to these models still exist, and in such cases may lead to traditional methods outperforming DL models. In this comparison study, we show that for a small sample, traditional statistical methods will outperform DL models in TSF by about 44 - 45 percent in proportional reduction in error (PRE), while for a sufficiently large sample, LSTM models will outperform traditional statistical methods by about 48 - 55 percent in PRE.

1. Introduction

Time-series analysis is the practice of analyzing a sequence of data points measured over a period of time, in which time-series forecasting (TSF) is the problem of predicting future values given previously measured observations. This has considerable implications in stock market forecasting, anomaly detection, traffic flow prediction, and many others. In the past, the problem of TSF has dominantly been handled in the fields of mathematics and statistics. In particular, the Auto-Regressive Integrated Moving Average (ARIMA) model have been widely popular for univariate TSF, along with its many variates such as the multivariate ARIMA model ARIMAX, as well as the Seasonal ARIMA model SARIMA.

As data becomes much more complex, high-dimensional, and non-linear, traditional methods are unable to capture these complex data patterns due to the

strict limitations and assumptions that they impose on themselves. As such, the need for models that are able to capture these complex data patterns have shifted the focus towards machine learning (ML) and especially deep learning (DL) models, due to their inherent non-linearity and processing capabilities. In particular, recurrent neural networks (RNNs) and Long-Short Term Memory (LSTM) models are specifically designed to be able to process and predict sequential data. In the context of TSF, RNNs and LSTMs have been shown to massively outperform traditional methods [6]. Moreover, specific LSTM architectures have been developed to explicitly tackle TSF by adapting the vanilla LSTM architecture to be able to better handle the intricacies of time-series data [1][4].

However, RNNs and LSTMs are not without limitations. In general, neural network architectures are computationally intensive, especially with a larger number of hidden layers, along with certain choices of hyperparameters, activation functions, optimizers, and so on and so forth [3]. In particular, LSTMs are prone to overfitting, especially in situations with sparse training data. In the context of TSF, this can particularly rise in situations of short-term forecasting, where the training sample size is relatively limited.

This then raises the question if DL models truly do dominate traditional methods in every single way, or if there are situations in which these traditional methods are still preferable to modern DL models.

In this comparison study, we will compare the performance of a chosen traditional forecasting model - the SARIMA model - against popular DL model choices for forecasting, which are the vanilla LSTM model and the sLSTM model [4] which is specifically designed for TSF. We test these models against each other in two different settings: a short-term univariate forecasting setting in which the training sample size is relatively small, and a long-term univariate forecasting setting in which the training sample size is much larger in magnitude.

2. Related Work

The application of DL models to time-series forecasting is still very much an active area of research, with one major area of focus being on adapting current RNN and LSTM models to explicitly process time-series, and another major area of focus being on adapting Transformers to tackle the TSF problem. In particular,

Beck et al. (2024) [1] have developed the xLSTM architecture, which sought to update the vanilla LSTM model to be able to compete with modern-day Transformers in the area of natural language processing (NLP) by mitigating many of the limitations inherent in LSTM models. This includes the scalar LSTM (sLSTM) block which motivated future development of LSTM architectures specifically designed for TSF [4].

With a recent paradigm shift to Transformer models, much of the DL TSF focus has been shifted towards applying Transformers to TSF as well, which is still a relatively new area of research [7]. Because of this, there has also been many points of contention in the field on whether or not Transformers are appropriate for TSF in the first place, with many empirical studies showing that traditional DL models ranging from stacked multi-layer perceptions (MLPs) to one-linear layer models are still able to outperform Transformers in TSF [2][8].

Likewise, when re-visiting traditional DL models in TSF, areas of contention have also been in the validity of DL models in certain situations [3], especially over traditional statistical models [6]. However, the literature in this area of comparing DL models to traditional models is relatively sparse. Our work seeks to contribute to this area of research, by providing a direct and extensive comparison of traditional forecasting models against traditional DL models in TSF. While our work is certainly not completely exhaustive, we hope that this may motivate future work in exploring this research topic much more rigorously in the future.

3. Methods

3.1. Models

3.1.1 SARIMAX

The Seasonal Auto-Regressive Integrated Moving Average (SARIMA) model is a generalization of the $ARIMA(p, d, q)$ model, which contains the following hyperparameters:

- p = order of the Auto-Regressive (AR) component, which represents the number of lagged observations that the model autoregresses on in order to predict the current value
- d = degree of differencing, which represents the number of times the data is differenced to remove non-stationarity
- q = order of the Moving Average (MA) component, which represents the number of lagged forecast errors in order to adjust the current forecast

The specific formulation of the $ARIMA(p, d, q)$ model is then given by:

$$x_t = c + \sum_{i=1}^p \phi_i x_{t-i} + \epsilon_t + \sum_{i=0}^q \theta_i \epsilon_{t-i}$$

Where $\phi_i \neq 0$, $\theta_i \neq 0$, and $\sigma_\epsilon^2 > 0$. One could then think of $ARIMA(p, d, q)$ as a linear combination between past observations and past error terms.

Because of the inherent seasonality of our dataset, we use the generalized $SARIMA(p, d, q)(P, D, Q, s)$ model, which adds incorporates seasonality components into the base $ARIMA(p, d, q)$ model, where P is the seasonal AR order, D is the seasonal differencing, Q is the seasonal MA order, and s is the number of periods in each season.

The associated hyperparameters were selected using a standard step-wise model selection procedure using the Akaike information criterion (AIC).

3.1.2 Vanilla LSTM

In our implementation, we use a standard LSTM architecture with two LSTM layers, a hidden state dimension of 64, and a dropout rate of 0.2. The hidden representation is then passed through a fully connected linear layer in order to produce an output forecast, given an input sequential time-series. We then use an Adam optimizer with a learning rate of 0.001 to train the model. All of our implementations are done using standard PyTorch libraries.

3.1.3 sLSTM

We use a sLSTM-based architecture as our TSF-specific DL model to compare both the SARIMA and the vanilla LSTM models to. Originally developed for NLP use, the sLSTM model has been shown to also be useful in TSF as well, due to its increased adaptability to process the intricacies of time-series data. sLSTM differs from a vanilla LSTM model in that instead of using sigmoid or tanh gates, sLSTM uses exponential gates instead. Exponential gating is chosen due to its ability to capture a broader range of values that are emblematic of time-series data, as opposed to sigmoid or tanh functions which compresses values to a much tighter range [4]. To prevent values from the exponential gating to become too big, sLSTM also introduces a new normalization gate which sums up the product of the input gate times all future forget gates [1] (see Figure 1 for sLSTM structure). Our implemented sLSTM architecture uses the same hyperparameter choices as our vanilla LSTM architecture.

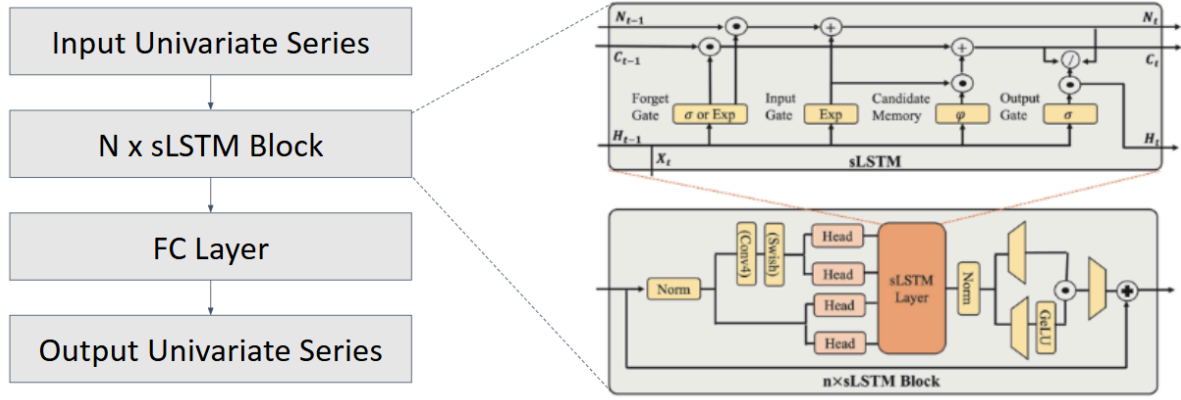


Figure 1: Overview of sLSTM architecture (Kong et al. 2025)

3.2. Dataset

We evaluate and compare the performance of our models on the popular Weather dataset, made available by the Max Planck Institute for Biogeochemistry [5]. This dataset has been extensively used in previous literature in assessing the performance of many forecasting models and TSF-specific DL models, among others [2][4]. The particular dataset we use contains observations of various weather indicators - such as temperature, air pressure, wind velocity, etc. - measured in 10 minute intervals over the span of a year. For simplicity of our experiments, we consider only the univariate case, where we only track the measurement of temperature ($^{\circ}\text{C}$).

We perform two different training and testing data splits in order to create two different forecasting situations. Our first split is representative of a short-term forecasting problem, where we resample our data into hourly intervals and consider a look-back window of $L = 72$ with prediction length $T = 24$. In this scenario, we train both DL models with a batch size of 32 and an epoch size of 50.

Our second split is representative of a long-term forecasting problem, where we resample our data into daily intervals and consider a look-back window of $L = 120$ with prediction length $T = 31$. In this scenario, we train both DL models with a batch size of 16 and an epoch size of 100. All experiments are conducted on the Google Colab Tesla T4 GPU.

4. Experiment & Results

4.1. Metrics

To assess the performance of our models, we use the Root Mean Squared Error ($RMSE$), which is a common metric used for assessing the accuracy of a model's prediction results [6]. The formulation of the $RMSE$ is as follows:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2}$$

Where N is the total number of observations, x_i is the true value, and $\hat{x}_i$ is the predicted value. We prefer to use the $RMSE$ because it penalizes large errors, and also scales the calculated scores back into the same units as the forecasted values, which aids in interpretability.

In addition, we also consider the Mean Absolute Error (MAE), which we see is another common metric to assess a model's predictive power. The formulation of the MAE is as follows:

$$MAE = \frac{1}{N} \sum_{i=1}^N |x_i - \hat{x}_i|$$

This provides another scale-dependent accurate metric that is useful in assessing our forecasting models.

To describe the comparative difference in predictive power, we use the Proportional Reduction in Error (PRE), which describes the percent reduction in predictive error of a model relative to another model's predictive error. The PRE is simply formulated as:

$$PRE = \frac{E_1 - E_2}{E_1}$$

Where for any two models Model 1 and Model 2, E_1 is the error metric associated with Model 1, and E_2 is the error metric associated with Model 2.

4.2. Results

Main Forecasting Results The quantitative results for TSF between our three models are shown in Table 1. We

find that in the short-term forecasting problem, the SARIMA model outperforms both the vanilla LSTM and the sLSTM model. Perhaps less interestingly, we also find that the vanilla LSTM outperforms the sLSTM, although this difference is very marginal. Comparing our SARIMA model with the other two models, we find that the *PRE* of SARIMA is about 44 to 45 percent, respectively to vanilla LSTM and sLSTM. Meanwhile, LSTM outperforms sLSTM with a *PRE* of about 0.6 percent.

TSF results for the short-term forecasting problem are also visualized in Figure 2, where we see that the forecast curve associated with the SARIMA model is marginally closer to the true prediction curve. We note that from an external validation perspective, while the SARIMA may outperform the DL models quantitatively, the forecast plot shows that the DL models may capture the overall structure of the true curve better than the SARIMA model.

When we consider the long-term forecasting problem, we see that both vanilla LSTM and sLSTM outperforms the SARIMA model, with a *PRE* of about 48 to 55 percent, respectively. Meanwhile, sLSTM outperforms vanilla LSTM with a *PRE* of about 4 percent.

Long-term forecast results are visualized in Figure 3, where we see that both DL models are able to follow closely to the true prediction curve, while the SARIMA model is very conservative in its forecasted values.

Model	MAE	RMSE
SARIMA	5.5410	6.5480
LSTM	7.9800	8.2750
sLSTM	8.0340	9.4430
SARIMA (Long)	5.7720	6.7890
LSTM (Long)	3.8980	4.9040
sLSTM (Long)	3.7160	4.6470

Table 1. MAE and RMSE Comparisons between SARIMA, vanilla LSTM, and sLSTM models. (Long) indicates that the model applied in a long-term forecasting setting.

Impact of Epoch Size In our vanilla LSTM and sLSTM implementations, we are also interested in seeing how our epoch size affects the performance of both models. From both Figure 4 and 5, we can see training loss comparisons between our vanilla LSTM and sLSTM models in the

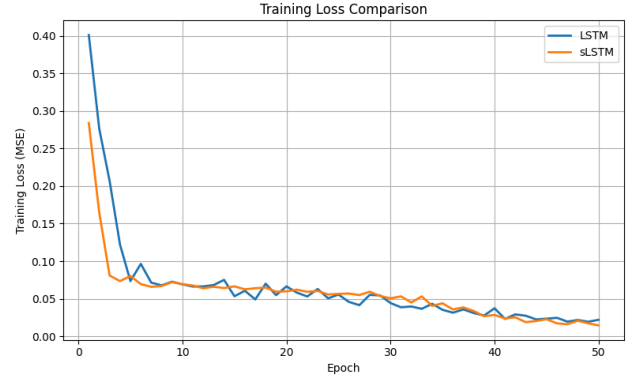


Figure 4. Exploration of epoch size on training loss of vanilla LSTM (blue) and sLSTM (orange) in short-term forecasting

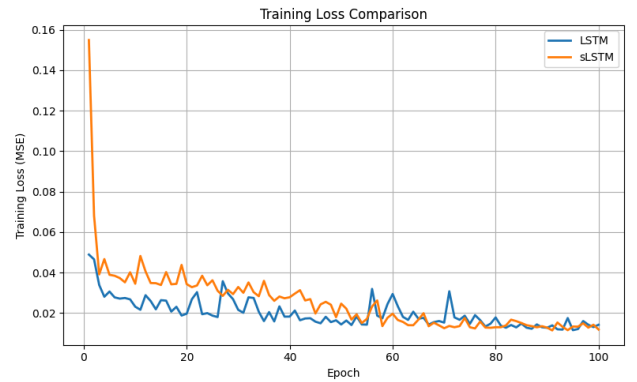


Figure 5. Exploration of epoch size on training loss of vanilla LSTM (blue) and sLSTM (orange) in long-term forecasting

short-term forecasting and the long-term forecasting situations, respectively.

From both of our training loss comparison plots, we see that training losses of both DL models decrease and stabilize with increasing epoch size, which aligns with what we expect in the relationship between training loss and epoch size.

We note that in the short-term forecast scenario, we see that while the vanilla LSTM is generally outperforming sLSTM, we see that sLSTM does start to stabilize better than vanilla LSTM at higher epoch sizes. We see that same type of behavior in the long-term forecast scenario, where sLSTM will display higher training losses than vanilla LSTM, but will gradually decrease and stabilize relative to vanilla LSTM with higher epoch sizes. This motivates a future need to conduct a more intensive comparison study between vanilla LSTM models and adjusted LSTM models such as sLSTM.

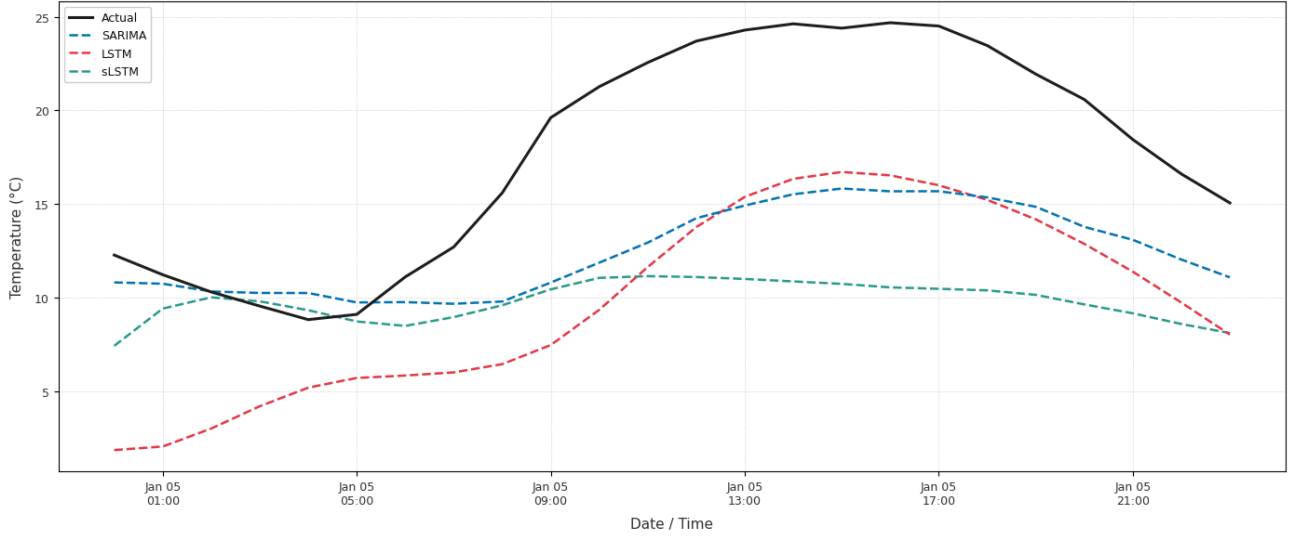


Figure 2. Short-term Forecast Results of SARIMA (blue), vanilla LSTM (red), and sLSTM (green)

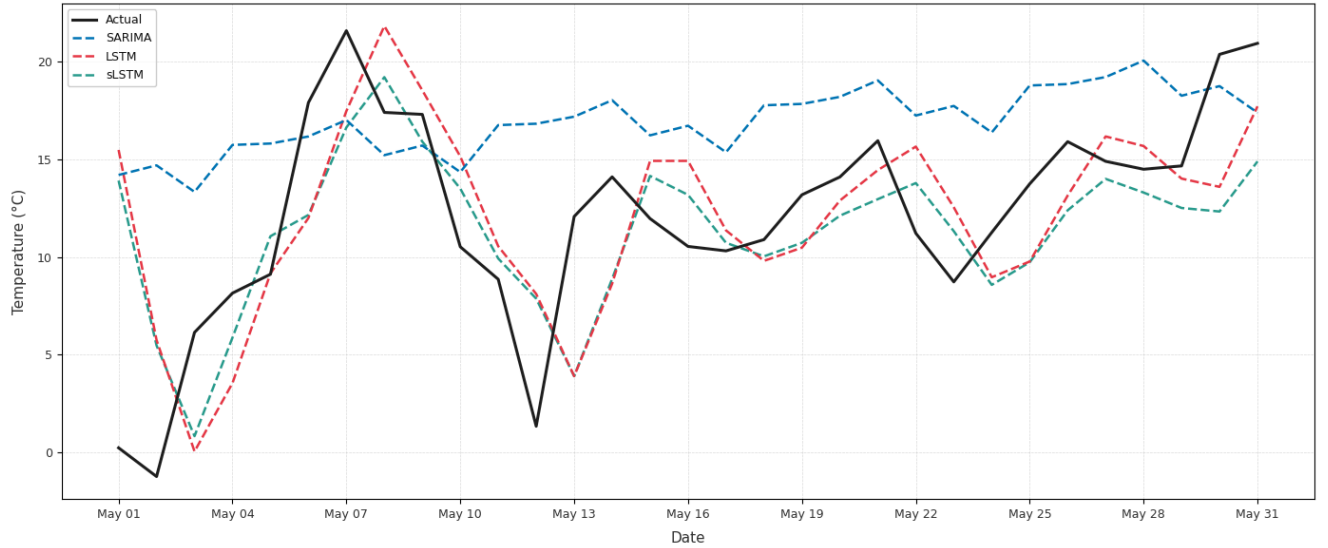


Figure 3. Long-term Forecast Results of SARIMA (blue), vanilla LSTM (red), and sLSTM (green)

5. Discussion

Our comparison study interestingly finds that in short-term forecasting situations, traditional statistical models that are appropriate for the specific forecasting task will outperform neural network models. This could be because in such cases, the structural limitations and assumptions that models such as ARIMA and SARIMA impose upon themselves are close to negligible, and can be taken for granted. An additional explanation could also be that neural network models rely on large training sample sizes to learn the complex data patterns that they are able to process. In cases where such sample sizes are relatively sparse, there may be too much noise for neural network models to accurately capture the true underlying pattern.

Vice versa, our study finds that in long-term forecasting situations, neural networks will massively outperform traditional statistical models. This is expected of our neural network models, which are known to be able to accurately process and predict long sequential data. This is equally expected of our SARIMA model, where the underlying assumptions are more likely to be violated with increasing sample sizes.

In addition to our main results, we also find that the choice of gating functions can have marginal effects on the predictive power of a LSTM. In particular, sLSTM's choice to use exponential gates instead of sigmoid or tanh gates leads to overall smaller training losses in long-term forecasting problems. Because this is not a main focus of our study, we only show these results

empirically. We hope that this motivates future work in exploring how LSTM design choices can lead to better performances, especially in the area of TSF.

It is worth noting the limitations of our current study. It is hard to tell what kind of forecasting counts as “short” and “long” - as is such the case of determining sufficiently large enough sample sizes - and is ultimately determined by the nature and dynamics of each individual dataset. Additionally, since this is purely an empirical study, we do not provide any theoretical guarantees. Therefore, a theoretical comparison between traditional statistical methods and neural network models in TSF is further needed to better understand just how well each of these models performs against each other. Finally, the forecasting situations that we have set up are very simplified types of forecasting. Specifically, we have only considered univariate time-series data in this study. A look at more complex types of data such as multivariate time-series would be the natural next step in comparing traditional statistical models against neural network models.

6. Conclusion

With a new paradigm shift into ML and DL models, these types of models are increasingly being applied in every domain imaginable at an alarming rate. With the high popularity of such models, one must ask if traditional methods still have uses in these domains today or if they have been entirely phased out by more modern methods. This study empirically shows that such traditional methods still have their uses, especially in cases where the limitations of modern ML methods are obvious. On the other hand, these modern methods are exceptional in their ability to process massive amounts of complex data, which are emblematic of contemporary real-world datasets. We hope that this study motivates a line of future work that more closely looks at how traditional methods compare to modern ML models, and also how ML models compare against each other, especially in situations that are more and more representative of real-life datasets.

References

- [1] Beck, Maximilian, et al. “xLSTM: Extended long short-term memory.” *Advances in Neural Information Processing Systems*, 37 (2024): 107547-107603
- [2] Chen, Si, Chun-Liang Li, Nate Yoder, Sercan Ö. Arik and Tomas Pfister. “TSMixer: An all-MLP Architecture for Time Series Forecasting.” *Trans. Mach. Learn. Res.* (2023): n. pag.
- [3] Kandadi, Thirupathi and Shankarlingam, G. “Drawbacks of LSTM algorithm: A case study” (2025)
- [4] Kong, Yaxuan, et al. “Unlocking the power of LSTM for long term time series forecasting.” *Proceedings of the AAAI Conference on Artificial Intelligence*, 39 (2025)
- [5] Kolle, Olaf, “Documentation of the weather station on top of the roof of the institute building of the Max-Planck-Institute for Biogeochemistry,” (2025)
- [6] S. Siامي-Namini, N. Tavakoli and A. Siامي Namin, “A comparison of ARIMA and LSTM in forecasting time series,” *17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, (2018): 1394-1401
- [7] Wen, Qingsong, et al. "Transformers in time series: A survey." *32nd International Joint Conference on Artificial Intelligence* (2022).
- [8] Zeng, Ailing, et al. "Are transformers effective for time series forecasting?." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 37. No. 9. 2023